VolantPCI Component Specification
Version 1.01

Document Number BOC-PCI-09003

August 1, 1997

Preface

This document describes the functions of the VolantPCI VLSI component. VolantPCI provides an efficient means of interfacing serial communication chips and other peripheral components to a high-speed bursting local bus. VolantPCI uses a dual-bus structure, containing eight independent DMA channels, with linked-list chaining and automatic appended I/O operations.

Contents

Figures

Tables

1.0 VolantPCI Component Overview

1.1 General

The major highlights of VolantPCI are listed below:

- Custom ASIC using LSI's 3.3v, LCB600K 0.6µ technology
- Eight-channel DMA controller
- One Sixteen-byte data buffer per channel
- PCI Local Bus interface
- One Application Interface Bus (AIBs)
- Support of Linked List Chaining
- Support of AIB I/O operations
- 32 General Purpose Input/Output Pins (PIO)
- 3.3V operation
- 208 PQFP
- up to 33 Mhz operation
- JTAG Support

VolantPCI interfaces a high-speed PCI Bus to one independent Application Interface Bus (AIB). Eight DMA channels provide support for transfers between the Local Bus and the AIB.

Each DMA channel supports Linked List Chaining, the ability for the channel to auto-initialize its registers from a predefined list in PCI address space. Each DMA also has the ability to append I/O operations within the list chain operation.

1.2 Performance

Each DMA channel is capable of supporting zero-wait-state data transfers to the PCI Bus for a maximum of 4 four-word transfers, or a peak throughput of 57 MBytes/sec at 25 Mhz, 75 Mbytes/sec at 33 Mhz (assuming AWDDDDT). VolantPCI rearbitrates for the Local Bus after each DMA channels' access.

Performance for DMA list chaining is slightly higher than for DMA data transfers. (Chaining operations burst up to 7 words.)

VolantPCI also accesses the AIB bus as a Bus Master. The AIB bus can support 4-state (160 ns at 25 Mhz and 121 ns at 33 Mhz) bus cycles to 8-bit devices, or a maximum rate of 6.25 MBytes/sec at 25 Mhz and 8.3 MBytes/sec at 33 Mhz (50 - 66 MBits/sec).

Sustained Throughput The following factors influence the amount of data throughput sustained between the PCI Bus and one of the AIBs:

- AIB device cycle time
- AIB device buffer size
- Frequency of linked list chaining

- Frequency of DMA executed AIB I/O operations (AIB OP's)
- PCI Bus utilization by other master devices
- The frequency of AIB Interrupt Acknowledge cycles

1.3 RAS Highlights

VolantPCI provides the following RAS support:

- Local Bus address and data parity generation and checking
- Gate Array ID (CCRID) register, providing a revision level for VolantPCI.
- JTAG Chip Testing Support

Note that VolantPCI does not generate or check parity for its internal structures.

1.4 VolantPCI Block Diagram

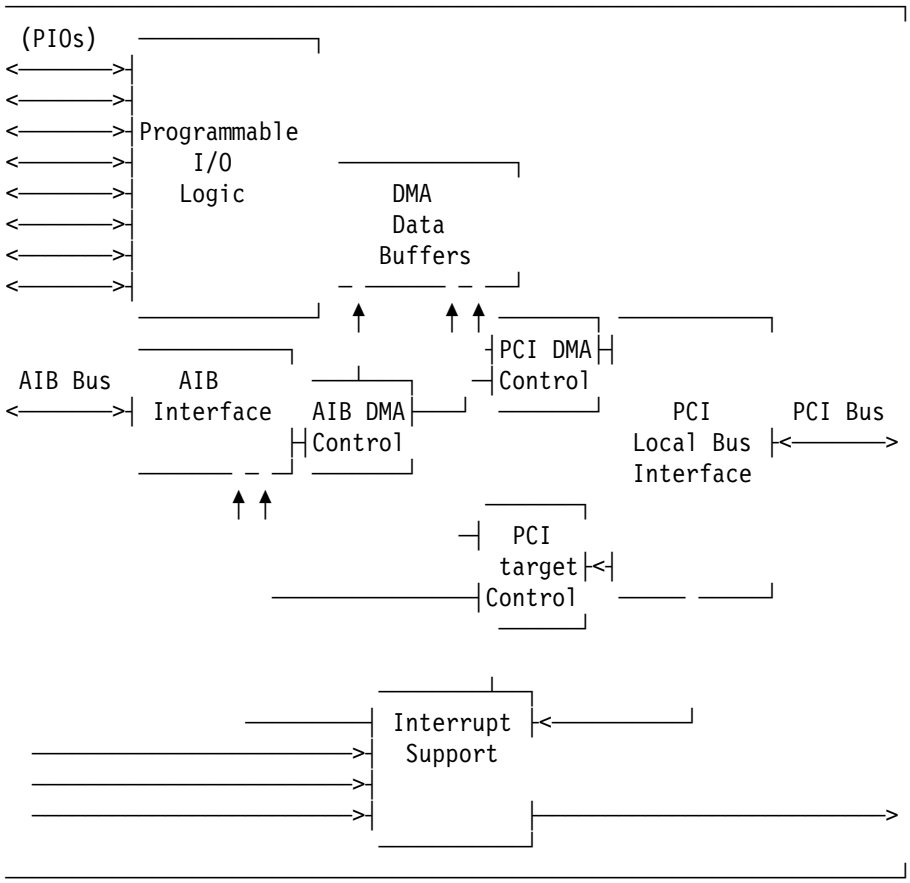


Figure 1. Major Functional Blocks in VolantPCI

1.5 VolantPCI Signal Description

The following tables provide a description of each signal in the VolantPCI module. See Appendix A, “VolantPCI Pin Name/Number Cross Reference” on page 77 for module pin number information.

