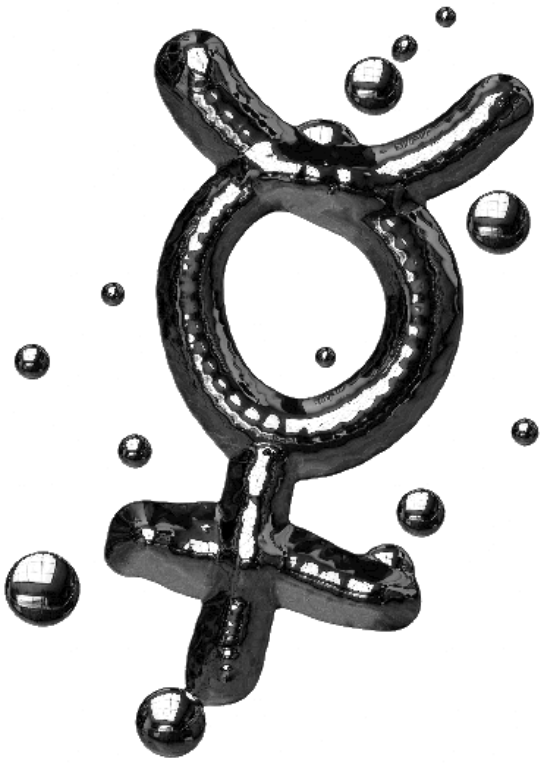
HP zx1 ioa ERS

External Reference Specification
Ropes to AGP/PCI/PCI-X Bridge



Technical Computing Division

Revision 3.2
April 18, 2003

A-2AG3-9002-1

Contents

List of Figures

List of Tables

List of Registers

Chapter 1

Introduction

1.1 Document Objectives

The primary objectives of this document are to:

- Describe the programming level visible functionality of the HP zx1 ioa
- Document the HP zx1 ioa programming information and connectivity guidelines for system level programming such as kernel work or driver writing

1.2 Definitions

Rope	An HP-proprietary connection between a rope host and rope guest. The HP zx1 ioa can use 1 or 2 ropes. Ropes are point to point connections, and are allocated to rope guests when the system is designed. The HP zx1 ioa supports the second generation (”enhanced”) version of ropes.
Rope Host	For the purposes of this document, this is the HP zx1 mio. In the HP zx1 mio-based systems, there is only 1 rope host.
Rope Guest	For the purposes of this document, this is the HP zx1 ioa. There are several rope guests in a HP zx1 mio-based system.
PIO	Processor I/O; these I/O transactions are initiated by the processor. An example could be the processor doing a 32-bit store to memory mapped I/O (MMIO) space.
OL*	OnLine replace/delete/add; replace/delete/add an I/O card while the system is up and running.
IOPA	I/O Physical Address.
IOVA	I/O Virtual Address; address falls into the HP zx1 mio’s Virtual I/O space.

1.3 Register Usage Note

Many registers in this document have reserved fields with values of “?”. These fields should not be changed. This can be done via byte enables or doing read-modify-writes to these registers, ensuring that the reserved fields are written back to the original values. Failing to do this could lead to all sorts of functional problems. Since the HP zx1 ioa registers are typically not written in performance paths, this should not cause performance problems.

1.4 Feature Set

1.4.1 The HP zx1 ioa is/does

- Ropes-to-PCI, PCI-X, and AGP 2.0 for enhanced-rope systems such as the HP zx1 mio.
- Optimized for DMA performance, even at expense of PIO.
- Supports AGP fast writes (only to addresses <4G).
- Peer-to-peer (P2P) for PCI and PCI-X transactions supported for writes only, low performance, “memory” only (no I/O port or config writes).
- Support 6 PCI slots and 10 wire interrupts. (For AGP and PCI-X, these are reduced to the limits imposed by the respective bus spec.)
- Support OL* in conjunction with the system design

1.4.2 The HP zx1 ioa is not/does not

- 5V PCI capable.
- Optimized for PIO performance.
- Compatible with follow-ons to AGP 2.0.
- Support OL* with AGP.
- Provide the “Hot Plug Controller” functionality described in the *PCI Hot Plug Specification Revision 1.0*. The HP zx1 ioa is an OL* capable bridge, not an OL* controller.
- General peer-to-peer solution.

Chapter 2

Operational Overview

2.1 I/O Bus Type Selection

2.1.1 How I/O bus type is determined

I/O Bus type is determined by a combination of external configuration pins sampled during power-on reset, and a software-initiated sampling of PCIXCAP. The external configuration pins control the selection of AGP or OLR capable buses. The software-initiated sampling of PCIXCAP doesn't select the PCI/PCI-X bus type — it just gives information to allow software to select the bus type. The definition of the configuration pins is covered in Section 3.2 on page 9.

2.1.2 When I/O bus type is determined

- The configuration bits are sampled during power-on reset. Their value is remembered through all subsequent resets. Reason: Changing things such as OLR mode after power-on makes no sense.
- PCIXCAP should be sampled during I/O bus reset. Reason: An initialization sequence must be driven on the bus when reset is removed, so bus type must only be changed during reset. In reality the sampling of PCIXCAP could be done with the bus not in reset, as long as the bus is reset before changing the bus type. Not every I/O bus reset will require re-sampling PCIXCAP, but re-sampling is required as part of an OLR operation as the card type may have changed.

2.2 Supported I/O Buses

2.2.1 AGP

AGP support includes Fast Writes (FW), DMA reads, and DMA writes. For optimal performance, AGP DMA transfers should be 64 or 128 bytes. PIO, configuration, message-signalled interrupt (MSI), and limited peer-to-peer (P2P) support is also included, but is detailed in Section 2.2.2 because it uses only the PCI subset of the AGP functionality.

AGP supports 1.5V signaling only.

The AGP spec allows PCI and AGP style cycles to be intermixed. Thus the AGP mode implicitly includes a complete 32-bit PCI implementation. AGP mode does not support PCI Interrupt Acknowledge.

2.2.2 PCI

Since the AGP spec requires PCI-style cycles as a subset of its functionality, operation on a pure PCI bus is largely just a matter of using the PCI-style subset with a different connector style. However, there are a few differences, which require some parts of the logic not to be shared directly.

- Pure PCI mode contains two submodes — NORMAL and OLR. In OLR mode, only one or two REQ#/GNT# pairs are active because only one or two I/O bus slots are supported. The remaining REQ#/GNT# pins take on new meanings related to online replacement functionality. In AGP mode, only one REQ#/GNT# pair is supported, and a considerably different arbitration algorithm is used.
- On a pure PCI bus, the AD bus may be wired as 64 or 32 bits. In the 64-bit case, transactions may negotiate dynamically and independently between the 32/64 widths for address and data using the REQ64# and ACK64# signals. On an AGP bus, only the 32-bit subset is used; the REQ64# and ACK64# signals are unused.
- On a pure PCI bus, more wire interrupts (INT#) are supported.

PCI PIO support, including both MMIO and legacy I/O port space is included, as are reads and writes of configuration space by the HP zx1 ioa. Supported peer-to-peer functionality includes only memory writes.

PCI signaling is 3.3 volt only.

2.2.2.1 PCI 2.2 specification exceptions

The HP zx1 ioa does not strictly follow the following sections of the *PCI Local Bus Specification*, Revision 2.2:

Section 3.7.2 requires that address parity be checked. Strictly speaking, all address phases visible to a target must be checked. The HP zx1 ioa follows the spirit of the specification that allows devices to only check parity when information is consumed. When a 64 bit master drives a 64 bit address, some of the address information is repeated. The HP zx1 ioa only checks parity on the address cycles that it uses to do decoding and ignores the redundant information. This exception was done intentionally since parity errors on cycles that we are not using cannot hurt us.

2.2.2.2 Optional PCI 2.2 specification exceptions

The HP zx1 ioa can be programmed to not follow the PCI specification. These options are provided for situations where strict adherence may not be the best solution. The sections of the *PCI Local Bus Specification*, Revision 2.2, that may optionally be violated are as follows:

Section 3.5.1.2 requires that no more then 8 subsequent wait states may be inserted by the target. The HP zx1 ioa can be programmed to insert more wait states then the maximum allowed. The default is to insert up to the legal maximum.

The wait state maximums were chosen to maximize bus efficiency. However, if a DMA read is going to be disconnected because the read stream has dried up due to a cache miss, inserting more wait states may allow one to avoid the disconnect. Not disconnecting a master that wants to continue his burst means avoiding the time it takes to restart the flow of DMA read data from the rope host.

Section 3.3.3.3.2 requires that a Delayed Completion Resource (DCR) compare:

- Address[63:0]
- State of REQ64#
- Bus Command (MR, MRL, or MRM)
- Address Parity

The HP zx1 ioa additionally compares Master ID, which is taken from the GNT# wires of the PCI arbiter. The HP zx1 ioa only uses a DCR on DMA reads. Since DMA reads are prefetchable, comparing Byte Enables is optional.

The HP zx1 ioa by default will strictly follow the PCI specification; optionally it may be programmed to use a relaxed comparison that ignores:

- Address[63:44] and Address[1:0]
- State of REQ64#
- Bus Command (MR, MRL, or MRM)

The HP zx1 ioa compares Master ID, which the spec does not require, but doing so helps to guarantee proper operation in most situations. The reader should keep in mind that Master ID does not help in determining the correct master if two masters are under the same PCI-to-PCI bridge.

It is easy to envision scenarios where a master might inadvertently change the bus command or REQ64# when a cycle is retried. For instance, more room may open up in an internal FIFO. This behavior seems much more likely

than the chance of a single card having 2 outstanding read requests to the same 8-byte address (but with a different bus command, state of REQ64#, etc.), having the value change between the 2 requests, and having it matter that the value changed. The kinds of cards that would even tend to support more than one outstanding request would be operating on multiple threads or tasks, in which case they probably wouldn't be sharing a cacheline of data.

2.2.3 PCI-X

When the HP zx1 ioa is wired to a PCI-X connector, it can end up using PCI style transactions or PCI-X style transactions. This selection is determined at reset time, so there is no cycle-by-cycle switching between these cycle styles.

Therefore, unlike AGP, the PCI-X functionality need *not* contain any PCI subset. When PCI-style signaling is required on a PCI-X bus, the HP zx1 ioa switches over entirely to the PCI mode described above.

Although it shares the same I/O bus pins and much of the same datapath with PCI/AGP modes, PCI-X mode is essentially independent of the other modes, sharing control logic only in cases where such sharing makes implementation or verification easier. It is expected that the master and slave controllers for PCI-X will be largely independent of their PCI/AGP counterparts, but as much as possible of the remaining the HP zx1 ioa logic will be shared and non-modal.

PCI-X support includes DMA reads and writes optimized for transfers of one cacheline (128 bytes) or longer. Shorter transaction sizes are supported as required by the PCI-X spec but are not optimized for performance. PIO support, including both MMIO and legacy I/O port space is included, as are reads and writes of configuration space by the HP zx1 ioa and MSI's.

PCI-X signaling is 3.3 volt only.

2.2.3.1 PCI-X 1.0a specification exceptions

The HP zx1 ioa does not strictly follow the following sections of the *PCI-X Local Bus Specification*, Revision 1.0a:

Section 5 requires that address parity be checked. The HP zx1 ioa only checks parity on the address cycles that it uses to do decoding, ignoring redundant address cycles. This exception (and justification) come from the HP zx1 ioa's PCI behavior. See Section 2.2.2.1 on the preceding page for more details.

Section 5.4.1 requires that parity be checked on the data phase of a read transaction which was terminated with a Split Response. As PCI-X 1.0a, Section 5.4.1.3 points out, "A data parity error in a Split Response indicates the existence of a serious problem in the system, but does not imply that the read data in the subsequent Split Completion is erroneous. It is possible that the read data in the subsequent Split Completion will arrive without error."

Using the same justification that we do not check parity when data is not consumed, and noting that nothing is consumed in the data phase of a Split Response, the HP zx1 ioa ignores data parity errors in Split Responses. Similarly, the HP zx1 ioa will ignore PERR assertions when driving a Split Response to a DMA read.

An additional justification for this exception is that some cards may not drive correct parity on Split Responses. Since they know no data is consumed, sloppy designers are likely to skip over this requirement. By not checking parity here, we can be tolerant of bad card behavior with no ill effect.[1]

Section 5.4.4 requires that if a DMA read split completion that the HP zx1 ioa masters, target-aborts or master-aborts, and reading memory has side effects, the Split Completion Discarded bit in the PCI-X status register should be set. Of course the HP zx1 ioa has no idea if a particular rope host is architected such that reads from memory have side effects. Under these conditions, the HP zx1 ioa will assert SERR#. This "exception" shouldn't really be an issue since the HP zx1 ioa relies on implementation specific error logging to a much greater extent than any of the PCI/PCI-X status register information.

2.3 Supported I/O Bus Cycles

Table 2.1 on the following page shows the cycle types that the HP zx1 ioa can *respond to* on the various I/O buses. Note that the entries labeled as PCI apply to both PCI and AGP buses. They also apply to the PCI-X bus if the reset-time

[1] It is wise to be strictly compliant in what you drive out on a bus, and very forgiving in what you receive back. The HP zx1 ioa attempts to do this as much as possible.

negotiation of bus type indicates that the bus is downgraded to PCI. These are mostly DMA related, but entries also exist for PIO split read returns and peer-to-peer writes.

Table 2.2 on the next page shows the cycle types that the HP zx1 ioa can *generate* on the various I/O buses. Note that the entries labeled as PCI apply to both PCI and AGP buses. They also apply to the PCI-X bus if the reset-time negotiation of bus type indicates that the bus is downgraded to PCI. These are mostly PIO related, but entries also exist for DMA read returns and peer-to-peer writes.

Bus type	Cycle Type	Length	HP zx1 ioa responds?	HP zx1 ioa non-error responses
PCI-X	Interrupt Acknowledge	DWORD	NO	n/a
PCI-X	Special Cycle	DWORD	NO	n/a
PCI-X	I/O Read	DWORD	NO	n/a
PCI-X	I/O Write	DWORD	NO	n/a
PCI-X	Memory Read DWORD	DWORD	YES	SR, R
PCI-X	Memory Write	Burst	YES†	DT, DNA, SDPD†, R†
PCI-X	Config Read	DWORD	NO	n/a
PCI-X	Config Write	DWORD	NO	n/a
PCI-X	Split Completion	Burst	If matches PIO	DT (PIO read)
PCI-X	Dual Address Cycle		*— See line for specific cycle —*	
PCI-X	Memory Read Block (or alias)	Burst	YES	SR, R
PCI-X	Memory Write Block (or alias)	Burst	YES†	DT, DNA, SDPD†, R†
PCI	Interrupt Acknowledge	DWORD	NO	n/a
PCI	Special Cycle	DWORD	NO	n/a
PCI	I/O Read	DWORD	NO	n/a
PCI	I/O Write	DWORD	NO	n/a
PCI	Memory Read	Variable Burst	YES	R, DLY
PCI	Memory Write	Variable Burst	YES†	R†, IMM†
PCI	Config Read	Variable Burst	NO	n/a
PCI	Config Write	Variable Burst	NO	n/a
PCI	Memory Read Multiple	Variable Burst	YES	R, DLY
PCI	Dual Address Cycle		*— See line for specific cycle —*	
PCI	Memory Read Line	Variable Burst	YES	R, DLY
PCI	Memory Write and Invalidate	Variable Burst	YES†	R†, IMM†
AGP	Read	Burst	YES	PIPE
AGP	Read (hi priority)	Burst	YES	See Read.
AGP	Write	Burst	YES	PIPE
AGP	Write (hi priority)	Burst	YES	See Write.
AGP	Long Read	Burst	YES	PIPE
AGP	Long Read (hi priority)	Burst	YES	See Long Read.
AGP	Flush	DWORD	YES	PIPE (Similar to Read)
AGP	Fence	No Data	YES	n/a. Effects HP zx1 ioa state only.
AGP	Dual Address Cycle		*— See line for specific cycle —*	

†Only these responses may occur for peer-to-peer on initiating bus.

PCI-X Response key:

SR	Split Response.
SDPD	Single Data Phase Disconnect.
DNA	Disconnect on next ADB.
DT	Data Transfer.
R	Retry.

PCI Response key: (includes PCI-style cycles on AGP)

R	Retry.
DLY	PCI delayed semantics. Card sees retry, sees immediate data on later attempt.
IMM	Immediate data transfer.

AGP Response key:

PIPE	Accept request. Enter into pipeline. (The only possible response on AGP cycles.)

Table 2.1: Cycles responded to by the HP zx1 ioa as an I/O bus target

Bus type	Cycle Type	Length	HP zx1 ioa generates?
PCI-X	Interrupt Acknowledge	DWORD	YES
PCI-X	Special Cycle	DWORD	NO
PCI-X	I/O Read	DWORD	YES
PCI-X	I/O Write	DWORD	YES
PCI-X	Memory Read DWORD	DWORD	YES
PCI-X	Memory Write	Burst	YES†
PCI-X	Config Read	DWORD	YES
PCI-X	Config Write	DWORD	YES
PCI-X	Split Completion	Burst	YES (DMA reads)
PCI-X	Dual Address Cycle	n/a	YES
PCI-X	Memory Read Block	Burst	YES
PCI-X	Memory Write Block (or alias)	Burst	NO
PCI	Interrupt Acknowledge	DWORD	YES††
PCI	Special Cycle	DWORD	NO
PCI	I/O Read	DWORD	YES
PCI	I/O Write	DWORD	YES
PCI	Memory Read	Variable Burst	YES
PCI	Memory Write	Variable Burst	YES†
PCI	Config Read	Variable Burst	YES
PCI	Config Write	Variable Burst	YES
PCI	Memory Read Multiple	Variable Burst	NO
PCI	Dual Address Cycle	n/a	YES
PCI	Memory Read Line	Variable Burst	NO
PCI	Memory Write and Invalidate	Variable Burst	NO
AGP	Fast Write	Variable Burst	YES
AGP‡	ST=000 LP Read Return	Burst	YES (DMA read or flush)
AGP‡	ST=001 HP Read Return	Burst	YES (DMA read)
AGP‡	ST=010 LP Write Data	Burst	YES (DMA write)
AGP‡	ST=011 HP Write Data	Burst	YES (DMA write)

†Only PCI/PCI-X Memory Write transactions may occur for peer-to-peer on target bus.

††Only in pure PCI mode; PCI on AGP does not support this.

‡AGP spec classifies these various ST+GNT permutations as “arbiter status signals,” rather than explicit cycle types. For all practical purposes, they are distinct cycle types on the bus, and are treated as such in this summary.

Table 2.2: Cycles mastered by the HP zx1 ioa as an I/O bus initiator

Chapter 3

Firmware Considerations

3.1 Hints to Driver Writers

3.1.1 AGP

For a DMA model, the HP zx1 ioa performs best using side band addressing (rather than PIPE#). If a card is capable of either, enable SBA. As far as request length, cache-line size transfers are the most efficient. A cache-line is 128 bytes for many of the newer processors. Normally, “bigger is better” for request size. However, this isn’t true beyond 128 bytes. There is no improvement going from 128 bytes up to about 192 bytes, and there is actually a loss of bandwidth as transfer sizes exceed 192 bytes. This is due to the amount of buffering in the HP zx1 ioa.

Of course, typical AGP DMA read transfers are 32 or 64 bytes long; 128 byte requests require using a “long read” and there are few AGP graphics cards that support them. The HP zx1 mio will handle naturally aligned[1], contiguous, consecutive 64 byte read requests at full bandwidth; naturally aligned contiguous, consecutive 32 byte reads will run at about 820 MB/s. The key here is to provide combinable request streams if possible. This could include aligning graphics visible data structures appropriately when performance is important. As always, benchmarking can help determine if this will be useful or not. As an aside, PCIX requests are typically much larger than a mere cache line, so this is not an issue with that protocol.

the HP zx1 ioa only supports AGP FWs to addresses below 4G. The AGP spec was unclear about if these should be supported. In addition, FW cycles to address above 4G would result in a 10% performance loss due to the extra address cycle. As a result, the HP zx1 ioa simply does not support them. A fatal error will be generated if they are attempted.

3.2 Power-up Initialization & Configuration

3.2.1 Power-up Configuration Bits

The following table describes the function of the HP zx1 ioa power-up configuration bits (PCI_INT_L[6:3]). When PWR_ON_RST_L deasserts, their value is latched inside the HP zx1 ioa. The interpreted value of these bits, along with other bus related functions can be found in the bus mode register outlined below.

[1] In other words, aligned to their size; a 64 byte request is 64-byte aligned, a 32 byte request is 32 byte aligned.

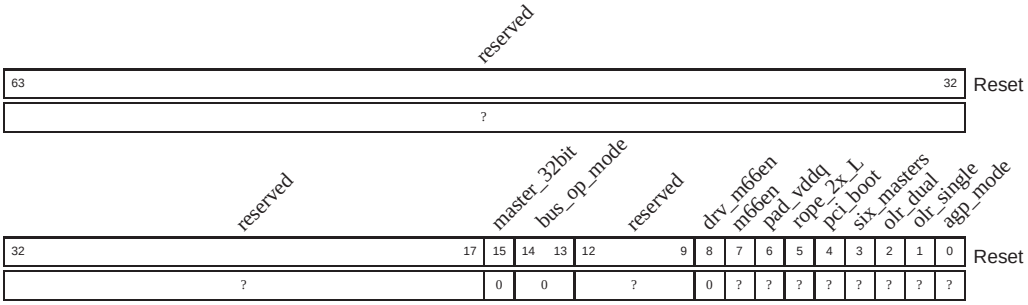
config[3:0]				OLR	Bus	Vddq	Ropes	Masters	PCI boot†
1	0	0	0	no	AGP	1.5V	dual	< 6 (1)	no
1	0	0	1	no	AGP	3.3V	dual	< 6 (1)	no
0	0	1	X	single	PCI[X]	3.3V	dual	< 6 (1)	no
0	1	1	X	single	PCI[X]	3.3V	single	< 6 (1)	no
0	0	0	X	dual	PCI[X]	3.3V	dual	< 6 (2)	no
0	1	0	X	dual	PCI[X]	3.3V	single	< 6 (2)	no
1	1	1	1	no	PCI[X]	3.3V	single	< 6	no
1	0	1	1	no	PCI[X]	3.3V	dual	< 6	no
1	1	1	0	no	PCI	3.3V	single	6	no
1	1	0	0	no	PCI	3.3V	single	6	yes
1	1	0	1	no	PCI[X]	3.3V	single	< 6	yes
1	0	1	0	— *unused* —					

†A “yes” indicates that PCI_RST_L does not remain asserted after a rope hard reset. This allows booting (fetching code) from PCI rather than from some dedicated PDH path.