1.5.1 PCI Bus Signals

Table 1 (Page 1 of 2). PCI bus specific signals

Name	Type	Description
FRAME#	**I/O PCI**	**Cycle Frame** indicates the beginning and duration of a PCI access. This signal is driven by VolantPCI as an initiator and received as a target.
IRDY#	**I/O PCI**	**Initiator Ready** indicates the initiator's ability to complete the current data phase of the transaction. This signal is driven by VolantPCI as an initiator and received by VolantPCI as a target.
TRDY#	**I/O PCI**	**Target Ready** indicates the target's ability to complete the current data phase of the transaction. This signal is driven by VolantPCI as a target and received by VolantPCI as an initiator.
STOP#	**I/O PCI**	**Stop** indicates the current target is requesting the initiator to stop the current transaction. This signal is driven by VolantPCI as a target and received by VolantPCI as an initiator.
pAD31:0	**I/O PCI**	**Address/Data Bits 31-0** are used by the PCI initiator to address memory and I/O targets. These signals are also used to select the VolantPCI chip for target operations.
CBE3:0#	**I/O PCI**	**Bus Command/Byte Enables** are driven by VolantPCI as a master and received as a target. During the address cycle they contain the bus command. During data cycles they are used as byte enables and determine which bytes of data are valid for each data cycle.
PAR	**I/O PCI**	**PCI Parity bit** provides a single bit of even parity across AD(31:0) and CBE(3:0)#. It is driven by the device supplying address or data and checked by the device receiving the address or data.
IDSEL	**I**	**Initialization Device Select** selects VolantPCI during configuration read and write transactions.
DEVSEL#	**I/O PCI**	**Device Select** is driven by VolantPCI to indicate that VolantPCI is selected as a target. This signal is received by VolantPCI as an initiator to indicate that a device has been selected.
PERR#	**I/O PCI**	**Parity Error** is used to report data parity errors during all PCI transactions (except Special Cycle). VolantPCI asserts this signal when detecting a parity error on received data, and receives this signal when driving data.
SERR#	**O PCI**	**System Error** is used to signal catastrophic errors. It is driven, when enabled, if an address parity error is detected.
CLK	**I**	**Clock** provides timing for all transactions on the PCI Bus and internal functions.
INTA#	**O PCI**	**Interrupt** is driven active when VolantPCI requires interrupt service by the system. This driver is open-drain.
RST#	**I**	**PCI Reset** when driven active immediately causes VolantPCI to tri-state all of its PCI drivers. All VolantPCI functions are reset and do not operate until this line is released.
REQ#	**O PCI**	**Request** is driven active when VolantPCI desires access to the bus as a master.

Table 1 (Page 1 of 2). PCI bus specific signals		
Name	**Type**	**Description**
GNT#	**I**	**Grant** indicates to VolantPCI that it has been granted master access to the PCI Bus.

1.5.2 VolantPCI AIB Bus Signals

Table 2. VolantPCI AIB Bus Signals

Name	Type	Description
a_A17:2	**O**	**AIB Address(17-2)** These pins function as the 16-bit address bus during an AIB bus cycle. All DMA data transfers use implicit addressing of the I/O device with the DACK signals and therefore these signals are unknown during DMA cycles. The bits are numbered from 17 to 2 to simplify conversion of PCI Bus addresses to AIB bus addresses. All access to the AIB must be byte accesses to byte-aligned addresses.
a_D7:0	**I/O**	**AIB Data(7-0)** These pins function as the 8-bit data bus during an AIB bus cycle. These pins are outputs during writes and inputs during reads.
-aWR	**O**	**-AIB Write** is driven low during a write operation to an AIB target device.
-aRD	**O**	**-AIB Read** is driven low during a read operation to an AIB target device.
-aCS3:0	**O**	**-AIB Chip Select(3-0)** are driven active when a valid address decode is detected for the address ranges defined in the CSD0-3 registers.
-aRESET	**O**	**-AIB Reset** is used to reset the AIB bus. It is driven active/inactive synchronous to the RST# input signal from the PCI Bus, and can also be driven active by writing the ACR register to reset AIB devices.
-aERROR	**I**	**-AIB Error input** is driven by AIB logic when any defined critical error occurs. This will cause vector #232 (E8h) to be generated. (see 3.2.3, “AIB ERROR Interrupt” on page 41).
aCLK	**O**	**AIB Clock** is the AIB Bus reference clock. This clock is equal in frequency to the local bus clock. See 4.2.3, “AIB Bus Configuration Register (ACR)” on page 57.
-aDREQ7:0	**I**	**-DMA Request(7-0)** are the 8 DMA request signals that AIB bus devices use to request DMA service. These signals are asynchronous inputs and should be driven inactive in response to the device receiving the corresponding DACK signal.
-aDACK7:0	**O**	**-DMA Acknowledge(7-0)** are the 8 DMA acknowledge signals that the DMA controller drives to the AIB Bus when the corresponding DREQ is being serviced. The duration of these signals are programmable within the DAPW registers. See 4.2.2, “DMA Acknowledge Pulse Width Registers (DAPW)” on page 56.
-aEOPTC	**I/O**	**-DMA End-of-Process/Terminal Count** is a synchronous signal used for termination of the DMA channels on a bus. This signal is an output "**TC**" for DMA transmit channels, and an input "**EOP**" for DMA receive channels. As an output, TC is driven active synchronous with the data DACK transfer on which a DMA terminal count condition occurs. As an input, EOP is received synchronously with the data DACK and can optionally cause a DMA channel to flush the current buffer, stop, interrupt, or chain.

1.5.3 VolantPCI Interrupt Controller Signals

Table 3. VolantPCI Interrupt Controller Signals

Name	Type	Description
-aINT3:0	**I**	**-AIB Interrupts(3-0)** are four separate interrupt input signals that are driven active by devices on the AIB bus when interrupt service is required. -A_INT(1-0) can be optionally programmed to require the interrupting device to supply an 8-bit interrupt vector when the corresponding INTACK signal is driven active. Otherwise these inputs can be used as direct interrupt inputs with the vector being automatically supplied by the VolantPCI chip. When operating -AIB_INT(0) or -AIB_INT(1) in interrupt acknowledge mode, the interrupt must be cleared at its source, AND the appropriate EOI to the interrupt controller must be issued, or interrupts below that prioritization level will be locked out (see 3.4.1, “AIB INT0/1 End-of-Interrupt (EOI0/1) commands” on page 48).
-aINTACK1:0	**O**	**-AIB Interrupt Acknowledge(1-0)** function as the interrupt acknowledge for the AIB_INT(1-0) signals when the AIB_INT(1-0) signals are programmed in vectored mode. When a device uses the -AIB_INT(1-0) signals and the VolantPCI chip is programmed to request an external vector, it must be capable of responding to the corresponding -AIB_INTACK(1-0) signal by driving an 8-bit interrupt vector on lines D(7-0) during the interrupt acknowledge cycle.

1.5.4 VolantPCI Test

Table 4. VolantPCI Clocks and Test

Name	Type	Description
JTAG-TDI	**I**	**Serial Test Data IN** is used during JTAG chip testing. This pin should be pulled up to Vdd external to the chip during normal operation.
JTAG-TDO	**O**	**Serial Test Data Out** is used during JTAG chip testing.
JTAG-TCK	**I**	**Test Clock** is used during JTAG chip testing. This pin should be pulled up to Vdd external to the chip during normal operation.
JTAG-MOD	**I**	**Test MODE** is used during JTAG testing of the chip. This pin should be pulled up to Vdd external to the chip during normal operation.
JTAG-RST#	**I**	**Test Logic Reset** is used during JTAG testing of the chip. This pin should be pulled down to Vss external to the chip during normal operation.
ScanMuxSel#	**I**	**Scan Mux Select** is used by the chip manufacturer to test the chip. This pin should be pulled up to Vdd external to the chip during normal operation.
ScanTestEn#	**I**	**Scan Test Enable** is used by the chip manufacturer to test the chip. This pin should be pulled up to Vdd external to the chip during normal operation.
PTSTOUT	**O**	**P Test Output** is used by the chip manufacturer to test the chip. This pin must be left unconnected.
IDDTST	**I**	**IDD Test** is used by the chip manufacturer to test the chip. When driven high, all chip outputs are tri-stated. This pin should be pulled down to Vss external to the chip during normal operation.

1.5.5 VolantPCI Miscellaneous Signals

Table 5. VolantPCI Miscellaneous Signals

Name	Type	Description
-LED_EN	**O**	**-LED Enable** is a signal dedicated to enabling an external light emitting diode for diagnostic purposes. (see 6.1.2, “LED Enable Register (LER)” on page 70).
REFCLK	**I**	**Reference Clock** is an input clock signal. It is used to allow software to determine the PCI CLK speed (see 6.1.3, “Clock Timer Register (CTR)” on page 71). It is also the clock used to generate the various programmable output clocks (see 6.2.1, “PIO Configuration Registers (PIOCFG)” on page 72).
PIO0(3-0)	**I/O**	**AIB Port 0 Programmable I/O(3-0)** provide 4 independent programmable I/O signals.
PIO0(7-4)	**O t.s.**	**AIB Port 0 Programmable I/O(7-4)** provide 4 independent programmable output signals.
PIO1(3-0)	**I/O**	**AIB Port 1 Programmable I/O(3-0)** provide 4 independent programmable I/O signals.
PIO1(7-4)	**O t.s.**	**AIB Port 1 Programmable I/O(7-4)** provide 4 independent programmable output signals.
PIO2(3-0)	**I/O**	**AIB Port 2 Programmable I/O(3-0)** provide 4 independent programmable I/O signals.
PIO2(7-4)	**O t.s.**	**AIB Port 2 Programmable I/O(7-4)** provide 4 independent programmable output signals.
PIO3(3-0)	**I/O**	**AIB Port 3 Programmable I/O(3-0)** provide 4 independent programmable I/O signals.
PIO3(7-4)	**O t.s.**	**AIB Port 3 Programmable I/O(7-4)** provide 4 independent programmable output signals.
MMI	**I**	**Multi-Master In** is a daisy-chain input connected to another chips MMO signal. If not using Multi-Master mode then it must be connected to its own MMO signal.
MMO	**O**	**Multi-Master Out** is a daisy-chain output connected to another chips MMI signal. If not using Multi-Master mode then it must be connected to its own MMI signal.
SED	**I/O**	**Serial Eprom Data** can be connected to the Data In (DI) and Data Out (DO) of a three-wire serial EPROM chip (e.g. Microchip 93C06). This pin has a weak internal pull-up. It should be pulled-up externally to Vdd by approx. 10kΩ.
SECS	**O**	**Serial Eprom Chip Select** can be connected to the Chip Select (CS) pin of a three-wire serial EPROM chip (e.g. Microchip 93C06).
SECLK	**O**	**Serial Eprom Clock** can be connected to the Clock (CLK) pin of a three-wire serial EPROM chip (e.g. Microchip 93C06).
Vss		All Vss pins should be connected to a ground plane.
Vdd		All Vdd pins should be connected to a 3.3V power plane.

1.6 Vero Compatibility

VolantPCI is intended to be highly software compatible with the functions performed by the Vero chip on the four/eight-port JUNO AIB cards. It is **not** intended to be a drop in replacement for Vero with a PCI Bus replacing the CFE Local Bus. Some Vero configuration registers will not be fully compatible with VolantPCI and many features have been added. The following lists which register groups have programming considerations (refer to the individual register description sections for more detail):

1.6.1 Interrupts

The PCI bus defines a single interrupt pin versus the 8-bit vector defined for Vero. When an interrupt is pending, VolantPCI will activate its pin. Software should then read the IVR (3.4.2, “Interrupt Vector Register (IVR)” on page 49) to determine the vector for the interrupt and branch to the appropriate handler. The vectors read in the IVR are the same values used for Vero's vector.

The DISR (2.3.9, “DMA Interrupt Status Registers (DISR)” on page 33) has changed to handle the additional errors due to the PCI Bus. PERR# has been mapped to Exception, parity errors remains the same and master and target abort errors have been added.

When operating in interrupt vector mode, there will no longer be a defined range for each vector or conversion of out-of-range vectors to the highest vector. VolantPCI will ignore the vector FFh (which is returned from the DUSCC when no interrupt is pending).

A bit in the IVR is equivalent to the INTA# pin and indicates an interrupt is pending in VolantPCI.

The EOI will now operate as either a write or read command. (Previously it was only a write command.) This is being done because it is possible that PCI bridge chips could post the EOI write command which would not allow a deterministic time between the EOI and the interrupt clearing to the PCI Bus. The user can now use the read EOI (which cannot be posted) to have a deterministic timing.

1.6.2 AIB Bus

- The bus now only supports 8-bit devices (vs. 8, 16 and 32)
- The bus is demultiplexed (16 address bits, 8 data bits)
- The READY and DEN signals have been deleted
- The separate EOP signals have been combined to a single pin.
- Parity support has been removed.

1.6.3 DMA

- DMA CCR Register
  - EOP no longer supported on Transmit channels (TC is supported)
  - Asynchronous DMA no longer supported. All DMA activity is synchronized to DREQ
  - Asynchronous EOP has been deleted.
  - Port size is fixed at 8-bit.
  - Added "write value read on first AIBOP masked by AIBOP2 data".
- AIB OPs on Transmit channels are no longer supported
- Chip can be configured to write DMA status at end of current CDB Status which includes byte count, OP data and termination status (TC, EOP, etc.)
- Added "flush FIFO" bit to 2.3.2, “DMA Channel Command Registers (DCCR)” on page 26.
- Fixed support of 64kB byte count.
- Added "quick start" and "do not chain if CPR=0" to allow simpler and faster chain control.
- Added 2.3.10, “Enhanced Status Pointer Register (ESP)” on page 34.