Table 3.1: HP zx1 ioa power-up configuration bits

3.2.2 Bus Mode Register

Register 3.1: BUS MODE (0x0620)



agp_mode (Read only) A 1 indicates that the HP zx1 ioa is connected to an AGP bus

olr_single (Read only) A 1 indicates that single-slot OLR functionality is available.

olr_dual (Read only) A 1 indicates that dual-slot OLR functionality is available.

six_masters (Read only) A 1 indicates there are 6 masters connected to the HP zx1 ioa, so all of the req/gnt pairs are in use. If this bit is a 1, m66en will always return 0, and writing to drv_m66en will not actually cause the HP zx1 ioa to drive. If this bit is 0, arbitration for the 6th master is prevented by hardware from being enabled.

pci_boot (Read only) A 1 indicates that booting from PCI (fetching code at power on) is possible as PCI_RST_L will be released after a rope hard reset command. A 0 for this bit will cause the HP zx1 ioa to be configured to keep PCI_RST_L asserted until specifically programmed by firmware to release it.

rope_2x_L (Read only) A 0 indicates that the HP zx1 ioa is connected to dual ropes.

pad_vddq (Read only) A 1 indicates the I/O pad voltage is 3.3V. A 0 indicates 1.5V. This field is used by the pad to determine which FETS to enable to achieve the desired output characteristics.

m66en (Read only) This bit reflects the state of the PCI M66EN signal.This bit will return 0 if the HP zx1 ioa is connected to 6 bus masters. A version of this bit is also available in the OLR control register, though it is not qualified by six_masters there.

drv_m66en A 1 causes the HP zx1 ioa to drive a low level on the M66EN signal. This could be useful when the HP zx1 ioa needs to operate a PCI bus at less than 33.3MHz.

bus_op_mode This field indicates the mode that the bus is operating in. Do not change this field, as there are a number of undocumented dependencies

00 PCI bus

01 PCI-X/100MHz (66MHz – 100MHz)

10 PCI-X/66MHz (50MHz – 66MHz)

11 PCI-X/133MHz (100MHz – 133MHz)

master_32bit This tells the master controller to always master 32-bit cycles rather than 64-bit cycles. If a bus were designed with only 32-bit devices on it, setting this bit would improve bus utilization somewhat.

Chapter 4

The HP zx1 ioa Address Map

4.1 Overview

This chapter describes the physical address map for the HP zx1 ioa, covering the address offsets for the configuration, address range, and status registers architected in the chip. Bit field descriptions of registers are given in the chapters describing the associated functionality.

4.2 Location of the HP zx1 ioa Configuration Space

The HP zx1 ioa configuration space is determined by a base register in the rope host. In the HP zx1 mio, this is the Rope Configuration Base register. The HP zx1 mio implements the rope guest configuration space as a distributed range, supplying 8KB of address space to each rope guest. See the rope host ERS for more information. Note that although the HP zx1 ioa only implements 4KB of address space, it looks at 13 bits of address, so it must either be allocated a full 8KB of space, or the rope host must zero out address bit 13 before sending a register address down the rope.

4.3 The HP zx1 ioa Register Map

Table 4.1 below shows all the registers in the HP zx1 ioa's configuration space in offset order. Reads of reserved registers will always return zeroes. Writes to reserved registers will occur on the HP zx1 ioa's internal register bus but to no effect; they are thus effectively discarded.

Byte writes are supported to all registers except the SAPIC registers which are 32-bit per the SAPIC spec.

Offset	Register Name	Page
0x0000	Function ID	45
0x0008	Function Class	46
0x0010 - 0x0028	*Reserved*	
0x0030	Capabilities Pointer	46
0x0038	*Reserved*	
0x0040	Configuration Address	49
0x0048	Configuration Data	49
0x0050	*Reserved for Elroy*	
0x0058	Bus Number	50
0x0060	Capabilities list and AGP status	47
0x0068	AGP command	47
0x0070	Outbound Error Address Log	22
0x0078	*Reserved for Elroy*	
0x0080	Arbitration Mask	64
0x0088	*Reserved for Elroy*	

continued on next page...

Table 4.1: the HP zx1 ioa Registers

. . . continued from previous page

Offset	Register Name	Page
0x0090	Arbitration Mode	64
0x0098	Multi-transaction Latency Timer	65
0x00A0	Capabilities list, command and status PCI-X	48
0x00A8 - 0x0100	*Reserved*	
0x0108	Status, Information, and Control	44
0x0110 - 0x01F8	*Reserved*	
0x0200	LMMIO Base	55
0x0208	LMMIO Mask	56
0x0210	GMMIO Base	54
0x0218	GMMIO Mask	55
0x0220	WLMMIO Base	55
0x0228	WLMMIO Mask	55
0x0230	WGMMIO Base	54
0x0238	WGMMIO Mask	54
0x0240 - 0x0248	*Reserved for Elroy*	
0x0250	ELMMIO Base	56
0x0258	ELMMIO Mask	56
0x0260 - 0x0268	*Reserved for Elroy*	
0x0270	*Reserved*	
0x0278	Slave Control	58
0x0280	MSI Base	56
0x0288	MSI Mask	57
0x0290	Inbound Error Address Log	20
0x0298	Inbound Error Attribute Log	20
0x02A0	Completion Message Log	21
0x02A8 - 0x03D8	*Reserved*	
0x03E0 - 0x03E8	*Reserved for Elroy*	
0x03F0 - 0x0600	*Reserved*	
0x0608	*Reserved for Elroy*	
0x0610 - 0x0618	*Reserved*	
0x0620	Bus Mode	10
0x0628 - 0x0678	*Reserved*	
0x0680	Error Configuration	17
0x0688	Error Status	18
0x0690	Error Master ID Log	19
0x0698 - 0x07F8	*Reserved*	
0x0800	I/O Register Select	68
0x0808	*Reserved*	
0x0810	I/O Window	68
0x0818 - 0x0838	*Reserved*	
0x0840	I/O EOI	69
0x0848	*Reserved*	
0x0850	Software Interrupt	69
0x0858 - 0x0FF8	*Reserved*	

Table 4.1: HP zx1 ioa Registers

Chapter 5

Reset Strategy

5.1 Overview

The HP zx1 ioa implements part of a hierarchical reset strategy. At the top of the structure is the processor bus to which the rope host chip is connected. The HP zx1 ioa and its I/O bus are near the bottom of the hierarchy. This chapter describes the various reset signals and how they affect hardware. See the appropriate rope host ERS for information on the rest of the reset hierarchy.

The Status, Information, and Control register (SIC) is integral to the HP zx1 ioa's reset functionality. See page 44 for the definition of this register and programming details.

5.2 Reset Inputs

The various sources of reset are listed below, in the order of most effect on logic to the least.

PWR_ON_RST_L Power reset pin. External input driven by the rope host. Driven low at power up. Deasserted asynchronously after power is good. Used to tristate most of the HP zx1 ioa's pads. Not used as a HP zx1 ioa functional reset. This signal is also used to sample the power-up configuration pins (see Section 11.4).

DLL_RESET_L DLL reset, low true. Resets the strobe and I/O bus DLL's in the HP zx1 ioa. Deasserted asynchronously causing the DLL's to move toward lock. The rope host may be able to assert this under software control to perform functions such as changing the rope frequency.

rope hard_reset Rope command resulting from power-on reset of the rope host. There may be other reset conditions the host sees that cause this as well, but that is rope-host dependent. This command is sent continuously across the rope as long as certain reset conditions exist in the rope host. It can also be software generated (at least in the HP zx1 mio) for purposes such as changing the rope frequency. Resets all of the HP zx1 ioa, including the registers, as well as the I/O bus.

rope soft_reset Rope command controlled by reset activity in the rope host. Sent continuously across the rope as long as the reset input is asserted in the host's corresponding rope controller. Resets all of the HP zx1 ioa (with the exception of registers) as well as the I/O bus. Causes the RF bit in the HP zx1 ioa's SIC register to be set. The RF bit must be cleared by software after the required I/O bus reset time.

func_reset Function reset, high true. Software reset generated within the HP zx1 ioa as a result of a write to the RF bit in the SIC register. Can also be set be certain OLR conditions. Remains asserted until cleared by software. Must stay asserted for minimum I/O bus reset time. **func_reset** causes only the arbiter, slave controller, and I/O bus to be reset.

5.3 Rules for Software Resets

The following rules and guidelines should be observed by software and firmware when exercising any of the resets:

- Rope resets are intended to work under *any* conditions. It would be reasonable to use them during warm boot to ensure that any DMAs that might be in progress are stopped. These resets are also used when recovering from rope errors. Note that the rope host is responsible for any necessary cleanup (read returns for outstanding PIO, etc.) when doing a rope reset.
- If the I/O bus needs to be reset as part of error recovery, a rope soft reset should be used rather than func_reset. This requirement ensures that all of the HP zx1 ioa will be in a known and consistent state. Prior to issuing the rope soft reset, interrupts that might be spurious can be masked off via the SAPIC registers.
- The func_reset is only intended to work when the bus is quiescent. Asserting this reset while the bus is active may put the HP zx1 ioa in a confused state. The only reason for making this reset software controllable is OLR. So, it is reasonable to assume that the bus will be quiesced before OLR operations. Prior to reseting the I/O bus, interrupts that might be spurious can be masked off via the SAPIC registers.
- func_reset is different from others in that it must be cleared by software. All others finish on their own. It was special-cased in this fashion to support OLR, where the bus must be held in reset indefinitely while an OLR operation is taking place. *Resetting a rope will set func_reset so it is necessary to clear func_reset before proceeding.*
- When clearing func_reset, software must ensure that the minimum I/O bus reset time is met. This minimum time should be 1ms.

5.4 Hardware Behavior

During a rope hard reset or rope soft reset, the HP zx1 ioa's state machines will be reset. As a result, it is the responsibility of the rope host to handle returning -1 (or hard fail) for previously issued PIO reads. Refer to the rope host ERS for details on how this is handled and any other caveats.

During the HP zx1 ioa's funct_reset, the I/O bus behaves as if in I/O bus fatal mode (see Section 6.6). While the bus is held in reset, all PIO reads will return -1's (or hard fails), and PIO writes will be bit-bucketed.

5.5 Impact of Resets

The impact of each reset input is shown in Table 5.1. A check mark indicates that the specified block or function is reset by the source at the top of that column.

Block	rope hard_reset	rope soft_reset	func_reset
HP zx1 ioa registers	✓		
All blocks except SC and arbiter	✓	✓	
SC and arbiter	✓	✓	✓
I/O bus	✓	✓	✓

Table 5.1: Reset impacts

5.6 What gets reset by pci_reset?

The arbiter is reset by pci_reset, so no one besides the HP zx1 ioa will be granted the bus. Parts of the slave controller that interface to the I/O bus will also be reset by pci_reset. For the master controller, its state machines that interface to the I/O bus are reset by pci_reset. Behavior such as driving 0's on the PCI_AD lines, and asserting PCI_REQ64_L are also controlled by pci_reset. The AGP command register also gets reset by pci_reset because its state must stay consistent with the state of an I/O device which would get reset by pci_reset.

Chapter 6

Error Handling

6.1 Overview

The HP zx1 ioa performs error detection and error logging. This functionality can be used to isolate the failing component in a system. Where appropriate, the HP zx1 ioa will also signal errors on the I/O bus. This chapter details the HP zx1 ioa's error architecture.

6.2 Supportability

To allow for software revision checking, the HP zx1 ioa contains chip identification and revision registers.

The HP zx1 ioa also supports deconfiguration of the attached I/O bus, via the reset function (RF) bit in the SIC register (see page 44). The HP zx1 ioa itself can be deconfigured by resetting the rope to which it is attached. See the appropriate rope host ERS for information on how to do this.

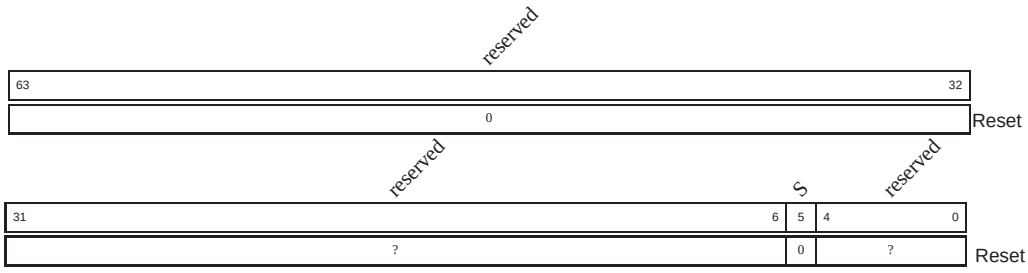
6.3 Error-related Registers

There are PCI and PCI-X architected status and command registers that also affect error behavior, or log error events. These registers are outlined in Chapter 8. Some important notes are that bits in these registers can override things like SERR# assertion, or change behavior in response to parity errors. In addition, there are a number of status bits that can be useful to pin down the exact cause of an error.

6.3.1 Error Config Register

This register is used to configure some of the HP zx1 ioa's error responses. The register is shown in diagram 6.1.

Register 6.1: ERROR CONFIGURATION (0x0680)

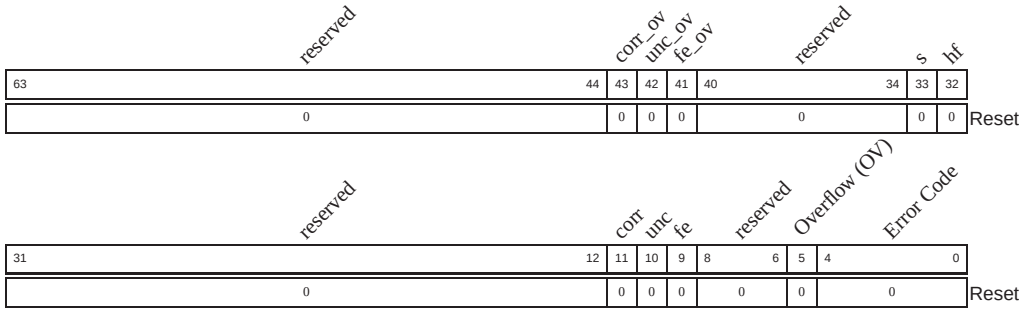


S Smart/Dumb mode bit. 1 indicates smart mode, while 0 indicates dumb mode.

6.3.2 Error Status Register

This is the main error log for I/O bus errors. Higher severity errors overwrite the error code for lower severity ones. Errors do not override higher severity ones, or ones of the same level. In other words, the first, most severe error has its error code logged. the HP zx1 ioa adds some bits beyond what Elroy had to try to give a more complete error picture.

Register 6.2: ERROR STATUS (0x0688)



error_code (Read only) Defines what kind of error occurred. A description of each error code is given towards the end of this chapter.

OV (Read only) See Section 6.3.2.1 below.

fe (Read only) Fatal error detected.

unc (Read only) Uncorrected error detected.

corr (Read only) Corrected error detected.

hf (Read only) Reflects state of hard-fail bit when first/most severe error was detected.

s (Read only) Reflects state of Smart-mode bit when first/most severe error was detected.

fe_ov (Read only) More than one fatal error detected.

unc_ov (Read only) More than one uncorrected error detected.

corr_ov (Read only) More than one corrected error detected.

The error status register is cleared using the CL and CE bits in the Status, Information, and Control Register. See Section 8.2.1 on page 43 for details.

Other related error logs such as the Master ID, inbound address and attribute, and outbound address registers follow the same rule regarding overwriting, so they contain information correlated with the error whose code appears in the error status register. In Section 6.9, the additional error logs that are significant for each error are specified. If not specified for a particular error, the additional error logs are don't cares.

In the case of the HP zx1 ioa detecting an address parity error in dumb mode, the fe_ov bit will get set. This is because the resulting assertion of SERR# by the HP zx1 ioa will result in a second fatal error.

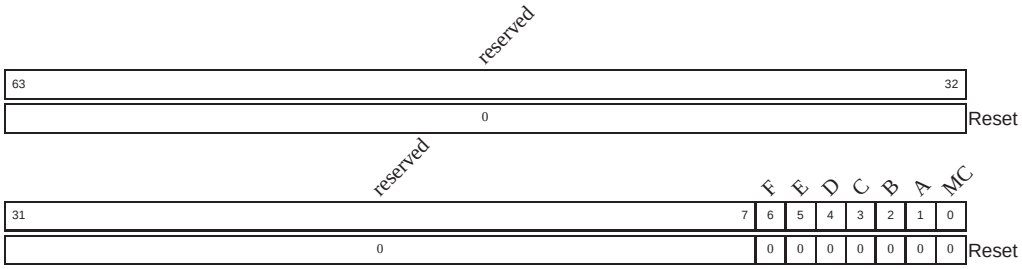
6.3.2.1 Overflow "Compatibility" Bit

The intent is that OV is set for uncorrectable or fatal errors when there have been two or more of the highest severity errors that are currently logged. One condition where OV would be set is if a fatal error is logged, and a second fatal error occurs. A second condition is if an uncorrectable error is logged, and a second uncorrectable error occurs, and no fatal errors have been logged. The OV bit will *not* be set for any number of uncorrectable errors followed by a single fatal error, or a single fatal error followed by any number of uncorrectable errors. This bit is *not* truly backward compatible with Elroy, as Elroy set its OV bit whenever any two errors had occurred of any severity (even correctable), not necessarily the highest severity.

6.3.3 Error Master ID Log Register

For those error types which specify logging of the I/O bus master ID, this register indicates which master owned the I/O bus during the transaction. The register contains valid information only when such an error has been logged. The encoding is one-hot — only one of the bits will be set at a time.

Register 6.3: ERROR MASTER ID LOG (0x0690)



Interpretation of the bits MC and A through F is shown in Table 6.1.[1]

Bit	PCI Mode	AGP Mode	PCI-X Mode
MC	HP zx1 ioa	HP zx1 ioa	HP zx1 ioa
A	Slot with reqa_L	AGP master	Slot with reqa_L
B	Slot with reqb_L	Never set	Slot with reqb_L
C	Slot with reqc_L	Never set	Slot with reqc_L
D	Slot with reqd_L	Never set	Slot with reqd_L
E	Slot with reqe_L	Never set	Slot with reqe_L
F	Slot with reqf_L	Never set	Never set

Table 6.1: Interpretation of Master ID bits.

The rules for when this register is overwritten or cleared are described in Section 6.3.2. This register is implemented in the slave controller block.

6.3.4 Inbound Error Address and Attribute Log Registers

The rules for when these registers are overwritten or cleared are described in Section 6.3.2. These register are implemented in the slave controller block. These registers are depicted in diagrams 6.4 and 6.5.

Note that the Inbound Error Attribute Log records transaction type (DMA Read, DMA Write, etc.). If no type was indicated, then the type was something else (such as an IOP-space transaction).

6.3.5 Completion Message Log Register

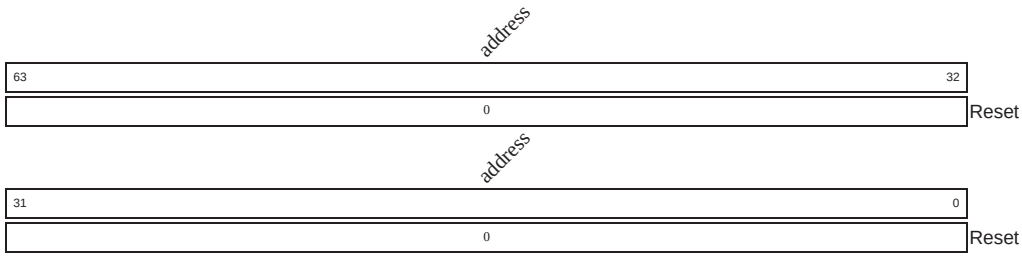
The rules for when this register is overwritten or cleared are described in Section 6.3.2. This register, shown in diagram 6.6, is implemented in the slave controller block.

6.3.6 Outbound Error Address Log Register

The rules for when this register is overwritten or cleared are described in Section 6.3.2. This register is implemented in the master controller block and shown in diagram 6.7.

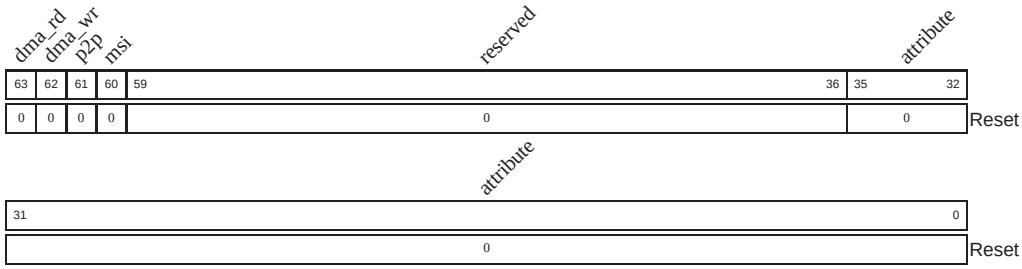
[1]The MC position is really only for historical purposes. As it turns out, none of the conditions under which the HP zx1 ioa is the master log info in the Master ID register. The error codes themselves distinguish the cycle as being HP zx1 ioa mastered.

Register 6.4: INBOUND ERROR ADDRESS LOG (0x0290)



address (Read only) Inbound address for the transaction. If transaction is not a dual-address cycle, the upper address bits will be forced to 0. For PCI, address[1:0] will always be zeroed, since those bits are not significant for PCI Memory space transactions. PCI-X, on the other hand, does allow a byte address to be specified, so all bits of this field will be significant on that bus.

Register 6.5: INBOUND ERROR ATTRIBUTE LOG (0x0298)



dma_rd† (Read only) This field is essentially an "extension" of the Inbound Error Address Log that indicates that the cycle with an error address logged was a DMA read.

dma_wr† (Read only) This field is essentially an "extension" of the Inbound Error Address Log that indicates that the cycle with an error address logged was a DMA write.