- Changed definition of 2.5.2, “DMA FIFO Residual Count registers (DFRC)” on page 39.
- DMA Queuing is no longer supported

1.6.4 Setup/Configuration

- 4.2.2, “DMA Acknowledge Pulse Width Registers (DAPW)” on page 56 (reduced to one for Tx, one for Rx) had some bits redefined or eliminated.
- Presence Detect Registers (deleted)
- 4.2.1, “Chip Select Definition registers (CSD0-3)” on page 53 changed.
- PCI Bus configuration is now required. For address compatibility, the registers and AIB address space may be programmed to the same address they occupied on ARTIC960 Adapter.
- LBCR deleted.
- 6.1.1, “Configuration Register (CFGR)” on page 69 added.

1.6.5 PIO's

A total of 32 programmable I/O pins have been added. Half of these are fully programmable for input/output, polarity and interrupt. The other half are output-only with programmable polarity (four of which can be programmed to output a divided version of REFCLK).

These are not program compatible with the FPGAs used on previous Vero-based daughter cards.

1.6.6 Multi-Master Mode

Multiple VolantPCI chips can be connected to a single PCI Bus request/grant signal pair without additional glue logic. See 5.3.2, “Multiple VolantPCI Chips on the PCI Bus” on page 60 for more details.

1.7 VolantPCI Register Address Map

The memory map of all registers addressable within the VolantPCI module is shown below. Detailed information is found in the indicated section. Each register is four byte aligned in the address space as noted.

Reserved bit locations in a register must always be written as '0' when programming a register. These bits are undefined. Users should not rely on reserve bits reading back as '0'.

1.7.1 Register Reset States

At the end of each register description section, the register's RESET conditions are shown. There are several different reset conditions. Some registers may have only one reset condition while others may have two or three. The following table describes the names and the function of each reset condition.

Table 6. VolantPCI Reset Conditions

Reset Condition	Description
CHIP RESET	CHIP RESET will reset the entire chip. This reset is active when the RST# signal is active. This condition overrides all other pending conditions within the chip and automatically forces all logic to its RESET state.
DMA RESET	A DMA RESET resets DMA channel related registers and logic. This reset condition is activated by either a write to the GDCR (see 2.5.1, “AIB Bus Global DMA Command Register (GDCR)” on page 37) or the DCCR (see 2.3.2, “DMA Channel Command Registers (DCCR)” on page 26).
AIB RESET	There is a separate AIB RESET for the AIB Bus. This reset affects AIB Bus related registers and associated logic. The AIB RESET is activated by writing to the ACR (see 4.2.3, “AIB Bus Configuration Register (ACR)” on page 57). **Note:** Currently there are no AIB bus related registers that are affected by this reset. Only the AIB RESET pin is affected.

A 'U' in the "Reset Conditions" section means undefined, and an 'S' means the value stays the same as before the reset command.

1.7.2 DMA Registers

Note: An 'n' in an address represents the channel number (0-7).

Table 7. VolantPCI DMA Register Set

Section, Name	Address Offset	Access Type
2.3.6, AIB Address 1/2 Register (AIB_ADDR 1/2)	+n000h	r/w
2.3.7, AIB_OP1 DATA Register (AIB_OP1)	+n004h	r/w
2.3.8, AIB_OP2 DATA Register (AIB_OP2)	+n008h	r/w
2.3.3, Memory Pointer Register (MPR)	+n00Ch	r/w
2.3.4, Transfer Count Register (TCR)	+n010h	r/w
2.3.5, Chain Pointer Register (CPR)	+n014h	r/w
2.3.1, Channel Control Register (CCR)	+n018h	r/w
2.3.2, DMA Channel Command Registers (DCCR)	+n020h	r/w
2.3.9, DMA Interrupt Status Registers (DISR)	+n028h	ro
2.5.2, DMA FIFO Residual Count registers (DFRC)	+n02Ch	ro
2.3.10, Enhanced Status Pointer Register (ESP)	+n038h	r/w
2.5.3, DMA Buffer Data registers (DBD)	+n100h- +n10Ch	r/w
2.5.1, AIB Bus Global DMA Command Register (GDCR)	+8000h	r/w

1.7.3 PCI Bus Related Registers

Table 8. PCI Bus Registers

Section, Name	Config Offset	Access Type
5.4.1, Device/Vendor ID (DEVID)	+00h	ro
5.4.2, Host Status/Command Register (HSCR)	+04h	r/w
5.4.3, Class Code/Revision ID Register (CCRID)	+08h	ro
5.4.4, Host Miscellaneous Functions Register (HMFR)	+0Ch	r/w
5.4.5, Register Base Address Register (REGBAR)	+10h	r/w
5.4.6, AIB Base Address Register (AIBBAR)	+14h	r/w
5.4.7, Subsystem ID (SSID)	+2Ch	ro
5.4.8, Latency/Grant/Interrupt Register (LGIR)	+3Ch	r/w

1.7.4 Interrupt Related Registers

Table 9. VolantPCI Interrupt Related Registers

Section, Name	Address Offset	Access Type
3.3.1, Interrupt Initialization register (IIR)	+8010h	r/w
3.3.2, Interrupt Mask register (IMR)	+8014h	r/w
3.3.3, Interrupt Status register (ISR)	+8018h	r/w
3.3.4, AIB Error Interrupt Mask register (EIMR)	+801Ch	r/w
3.4.1, AIB INT0/1 End-of-Interrupt (EOI0/1) commands	+9000h +A000h	r/w
3.4.2, Interrupt Vector Register (IVR)	+F000h	r/w

1.7.5 AIB Bus Related Registers

Table 10. VolantPCI AIB Bus Registers

Section, Name	Address Offset	Access Type
4.2.1, Chip Select Definition registers (CSD0-3)	+B000h- +B00Ch	r/w
4.2.2, DMA Acknowledge Pulse Width Registers (DAPW)	+B030h, +B038h	r/w
4.2.3, AIB Bus Configuration Register (ACR)	+C000h	r/w

1.7.6 Miscellaneous Registers

Table 11. VolantPCI Miscellaneous Registers

Section, Name	Address Offset	Access Type
6.1.2, LED Enable Register (LER)	+D004h	r/w
6.1.1, Configuration Register (CFGR)	+D008h	r/w
6.1.3, Clock Timer Register (CTR)	+D00Ch	ro
6.2.1, PIO Configuration Registers (PIOCFG)	+D100h- +D137h	r/w
6.2.2, PIO Status Registers (PIOSTAT)	+D200h- +D237h	r/w
6.2.3, Serial EPROM Register (SER)	+F100h	r/w

2.0 VolantPCI DMA Controller

2.1 General

The functions of the DMA portion of the VolantPCI module are highlighted below.

- 8 independent DMA channels (any channel may operate as transmit or receive)
- split bus implementation (8-bit AIB Bus to 32-bit Local Bus)
- support of 16-bit Address, 8-bit Data AIB Bus devices
- 16 byte fifo per channel
- 16 byte burst capability on Local Bus for Data
- 3 Write / 7 Read List Chain Operation on Local Bus for Chaining
- 32-bit 4GB addressability on Local Bus
- 16-bit 64KB addressability on AIB Bus for AIB OP's
- separate DREQ and DACK signals for each DMA channel
- programmable DACK cycle time for transmit and receive DACK cycles
- 64KB byte count capability
- Descriptor Table for each channel
- Linked list chaining of buffers on all channels
- Chaining support for end-of-process and terminal count condition
- Automatic storage of residual transfer count and termination status on chain event
- Automatic storage of AIB OP reads on chain event
- Up to two programmable auto I/O operations on chain event (Receive DMA channels only)
- Six interrupt sources for each channel

Each channel is controlled by a Channel Descriptor Table (CDT) register set. The program normally writes a block of the CDT register values to memory resident structures called Channel Descriptor Blocks (CDB's). The program then writes the starting address of a CDB to the Chain Pointer Register (CPR) with bit 0 set to '1'. At this point, the CDB is automatically fetched from memory and loaded into the CDT. With the linked list chain option, once the DMA channel is started, the CDBs can be automatically fetched by the DMA channel's state machine. Alternatively, the CDT registers can be programmed directly.

Once enabled, a channel will service DMA requests from the AIB Bus until one of a number of programmable conditions is reached. If the DMA channel is programmed to stop on one of these conditions, the channel can be re-enabled with a write to the Channel Control Register (CCR). Any condition that can stop the channel can also interrupt the PCI Bus. Also, the channel has the ability to interrupt without stopping the channel. All of the options are programmable in the CCR and are described later.

Devices on the AIB bus assert two types of requests: Transmit (TR) and Receive (RR).

Once a Transmit channel is started, the DMA controller arbitrates for control of the Local Bus, and when granted control, bursts up to 16 bytes of data from memory into that channel's FIFO. The data is then transferred to the requesting device across the AIB Bus until the FIFO is empty. This allows data transfer to the AIB device to occur in the background of Local Bus activity.

For a receive DMA channel, the DMA controller will receive up to 16 bytes of data across the AIB Bus. It then arbitrates for control of the Local Bus. Once granted, the DMA controller bursts all data into memory. This, again, allows data transfer from the AIB device to occur in the background of Local Bus activity.

2.2 DMA FIFO

As mentioned above, data transfer from Local Bus to AIB Bus is not direct, but rather, is buffered in a set of fifos. There is a 16 byte deep fifo associated with each DMA channel that acts as an intermediate storage area for data. The fifos support high speed access so that all accesses to the Local Bus will move data into and out of the fifos instead of directly to the AIB Bus, which might support only slower devices. This allows all Local Bus DMA data transfer initiated by VolantPCI to be high speed bursted accesses.

The fifos are configured such that AIB Bus accesses and Local Bus accesses can occur simultaneously for different channels. A DMA channel can be performing a read or write on the AIB Bus, while at the same time, another channel is performing a read or write on the Local Bus.

2.3 Channel Descriptor Register Set

Table 12 on page 22 shows the organization of the channel descriptor table (CDT) registers for one of the DMA channels. This table is duplicated for each of the 8 DMA channels. Each register is fully addressable on the PCI Bus.

Table 12. VolantPCI DMA Channel Descriptor Table Registers

Register Name	**Register Function**	**Valid Bits**
CCR	Channel Control Register	31-0
CPR	Chain Pointer Register	31-2
TCR	Transfer Count Register	15-0
MPR	Memory Pointer Register	31-0
AIB_OP2	AIB OP2 Data Register	7-0
AIB_OP1	AIB OP1 Data Register	7-0
AIB_ADDR2	AIB OP2 Address Register	31-16
AIB_ADDR1	AIB OP1 Address Register	15-0
ESP	Enhanced Status Pointer Register	31-2

Note: The ESP is not part of a CDB and normally is never written by software. The exceptions would be: 1) when running in Enhanced Status Mode List Chaining and the first CDB is programmed directly, 2) diagnostics and other test code.

2.3.1 Channel Control Register (CCR)

2.3.1.1 Description:

The CCR register controls the operational personality for a DMA channel.

2.3.1.2 Register Addressing

DMA Channel	Address Offset
0	+00018h (read/write)
1	+01018h (read/write)
2	+02018h (read/write)
3	+03018h (read/write)
4	+04018h (read/write)
5	+05018h (read/write)
6	+06018h (read/write)
7	+07018h (read/write)

2.3.1.3 Register Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
						Reserve		Reserve		AIB write control		AIB operation control			
GEN STA	Reserved					x	x	x	x	WR 1	WR 0	OP #2	OP #1	# of I/O OPs	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
channel stopping options			channel interrupt options			channel chaining options				channel definition options					
STP 2	STP 1	STP 0	INT 2	INT 1	INT 0	LCH 2	LCH 1	LCH 0	Rsvd.	EOP DIR	Rsvd.		+T/ -R	EN/ DIS	

2.3.1.4 Bit Descriptions

- Bit 31. General purpose status bit. This bit can be used for any application dependent purpose. It has no function internal to the chip. It is updated by I/O writes, and during list chaining fetches.
- Bits 30-22. Reserved. Always write these bits to '0'.
- Bits 21-20. These two bits are encoded to indicate what data will be written to the AIB Bus on the second of two AIB OP's, if the first of the two AIB OP's is a read.

```
WR1    WR0
---    ---
 0      0     -- Write the value of AIB_OP2 Data Register
 0      1     -- Write the value that was read on first OP
 1      0     -- Write the complement of value read on first OP
 1      1     -- Write the value that was read on first OP masked
                 by the value of AIB_OP2 Data Register
```

- Bit 19. AIB OP #2, read or write. This bit is coupled with bits 16 and 17. It defines the second I/O operation to the AIB Bus after a chain event to be a read or write. If the operation is a write, the data written is defined by bits 20 and 21 of the CCR. If the operation is a read, the data will be stored in memory.
  - 0 = Read, 1 = Write
- Bit 18. AIB OP #1, read or write. This bit is coupled with bits 16 and 17. It defines the first I/O operation to the AIB Bus after a chain event to be a read or write. If the operation is a write, the data written is defined by bits of the AIB_OP1 register. If the operation is a read, the data will be stored in memory.
  - 0 = Read, 1 = Write
- Bits 17-16. Number of I/O operations to AIB Bus upon chain event. These two bits are encoded to provide the DMA controller with the ability to execute up to 2 discrete I/O operations to the AIB Bus following the recognition of a chain event. The address to which the first I/O operation occurs is defined by bits 0-15 in the AIB_ADDR 1/2 register. The address to which the second I/O operation occurs is define by bits 16-31 in the AIB_ADDR 1/2 register.