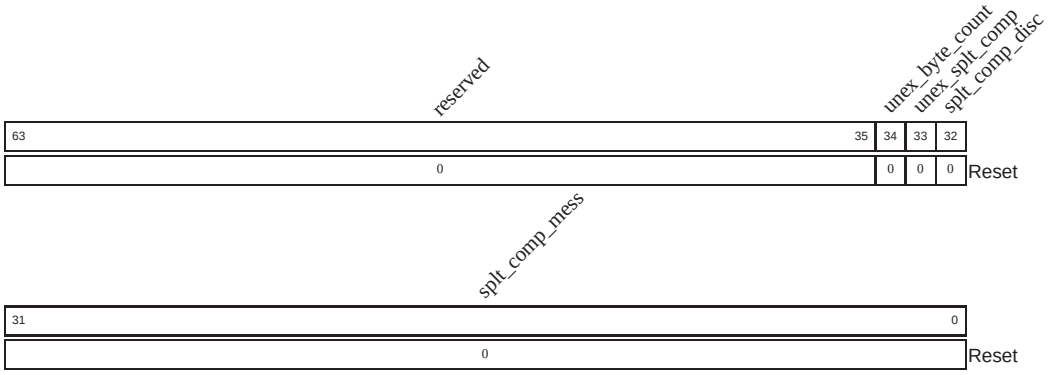
p2p† (Read only) This field is essentially an "extension" of the Inbound Error Address Log that indicates that the cycle with an error address logged was a peer-to-peer write.

msi† (Read only) This field is essentially an "extension" of the Inbound Error Address Log that indicates that the cycle with an error address logged was a message signalled (transaction based) interrupt.

attribute (Read only) PCI-X only. The PCI-X spec defines the meaning of the attribute field based on the type of transaction (burst, dword, Split Completion).

†If none of these types are indicated, then the type was something else (such as an IOP-space transaction).

Register 6.6: COMPLETION MESSAGE LOG (0x02A0)



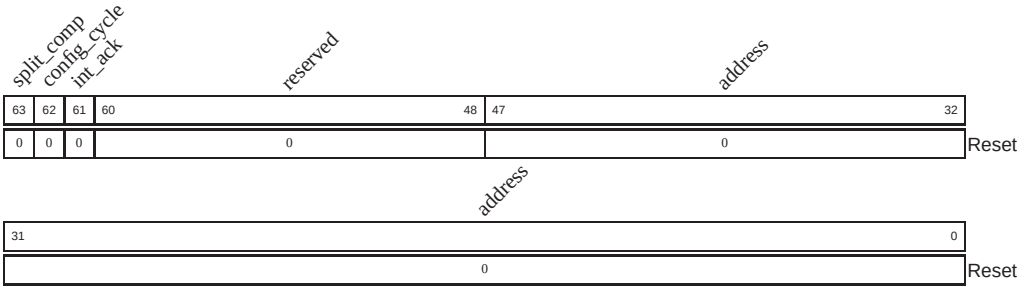
unex_byte_count (Read only) HP zx1 ioa sees a Split Completion with a Sequence ID matching an outstanding transaction, but the byte count is incorrect.

unex_splt_comp (Read only) HP zx1 ioa sees a Split Completion with its Requester ID, but which doesn't have a Tag that matches an outstanding HP zx1 ioa request.

splt_comp_disc (Read only) Split completion that the HP zx1 ioa mastered either master or target aborted.

splt_comp_mess (Read only) Split completion message defined in the PCI-X spec. This field can only be assumed to be valid when the error the HP zx1 ioa logged was for the receipt of a PIO Split Completion error message.

Register 6.7: OUTBOUND ERROR ADDRESS LOG (0x0070)



address (Read only) Address of outbound transaction. This would normally be a PIO read or write, in which case the lower 2 bits will always return 0. However, if it isn't a PIO read or write, one of bits 63:61 will be set. Note that a peer-to-peer write transaction coming from another HP zx1 ioa will look just like a PIO write to this HP zx1 ioa.
The address logged for AGP FWs will only be approximate. It should still be an address that is within the block where the error occurred.
The address logged for a configuration cycle is the address that appears on the IO bus (device number is one-hot decoded etc.)
The address logged for PCI-X DMA read return split completions will be the starting address of the transaction if the error occurred in the 1st, 2nd, or 3rd data phase of the transactions. Errors on later data phases will have the appropriate address logged.

int_ack (Read only) Indicates outbound transaction was an interrupt acknowledge and thus the address field isn't meaningful.

config_cycle (Read only) Indicates outbound transaction was a PCI configuration cycle.

split_comp (Read only) Indicates outbound transaction was a PCI-X DMA read return Split Completion. The address has a different meaning in this case. It has sequence ID and other information that can help identify the master that requested the DMA read. The PCI-X spec outlines what information is driven during the address phase of a split completion. The HP zx1 ioa logs extra internal state for the reserved bit 7. If it is set it indicates the original request was a burst read, if it is not set it indicates the original request was a DWORD read.

6.4 I/O Bus Availability

The HP zx1 ioa includes an HF bit in its status/control register. If this bit is set, the HP zx1 ioa returns a hard-fail response on PIO reads (or I/O port writes since they require a completion message) as an error indication. If not set, the HP zx1 ioa returns the data -1.

The HF or -1 error indication is returned whenever a PIO read or I/O port write terminates in a master abort or target abort, when a PIO read receives bad parity, or when an I/O port write sees PERR# asserted. It is also returned for all PIO reads and I/O port writes when the I/O bus is already in I/O bus fatal mode.

This only applies to PIO reads that reach the I/O bus. PIO reads reaching internal HP zx1 ioa registers always attempt to return register data. Of course, there are cases where the HP zx1 ioa's PIO read data can not be successfully returned to the rope host, either because the rope is already considered fatal by the rope host, or because the HP zx1 ioa has reset itself and stopped sending any rope messages as a containment method for rope errors. In the latter case, the rope host must detect the situation using a timeout.

In either of these cases, it is the rope host's responsibility to log the error.

The S bit in the Error Configuration register specifies whether the I/O bus to which the HP zx1 ioa connects is considered "Smart" or "Dumb."

HP zx1 ioa's response to

1. internal outbound parity errors,
2. inbound data parity errors,
3. address parity errors,
4. master aborts,
5. target aborts,
6. reads from write-only addresses,
7. PIO Split Completion error messages,
8. PIO Split Completion timeouts,
9. SERR# assertions, and
10. PERR# assertions

is affected by the selection of Smart or Dumb mode. the HP zx1 ioa handling of rope errors is unaffected by the smart/dumb mode selection, as are the few other errors not explicitly mentioned above.

6.4.1 Smart I/O buses

A smart I/O bus is one containing only smart I/O cards.

An I/O card is smart if it reliably detects, signals, and logs all address and data parity errors. If the card detects a parity error, it is responsible for not using the address and data associated with the transaction. A smart card is required to perform graceful transaction completion on transactions with error conditions and is required not to hang the bus.

6.4.2 Dumb I/O buses

A dumb I/O bus is one containing one or more dumb I/O cards.

A dumb card is defined by its inability to guarantee detection, signaling, containment and logging of address and/or data parity errors or inability to guarantee graceful completion of transactions containing parity errors. A dumb card could potentially use data or addresses with bad parity.

The HP zx1 ioa assists in the containment for dumb I/O buses by providing an I/O bus fatal mode. This mode is entered when certain types of errors occur. I/O bus fatal mode is described in detail later in this chapter.

6.4.3 Smart/Dumb Mode Special Cases

6.4.3.1 AGP buses

AGP buses must be configured as dumb.

For the most part, this makes little difference; AGP cycles are not parity protected and have no concept of master or target aborts, so smart AGP cards are not a meaningful concept. Revision 2.0 of the AGP specification, Section 3.7 says:

> Since the sole purpose of the A.G.P. is the connection of video display devices, no provision has been made in this interface specification for detection or reporting of any bus errors.

AGP has no data parity protection, so the HP zx1 ioa internal errors on outbound data are not containable by poisoning. Thus, dumb mode is mandatory.

AGP drivers must reset the HP zx1 ioa's state machines and the AGP bus (rope soft reset does both) for fatal AGP buses as part of any recovery sequence, since AGP pipeline state is flushed as part of the fatal sequence. This should not be an issue, since AGP has no recoverable error conditions. AGP I/O card status registers may be read before the reset is issued.

6.4.3.2 PCI-X buses

In theory, all PCI-X buses that run PCI-X protocol (i.e., do not contain PCI cards) should be smart. The PCI-X specification's requirements regarding error containment, detection, and logging should ensure this.

Nonetheless, the HP zx1 ioa permits PCI-X buses to be configured as smart or dumb. Dumb PCI-X buses are useful, since PCI-X cards have been discovered that are not as smart as theory would indicate!

PCI-X buses are not guaranteed to be left in a stable, operational state after entering I/O bus fatal mode. This is because both DMA and PIO split reads could be outstanding, with no fully reliable means to allow for Split Completions. The HP zx1 ioa manufactures a fake read return (-1 or HF, as usual) for any PIO split read that was outstanding at the time of going fatal. It also discards the read returns from the rope host for any split DMA reads that were outstanding at the time.

Reading card status after going fatal is a little more complicated for PCI-X. We need to consider the case with and without a PCI-X to PCI-X bridge. Without a bridge, one can write to the card's bus master bit (PCI command register) to clear it. This will prevent the card from mastering any cycles except Split Completions. Then one can re-enable arbitration for the card, and go about reading registers. Out of paranoia, it is also probably best to clear the HP zx1 ioa's "Memory Space" enable bit (in the Control field of the FID, see page 45). This prevents the HP zx1 ioa from responding to any DMA traffic just in case the card violates the PCI-X spec in how its bus master bit works. We must also be aware that the HP zx1 ioa may detect an unexpected split completion when we re-enable arbitration for the card, as it is possible that there was an outstanding completion when the HP zx1 ioa went fatal.

For the case with a bridge, Split Completions will get stuck behind DMA writes. The only possible way to let them drain is to disable the HP zx1 ioa's "Memory Space" bit, enable arbitration, and let the HP zx1 ioa master abort all DMA cycles. The hope is that will drain the DMA writes, allowing PIO reads to complete with Split Completions. However, there is no guarantee that the bridge won't become very upset about the master aborts and still not drain the DMA writes.

After reading card status, drivers *must* reset the HP zx1 ioa's state machines and the PCI-X bus (rope soft reset does both) for fatal PCI-X buses. The HP zx1 ioa and the I/O cards can still be out of sync with regard to outstanding transactions. The HP zx1 ioa's "Memory Space" bit can be re-enabled at this time as well.

6.4.3.3 PCI buses

For the PCI case with a bridge, read returns will get blocked behind DMA writes, just like the Split Completions in PCI-X. The only possible way to let them drain is to disable the HP zx1 ioa's "Memory Space" bit (in the Control field of the FID, see page 45), enable arbitration, and let the HP zx1 ioa master abort all DMA cycles. The hope is that will drain the DMA writes, allowing the PIO reads to complete. Again, there is no guarantee that the bridge won't become very upset about the master aborts, and still not drain the DMA writes.

There are even non-bridge cases where the PCI bus is not really left in an operational state when I/O fatal mode is entered. If the HP zx1 ioa issues a transaction that is retried, and the HP zx1 ioa enters I/O fatal mode before having an opportunity to complete the transaction, the card state and the HP zx1 ioa's state are out of sync. Attempts to read card status may be unsuccessful as the card may be waiting for the retried cycle to be attempted. There are likely other examples like this.

6.4.3.4 Containment Mode Summary

Table 6.2 shows the error containment modes that can be used for each of the buses. The columns are interpreted as follows:

Containment Specifies which entity is responsible for containing the errors.

Card status Specifies whether it is possible to read status registers on the I/O card(s) after the error occurs.

Bus Type	Containment	Card status	Reset Required
PCI Smart Mode	Card	Readable	No
PCI Dumb mode	HP zx1 ioa	Maybe readable	Yes
PCI-X Smart Mode	Card	Readable	No
PCI-X Dumb Mode	HP zx1 ioa	Maybe readable	Yes
AGP Dumb Mode	HP zx1 ioa	Readable	Yes

Table 6.2: Error containment modes for each I/O bus.

Reset Required Specifies whether a rope soft reset is a mandatory part of the error recovery sequence.[2] Of course, if an intervening OL* operation is needed, a reset is unavoidable.

Dumb mode containment is the “big hammer” that guarantees containment even for cards that are designed with minimal error savvy. By contrast, smart mode containment is more surgical, disrupting the I/O subsystem as little as possible, but requiring well-designed I/O cards.

6.5 General Error Handling Behavior

The HP zx1 ioa drives SERR# and PERR# in response to address and data parity errors respectively, and sets the various PCI architected error bits in the PCI Status register, as specified by the PCI specification. In addition, it performs other logging and containment actions not architected in the PCI/PCI-X specification documents. A summary of these follows:

- Bad parity received on rope command or data can be enabled to cause the HP zx1 ioa to reset (core_reset asserted) and to stop responding to the rope until it receives a soft or hard reset. The HP zx1 ioa also sends a poisoned rope command to goad the host chip into entering its rope fatal mode quickly.

- DMA write data received with bad parity is contained by the HP zx1 ioa by forwarding it with all byte enables turned off. This type of containment continues for the entire remainder of the data in the effected burst. In dumb mode, the I/O bus is also put into I/O bus fatal mode.

  In the case of PCI-X, such DMA write containment squashes only the remainder of the burst, and not necessarily the entire transaction. In other words, on a smart PCI-X bus, if the transaction disconnects on an ADB with partially satisfied byte count and then resumes, the resumed write is treated as a separate transaction with no containment precautions.[3]

- Bad data parity received on a PIO read return (or on a PCI-X Split Completion for a PIO cycle) results in a -1 or Hard-fail return, depending on the state of the HF mode bit. In dumb mode, the I/O bus is also put into I/O bus fatal mode.

- Addresses with bad parity (or attributes with bad parity for PCI-X) are decoded normally. By the time the HP zx1 ioa is aware of the error, it may have already asserted DEVSEL#. The HP zx1 ioa then target aborts the transaction. The event is logged, and the address information is discarded. In dumb mode, the I/O bus is put into I/O bus fatal mode. Regardless of whether or not the HP zx1 ioa responded to the address, it pulls SERR# in dumb mode.

- The assertion by any agent of SERR# is logged by the HP zx1 ioa. The I/O bus is also put into I/O bus fatal mode.

- The assertion by another agent of PERR# while the HP zx1 ioa is driving data is logged by the HP zx1 ioa. In dumb mode, the I/O bus is also put into I/O bus fatal mode. PCI-X write data parity error split completion error messages are treated just as if PERR# was asserted.

[2] It is not clear whether dumb mode recovery without an intervening bus reset would ever be used in practice. It was “supported” in Elroy, but in reality could never be guaranteed to work. The HP zx1 ioa does not support it for PCI buses. In reality, there are a number of cases in PCI that will deadlock such that a reset is required. A reset is always required for PCI-X or AGP because they lose pipeline/split transaction state upon going fatal. There is less such state for PCI, but the same issue applies to PCI as well.

[3] The alternative, i.e. containing the entire transaction, would require an associative table of remaining byte count lengths by transaction ID, and even then would not be rigorous. For example, if the data parity error occurred in the initial burst of a transaction that disconnects on the first ADB, which is three or fewer data phases from the start of the transaction, the entire transaction’s byte count would be unknown to HP zx1 ioa.

- If an agent requests the I/O bus and then does not use it within 16 clocks of getting a grant, the event is logged and the grant is removed if the master doesn't assert FRAME# at that time. The event will be logged at 15 clocks, although the master still has one more clock to assert FRAME# before losing the grant. The 16/15 clocks were intended to be 8/7 for PCI-X. Due to an errata, the error is signaled prematurely (as early as 5 clocks) though the grant is not removed for 7 clocks.
- If no agent responds to a transaction mastered by the HP zx1 ioa, the transaction is master aborted and discarded. If it is a PIO read, an I/O port read or a Configuration read, a fake return (-1 or HF) is created. In dumb mode, I/O bus fatal mode is entered unless the transaction was to Configuration or I/O space. PCI-X master abort split completion error messages are treated just like any other master abort.
- If an agent terminates a transaction mastered by the HP zx1 ioa using a target abort, the transaction is discarded. If it is a PIO read or I/O port write, a fake return (-1 or HF) is created. In dumb mode, I/O bus fatal mode is entered. PCI-X target abort split completion error messages are treated just like any other target abort.
- If the HP zx1 ioa receives a split response to a PIO, and no Split Completion transaction occurs within a programmable timeout period, a fake return (-1 or HF) is created. In dumb mode, I/O bus fatal mode is entered.
- If the HP zx1 ioa receives a read from a write-only space, it will target abort and discard the transaction. The event is logged, and in dumb mode, I/O bus fatal mode is entered.
- If the HP zx1 ioa gets a Split Completion error message (SCE) when running a PIO cycle, the event is logged, and a fake return (-1 or HF) is generated. In dumb mode, I/O bus fatal mode is entered. Only split completion error messages other than master abort, target abort, or write data parity error are treated as this error class.
- If the HP zx1 ioa masters a DMA read Split Completion, and that cycle is master or target aborted, the HP zx1 ioa discards the transaction and logs the event. No further action is required. The PCI-X "Split Completion Discarded" status bit will *not* be set since the read has no side effects. The HP zx1 ioa will assert SERR# for this case. Detecting the SERR# assertion would cause the HP zx1 ioa to enter I/O bus fatal mode .
- If the HP zx1 ioa receives a Split Completion with its Requester ID but with a Tag that does not match any outstanding requests, it will not assert DEVSEL# and the transaction will master abort. The event is logged, but no further action is required. The PCI-X "Unexpected Split Completion" status bit will *not* be set since the HP zx1 ioa never responded to the cycle.
- If the HP zx1 ioa receives a Split Completion with a correct Sequence ID (Requester ID and Tag match an outstanding the HP zx1 ioa transaction) but with an unexpected byte count, the HP zx1 ioa will claim the transaction and discard it internally. A fake read return (-1 or HF) will be sent for the deferred completion. The PCI-X "Unexpected Split Completion" status bit *will* be set since the HP zx1 ioa responded to the cycle. In dumb mode, I/O bus fatal mode is entered.
- If the HP zx1 ioa sees a reserved command issued by an AGP master, it will log an AGP error and enter I/O bus fatal mode.
- If the HP zx1 ioa is asked to send an AGP FW command to an address above 4G, it will log an AGP error and enter I/O bus fatal mode.

6.6 I/O Bus Fatal Mode

I/O bus fatal mode is a containment mechanism in which the HP zx1 ioa quiesces the I/O bus following an error, for cases where the I/O card/driver combination is not able to guarantee containment any other way. I/O bus fatal mode should not be confused with *rope fatal* mode.

I/O bus fatal mode is mostly used for I/O buses configured as "dumb," since "smart" I/O cards can handle most containment scenarios on their own. However, in cases where there is no reliable way to assign containment responsibility to a particular card, an error can cause I/O bus fatal mode to occur *regardless of the smart/dumb mode of this bus.* Two such error cases exist:

- internal bad parity on a the HP zx1 ioa register write and
- someone pulls SERR#. SERR# assertion is non-specific enough that there is no unambiguous way for cards to identify which card is responsible for containment. The timing relationship between SERR# and address parity errors is open-ended, and the signal may be used for various other unrelated errors as well.

The contents of I/O card status registers may be of interest to drivers following an error, so it is desired that I/O bus fatal mode disturb the I/O card state as little as possible. For this reason, entering I/O bus fatal mode does not cause an I/O bus reset, nor any type of unusual cycle termination (master or target abort). It quiesces traffic on the bus in the least disruptive way possible.

6.6.1 I/O Bus Fatal Mode Behavior

Characteristics of I/O Bus Fatal mode are:

- Arbitration for the I/O bus is disabled as soon as possible for all bus agents. This may or may not be soon enough to guarantee that no further transactions occur on the bus.
- No further transactions are mastered by the HP zx1 ioa on the I/O bus. PIO and P2P transactions that would have caused the HP zx1 ioa to master the I/O bus are silently discarded. In the case of PIO reads or I/O port writes, dummy returns are provided, consisting of either the value -1 (all ones), or a hard fail indication, depending upon the HF mode.
- Regardless of whether it is DMA, PIO, or P2P, the burst in progress at the time of going fatal continues to a normal termination. If the transaction is a DMA read, it continues normally. If convenient to do so, DMA reads may be disconnected (not aborted) at the next legal opportunity, rather than waiting for the entire transfer to complete. If a DMA write, all of the remaining data in that burst is forwarded to the rope host chip with its byte enables squashed, effectively discarding it. If it is an interrupt transaction or P2P write, it is discarded.
- I/O bus fatal mode causes all I/O SAPIC interrupt mask bits to be set, thereby containing any wire-based interrupts that may be driven after the error event.
- PCI-X split transaction state for PIO cycles is cleared in the HP zx1 ioa. However, I/O cards still remember this state and are thus out of sync with the HP zx1 ioa. For this reason, dumb PCI-X buses must be reset (as well as the HP zx1 ioa state machines — rope soft reset does both) as part of the recovery sequence.
- PCI-X DMA read return state is kept in the HP zx1 ioa because the rope host will send data for all outstanding requests which the HP zx1 ioa must then internally discard.
- On an AGP bus, no further pipelined DMA requests are enqueued. Any requests currently enqueued (but not yet sent up the rope) are discarded. DMA writes in progress while entering I/O fatal mode will have their byte enables squashed. DMA read returns in progress on the I/O bus will complete as normal. Dummy data (no byte enables set) with EOT indicated will be sent up for each outstanding DMA write request. RDF data received for outstanding DMA read requests will just sit there and back up the RDF until cleared by a rope soft reset. This behavior continues until both the HP zx1 ioa and the AGP bus are reset with a rope soft reset.

By disabling arbitration in I/O bus fatal mode, further DMA, P2P, and MSI traffic is prevented. However, there is a race conditionwhereby it is possible for one further transaction to get started too early to be contained in this manner. Any such cycle is retried by the HP zx1 ioa.

I/O bus fatal mode is generated slightly differently to different blocks. When the error block sees a fatal error, the **arb_enable** bit in the arbitration mask register will be cleared, causing all blocks that know about I/O bus fatal mode to indeed be in I/O bus fatal mode. For most of these blocks, the I/O bus being reset will also cause them to be in I/O bus fatal mode. The exceptions are the arbitration mask register, the SAPIC mask bits, and the AGP slave controller.[4] They do not “go fatal” simply by the I/O bus being in reset. In the case of the HP zx1 ioa being in I/O bus fatal mode only because of the I/O bus reset, simply bringing the bus out of reset will bring the HP zx1 ioa out of I/O bus fatal mode. For the case of I/O bus fatal mode being entered due to an error condition, the sequence outlined below will be necessary.

6.6.2 I/O Bus Fatal Mode Recovery

Depending upon the rope host chip in use, there may be additional steps required to keep the rope host cache and I/O TLB happy. This list only covers actions that the HP zx1 ioa requires. See the appropriate rope host ERS for more information.

[4] As mentioned earlier, for the AGP slave controller I/O fatal mode lasts until a rope soft reset occurs.