  Note: AIB OPs are supported only on Receive DMA channels. Write these as '00' for transmit channels.

  ```
  00 = No I/O operation upon chain event.
  01 = One I/O operation upon chain event.
  10 = Two I/O operations upon chain event.
  11 = Reserved.
  ```

- Bit 15-13. Encoded DMA channel stopping options. Bit 15 is always '0'.

STP1	STP1	STP0	Stopping Option
0	0	0	Do not stop
0	0	1	TC=0
0	1	0	EOP (Receive Channels Only)
0	1	1	TC=0 "or" EOP
1	X	X	Reserved

  When a channel is stopped, the CCR enable/disable bit is reset. Also, a chaining condition takes precedence over a stopping condition if both occur at the same time since the chaining condition causes a new CCR enable/disable bit to be fetched from memory.

  Note: When the 'Do not stop' option is selected and the channel is programmed as a transmit channel, the TCR must be initialized with a value of multiple of 16 bytes.

- Bit 12-10. Encoded DMA interrupt options. Bit 12 is always '0'.

  Error interrupts are not separately maskable in the CCR. To eliminate error related DMA interrupts, detection of the error inputs (PERR and Parity) should be disabled in the appropriate registers.

INT2	INT1	INT0	Interrupt Option
0	0	0	Normal Termination Interrupts disabled
0	0	1	TC=0
0	1	0	EOP (Receive Channels Only)
0	1	1	TC=0 "or" EOP (Receive Channels Only)
1	X	X	Reserved

- Bit 9-7. Encoded list chaining enabling options.

```
LCH2 LCH1 LCH0   Chaining Option
---- ---- ----   ---------------
 0    0    0     Disabled
 0    0    1     TC=0
 0    1    0     EOP  (Receive Channels Only)
 0    1    1     TC=0 "or" EOP (Receive Channels Only)
 1    0    0     Reserved
 1    0    1     Reserved
 1    1    0     Reserved
 1    1    1     NOP
```

The chaining NOP selection only functions when it is chained in. AIB OPS will not occur for a NOP.

VolantPCI will not chain if the CPR register for the channel points to 00000000h.

- Bit 6-5. Reserve. Always write this to 0.

  Note: These bits were used in the Vero chip. Bits '6-5' set the Port Size for the DMA transfer. VolantPCI has a fixed 8-bit port size for all DMA channels.

- Bit 4. This bit, when '0', enables the EOP(TC) as an output pin. This is the normal mode for Tx channels. When set to '1', the pin is an input. This mode should be used for Tx channels when the user does not want a TC indication to the external device. This mode ('1') **must** be used for all Rx channels (EOP detection can be disabled using other CCR bits).

- Bit 3-2. Reserve. Always write this to 0.

  Note: These bits were used in the Vero chip. Bit '3' enabled the Asynchronous EOP option. Bit '2' enabled the Asynchronous DMA option. These Vero options are not supported in the VolantPCI chip.

- Bit 1. +Transmit/-Receive indicator. This bit indicates if the DMA channel is a Transmit Channel "1", or a Receive Channel "0". For receive channels, data transfer is **from AIB Bus** to Local Bus. For transmit channels, data transfer is from Local Bus to AIB Bus.

  Previous Vero based ARTIC cards allocated the channels as follows:

  ```
  TRANSMIT CHANNELS:  2,3,6,7
  RECEIVE  CHANNELS:  0,1,4,5
  ```

- Bit 0. +Enable/-disable channel.

  – 0 = DMA Channel Disabled
  – 1 = DMA Channel Enabled

  This bit reflects the status of the DMA channel. When enabled, the DMA channel can be stopped by writing this bit to a '0', however, this operation is dangerous because the other CCR bits will also be written. The DMA channel should be stopped by writing to the DCCR (see 2.3.2, “DMA Channel Command Registers (DCCR)” on page 26).

  This bit is updated during list chaining.

  Note: This bit will not read back '0' unitl the DMA channel has actually stopped.

  The dummy CDB along with bit 31 of the CCR used to determine the DMA channel's status (running, chaining, etc.) for Vero is no longer required to ensure that the channel is stopped.

2.3.1.5 Reset Conditions

```
CHIP RESET:     U000 0000 00UU UUUU 0000 0000 0001 00U0
DMA RESET:      S000 0000 00SS SSSS 0SS0 SSSS S00S 00S0
```

2.3.2 DMA Channel Command Registers (DCCR)

2.3.2.1 Description:

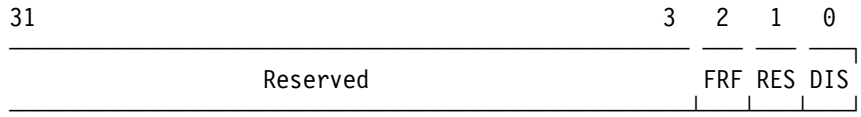
The DCCR supports three commands. One allows for the bit manipulation of the CCR enable/disable bit. This bit allows CCR bit 0 to be set/reset without affecting the other CCR bits. A second command allows internal logic to be reset to a known state if an error condition causes a channel to stop before completion. Also, a third command is available for receive DMA channels only. This command forces the receive fifos to be flushed to local bus memory. A separate command register exists to control each DMA channel.

The command to enable a channel should not be issued with the same write that releases the channel from a reset command.

2.3.2.2 Register Addressing

DMA Channel	Address Offset	
0	+00020h	(read/write)
1	+01020h	(read/write)
2	+02020h	(read/write)
3	+03020h	(read/write)
4	+04020h	(read/write)
5	+05020h	(read/write)
6	+06020h	(read/write)
7	+07020h	(read/write)

2.3.2.3 Register Format



2.3.2.4 Bit Descriptions

- Bit 31-3. Reserved. Always write these bits to '0'.
- Bit 2. Flush Receive Fifo. (Receive Channels Only)

  Setting this bit to a '1' forces the residual bytes held in the receive DMA channel's fifo to be flushed to local bus memory. When the bytes are flushed, the FRF bit will reset back to '0'. The FRF command is only valid for receive channels when the channel is stopped. Writes to this bit are blocked when the DIS bit = '1' (channel is running). Users should stop the channel by writing to the DIS bit and verify that the channel is stopped (DIS = '0') before flushing the fifo.
- Bit 1. Reset channel command. Setting this bit to '1' causes a channel to be reset as shown by the Reset Command for each register. The channel is held in the reset state until the bit is set back to '0'.
  - 1 = Reset Active
  - 0 = Reset Inactive

- Bit 0. Disable channel command. Bit 0 = '0' defines the command to disable the channel. Bit 0 = '1' defines the command to enable the channel. Bit 0 of the CCR, if read, will reflect an enable or disable command if it is issued through the DCCR.
  - 1 = DMA Channel Enable
  - 0 = DMA Channel Disable

  Note: When stopping the channel, a write to this bit causes the DMA channel to complete any pending operations and then stop. This bit should be read to verify that the channel is stopped since there may be a delay between the writing of this bit to stop the channel, and when the DMA channel actually stops.

2.3.2.5 Reset Conditions

```
CHIP RESET:     0000 0000 0000 0000 0000 0000 0000 0000
DMA RESET:      0000 0000 0000 0000 0000 0000 0000 00S0
```

2.3.3 Memory Pointer Register (MPR)

2.3.3.1 Description:

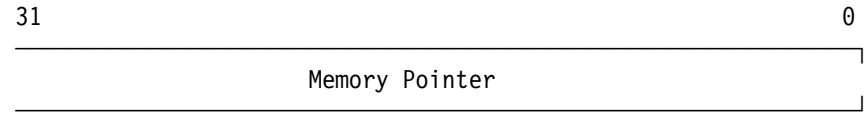
The MPR contains the 32-bit Local Bus address of the next data byte to be DMA'ed. There are no alignment restrictions on the address programmed into this register.

Note: This register **must** be written before the TCR is written when directly programming a DMA transfer.

2.3.3.2 Register Addressing

DMA Channel	Address Offset
0	+0000Ch (read/write)
1	+0100Ch (read/write)
2	+0200Ch (read/write)
3	+0300Ch (read/write)
4	+0400Ch (read/write)
5	+0500Ch (read/write)
6	+0600Ch (read/write)
7	+0700Ch (read/write)

2.3.3.3 Register Format



2.3.3.4 Bit Descriptions

- Bits 31-0. PCI Bus Address.

2.3.3.5 Reset Conditions

```
CHIP RESET:     UUUU UUUU UUUU UUUU UUUU UUUU UUUU UUUU
DMA RESET:      SSSS SSSS SSSS SSSS SSSS SSSS SSSS SSSS
```

2.3.4 Transfer Count Register (TCR)

2.3.4.1 Description:

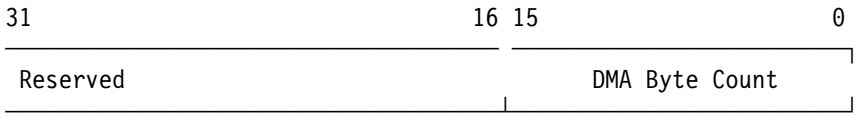
Bits 0-15 contain the current DMA transfer count in bytes. Values from 0000h to FFFFh can be programmed. This allows DMA transfers of from '1' (0001h) to '64KB' (0000h) to be transferred. This register is automatically saved into memory when a chain event occurs. This is shown in Figure 2 on page 35.

The Terminal Count (TC) condition occurs when the Transfer Count decrements to zero.

2.3.4.2 Register Addressing

DMA Channel	Address Offset
0	+00010h (read/write)
1	+01010h (read/write)
2	+02010h (read/write)
3	+03010h (read/write)
4	+04010h (read/write)
5	+05010h (read/write)
6	+06010h (read/write)
7	+07010h (read/write)

2.3.4.3 Register Format



2.3.4.4 Bit Descriptions

- Bits 31-16. Reserved. Always write these bits to '0'.

  Note: Vero compatible code may write these bit to a non-zero value. All new code should write these bits as '0'.

- Bits 15-0. Transfer count in bytes (1 to 64K).

  Note: A zero value equals a count of 64k. For future compatibility it is suggested that the value of 00010000h be used for 64k transfers.

2.3.4.5 Reset Conditions

```
CHIP RESET:     0000 0000 0000 0000 UUUU UUUU UUUU UUUU
DMA RESET:      0000 0000 0000 0000 SSSS SSSS SSSS SSSS
```

2.3.5 Chain Pointer Register (CPR)

2.3.5.1 Description:

The CPR contains the 32-bit address pointer that points to the memory location where the DMA controller will fetch the CDB for the next buffer when a chain event occurs. The lower 2 bits of this register will always read '00' forcing the chain pointer to be 4 byte aligned. The following two conditions define a chaining event.

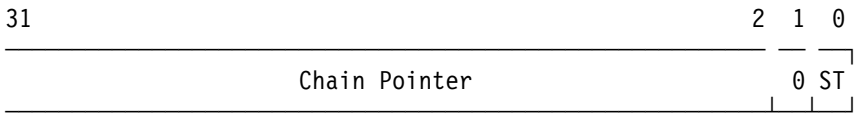
1. terminal count has been reached and list chaining for terminal count is enabled in the CCR,
2. an End-Of-Process condition has occurred and list chaining for end-of-process is enabled in the CCR.

When writing this register, if a '1' is written to bit 0 and the channel is stopped, the channel will start and immediately chain from the address written. This allows the DMA to be started with a single write. If the channel is running, the CPR will be replaced. **The CPR must only be written if the channel is stopped or this register currently contains 00000000h. Also, software should never start a chain to address 0 by writing 00000001h to this register.** See 2.4.2, “Linked List Chaining/Stopping” on page 36.

2.3.5.2 Register Addressing

DMA Channel	Address Offset
0	+00014h (read/write)
1	+01014h (read/write)
2	+02014h (read/write)
3	+03014h (read/write)
4	+04014h (read/write)
5	+05014h (read/write)
6	+06014h (read/write)
7	+07014h (read/write)

2.3.5.3 Register Format



2.3.5.4 Bit Descriptions

- Bits 31-2. Chain Pointer Address
- Bit 1. Reserved. This bit always reads '0'.
- Bit 0. Chain and start. If when writing the CPR this bit is written as a '1' and the channel is stopped, the channel will immediately chain to the address written. If the channel is running this bit is ignored. This bit always reads '0'.