6.6.2.1 For a PCI bus

1. Read the various error logging registers. Clear the error code in the Error Status register. This is done using the CL and CE bits of the the HP zx1 ioa Status, Information, and Control register (see page 44).
2. Enable PIO cycles. This is done by writing a 1 to bit 0 of the Arbitration Mask register (see page 64). The other master's mask bits should be cleared as part of this write to the mask register.
3. Now I/O bus configuration cycles and other PIO accesses may be performed to card registers in order to diagnose the error and take corrective action where possible. However, recall that even PIO may not be possible, and if there is an intervening PCI to PCI bridge, the situation is more complicated. See Section 6.4.3.3 for details.
4. Issue a rope soft reset to reset the HP zx1 ioa state machines and the I/O bus.
5. Re-enable arbitration via the Arbitration Mask register. Re-enable any SAPIC wire interrupts.
6. Bring the I/O bus out of reset via the Status, Information, and Control register. The order of this step with the previous step can probably be swapped.
7. Do any required re-initialization of cards via configuration and PIO cycles.

If an individual card or a whole I/O bus is being deconfigured, some of these steps will be skipped. If an online replacement operation is being performed, there will be additional steps .

6.6.2.2 For a PCI-X bus

Recovery is similar to the PCI case, except that step 3 is more involved. It will need to be expanded to disable the bus master bit on the card, potentially clear the "Memory Space" bit in the HP zx1 ioa, re-enable arbitration for the card, and then read card status. Step 4 is of course mandatory for PCI-X. If the HP zx1 ioa's "Memory Space" bit was cleared, it will also have to be re-enabled (see Section 6.4.3.2). In addition, the HP zx1 ioa must **not** be brought out of I/O fatal mode until after all DMA read data has been returned from the rope host and discarded internally by the HP zx1 ioa.

6.6.2.3 For an AGP bus

Recovery is similar to the PCI case, Step 4 is of course mandatory, and online replacement is not possible for AGP.

6.7 PCI-X Error Handling

New error logging is needed to log the attributes of the offending cycle:

- When the HP zx1 ioa masters PCI-X, it is only doing PIO or DMA read Split Completions. (P2P targets are on AGP only). Thus the attribute bus/device/function fields are not very interesting — for PIO they are always the same (requester), and for Split Completions, they are always the same (completer). Boring! For this reason, attributes are not logged for PIO.
- When the HP zx1 ioa responds as a slave on PCI-X, the entire address and attribute fields are of interest, so both are logged.
- When the HP zx1 ioa receives a Split Completion Error Message in response to a PIO transaction, the message is logged if the message is not a master abort, target abort, or write data parity error. These 3 exceptions are handled just like regular master aborts, target aborts, or the assertion of PERR# on write data.

The HP zx1 ioa never returns a Split Completion Message or Split Completion Error Message (SCM or SCE bit set in Split Completion attribute) to a PCI-X agent. This is because, as a host bridge, there are no completer-bus exceptions that could give rise to it.

When the HP zx1 ioa receives a split response to a PIO read, it waits for a Split Completion without issuing any further requests. If the Split Completion received is a Split Completion Error Message (except for master abort, target abort, or

write data parity error), the HP zx1 ioa translates this to a -1 or HF response. In dumb mode, it also enters I/O bus fatal mode.

A Split Completion timer is used to terminate a split PIO read if no response is received within a specified time. The intent is to set this timer to be shorter than any timeouts closer to the system core (i.e., the IOC or CPU), so that this type of error does not have to bring down an entire rope or possibly the entire system. Split Completion timer expiration causes an error to be logged. In dumb mode, the HP zx1 ioa also enters I/O bus fatal mode.

If the HP zx1 ioa is the target to a transaction that starts within its address range, but that would extend beyond the HP zx1 ioa's address range, the HP zx1 ioa will claim the transaction, but then immediately signal target abort. This scenario should never arise for properly programmed cards.

6.8 PCI[X] Architected Error Log Registers

The PCI and PCI-X specifications document various error log bits in the PCI Status register. The bus specifications serve as the definitive definition of the behavior of these bits, and they are not repeated here. Note that the operation of these bits is significantly different in PCI[5] mode than PCI-X mode. For example, comparing the Master Data Parity Error bit in PCI-X vs PCI, the behavior differs as documented in the PCI-X spec section 5.4.1.

One case of particular interest is the "Split Completion Discarded" bit for PCI-X. Section 5.4.4 of the PCI-X spec says to set this bit and pull SERR# if a Split Completion is discarded because of a master or target abort, but only if the completion is for a read from an address *with read side effects.* DMA reads should not have side effects, so according to the PCI-X spec, the log bit should *not* be set, and SERR# should not be pulled. It is possible to design a host bridge such that DMA reads do have side effects. As a result, the HP zx1 ioa will assert SERR#. However, the HP zx1 ioa still doesn't set the "Split Completion Discarded" bit, as the HP zx1 ioa has its own way to log that this error has occurred. . Clarifications underway in the PCI-X workgroup are leaning toward suggesting the HP zx1 ioa should really set the Split Completion Discarded bit regardless of side effects or not.

A possible scenario where a split read completion would be discarded would be the case of a PCI-X device whose in-progress DMA has been terminated by a command issued through a PIO cycle, so this silent discarding of the completion seems reasonable. As mentioned before, the HP zx1 ioa provides an advisory error log for this event, to assist in debugging.

The mirror image of this case is if the HP zx1 ioa sees a Split Completion with its Requester ID, but with a Tag that doesn't match. The HP zx1 ioa will not assert DEVSEL#, so according to the PCI-X spec, the HP zx1 ioa should *not* set the "Unexpected Split Completion" bit. However, the HP zx1 ioa will make an advisory log of this event, sharing the error code with the above Split Completion discarded scenario.

Another case is if the HP zx1 ioa receives a Split Completion with the correct Sequence ID, but with an unexpected byte count. The HP zx1 ioa will claim the transaction and discard it internally. A fake (-1 or HF) read return will be generated. Since the HP zx1 ioa asserted DEVSEL# but discarded the transaction, it *will* set the "Unexpected Split Completion" bit.

Finally, if the HP zx1 ioa receives a Split Completion Error Message that it doesn't otherwise handle, it will log the message in the Completion Message Log, generate a fake return (-1 or HF), and set the "Received Split Completion Error Message" bit in the PCI-X Status register.

6.9 Error List

In addition to the error logging architected in the PCI[X] specs, the HP zx1 ioa contains a set of more detailed error logs. Unlike the PCI Status register log bits, these logs are designed to be identical in PCI and PCI-X mode. They are a superset of the Elroy error logs. The only exception is that the "card asserted LOCK#" error which no longer exists.

The following list describes the log information, severity, and containment for each type of error that the HP zx1 ioa can detect.

6.9.1 Reading the Error List

Where logged: PI indicates the error has been logged in the Error Status register.

[5]Including the PCI subset of AGP and PCI-X buses containing a PCI card.

Severity: The severity field only describes the HP zx1 ioa's part in the error. So for example, an error which will ultimately bring down the system would be viewed as "Fatal" by the system, but the HP zx1 ioa may classify it as "Uncorrected" because its only part in the proceedings is to forward the bad data on and let another chip or card contain it.

- Corr — Corrected. Used for errors that don't need any particular containment.
- Unc — Uncorrected. Used for errors that may need containment, but the containment is not explicitly done by the HP zx1 ioa. Another agent in the system may subsequently correct the error (e.g. smart I/O card discards and re-fetches bad data, smart driver sees -1 return and cleans up), or may contain it in another way (e.g. IOC makes rope fatal when it sees the uncorrected bad parity data forwarded to it from a the HP zx1 ioa inbound internal parity error).
- Fatal — Errors which the HP zx1 ioa contains using a big hammer — either using I/O bus fatal mode, or by resetting itself and forcing the IOC to go rope fatal.

Log code: For PI, a 5-bit error code that will appear in the log register. Values 0x12 or higher are reserved for errors that cause I/O bus fatal mode. Values less than 0x12 are used for all other errors. Where the same error is considered fatal in dumb mode and non-fatal in smart mode, the same code is assigned to both, with the exception of the MSB.

Other Info Logged: Additional information logged with the error.

- IAddress — The Inbound Error Address Log register and Inbound Error Attribute Log register are set in the Slave Controller.
- OAddress — The Outbound Error Address Log register is set in the Master controller.
- MasterID — The Error Master ID Log register is set in the Slave controller.
- CompMsg — The Completion Message Log register is set in the Slave controller.

These registers are significant only if they are specified for a particular error. If they are not specified, then these registers are considered don't cares.

Containment: This should be self-explanatory.

6.9.2 Errors independent of smart/dumb mode

6.9.2.1 I/O card requested bus, but did not use it within 16 clock cycles (8 for PCI-X)

Where logged: PI

Severity: Corr

Log code: 0x01

Other info logged: MasterID

Containment: None needed.

Rationale: PCI spec says that if a card requests and is granted the bus, it must start a transaction within 16 clock cycles after the GNT# is asserted. If not, the card may be assumed to be broken, and the arbiter may henceforth ignore its requests. The HP zx1 ioa itself just reports this condition, and it is up to software to decide how to handle it. The HP zx1 ioa flags the error after 15 clocks, but the master actually has 16 clocks to assert FRAME#.

6.9.2.2 The HP zx1 ioa internal error during register access

Where logged: PI

Severity: Fatal

Log code: 0x12

Other info logged: OAddress

Containment: I/O bus fatal mode is entered *regardless of whether in dumb or smart mode*. Register access is still performed. Register address is logged.

Note: This error can only occur for two reasons: (1) a parity error occurred in the data for a register write or (2) the HP zx1 ioa's internal regbus daisy chain suffered a fault during a register read or write.

Rationale: Fatal mode containment is needed even in smart mode, since the usual smart mode containment mechanism of poisoning outbound I/O bus parity can't be used on a write to an internal register.

6.9.2.3 DMA split read completion target- or master- aborted

Where logged: PI

Severity: Corr

Log code: 0x02

Other info logged: OAddress, CompMsg

Containment: No containment is likely to be needed. This error will cause SERR# to assert.

Rationale: Advisory error only. When a DMA split read completion terminates in a master abort or target abort, the PCI-X architected "Split Completion Discarded" register bit is *not* set, because DMA addresses memory without read side effects. This log code records the event in case it is useful for debugging purposes. CompMsg must be checked to distinguish this error from an unexpected split completion.

6.9.2.4 Unexpected Split Completion

Where logged: PI

Severity: Corr

Log code: 0x02

Other info logged: MasterID, IAddress, CompMsg

Containment: No containment is needed.

Rationale: Advisory error only. PCI-X architected "Unexpected Split Completion" register bit is *not* set since the HP zx1 ioa never asserts DEVSEL#. CompMsg must be checked to distinguish this error from a DMA split read completion that target or master aborted.

Note: The Inbound Address Log will not be extremely interesting, as it will record a Split Completion targeting the HP zx1 ioa; the Tag field may be of interest. The Inbound Attribute Log will be more useful.

6.9.2.5 The HP zx1 ioa observes SERR# asserted

Where logged: PI[6]

Severity: Fatal

Log code: 0x1f

Other info logged: None

Containment: I/O bus fatal mode entered.

Note: The HP zx1 ioa can not discern the reason for the SERR#. However, if it was an address, command, or attribute parity error, the card that detected it should set both the “Signaled System Error” and “Detected Parity Error” bits. Otherwise, only the first bit will be set. This is how software can tell these apart.

6.9.2.6 The HP zx1 ioa detects an AGP error

Where logged: PI

Severity: Fatal

Log code: 0x11

Other info logged: None

Containment: I/O bus fatal mode entered.

Note: All AGP errors are fatal as no error checking or recovery is defined for AGP. Currently the HP zx1 ioa seeing a reserved command, or the HP zx1 ioa attempting to generate a FW to an address $\geq$4G are the only AGP errors detected.

6.9.3 Errors in smart I/O bus mode

6.9.3.1 The HP zx1 ioa internal parity error in PIO write or DMA read Split Completion data

Where logged: PI

Severity: Unc

Log code: 0x05

Other info logged: OAddress

Containment: Data is driven out on I/O bus with bad parity. I/O card detects this and contains it. The card should pull PERR# as well, but the HP zx1 ioa will ignore this PERR# assertion error so as to not set the overflow bit in a scenario that is just a single error.

6.9.3.2 The HP zx1 ioa internal parity error in connected DMA read return data

Where logged: PI

Severity: Unc

Log code: 0x08

Other info logged: IAddress, MasterID

Containment: Data is driven out on I/O bus with bad parity. I/O card detects this and contains it. The card should pull PERR# as well, but the HP zx1 ioa will ignore this PERR# assertion error so as to not set the overflow bit in a scenario that is just a single error.

[6]This log is attempted even if the HP zx1 ioa itself is the agent pulling SERR#. But in this case, the HP zx1 ioa will already have logged the underlying address/command/attribute parity error. Since both of these errors are of the same severity, only the parity error is logged, and the SERR# assertion causes the OV bit to be set in the error log. This is a little (!) ugly, but consistent with Elroy behavior.

6.9.3.3 The HP zx1 ioa observes PERR# asserted while driving out PIO write, DMA read Split Completion, or P2P write data, or receives a PCI-X write data parity error split completion error message

Where logged: PI

Severity: Unc

Log code: 0x03

Other info logged: OAddress

Containment: PERR# indicates that the card has already detected the error. Being a smart card, it will handle its own containment, and set the “Detected Parity Error” bit in its PCI Status register. This log identifies the requester as the HP zx1 ioa. In the case of an I/O port write, a hard-fail or -1 response is given as usual.

6.9.3.4 The HP zx1 ioa observes PERR# asserted while driving out connected DMA read return data, or while giving a split response to a PCI-X DMA read

Where logged: PI

Severity: Unc

Log code: 0x09

Other info logged: MasterID, IAddress

Containment: PERR# indicates that the card has already detected the error. Being a smart card, it will handle its own containment, and set the “Detected parity error” bit in its PCI Status register. This log identifies the responder as the HP zx1 ioa.

6.9.3.5 The HP zx1 ioa observes bad data parity while accepting PIO read return data, or while receiving a PCI-X Split Completion for a PIO cycle

Where logged: PI

Severity: Unc

Log code: 0x06

Other info logged: OAddress

Containment: The HP zx1 ioa pulls PERR# as required by the spec. This tells the smart I/O card to take whatever containment measures may be needed. The HP zx1 ioa substitutes hard-fail or -1, depending on the setting of the HF bit, in place of the return value, and this informs the smart driver of the event.

6.9.3.6 The HP zx1 ioa observes bad data parity while accepting DMA or P2P write data, or a transaction interrupt

Where logged: PI

Severity: Unc

Log code: 0x04

Other info logged: MasterID, IAddress

Containment: The HP zx1 ioa pulls PERR# as required by the spec. This tells the smart I/O card to take whatever containment measures may be needed. If the transaction is DMA, the HP zx1 ioa masks off the byte enables when forwarding the data to the rope host, for the data phase with the bad parity and all subsequent data in the same burst — i.e. until the next disconnect. If it is a transaction interrupt or Peer-to-peer, the HP zx1 ioa discards it, preventing it from ever reaching the rope.

6.9.3.7 The HP zx1 ioa observes bad address, command, or attribute parity on an I/O bus cycle

Where logged: PI

Severity: Unc

Log code: 0x0b

Other info logged: MasterID, IAddress

Containment: The HP zx1 ioa pulls SERR# (if enabled) as required by the spec. This tells the smart I/O card to take whatever containment measures may be needed. If the HP zx1 ioa claims the transaction, it will terminate with target abort and the cycle will not be forwarded to the ropes.

Rationale: PCI spec allows either target abort, or no response leading to master abort. The HP zx1 ioa decodes address, and may respond with DEVSEL# before it has time to realize there was an error. Thus, the HP zx1 ioa will target abort if the transaction appeared to belong to the HP zx1 ioa; otherwise no response will be given.

Note: IAddress log may not be meaningful. After all, there was an address parity error.

6.9.3.8 The HP zx1 ioa receives no DEVSEL# when mastering I/O bus, or receives a PCI-X master abort split completion error message

Where logged: PI

Severity: Unc

Log code: 0x0c

Other info logged: OAddress

Containment: The HP zx1 ioa master aborts as required by the spec. Transaction is discarded. If transaction is a PIO read or I/O port write, a fake return (-1 or HF as usual) is generated.

Note: This case does not apply when the HP zx1 ioa is mastering a DMA read Split Completion. In that case the error condition is logged but is considered a correctable error.

6.9.3.9 The HP zx1 ioa receives target abort response when initiating I/O bus cycle, or receives a PCI-X target abort split completion error message

Where logged: PI

Severity: Unc

Log code: 0x0d

Other info logged: OAddress

Containment: Transaction is discarded. If transaction is a PIO read or I/O port write, a fake return (-1 or HF as usual) is generated.

Note: This case does not apply when the HP zx1 ioa is mastering a DMA read Split Completion. In that case the error condition is logged, but is considered a correctable error.

6.9.3.10 The HP zx1 ioa target aborts an attempt to read from a write-only address

Where logged: PI

Severity: Unc

Log code: 0x0a

Other info logged: MasterID, IAddress

Containment: Transaction is discarded.

Note: In Elroy, this log code meant that Elroy had signaled a target abort. Elroy signals target abort for other reasons, but in the HP zx1 ioa there are only two reasons: (1) an I/O card attempted to read from a write-only address or (2) the HP zx1 ioa responded to a transaction with bad address/command/attribute parity. The latter already has a dedicate log code of its own, so in the HP zx1 ioa this code has been defined to cover only the former. Write-only areas are transaction interrupt and peer-to-peer spaces.

6.9.3.11 The HP zx1 ioa receives PIO Split Completion error message or unexpected byte count

Where logged: PI

Severity: Unc

Log code: 0x07

Other info logged: OAddress, CompMsg (also MasterID, IAddress for byte count error)

Containment: This error indicates that the HP zx1 ioa received either (1) a Split Completion error message it doesn't already handle in response to a PIO transaction or (2) a Split Completion transaction with an incorrect byte count. In the event of an error message, the message is stored in the Completion Message Log. If the byte count is incorrect, then the Unexpected Byte Count bit is set in the Completion Message Log, and the inbound address/attributes are logged. In either case, the outbound address of the original PIO transaction is logged and a fake read return (-1 or HF as usual) is generated. Driver must contain the problem upon seeing the read return.

The HP zx1 ioa also sets the architected "Unexpected Split Completion" bit in its PCI-X Status register for the byte count error.

Note: If a Split Completion message has a corrupt byte count, then the byte count error condition takes precedence. The message logged in the Completion Message register is most likely not reliable in such a scenario.

Note also that the Inbound Address Log is *not* meaningful for Split Completions. Only the Inbound Attribute Log will actually be of interest for the byte count error.

6.9.3.12 The HP zx1 ioa times out waiting for a PIO Split Completion

Where logged: PI

Severity: Unc

Log code: 0x0e

Other info logged: OAddress

Containment: Transaction is discarded. A fake return (-1 or HF as usual) is generated.

Note: The timeout can be disabled by programming a zero as the timeout period — the default behavior for Elroy compatibility. The purpose is to deal with cards that issue split responses and then never do Split Completions.

6.9.4 Errors in dumb I/O bus mode

6.9.4.1 The HP zx1 ioa internal parity error in PIO write, DMA read Split Completion, or P2P write data

Where logged: PI

Severity: Fatal.

Log code: 0x15

Other info logged: OAddress

Containment: I/O Bus fatal mode. The card should pull PERR# as well, but the HP zx1 ioa will ignore this PERR# assertion error so as to not set the overflow bit in a scenario that is just a single error.

6.9.4.2 The HP zx1 ioa internal parity error in connected DMA read return data

Where logged: PI

Severity: Fatal

Log code: 0x18

Other info logged: IAddress, MasterID

Containment: I/O Bus fatal mode. The card should pull PERR# as well, but the HP zx1 ioa will ignore this PERR# assertion error so as to not set the overflow bit in a scenario that is just a single error.

6.9.4.3 The HP zx1 ioa observes PERR# asserted while driving out PIO, DMA read Split Completion, or P2P write data, or receives a PCI-X write data parity error split completion error message

Where logged: PI

Severity: Fatal

Log code: 0x13

Other info logged: OAddress

Containment: I/O bus fatal mode, -1 or HF returned for I/O port write.

6.9.4.4 The HP zx1 ioa observes PERR# asserted while driving out connected DMA read return data, or while giving a split response to a PCI-X DMA read

Where logged: PI

Severity: Fatal

Log code: 0x19

Other info logged: MasterID, IAddress

Containment: I/O bus fatal mode.

6.9.4.5 The HP zx1 ioa observes bad data parity while accepting PIO read return data, or while receiving a PCI-X Split Completion for a PIO cycle

Where logged: PI

Severity: Fatal

Log code: 0x16

Other info logged: OAddress

Containment: I/O bus fatal mode. The HP zx1 ioa pulls PERR# as required by the spec. The HP zx1 ioa substitutes hard-fail or -1 as usual, in place of the return value.

6.9.4.6 The HP zx1 ioa observes bad data parity while accepting DMA or P2P write data, or a transaction interrupt

Where logged: PI

Severity: Fatal

Log code: 0x14

Other info logged: MasterID, IAddress

Containment: I/O bus fatal mode entered. The HP zx1 ioa pulls PERR# as required by the spec. If the transaction is DMA, the HP zx1 ioa masks off the byte enables when forwarding the data to the rope host; the BE masking starts for the data phase with the bad parity and continues for all subsequent data in the same burst — i.e. until the next disconnect. If it is a transaction interrupt or Peer-to-peer, the HP zx1 ioa discards it, preventing it from ever reaching the rope. I/O bus fatal mode retries any subsequent transaction, and eventually disables arbitration.

6.9.4.7 The HP zx1 ioa observes bad address, command, or attribute parity on an I/O bus cycle

Where logged: PI

Severity: Fatal

Log code: 0x1b

Other info logged: MasterID, IAddress

Containment: The HP zx1 ioa pulls SERR# (if enabled) as required by the spec. If the HP zx1 ioa claims the transaction, it will terminate with target abort and the cycle will not be forwarded to the ropes. I/O bus fatal mode entered.

Rationale: PCI spec allows either target abort, or no response leading to master abort. The HP zx1 ioa decodes address, and may respond with DEVSEL# before it has time to realize there was an error. Thus, the HP zx1 ioa will target abort if the transaction appeared to belong to the HP zx1 ioa; otherwise no response will be given.

Note: IAddress log may not be meaningful. After all, there was an address parity error

6.9.4.8 The HP zx1 ioa receives no DEVSEL# when mastering I/O bus, or receives a PCI-X master abort split completion error message

Where logged: PI

Severity: Fatal

Log code: 0x1c

Other info logged: OAddress

Containment: The HP zx1 ioa master aborts as required by the spec. Transaction is discarded. If transaction is a PIO read or I/O port write, a fake return (-1 or HF as usual) is generated. I/O bus fatal mode entered.

Note: This case does not apply when the HP zx1 ioa is mastering a DMA read Split Completion. In that case the error condition is logged but is considered a correctable error.

6.9.4.9 The HP zx1 ioa receives target abort response when initiating I/O bus cycle, or receives a PCI-X target abort split completion error message

Where logged: PI

Severity: Fatal

Log code: 0x1d

Other info logged: OAddress

Containment: Transaction is discarded. If transaction is a PIO read or I/O port write, a fake return (-1 or HF as usual) is generated. I/O bus fatal mode entered.

Note: This case does not apply when the HP zx1 ioa is mastering a DMA read Split Completion. In that case the error condition is logged but is considered a correctable error.

6.9.4.10 The HP zx1 ioa target aborts an attempt to read from a write-only address

Where logged: PI

Severity: Fatal

Log code: 0x1a

Other info logged: MasterID, IAddress

Containment: Transaction is discarded. I/O bus fatal mode

Note: In Elroy, this log code meant that Elroy had signaled a target abort. Elroy signals target abort for other reasons, but in the HP zx1 ioa there are only two reasons: (1) an I/O card attempted to read from a write-only area or (2) the HP zx1 ioa responded to a transaction with bad address/command/attribute parity. The latter already has a dedicated log code of its own; so in the HP zx1 ioa this code has been defined to cover only the former. Write-only areas are transaction interrupt and peer-to-peer spaces.

6.9.4.11 The HP zx1 ioa observes SERR# asserted

Where logged: PI[7]

Severity: Fatal

Log code: 0x1f

Other info logged: None

Containment: I/O bus fatal mode entered.

Note: The HP zx1 ioa can not discern the reason for the SERR#. However, if it was an address, command, or attribute parity error, the card that detected it should set both the “Signaled System Error” and “Detected Parity Error” bits. Otherwise, only the first bit will be set. This is how software can tell these apart.

[7]This log is attempted even if the HP zx1 ioa itself is the agent pulling SERR#. But in this case, the HP zx1 ioa will already have logged the underlying address/command/attribute parity error. Since both of these errors are of the same severity, only the parity error is logged, and the SERR# assertion causes the OV bit to be set in the error log. This is a little (!) ugly, but consistent with Elroy behavior.

6.9.4.12 The HP zx1 ioa receives PIO Split Completion error message or unexpected byte count

Where logged: PI

Severity: Fatal

Log code: 0x17

Other info logged: OAddress, CompMsg (also MasterID, IAddress for byte count error)

Containment: This error indicates that the HP zx1 ioa received either (1) a Split Completion error message it doesn't already handle in response to a PIO transaction or (2) a Split Completion transaction with an incorrect byte count. In the event of an error message, the message is stored in the Completion Message Log. If the byte count is incorrect, then the Unexpected Byte Count bit is set in the Completion Message Log, and the inbound address/attributes are logged. In either case, the outbound address of the original PIO transaction is logged and a fake read return (-1 or HF as usual) is generated. I/O bus fatal mode is entered.

the HP zx1 ioa also sets the architected "Unexpected Split Completion" bit in its PCI-X Status register for the byte count error.

Note: If a Split Completion message has a corrupt byte count, then the byte count error condition takes precedence. The message logged in the Completion Message register is most likely not reliable in such a scenario.

Note also that the Inbound Address Log is *not* meaningful for Split Completions. Only the Inbound Attribute Log will actually be of interest for the byte count error.

6.9.4.13 The HP zx1 ioa times out waiting for a PIO Split Completion

Where logged: PI

Severity: Fatal

Log code: 0x1e

Other info logged: OAddress

Containment: Transaction is discarded. A fake return (-1 or HF as usual) is generated. I/O bus fatal mode is entered.

Note: The timeout can be disabled by programming a zero as the timeout period — the default behavior for Elroy compatibility. The purpose is to deal with cards that issue split responses, and then never do Split Completions.

6.10 Summary

Table 6.3 summarizes the error list.

Logged	Code	Severity	Section	Page	Description	MasterID	IAddress	OAddress	CompMsg
Independent of smart/dumb mode:									
PI	0x01	Corr	6.9.2.1	30	No transaction in 16 clocks	•			
PI	0x02	Corr	6.9.2.3	31	DMAR split completion aborted			•	•
PI	0x02	Corr	6.9.2.4	31	Unexpected split completion	•	•		•
PI	0x11	Fatal	6.9.2.6	32	AGP error				
PI	0x12	Fatal	6.9.2.2	31	Register error			•	
PI	0x1f	Fatal	6.9.2.5	32	SERR# asserted				
Smart mode:									
PI	0x03	Unc	6.9.3.3	33	PERR# during PIOW			•	
PI	0x04	Unc	6.9.3.6	33	Bad DMAW parity	•	•		
PI	0x05	Unc	6.9.3.1	32	Bad internal PIOW parity			•	
PI	0x06	Unc	6.9.3.5	33	Bad PIORR parity			•	
PI	0x07	Unc	6.9.3.11	35	PIO split completion error	•	•	•	•
PI	0x08	Unc	6.9.3.2	32	Bad internal DMARR parity	•	•		
PI	0x09	Unc	6.9.3.4	33	PERR# during DMARR	•	•		
PI	0x0a	Unc	6.9.3.10	35	Read of int/P2P space	•	•		
PI	0x0b	Unc	6.9.3.7	34	Bad address phase parity	•	•		
PI	0x0c	Unc	6.9.3.8	34	No DEVSEL#			•	
PI	0x0d	Unc	6.9.3.9	34	Target abort received			•	
PI	0x0e	Unc	6.9.3.12	35	Timeout on PIO completion			•	
Dumb mode:									
PI	0x13	Fatal	6.9.4.3	36	PERR# during PIOW			•	
PI	0x14	Fatal	6.9.4.6	37	Bad DMAW parity	•	•		
PI	0x15	Fatal	6.9.4.1	36	Bad internal PIOW parity			•	
PI	0x16	Fatal	6.9.4.5	36	Bad PIORR parity			•	
PI	0x17	Fatal	6.9.4.12	39	PIO split completion error	•	•	•	•
PI	0x18	Fatal	6.9.4.2	36	Bad internal DMARR parity	•	•		
PI	0x19	Fatal	6.9.4.4	36	PERR# during DMARR	•	•		
PI	0x1a	Fatal	6.9.4.10	38	Read of int/P2P space	•	•		
PI	0x1b	Fatal	6.9.4.7	37	Bad address phase parity	•	•		
PI	0x1c	Fatal	6.9.4.8	37	No DEVSEL#			•	
PI	0x1d	Fatal	6.9.4.9	38	Target abort received			•	
PI	0x1e	Fatal	6.9.4.13	39	Timeout on PIO completion			•	
PI	0x1f	Fatal	6.9.4.11	38	SERR# asserted				

Table 6.3: Summary of HP zx1 ioa error codes

Chapter 7

Online Replacement Support

7.1 Overview

OL* support is a platform issue. While the HP zx1 ioa contains elements needed to support OLR, that is only a small piece of the OL* puzzle.

OL* is, therefore, beyond the scope of this ERS.

Chapter 8

Master Controller

8.1 Overview

The master controller generates transactions to targets on the I/O bus (PIO reads and writes).

8.1.1 Configuration Reads and Writes

Configuration reads and writes on the PCI bus are generated through the configuration address register and configuration data register in the HP zx1 ioa. To generate a configuration cycle, the configuration address register must first be properly set up, then an access to the configuration data register will initiate a corresponding configuration read or write on the PCI bus.

Care must be taken to make sure only one configuration cycle is being attempted at once for two reasons:

1. It is important to make sure a configuration cycle has finished before the configuration address is changed. This can be done by waiting for the deferred completion on a configuration read or doing a read to the configuration address register after a configuration write and waiting for that deferred completion before going to the next configuration cycle.
2. Configuration cycles are not flow controlled by the chipset, so having too many outstanding can cause the delayed transaction FIFO to overflow.

8.2 Registers

This section describes the master controller registers.

8.2.1 Status, Information, and Control Register

This register is used to reset the I/O bus, control the HP zx1 ioa's response to error conditions on PIO reads, clear the error log, and control the VGA address range on this I/O bus. Bits implemented by the HP zx1 ioa are defined in Register 8.1 on the following page.

The RC and RF bits are used to reset the I/O bus. See Chapter 5 for more details on resets.

The HF bit is used to control the HP zx1 ioa's response to errors on PIO reads. The CE and CL bits are used to clear the HP zx1 ioa's error logs without erasing errors that software hasn't read yet. See Chapter 6 for more information on the HP zx1 ioa error strategy. Table 8.1 on the next page summarizes their behavior in the HP zx1 ioa.

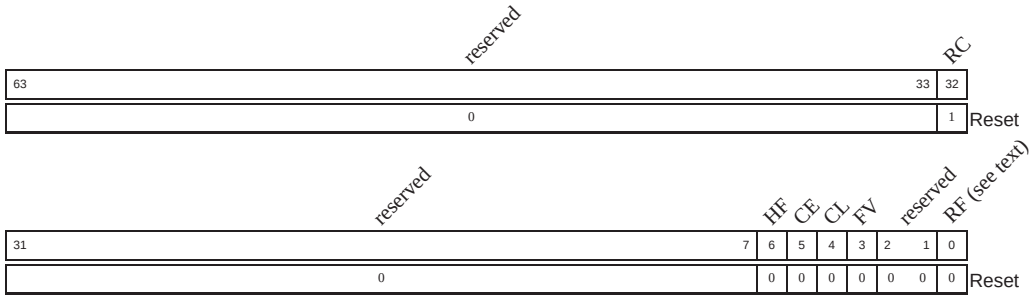
Finally, the FV bit is used to alert the HP zx1 ioa to the presence of a VGA compatible graphics card. This is necessary because VGA cards are allowed to claim their legacy address space outside the normal PCI auto-configured space. This is defined from the perspective of CPU bus accesses being forwarded to the I/O bridge that has the VGA card beneath it. For the HP zx1 ioa, setting FV = 1 means that transactions to VGA cards are forwarded *to this function*. Therefore,

Value written CE	Value written CL	Previous value CE	Previous value CL	Value after write CE	Value after write CL	What happened to logs
0	0	X	X	0	0	unchanged or new error logged
1	0	X	X	0	0	new error logged
1	0	X	X	1	0	unchanged
0	1	0	X	0	0	unchanged or new error logged
0	1	1	X	0	0	not cleared — new error logged
0	1	1	X	0	1	cleared
1	1	X	X	*undefined*		*undefined*

Table 8.1: Behavior of the CE and CL bits in clearing error logs.

when FV = 1, the HP zx1 ioa will not claim as a target any transaction that falls within VGA legacy address space. See Section 9.2.1 on page 53 for more information on the VGA space.

Register 8.1: STATUS, INFORMATION, AND CONTROL (0x0108)



Reserved Must be set to 0 for writes, must be ignored for reads.

RC Reset Complete. (Read-only)

This bit returns 1 while the PCI bus is held in reset. To change the reset status of the I/O bus, write to the Reset Function bit below.

HF Hard-Fail Response Control.

If HF = 1, a Hard-Fail response is returned on the CPU bus if there is an error on a PIO read or I/O port write. Examples include parity errors, master-aborts, target-aborts, or the HP zx1 ioa in fatal mode. If HF = 0, then a −1 is returned instead.

CE Clear Error Log Enable.

Clearing of error logs is enabled if CE = 1. If CE = 0, error log clearing is disabled. Whenever a new error is logged by this function, the CE bit is reset to 0. See Table 8.1 for more details.

CL Clear Error Log.

If a 1 is written to the CL bit and CE = 1, then the error logs are cleared. Otherwise, no action is taken.

Reads of this bit indicate whether the clear was successful. See Table 8.1.

FV Forward VGA. (Set to indicate that VGA transactions are forwarded *to* this HP zx1 ioa.)

If FV = 1, VGA Frame Buffer space will be considered Local Bus Peer-to-peer, which will not be claimed by the HP zx1 ioa. This bit must be set if a VGA device is located on this PCI[X]/AGP bus. Only one Rope Guest (the HP zx1 ioa or Elroy) in the system should have its FV bit set. See Section 9.2.1 on page 53 for more details on address decoding of the VGA Frame Buffer.

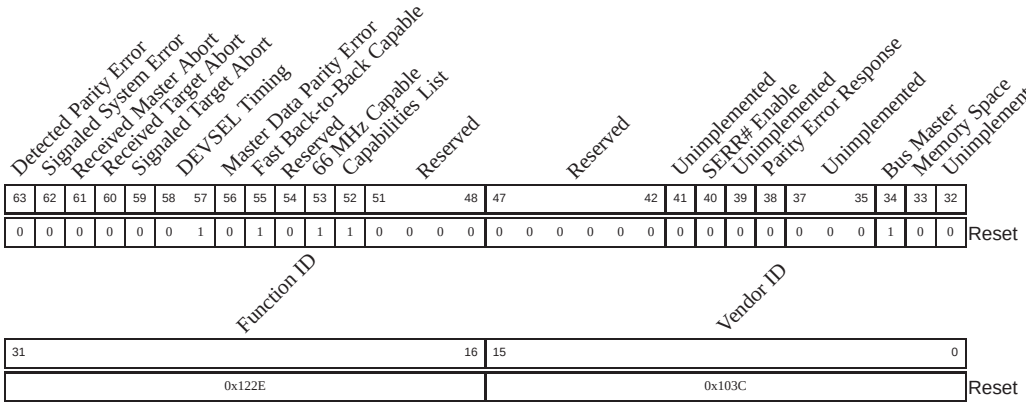
⚠ **RF** Reset Function. (Write-only)

Writing a 1 to this bit resets the PCI[X]/AGP bus. Bus will be held in reset until a subsequent write sets this bit to 0. This reset is intended to be used for OLR operations only; see Chapter 5, Reset Strategy, for more information.

Reads of this bit will always return 0. To check the reset status of the I/O bus, read from the Reset Complete bit above.

8.2.2 PCI Registers

Register 8.2: FUNCTION ID (0x0000)



Function ID (Read only) Identifies function. The value is unique to the HP zx1 ioa.

Vendor ID (Read only) Identifies vendor as HP.

PCI Status See PCI 2.2 Section 6.2.3 and PCI-X 1.0 Section 7.1 for more information.

Detected Parity Error When set, this status bit indicates that the HP zx1 ioa has detected an address or data parity error. This bit is the only indication that a parity error occurred if parity checking was disabled.

Signaled System Error When set, this status bit indicates that the HP zx1 ioa has asserted SERR#.

Received Master Abort When set, this status bit indicates that a transaction the HP zx1 ioa mastered was Master-Aborted.

Received Target Abort When set, this status bit indicates that a transaction the HP zx1 ioa mastered was Target-Aborted.

Signaled Target Abort When set, this status bit indicates that the HP zx1 ioa has terminated a transaction with Target-Abort.

DEVSEL Timing (Read only) Indicates that the HP zx1 ioa is a Medium speed decoder.

Master Data Parity Error When set, this status bit indicates that a transaction mastered by the HP zx1 ioa had a data parity error.

Fast Back-to-Back Capable (Read only) Indicates that the HP zx1 ioa is capable of understanding PCI Fast Back-to-Back transactions.

66 MHz Capable (Read only) Indicates that the HP zx1 ioa is capable of 66 MHz PCI operation.

Capabilities List (Read only) Indicates that the HP zx1 ioa implements a New Capabilities linked list at offset 0x0034.

PCI Control See PCI 2.2 Section 6.2.2 and PCI-X 1.0 Section 7.1 for more information.

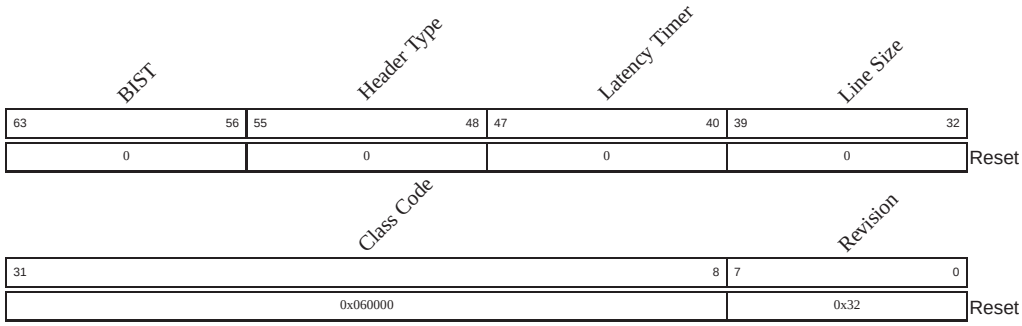
SERR# Enable When set to 1, the HP zx1 ioa may drive SERR#.

Parity Error Response When set to 1, the HP zx1 ioa takes normal action to parity errors. When 0, the HP zx1 ioa ignores all parity errors.

Bus Master The HP zx1 ioa ignores this bit; as host bridge, it can always master the bus.

Memory Space When set to 1, the HP zx1 ioa may claim Memory Space transactions as a target.

Register 8.3: FUNCTION CLASS (0x0008)



BIST (Read only) Always returns 0.

Header Type (Read only) Always returns 0.

Latency Timer PCI Latency Timer value (PCI 2.2 spec Section 6.2.4).

Since the HP zx1 ioa gets its data in 16 byte packets when the latency timer expires the master controller may put out as many as the current data transfer plus four more if on a 32 bit bus (current transfer plus two more if on a 64 bit bus) instead of the current transfer plus one more that the spec indicates.)

Line Size PCI Cache Line Size (PCI 2.2 spec Section 6.2.4). Valid values are listed below; any other value will be treated as indicating 64 byte cache lines.

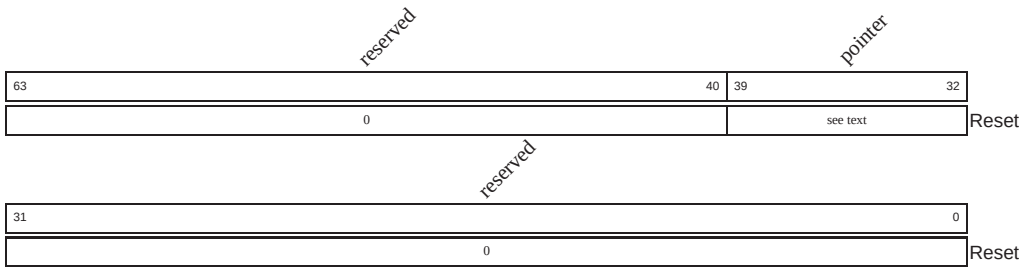
0010_0000 128 bytes

0001_0000 64 bytes

Class Code (Read only) PCI Class Code (PCI 2.2 spec Section 6.2.1). The HP zx1 ioa identifies itself as a Host bridge.

Revision (Read only) Revision number. Bits 4–7 provide the major revision number (starting with 0), and bits 0–3 provide the minor revision number. 0x00=Rev 1.0, 0x01=Rev1.1(metal mask change for config cycle trapping), 0x20= Rev 2.0 (many changes including change to revision number definition major revision will now match up with “common”revision name (i.e. start with 1 not 0).

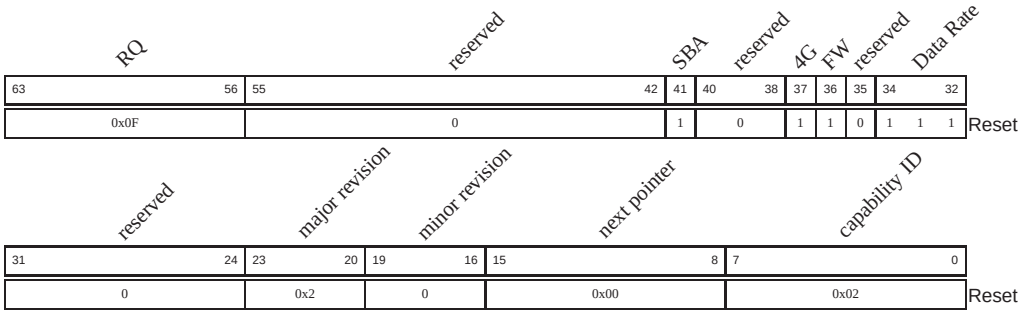
Register 8.4: CAPABILITIES POINTER (0x0030)



Pointer (Read only) Points to a list of registers describing extra capabilities. If this HP zx1 ioa is configured to be in AGP mode this points to register 0x60 which indicates AGP as an extra capability. If this the HP zx1 ioa is not configured to be in AGP mode this points to register 0xA0 which indicates PCI-X as an extra capability.

8.2.3 AGP Registers

Register 8.5: CAPABILITIES LIST AND AGP STATUS (0x0060)



See AGP 2.0 spec Section 6.1.9 for more information.

RQ (Read only) Specifies the maximum number of outstanding AGP requests supported by the HP zx1 ioa. The value of 0x0F indicates a pipeline depth of 16 entries.

SBA (Read only) Indicates the HP zx1 ioa supports AGP sideband addressing.

4G (Read only) Indicates the HP zx1 ioa supports addresses above 4G.

FW (Read only) Indicates the HP zx1 ioa supports FW PIO transactions.

Data Rate (Read only) Indicates the HP zx1 ioa can handle all three (1x, 2x, 4x) data rates.

See AGP 2.0 spec Section 6.1.8 for more information.

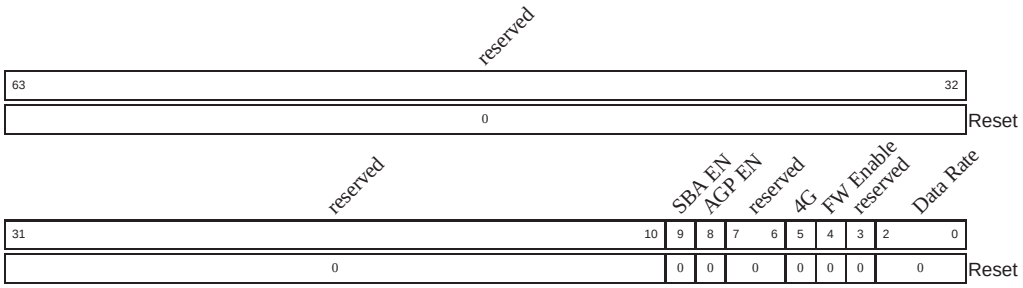
Major Revision (Read only) Major revision of the capability being implemented (2 for the HP zx1 ioa AGP).

Minor Revision (Read only) Minor revision of the capability being implemented (0 for the HP zx1 ioa AGP).

Next Pointer (Read only) Pointer to next capability identifier in the list. Indicates no other capabilities.

Capability ID (Read only) Indicates AGP (2).

Register 8.6: AGP COMMAND (0x0068)



See AGP 2.0 spec Section 6.1.10 for more information

SBA EN Enables AGP sideband address mechanism.

AGP EN Enables the HP zx1 ioa to participate in AGP transactions. (This should always be set if FW-Enable is set)

4G Enables the HP zx1 ioa to accept AGP transactions to addresses above 4G

FW Enable Enables the HP zx1 ioa to use Fast Writes on PIO writes.

Data Rate Indicates data rate to be used. Rate applies to both AD and SBA buses.

001 AGP 1x

010 AGP 2x

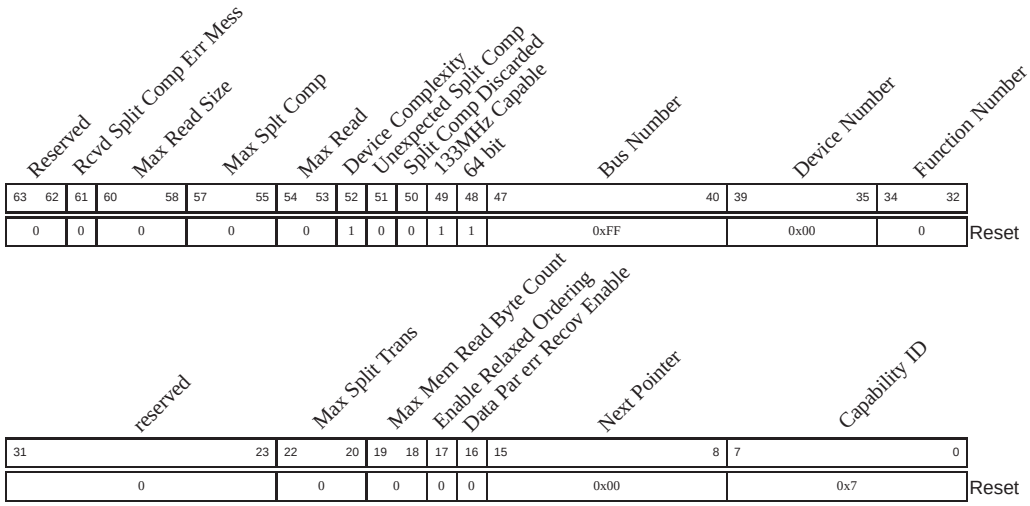
100 AGP 4x

This register will be reset with the AGP (PCI) RESET signal. Therefore this register must be programed after RESET has been released.

8.2.4 PCI-X Registers

The Capabilities list, command and status register for PCI-X is byte accessible to allow access by ACPI.

Register 8.7: CAPABILITIES LIST, COMMAND AND STATUS PCI-X (0x00A0)



See Section 7.2 of the PCI-X 1.0 spec for more information.

Received Split Completion Error Message A 1 indicates an error message was returned with a Split Completion. Once set, this bit remains set until a 1 is written to this bit.

Designed Maximum Cumulative Read Size (Read only) always returns 0

Designed Maximum Outstanding Split Transactions (Read only) always returns 0

Designed Maximum Memory Read Byte Count (Read only) always returns 0

Device Complexity (Read only) Value indicates that the HP zx1 ioa is a bridge; this isn't really very meaningful for a host bridge.

Unexpected Split Completion A 1 indicates the error condition that an unexpected split completion with the HP zx1 ioa's initiator ID was received, or a split completion that didn't match the request. Once set, this bit remains set until a 1 is written to this bit.

Split Completion Discarded (Read Only) always returns 0.

133 MHz Capable (Read Only) always returns 1.

64 bit (Read only) Returns a 1 indicating that the HP zx1 ioa is a 64-bit device.

Bus Number (Read only) Always returns 0xFF since the HP zx1 ioa is never the target of a configuration write.

Device Number (Read only) Always returns 0x00.

Function Number (Read only) Always returns 0.

Maximum Outstanding Split Transactions (Read only) Always returns 0.

Maximum Memory Read Byte Count (Read only) Always returns 0. (The largest read transaction the HP zx1 ioa will generate is 16 bytes.)

Enable Relaxed Ordering (Read only) Always returns 0

Data Parity Error Recovery Enable Enabled with a 1.

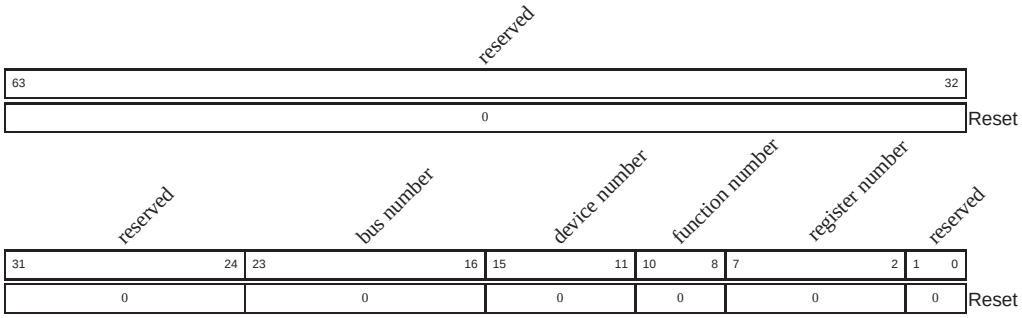
Next Pointer (Read only) Points to NULL (end of capabilities list).

Capability ID (Read only) Indicates PCI-X.

8.2.5 I/O Configuration Space Registers

See Section 8.1.1 on page 43 for details on using these registers.

Register 8.8: CONFIGURATION ADDRESS (0x0040)



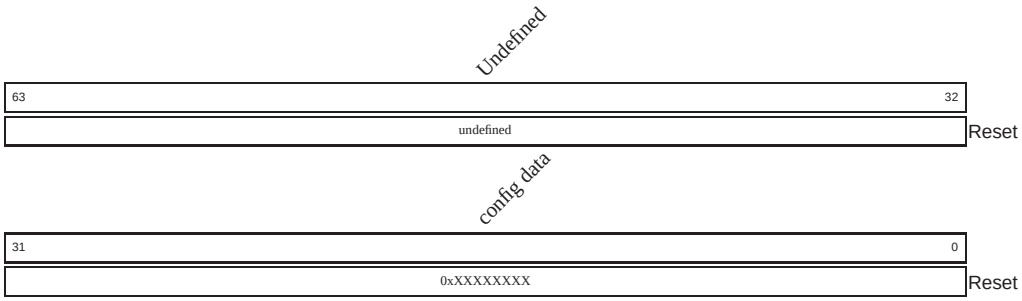
Bus Number Identifies the PCI bus number the configuration access is intended for. A bus number of 0 indicates that the PMC is to do a type 0 (destined for the HP zx1 ioa's bus) configuration transaction, and any other bus number will result in a type 1 configuration cycle with the address information passed on for the bridge to use.

Device Number This determines which AD line is asserted to generate an IDSEL assertion. The AD line that will be asserted for a given device number is AD(device_number + 16) and the rest of AD[31:16] will be zero. A device number greater than 15 will result in no AD lines being driven for use as an IDSEL line. AD lines used for IDSEL will be pre-charged for 4 PCI cycles before Frame# is asserted (address stepping). This will allow 60ns pre-charge time for a 66 MHz system and 120ns for a 33 MHz system to resistively decouple the IDSEL line from the AD line.

Function Number Identifies the function number within a physical PCI device.

Register Number Identifies the word address of the target function register.

Register 8.9: CONFIGURATION DATA (0x0048)



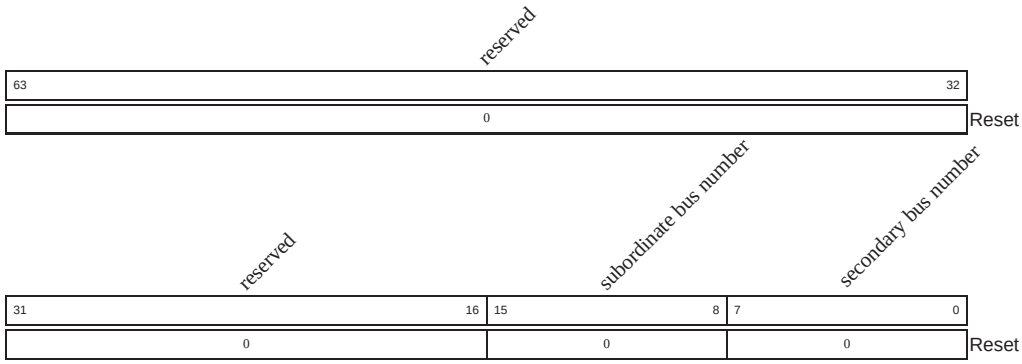
Undefined Reads must be ignored and writes have no effect.

Config Data Reading or writing this register causes the corresponding read or write configuration cycle to be generated. The behavior of the HP zx1 ioa when it master aborts (nobody responds to the operation) will vary depending on the state of the CM bit in the Error Configuration Register (see Section 6.3.1).

Care must be taken to make sure that the read or write to the config_data register is paired with that value desired in the config_address register. In general it is desirable to have a lock so only one processor is allowed to do config cycles at a time. The config_address register should be set up, then the config_data register accessed. If the access to the config_data access was a write, it should be followed by a read of the config_address register to make sure the config cycle completes before writing the config_address register for the next config cycle. Flow control in the IOC/the HP zx1 ioa requires that there be only one write to the config data register outstanding at a time.

8.2.6 Miscellaneous Registers

Register 8.10: BUS NUMBER (0x0058)



Subordinate_bus_number Highest number bus below this the HP zx1 ioa. This has no effect on chip operation and is just for firmware convenience.

Secondary_bus_number Number of the bus directly below this the HP zx1 ioa. This has no effect on chip operation for PCI and AGP and is just for firmware convenience for those buses. For PCI-X this number will appear as the “secondary bus number” (bits 7:0) in the attribute phase of configuration cycles, which in turn becomes the bus number for devices on that bus.

Chapter 9

Slave Controller

9.1 Overview

The Slave Controller (SC) is responsible for responding to PCI, PCI-X, and AGP transactions to Memory-space as a target. No I/O Port space (IOP) transactions are claimed. Supported transactions include DMA, Message Signalled Interrupts (MSI), Memory Mapped I/O (MMIO) Remote Bus Peer-to-peer writes, PIO read split completions, Flush, and Fence. See Table 9.1 for a list of supported transaction types on PCI, PCI-X, and AGP buses.

Target Transaction Types	PCI	PCI-X	AGP
DMA Read	**yes**	**yes**	**yes**
DMA Write	**yes**	**yes**	**yes**
Message Signaled Interrupt Write	**yes**	**yes**	n/a
Remote Bus MMIO Peer-to-peer Write	**yes**	**yes**†	n/a
PIO Read Split Completions	n/a	**yes**	n/a
Flush	n/a	n/a	**yes**
Fence	n/a	n/a	**yes**

†Single Dataphase only.

Table 9.1: Supported transaction types, the HP zx1 ioa as Target.

9.1.1 VGA MMIO Frame Buffer

VGA frame buffer space is the 128 KB region from address 640K to (768K−1). It receives some special treatment because VGA compatible cards are allowed to claim their classic hard-coded address space from the primitive days before auto-configuration. As such, the HP zx1 ioa must know not to claim the VGA frame buffer if a VGA card is present on this the HP zx1 ioa's bus. In addition, if there is a VGA card in the system, but not on this the HP zx1 ioa's bus, then VGA frame buffer peer-to-peer writes may be done if they are enabled.

9.2 Address Decoding

Memory-space bus transactions claimed by the Slave Controller include DMA, Message Signalled Interrupt writes, and Remote Bus Peer-to-peer writes. Check Table 9.1 to see which I/O buses support which transaction types. No I/O Port space transactions are ever claimed by the HP zx1 ioa (see Table 2.1 on page 6).

Figure 9.1 on the following page illustrates how HP zx1 ioa decides what the type of a memory-space transactions is. The *sets* in Figure 9.1 represent regions of memory that the HP zx1 ioa is programmed to recognize. The colored *intersections* between sets determine the type of memory-space transaction (DMA, MSI, etc.). For example, the entire diagram is colored "DMA (not IOVA)" except for areas covered by some other set. Therefore, the HP zx1 ioa considers all addresses to be non-IOVA DMA except those that fall in a region you told it are *not*.

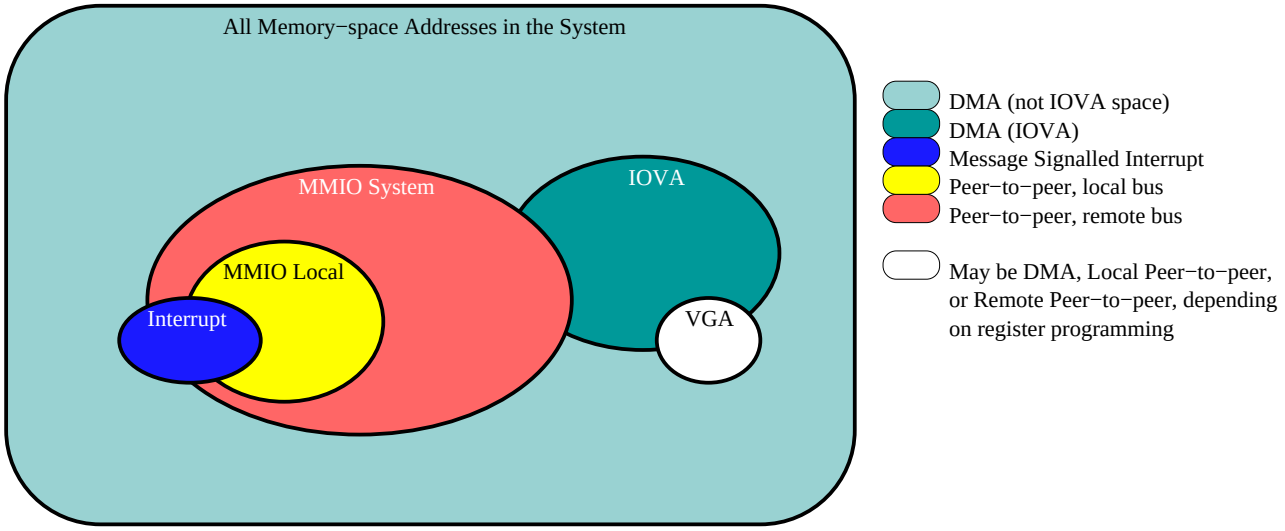


Figure 9.1: HP zx1 ioa Slave Controller's view of memory-space addresses.

The HP zx1 ioa is programmed to recognize the sets from Figure 9.1 by using the following registers. Each category below directly corresponds to one set. The registers listed under each category define the subsets of memory that make up the complete set.

MMIO System Space: All the MMIO PCI[X] devices in the system.

- WGMMIO for addresses $\geq$ 4 GB (see Section 9.3.1, Registers 9.1 and 9.2)
- WLMMIO for addresses $<$ 4 GB (see Section 9.3.1, Registers 9.5 and 9.6)

MMIO Local Space: All the MMIO PCI[X] devices below this HP zx1 ioa. This region ***must*** be a subset of MMIO System Space.

- GMMIO for addresses $\geq$ 4 GB (see Section 9.3.1, Registers 9.3 and 9.4)
- LMMIO for addresses $<$ 4 GB (see Section 9.3.1, Registers 9.7 and 9.8)
- ELMMIO for additional addresses $<$ 4 GB (see Section 9.3.1, Registers 9.9 and 9.10)

Interrupt Space: Message Signalled Interrupt space.

- MSI (see Section 9.3.1, Registers 9.11 and 9.12)

IOVA Space: I/O Virtual Address DMA space.

VGA Space: VGA MMIO Frame Buffer. This region receives special treatment. See Section 9.2.1 below for details.

Now that we have defined the address regions, the intersections of these regions determine the transaction type. The HP zx1 ioa uses the following rules to make that determination.

DMA if the address is *not* within **MMIO System Space** and *not* within **Interrupt Space**.

Message Signalled Interrupt if the address falls within **Interrupt Space**.

- To disable MSI's, disable the Range Enable bit of the MSI Base register.

Remote Bus Peer-to-peer if the address falls within **MMIO System Space**, *not* within **MMIO Local Space**, and *not* within **Interrupt Space**.

- To disable Remote Bus Peer-to-peer to VGA frame buffer space, clear the VPE bit of Register 9.13 on page 58.

- To disable Remote Bus Peer-to-peer elsewhere, program the range registers so that MMIO System Space exactly equals MMIO Local Space.

Local Bus Peer-to-peer if the address falls within **MMIO Local Space**. The HP zx1 ioa will not claim these transactions since the target is some other PCI[X] device below this the HP zx1 ioa.

Note that PIO read split completions are a special case of transaction claimed by the HP zx1 ioa . Since they are actually the completion of transactions generated by the HP zx1 ioa's Master Controller, they are not discussed here.

9.2.1 VGA MMIO Frame Buffer

The VGA MMIO Frame Buffer is located at the memory-space addresses from 640 KB to (768 KB − 1). It may be decoded as DMA, Remote Bus Peer-to-peer, or Local Bus Peer-to-peer, depending on the programming of the FV bit of the Status, Information, and Control Register (page 44) and VPE bit of the Slave Control Register (page 58). The default decoding is DMA. See Table 9.2 for details.

FV bit of Register 8.1	VPE bit of Register9.13	Transaction Type
0	0	DMA (default)
0	1	Remote Bus Peer-to-peer
1	X	Local Bus Peer-to-peer

Table 9.2: Programming of VGA MMIO Frame Buffer.

9.3 Registers

This section describes all the Slave Controller registers. All register fields marked "Reserved" must be set to zero for writes and ignored for reads.

9.3.1 Address Range Registers

Address range registers define regions of address space. They exist in *Base* and *Mask* pairs. The Mask register determines what bits of the Base register are compared to the incoming address. If this comparison matches, then a given address is within the region defined by the range register pair. Each register pair has a *Range Enable* bit. The pair is ignored if their Range Enable is 0; only when the Range Enable is set to 1 are incoming addresses compared with the range register pair.

Therefore, when the following pseudo-code evaluates to TRUE, the `Address` is within the region defined by the `BASE` and `MASK` pair:

```
(BASE == (MASK && Address)) && Range_Enable
```

There are some restrictions on how Address Masks may be programmed. In order to prevent aliasing, Masks must be set so that there are no deasserted bits in more significant positions then asserted bits. Table 9.3 on the next page shows examples of acceptable and illegal Mask programming.

In addition, the Base Address must be programmed so that if a bit is asserted in the Base, it must be asserted in the Mask. See Table 9.4 on the following page for examples of correct and illegal Base programming.

Using this technique, the first address of the region will be naturally aligned to the size of the region. For example, an 8 MB region will be aligned on an 8 MB address boundary.

The WLMMIO, LMMIO, and ELMMIO range registers can only be used to specify MMIO addresses below 4 GB. The WGMMIO and GMMIO range registers are only to be used to specify MMIO addresses above 4 GB. To ensure that WGMMIO and GMMIO only refer to address ranges over 4 GB, at least one bit in their Base Address[43:32] must be asserted. Obviously, that also means that at least one Address Mask bit must be asserted, too.

The MSI range registers are unique in that they may be used to specify a memory region above or below 4 GB. However, they may not be used to specify a memory region in the first 2 GB of memory. Therefore, at least one bit in MSI Base [43:31] must be set.

Unimplemented bits in address range registers are hardwired to 0; reads return 0, writes have no effect.

Address Mask	
0b 1000 0000	Okay
0b 1111 1100	Okay
0b 0100 0000	Illegal
0b 1100 1000	Illegal

The underlined bits cannot be set to 0, since there are less significant bits set to 1.

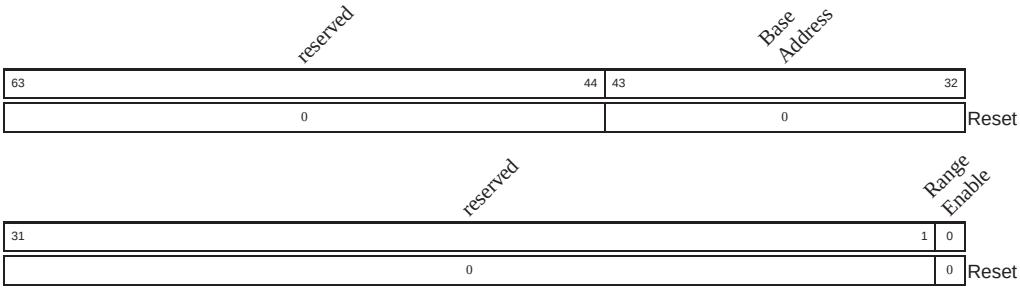
Table 9.3: Examples of Address Mask programming.

Address Mask	Base Address	
0b 1110 0000	0b 0000 0000	Okay
0b 1110 0000	0b 0100 0000	Okay
0b 1110 0000	0b 0100 0100	Illegal
0b 1110 0000	0b 0001 0000	Illegal

The underlined bits cannot be set to 1, since the corresponding Mask bits are set to 0.

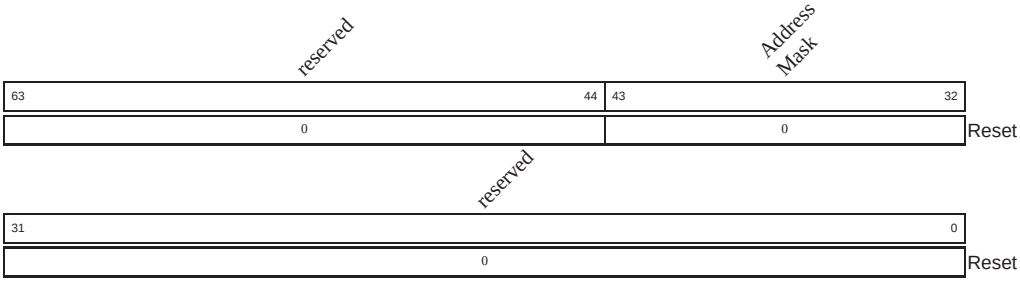
Table 9.4: Examples of Base Address programming.

Register 9.1: WGMMIO BASE (0x0230)

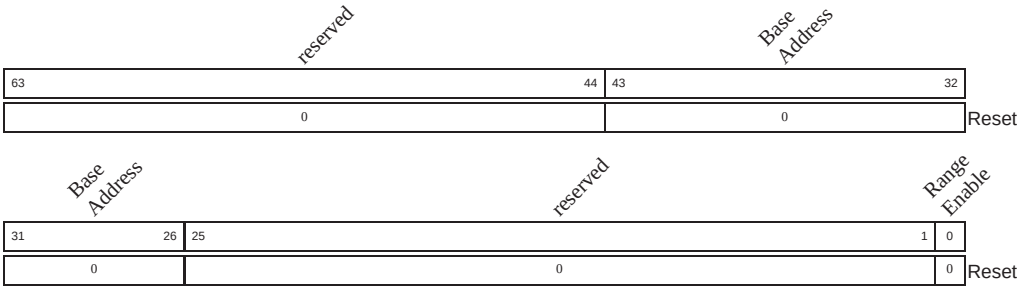


Base Address[43:32] must have at least one bit asserted.

Register 9.2: WGMMIO MASK (0x0238)

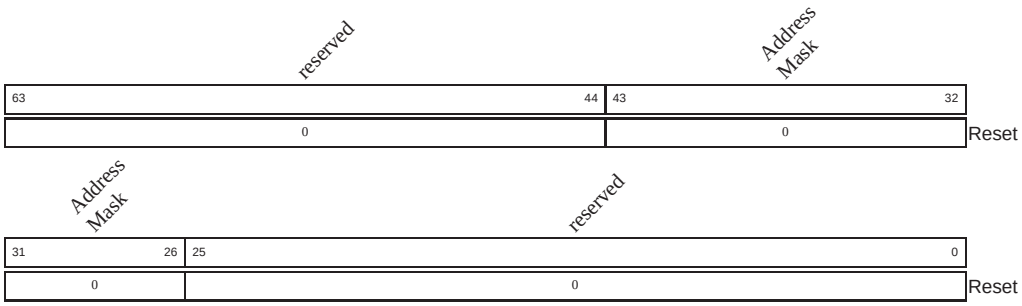


Register 9.3: GMMIO BASE (0x0210)

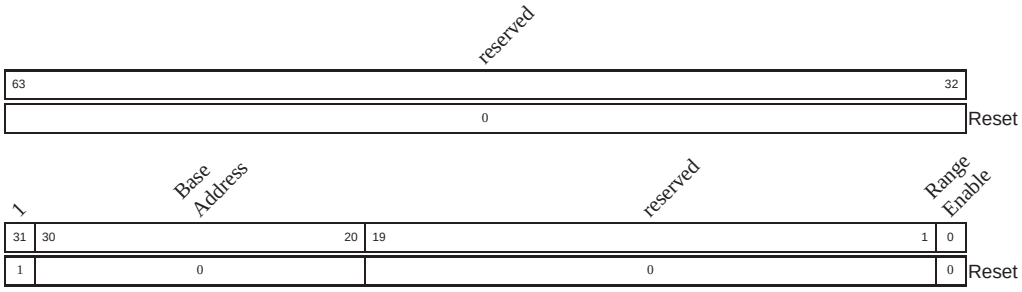


Base Address[43:32] must have at least one bit asserted.

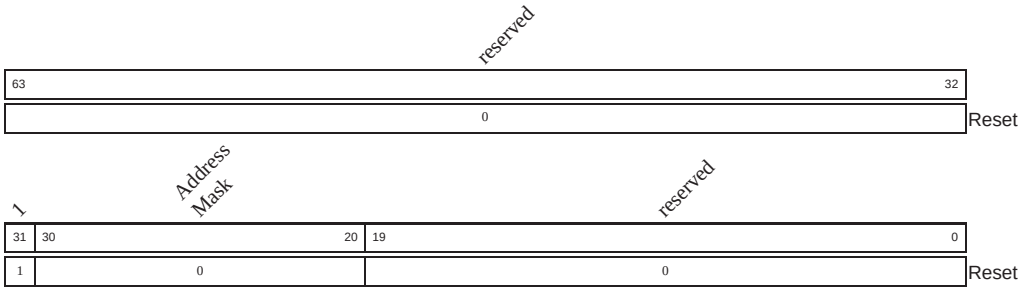
Register 9.4: GMMIO MASK (0x0218)



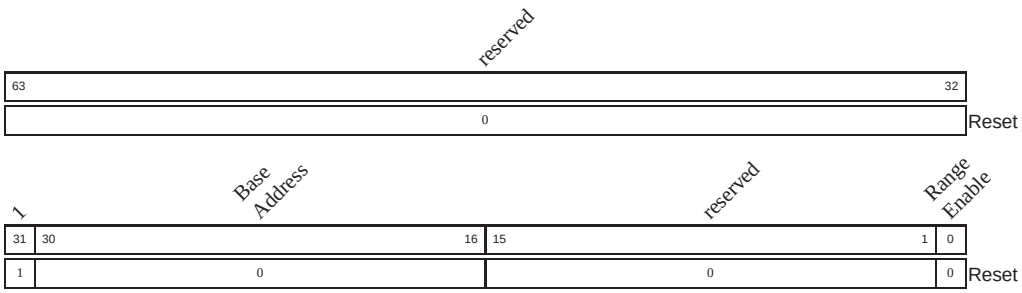
Register 9.5: WLMMIO BASE (0x0220)



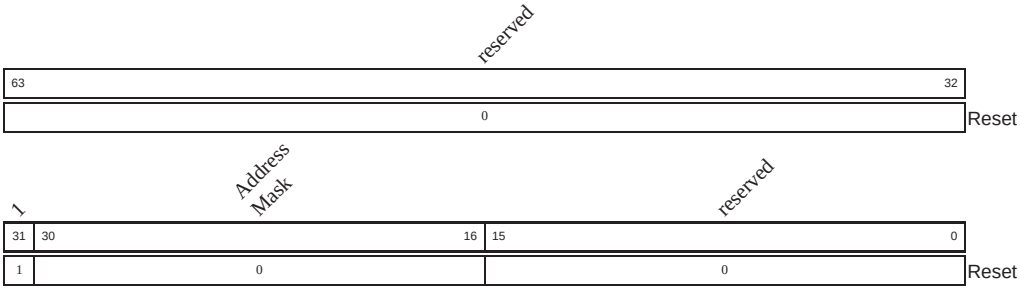
Register 9.6: WLMMIO MASK (0x0228)



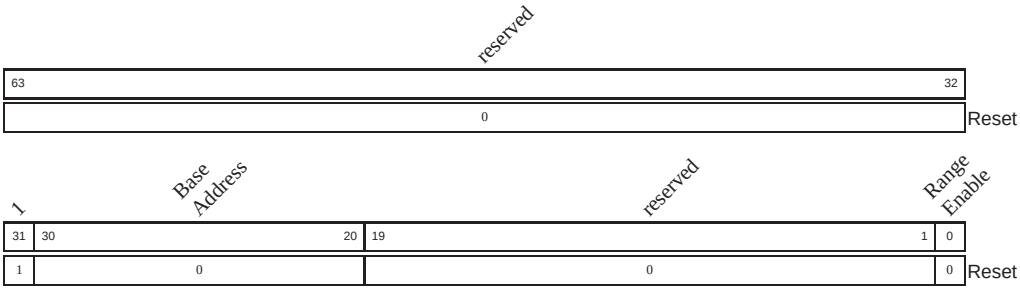
Register 9.7: LMMIO BASE (0x0200)



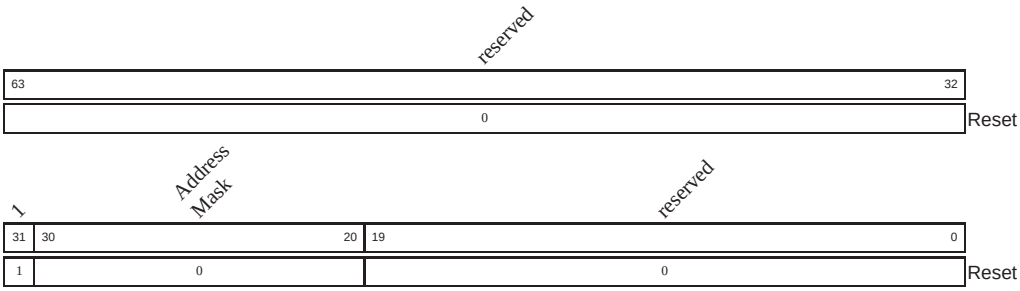
Register 9.8: LMMIO MASK (0x0208)



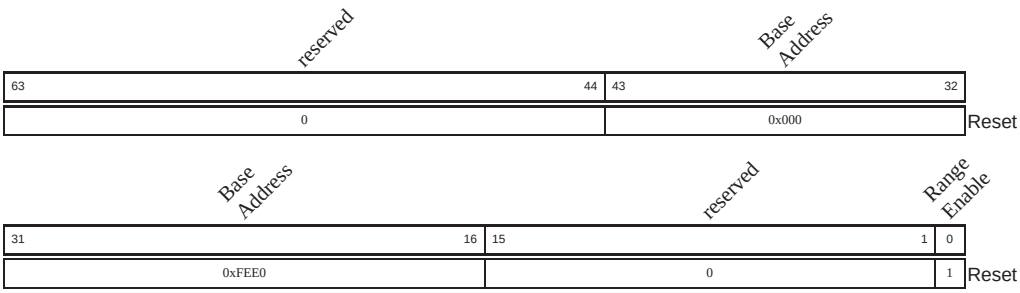
Register 9.9: ELMMIO BASE (0x0250)



Register 9.10: ELMMIO MASK (0x0258)

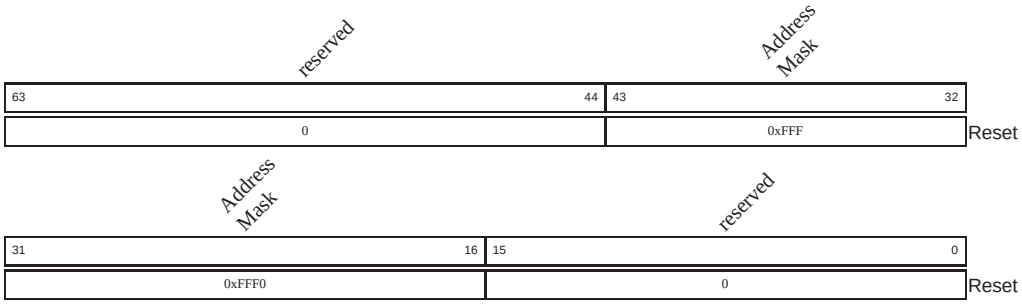


Register 9.11: MSI BASE (0x0280)



Base Address[43:31] must have at least one bit asserted.

Register 9.12: MSI MASK (0x0288)



9.3.2 Slave Control Register

Register 9.13 controls:

- how many subsequent wait states are inserted by the PCI slave controller.
- what algorithm is used to match PCI masters with delayed completions.
- how it responds to VGA frame buffer accesses.

Register 9.13: SLAVE CONTROL (0x0278)

See Section 9.3.2

Strict_DCR If set to 1, the HP zx1 ioa will strictly follow Section 3.3.3.3 of the *PCI Local Bus Specification*, Revision 2.2.

If set to 0, the HP zx1 ioa will follow the relaxed Delayed Completion Resource comparison used by Elroy.

Disable_64_bit If set to 1, the HP zx1 ioa will only respond as a 32 bit target.

VPE VGA Peer-to-peer Enable.

If VPE = 1 then access to VGA Frame Buffer space may be decoded as Remote Bus Peer-to-peer. See Section 9.2.1 on page 53 for more details on address decoding of the VGA Frame Buffer.

SLT Subsequent Latency Timer.

Controls the maximum number of PCI wait states after the first data phase of the transaction. The HP zx1 ioa will complete all subsequent data phases within SLT+2 clocks. This field has no effect on PCI-X or AGP.

Set this field to 6 or less to be strictly compliant with the *PCI Local Bus Specification*, Revision 2.2.

The Subsequent Latency Timer (SLT) controls the number of PCI[1] wait states that can be inserted in a DMA read stream after the first data phase. If the first data phase of a PCI DMA read is not available when a Master requests it, he will be immediately Retried. After the first data phase, the HP zx1 ioa will attempt to burst as long as possible. If prefetching hints have been tuned to the bandwidth needs of the PCI master, then DMA reads should be able to burst efficiently. If sufficient data is not available to continue the burst, the HP zx1 ioa will wait for more data by inserting wait states on the PCI bus. If SLT+1 wait states have been inserted before more data is available, the HP zx1 ioa will Disconnect the transaction on the following clock.

Section 3.5.1.2 of the *PCI Local Bus Specification*, Revision 2.2 requires that all subsequent data phases complete in 8 or less clocks. Since the maximum number of clocks the HP zx1 ioa will take to complete a data phase is SLT+2, set the SLT to 6 or less to be compliant. If strict PCI specification compliance is not required, it is recommended that the SLT be set to the average number of clocks of overhead that will be inserted if a DMA master is disconnected and has to re-start its transaction.

[1]The SLT has no effect on PCI-X or AGP; those buses do not allow optional wait-states.

The Strict_DCR bit is used to control the comparison used by the DMA Read Delayed Completion Resource (DCR). In relaxed mode (Strict_DCR = 0), the DCR compares:

- Address[43:2] (parity must be good)
- Master ID (taken from GNT#)

In strict mode (Strict_DCR = 1), the DCR additionally compares:

- State of REQ64#
- Bus Command (MR, MRL, or MRM)
- Address[63:44] and Address[1:0] (parity must be good)

Section 3.3.3.3.2 of the *PCI Local Bus Specification*, Revision 2.2 requires all of the above to be compared by a Delayed Completion Resource, with the exception of Master ID. Comparison of Byte Enables is optional, since DMA Reads are prefetchable.

The VGA Peer-to-peer Enable bit controls whether VGA frame buffer space can be forwarded as Remote Bus Peer-to-peer or not. Deasserting this bit disables VGA remote peer-to-peer from this bus; asserting enables it. Note that the FV bit in the Status, Information, and Control Register controls whether there is a VGA compatible card on this the HP zx1 ioa's PCI[X]/AGP bus; if that bit is asserted, then the VGA Peer-to-peer Enable bit has no effect, since the HP zx1 ioa will never claim the VGA frame buffer in that case. Also, this bit has no effect if Remote Bus Peer-to-peer is not supported by this bus type. Table 9.1 on page 51 lists the buses for which remote peer-to-peer is supported.

Chapter 10

Arbiter

10.1 Overview

The HP zx1 ioa chip has the ability to connect to three different buses: PCI, PCI-X, and AGP. Each of these buses has its own specifications that govern the behavior of its arbiter. What this really means is that the HP zx1 ioa will have three different arbiters, each optimized for its particular bus. The AGP arbiter will actually be implemented as part of the AGP logic in the slave controller as arbitration in AGP is intimately involved with data transfers, not just bus ownership.

10.1.1 Arbitration Algorithms

The HP zx1 ioa will use two different algorithms for arbitration.

Limited/Unfair Once the master is granted the bus, the arbiter will keep the grant asserted (assuming request is still asserted) for a programmable number of clocks to allow a master to do multiple bus transaction per turn on the bus. This could be used to allow masters who tend to do shorter transfers to still get their fair share of the bus, or to slant priority towards particular masters. Limited/Unfair can be selected on a per master basis.

Round Robin This is a fair algorithm in which all the masters on the bus have equal priority. Requests will be serviced in a circular manner, so that even if every master is requesting the bus, they will all get their turn. This is probably the preferred algorithm.

Note that the HP zx1 ioa does not allow an external arbiter to replace its internal arbiter.

10.2 PCI Arbitration

The PCI arbiter controls which master has ownership of the bus at any time. When no one is requesting the bus, it controls how the bus is parked. The arbiter accomplishes this task through the use of request/grant pairs. A master will assert its request line when it needs the bus. The arbiter will then make a decision as to whether that master will be granted permission to start a transaction. A typical bus arbitration is shown in Figure 10.1 on the following page.

Figure 10.1 illustrates basic PCI arbitration between two masters. Ownership of the bus is granted to master A in clock two by asserting GNT#-A. If master A does not begin a transaction (by asserting FRAME#) within 16 clocks of GNT#-A being asserted, then the arbiter can assume that master A is broken. Master A begins a transaction on the rising edge of clock three, and finishes it in clock four. Master A deasserted its request line after clock two, and master B asserted its request. In clock five, the arbiter grants master B the bus, and master B begins its transaction in cycle six. The arbiter notes that nobody else is requesting the bus, and leaves the bus parked on master B until such time as another master needs the bus.

The HP zx1 ioa will always park the bus on its master controller when no one else wants the bus.

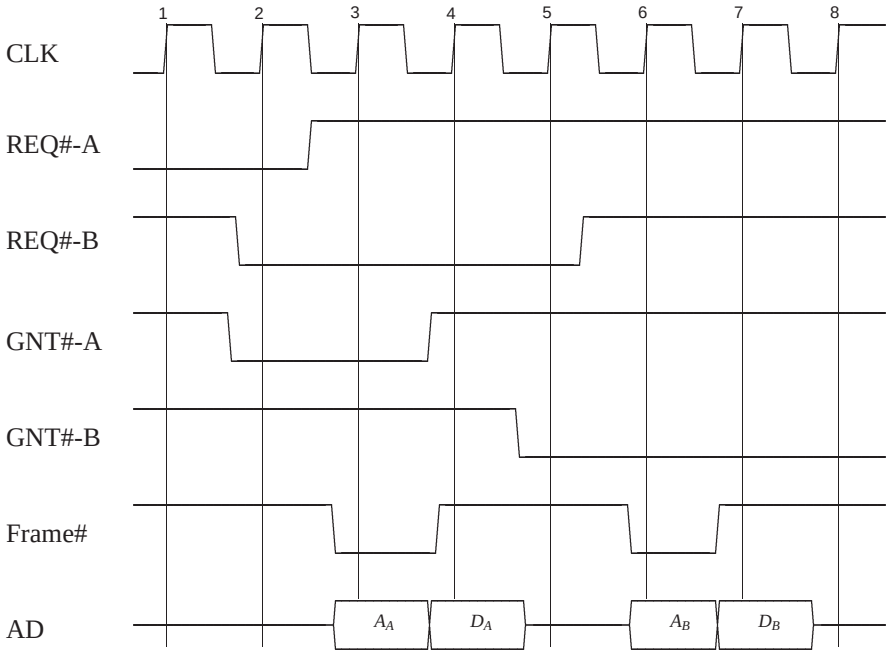


Figure 10.1: Simple PCI arbitration

10.3 PCI-X Arbitration

The signalling protocol and operation for the PCI-X arbitration is the same as it is for PCI, except in a few areas:

1. Requests and grants will be registered.
2. After the arbiter asserts a grant and the bus is idle, it must keep it asserted for at least five clocks or until the initiator drives FRAME# or deasserts REQ#.
3. There must always be a clock when no grant is asserted when deasserting the grant from one master and asserting the grant to the next master. For performance purposes, the HP zx1 ioa violates this when deasserting its own internal grant when the bus is parked on the HP zx1 ioa. The HP zx1 ioa's master controller has been designed to handle this, and since the HP zx1 ioa's grant isn't externally visible, this "violation" can't be detected.
4. A master has only 6 clocks in which the bus is idle before it asserts FRAME#, rather than the 16 allowed for PCI. The HP zx1 ioa allows 8 for PCI-X.
5. Latency timers in bus masters (doesn't really affect the HP zx1 ioa arbiter)
   (a) The default for a master is now 64 (instead of 0 in PCI mode). This is intended to allow the device to transfer at least two ADBs worth of data at a shot.

10.4 AGP Arbitration

An AGP arbiter is really a superset of a PCI arbiter. It implements a full PCI arbiter as well as controlling data transfers on the AGP bus. The HP zx1 ioa will only support a single device on the AGP bus. This means that the arbiter will only need to manage two request/grant pairs (one to the internal master, and one to the external card).

One of the new functions that the arbiter takes on is to control the data transfer for DMA transactions. When the arbiter sees that a DMA transaction is outstanding it asserts a GNT and places an appropriate value on the ST[2:0] wires (see Table 10.1 on the next page for values and descriptions).This signals the card to either sink (in the case of a DMA read return) or source (in the case of a DMA write) data from the HP zx1 ioa. Figure 10.2 on the facing page shows such a transaction.

ST[2:0]	Description
000	LP Read data
001	HP read data
010	LP write data
011	HP write data
100	Reserved
101	Reserved
110	Reserved
111	Begin bus transaction

Table 10.1: Values for ST[2:0]

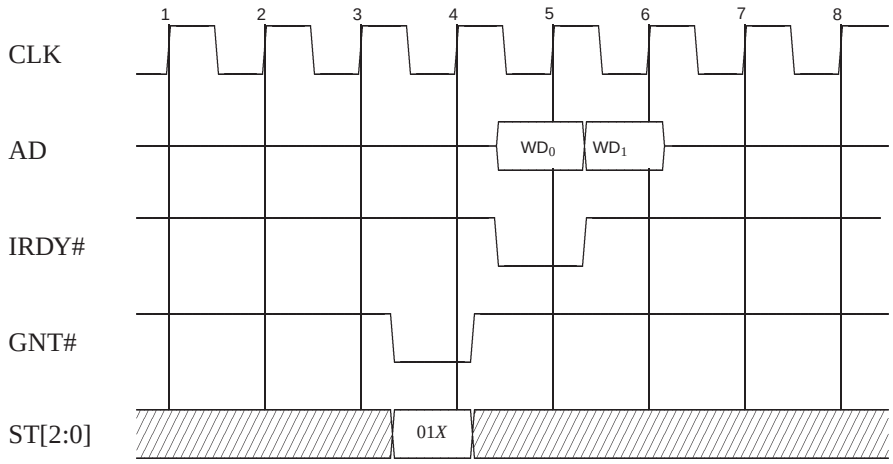


Figure 10.2: DMA write transaction on AGP

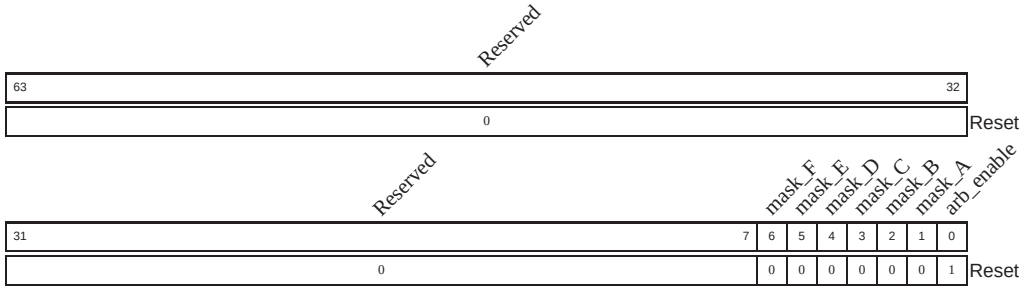
The arbiter will also need to handle both PCI and PIPE# requests on the bus. A device begins these requests by asserting the REQ# signal. The arbiter will respond by asserting GNT# and placing the value of 3'b111 on the ST[2:0] wires. This indicates to the device that it is allowed to begin either a PCI transaction or a PIPE# AGP request. The master must begin the PIPE# request within one clock or a FRAME# (PCI style) request within one or two clocks.

The HP zx1 ioa will actually implement the AGP arbiter as part of the Slave Controller.

10.5 Registers

The registers which control the HP zx1 ioa's arbiter are described below.

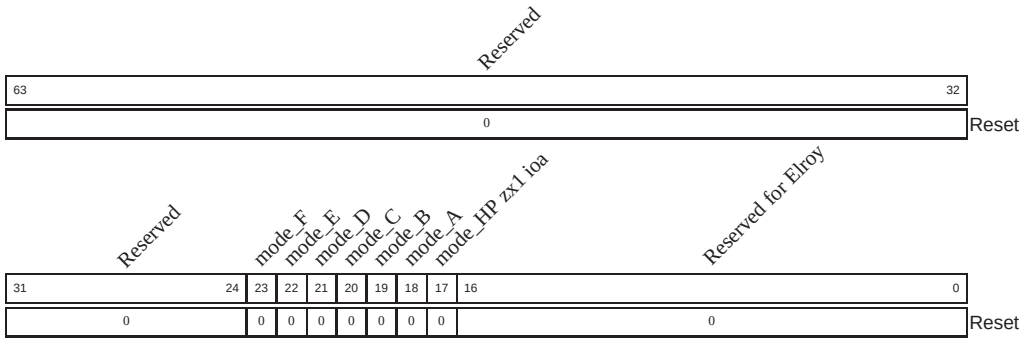
Register 10.1: ARBITRATION MASK (0x0080)



mask_A-F A master is enabled when its corresponding bit is set to one. When a master is disabled, it will not be granted ownership of the bus. If the config bits indicate that the system has less than 6 masters, hardware will prevent the setting of mask_F. In the case of AGP arbitration, the arbiter isn't looking at anything but "A," so the B-F mask bits aren't important.

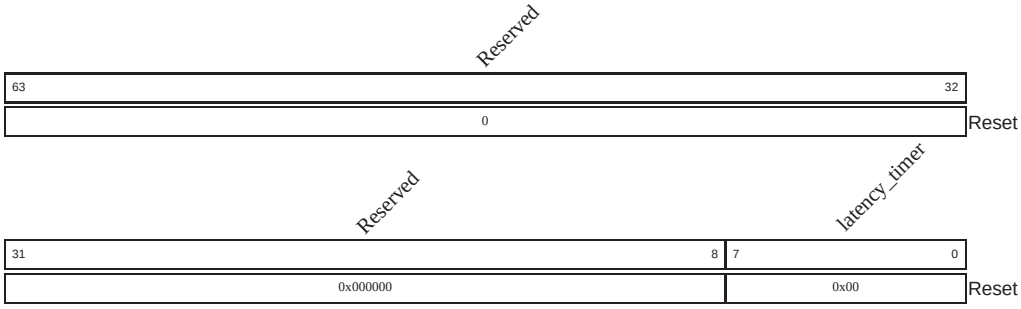
arb_enable Overall Arbitration enable. If this bit is 0, the HP zx1 ioa is in I/O bus fatal mode, and all other mask bits are "masked" to the arbiter. arb_enable acts as an overall arbitration mask. See the Error chapter for information on I/O fatal mode behavior.

Register 10.2: ARBITRATION MODE (0x0090)



mode_*X* When mode is low (default), the master will not use the multiple transaction latency timer. It will get the bus for a single transaction unless others are not requesting the bus. When mode is high, that master will be granted the bus for the number of clocks indicated by the latency timer. This is the limited/unfair mode.

Register 10.3: MULTI-TRANSACTION LATENCY TIMER (0x0098)



latency_timer The latency timer will start counting when a master is granted the bus. That master will not have its grant removed (unless it removes its request) until the timer expires. A setting of 0 for the latency timer is just like having normal round-robin arbitration. This latency timer should not be confused with the PCI architected latency timer that each master has in its PCI config space register set.

Chapter 11

Interrupts

11.1 Overview

The interrupt architecture in IPF platforms is known as SAPIC (Streamlined Advanced Programmable Interrupt Controller). As the name suggests, it is a simplified version of the APIC found in later x86 systems. The SAPIC is split into two major parts: an I/O unit and a local unit. The I/O unit is frequently called the I/O SAPIC, and it resides somewhere in the I/O subsystem. Its primary task is to convert interrupts that come in over interrupt wires (such as the INTx# signals on PCI) to a PCI transaction that serves as an interrupt message. The local unit is implemented in the processor. It receives the interrupt messages sent by the I/O SAPIC and delivers them to the processor core. When multiple interrupts are pending, the local unit prioritizes them. It can also mask specific interrupts and is used by the processor to generate interprocessor interrupts.

There is a local unit in every processor, so an MP system will contain multiple local units. Similarly, multiple I/O SAPIC's can exist in a system. In general, the number of I/O SAPIC's is not dictated by the number of processors or any other system parameters. For implementation convenience, there is one I/O SAPIC per the HP zx1 ioa.

Figure 11.1 shows an example interrupt path. The interrupt begins with a PCI card pulling an interrupt wire low. That change is seen by the I/O SAPIC, which then uses an internal look-up table to determine the content of the interrupt message it must send. In the HP zx1 ioa, the I/O SAPIC delivers the interrupt message directly to the write-posting FIFO. The message proceeds through the HP zx1 mio, which converts it to the appropriate transaction type on the processor bus. The local unit in the addressed processor accepts the message and delivers it to the processor itself.

Note that the figure depicts the I/O SAPIC as part of the HP zx1 ioa. The I/O SAPIC can be implemented as a stand-alone unit on the bus. Also, PCI cards are allowed to master their own interrupt message transactions, thus bypassing the I/O SAPIC entirely.

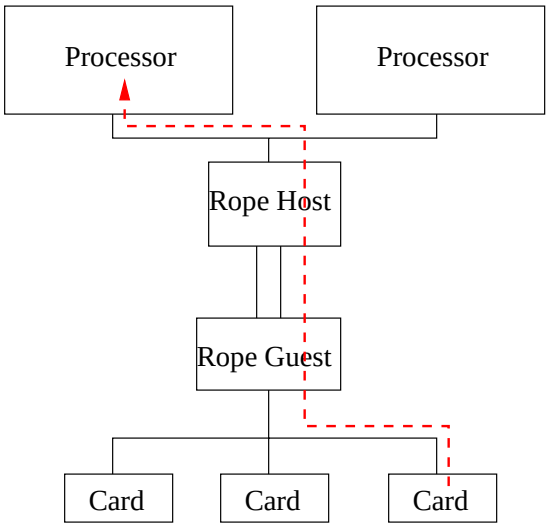


Figure 11.1: Interrupt example

11.2 The HP zx1 ioa support for SAPIC

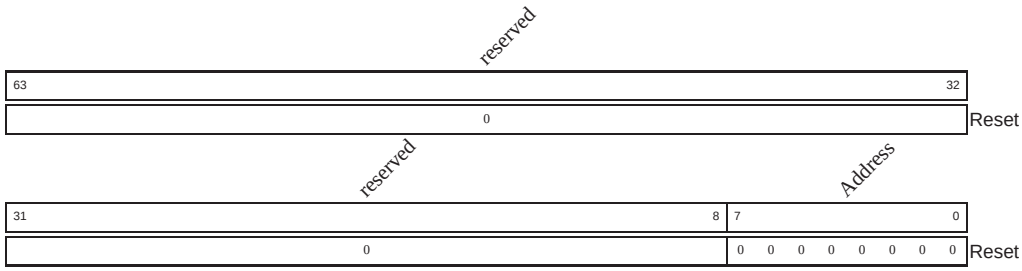
The HP zx1 ioa implements an I/O SAPIC, as described above. The HP zx1 ioa provides seven external interrupt lines via the PCI_INT_L[6:0] pins. The HP zx1 ioa supports three additional interrupt lines in PCI[X] modes providing INT7#, INT8# and INT9# . Finally, the HP zx1 ioa provides a register to which software can write to generate an interrupt.

11.3 I/O SAPIC Definition

11.3.1 I/O Select

The I/O SAPIC has a register indirect interface to software, i. e. the actual registers are not mapped into memory space. Instead, two indirection registers are mapped into memory space. One is essentially an address register and the other is a data register. The address register decides which I/O SAPIC register is actually being targeted. A read or write to the data register actually reads or writes the selected target. The address register is called the I/O Register Select, and the data register is the I/O Window. These registers are depicted in register diagrams 11.1 and 11.2.

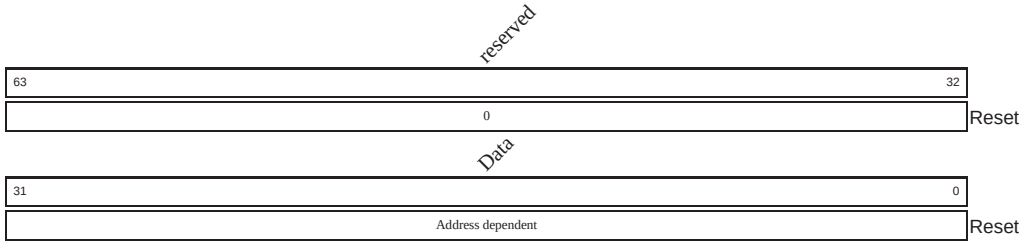
Register 11.1: I/O REGISTER SELECT (0x0800)



The 8 least-significant bits of the I/O Register Select determine which I/O SAPIC register is being accessed via the I/O Window. The HP zx1 ioa will return zeros for the reserved portion of this register.

11.3.2 I/O Window

Register 11.2: I/O WINDOW (0x0810)

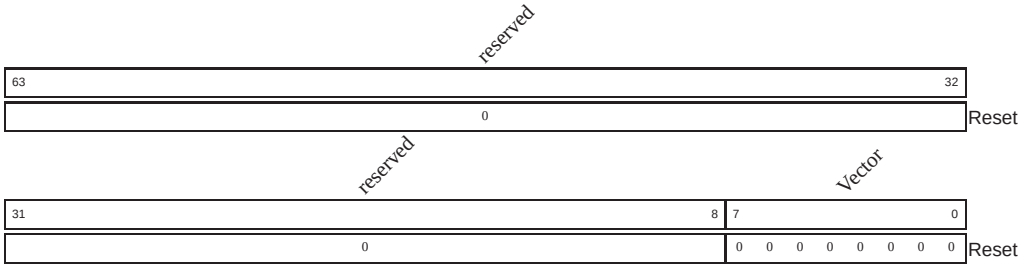


Reading the I/O Window register will return the value of the register selected by the address in I/O Register Select. Writing to the Window will similarly write to the selected register. If the selected register is read-only, the write will have no effect. Note that the Window provides only 32-bit access to the I/O SAPIC's indirect registers.

11.3.3 I/O EOI

The EOI (End-of-Interrupt) register is written when a processor has finished servicing a level-sensitive interrupt. If the interrupt signal corresponding to the written vector is still asserted, another interrupt message will be delivered. It is shown in register diagram 11.3.

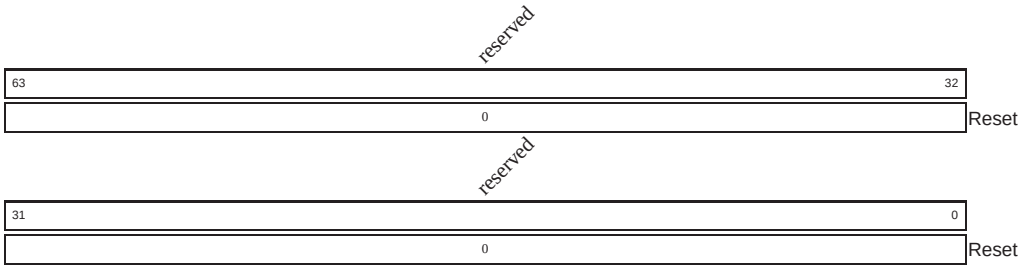
Register 11.3: I/O EOI (0x0840)



11.3.4 Software Interrupt

The final SAPIC register which is directly mapped into memory space is the Software Interrupt register. Writing to the Software Interrupt with any value will generate an interrupt. Reading this register will always return zero. The Software Interrupt is shown in register diagram 11.4. Note that, for this interrupt to occur, the corresponding SAPIC table entry must be unmasked and programmed to be **high-true**, **edge-sensitive**. The Software Interrupt, since it is independent of I/O bus activity, does not get masked upon entering I/O fatal mode.

Register 11.4: SOFTWARE INTERRUPT (0x0850)



11.3.5 I/O SAPIC Internal Registers

The I/O SAPIC internal registers accessed by the Register Select and Window registers are shown in Table 11.1 on the following page. Registers that are described as 64 bits wide are accessed as multiple independent 32-bit registers.

11.3.5.1 I/O SAPIC Version

Each I/O SAPIC unit contains a Version register that identifies different implementations. Software can use this register to provide compatibility between different I/O SAPIC versions. Since the HP zx1 ioa supports SAPIC, the version field is hardwired to 0x20. The Version register also contains the number of the highest Redirection Table Entry. For the HP zx1 ioa, this number is hardwired to 0x0A. This register is shown in Figure 11.2

11.3.5.2 Redirection Table Entry

The I/O Redirection Table has a dedicated entry for each interrupt input pin, as well as the Software Interrupt. The HP zx1 ioa has ten interrupt pins which can be connected to interrupt request lines on the I/O bus. How these interrupt lines are connected to and shared among the PCI slots is up to the system designer; see Chapter 3 for more information.

Each entry in the Redirection Table has fields controlling interrupt polarity, interrupt masking, edge/level sensitivity, etc. These fields are defined in Figure 11.3 on page 71. Note that changing the polarity field or mask field of a Redirection Table Entry may cause the immediate generation of an interrupt by the SAPIC. Note also that all mask bits in the table return to their reset value upon a the HP zx1 ioa function reset (see Chapter 5 for more information).

The type of the interrupt is determined by the Delivery Mode field. The fixed delivery mode is defined is 3'b00*x*, where *x* is the redirectable bit. If the redirectable bit is set to 0, then the interrupt is directed to the destination processor specified

Address	Register name	Reset Value	R/W
0x01	I/O SAPIC Version	0x000A_0020	Read only
0x10	Redirection Table Entry for int0[31:0]	0x0001_0000	R/W
0x11	Redirection Table Entry for int0[63:32]	0x0000_0000	R/W
0x12	Redirection Table Entry for int1[31:0]	0x0001_0000	R/W
0x13	Redirection Table Entry for int1[63:32]	0x0000_0000	R/W
0x14	Redirection Table Entry for int2[31:0]	0x0001_0000	R/W
0x15	Redirection Table Entry for int2[63:32]	0x0000_0000	R/W
0x16	Redirection Table Entry for int3[31:0]	0x0001_0000	R/W
0x17	Redirection Table Entry for int3[63:32]	0x0000_0000	R/W
0x18	Redirection Table Entry for int4[31:0]	0x0001_0000	R/W
0x19	Redirection Table Entry for int4[63:32]	0x0000_0000	R/W
0x1a	Redirection Table Entry for int5[31:0]	0x0001_0000	R/W
0x1b	Redirection Table Entry for int5[63:32]	0x0000_0000	R/W
0x1c	Redirection Table Entry for int6[31:0]	0x0001_0000	R/W
0x1d	Redirection Table Entry for int6[63:32]	0x0000_0000	R/W
0x1e	Redirection Table Entry for int7[31:0]	0x0001_0000	R/W
0x1f	Redirection Table Entry for int7[63:32]	0x0000_0000	R/W
0x20	Redirection Table Entry for int8[31:0]	0x0001_0000	R/W
0x21	Redirection Table Entry for int8[63:32]	0x0000_0000	R/W
0x22	Redirection Table Entry for int9[31:0]	0x0001_0000	R/W
0x23	Redirection Table Entry for int9[63:32]	0x0000_0000	R/W
0x24	Redirection Table Entry for SW_int[31:0]	0x0001_0000	R/W
0x25	Redirection Table Entry for SW_int[63:32]	0x0000_0000	R/W

Table 11.1: I/O SAPIC internal registers

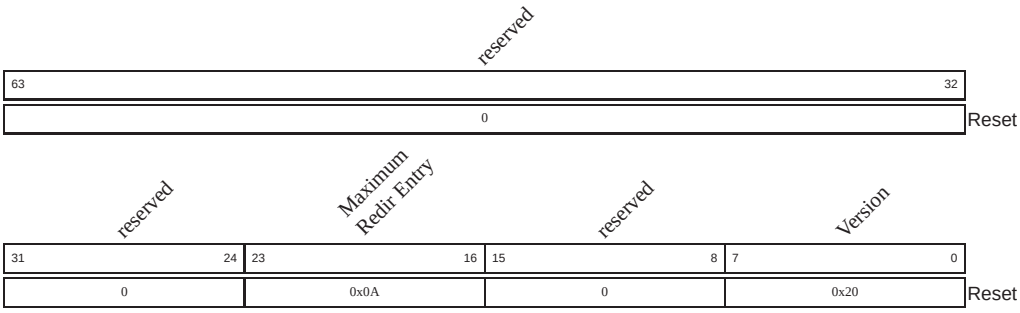
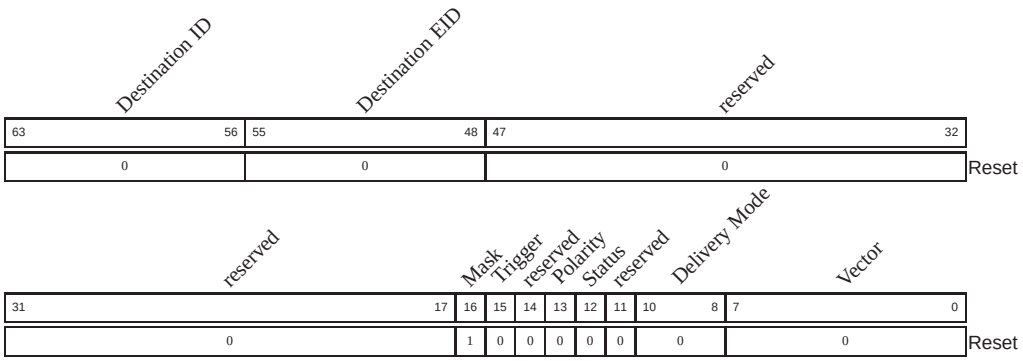


Figure 11.2: I/O SAPIC Version register

by the ID/EID fields. If the redirectable bit is set, then the interrupt may be redirected to one of the other processors on the same system bus as the one identified in the ID/EID fields.



Vector 8-bit interrupt vector

Delivery Mode Specifies how destination processors should act upon reception of interrupt. Values not specified below are reserved.

000 Fixed

001 Fixed with redirection

010 PMI

100 NMI

101 INIT

111 External INT

Status (Read only) Delivery status of interrupt. A 0 indicates that the interrupt is currently idle. A 1 indicates that a send is pending.

Polarity Assertion level of interrupt pin. A 0 indicates active high, while 1 indicates active low.

Trigger Interrupt sensitivity. A 0 indicates edge sensitive, while 1 indicates level sensitive.

Mask Masks interrupt delivery. The reset value of 1 masks all interrupt activity. Resets to 1 upon the HP zx1 ioa entering I/O fatal mode under some OLR conditions. The SW interrupt is an exception to this behavior.

Destination ID/EID Specifies destination processor

Figure 11.3: I/O SAPIC Redirection Table Entry

Since PCI, PCI-X, and AGP all define the interrupt wires to be level-sensitive, low true, the majority of the HP zx1 ioa testing will be done with Polarity and Trigger both set to 1.

11.3.6 Interrupt Message Conversion

To deliver an interrupt, the I/O SAPIC will generate a 32-bit memory write cycle which will be written into the HP zx1 ioa's inbound data FIFO. Figure 11.4 on the following page shows the memory write transaction. It is the responsibility of the rope host device to convert this memory write cycle into the appropriate transaction on the processor bus. The address of the 32-bit memory write is defined by the destination fields of the corresponding Redirection Table Entry in the I/O SAPIC. The RH (Redirectable Hint) is set to 1 only if the delivery mode is set to 3'b001 in the corresponding entry.

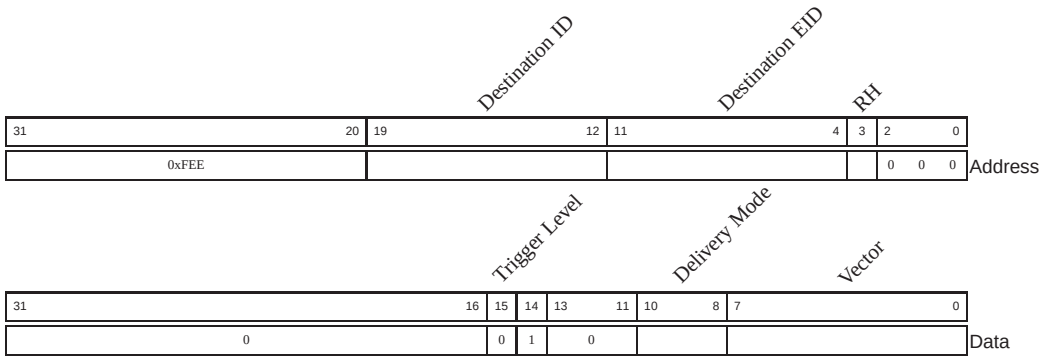


Figure 11.4: Address & data for interrupt transaction (32-bit PCI memory write)

11.4 Power-on Configuration

Several of the interrupt lines are also used to provide configuration information to the HP zx1 ioa at power-up. The interrupt lines are convenient for this purpose since they are often unused in certain I/O slot configurations. The rising edge of the PWR_ON_RST_L signal is used to latch their value. The signals are latched in the pads themselves and another block is responsible for driving the derived configuration signals.

For more information, see Section 3.2.1 on page 9.

Chapter 12

Rope Controller

12.1 Overview

The rope controller provides the interface between the HP zx1 ioa and the rope host chip.

12.2 I/O Rope

The rope connection to the host chip can be either a single rope or multiple bundled ropes.