2.3.5.5 Reset Conditions

```
CHIP RESET:     UUUU UUUU UUUU UUUU UUUU UUUU UUUU UU00
DMA RESET:      SSSS SSSS SSSS SSSS SSSS SSSS SSSS SS00
```

2.3.6 AIB Address 1/2 Register (AIB_ADDR 1/2)

2.3.6.1 Description:

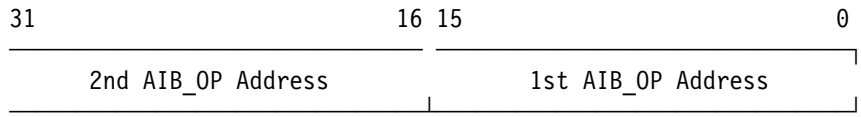
The AIB_ADDR 1/2 register contains two separate 16-bit address fields. Bits 0-15 defines the first AIB address of two possible I/O operations that the hardware will perform when a chain event occurs on receive DMA channels. Bits 16-31 defines the second AIB address of two possible I/O operations that the hardware will perform when a chain event occurs. The first operation (read or write) is directed at the address defined by bits 0-15 of this register. For a write operation, the data is defined by the value in the AIB_OP1 DATA register. For read operations, the data read is stored in memory. See 2.4.1, “Modes of List Chaining” on page 35 for more information. The second operation (read or write) is directed at the address defined by bits 16-31 of this register. This operation is the same as the first except that the data written to the AIB is that contained in the AIB_OP2 DATA register or as defined in CCR bits 20-21.

Note: VolantPCI only supports AIB OPs on receive channels. The Vero chip supported AIB OPs on receive and transmit channels.

2.3.6.2 Register Addressing

DMA Channel	Address Offset
0	+00000h (read/write)
1	+01000h (read/write)
2	+02000h (read/write)
3	+03000h (read/write)
4	+04000h (read/write)
5	+05000h (read/write)
6	+06000h (read/write)
7	+07000h (read/write)

2.3.6.3 Register Format



2.3.6.4 Bit Descriptions

- Bits 31-16. AIB OP Address for 2nd AIB Chain Operation
- Bits 15-0. AIB OP Address for 1st AIB Chain Operation

2.3.6.5 Reset Conditions

```
CHIP RESET:     UUUU UUUU UUUU UUUU UUUU UUUU UUUU UUUU
DMA RESET:      SSSS SSSS SSSS SSSS SSSS SSSS SSSS SSSS
```

2.3.7 AIB_OP1 DATA Register (AIB_OP1)

2.3.7.1 Description:

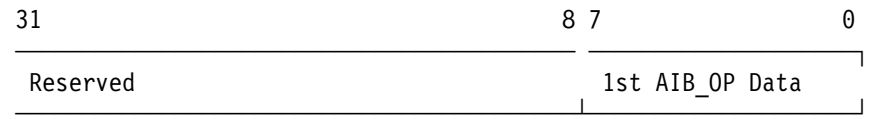
The AIB_OP1 register contains the data field of the first of two operations upon a chain event for receive DMA channels. When a chain event occurs, the DMA controller can be programmed to perform up to two I/O operations to any AIB Bus address in the 64K of the AIB address space. The first operation (read or write) is directed at the address defined by bits 0-15 of the AIB_ADDR 1/2 register. For a write operation, the data is defined by this register. For reads, the data is placed in this register.

Note: VolantPCI only supports AIB OPs on receive channels. The Vero chip supported AIB OPs on receive and transmit channels.

2.3.7.2 Register Addressing

DMA Channel	Address Offset
0	+00004h (read/write)
1	+01004h (read/write)
2	+02004h (read/write)
3	+03004h (read/write)
4	+04004h (read/write)
5	+05004h (read/write)
6	+06004h (read/write)
7	+07004h (read/write)

2.3.7.3 Register Format



2.3.7.4 Bit Descriptions

- Bits 31-8. Reserved. Always write these bits to 0.
- Bits 7-0. Data for 1st. AIB Chain Operation

2.3.7.5 Reset Conditions

```
CHIP RESET:     0000 0000 0000 0000 0000 0000 UUUU UUUU
DMA RESET:      0000 0000 0000 0000 0000 0000 SSSS SSSS
```

2.3.8 AIB_OP2 DATA Register (AIB_OP2)

2.3.8.1 Description:

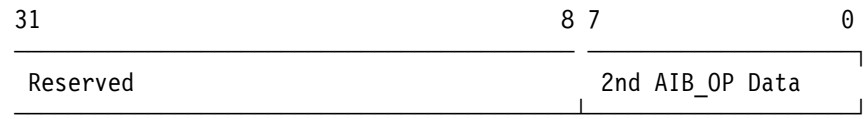
The AIB_OP2 register contains the data field of the second of two operations upon a chain event of a receive DMA channel. When a chain event occurs, the DMA controller can be programmed to perform up to two I/O operations to any AIB Bus address in the 64K of the AIB address space. The second operation (read or write) is directed at the address defined by bits 16-31 of the AIB_ADDR 1/2 register. For a write operation, the data is defined by this register or may be defined by bits 20-21 of the CCR. For reads, the data is placed in this register.

Note: VolantPCI only supports AIB OPs on receive channels. The Vero chip supported AIB OPs on receive and transmit channels.

2.3.8.2 Register Addressing

DMA Channel	Address Offset
0	+00008h (read/write)
1	+01008h (read/write)
2	+02008h (read/write)
3	+03008h (read/write)
4	+04008h (read/write)
5	+05008h (read/write)
6	+06008h (read/write)
7	+07008h (read/write)

2.3.8.3 Register Format



2.3.8.4 Bit Descriptions

- Bits 31-8. Reserved. Always write these bits to 0.
- Bits 7-0. Data for 2nd. AIB Chain Operation

2.3.8.5 Reset Conditions

```
CHIP RESET:     0000 0000 0000 0000 0000 0000 UUUU UUUU
DMA RESET:      0000 0000 0000 0000 0000 0000 SSSS SSSS
```

2.3.9 DMA Interrupt Status Registers (DISR)

2.3.9.1 Description:

Each DMA channel has a DMA Interrupt Status register (DISR) associated with it. This status register is cleared when read. The EOP and TC status will also clear during a list chain operation if the previous CDB did not have interrupts enabled (the EOP/TC status can be checked in the CDB). If interrupts are enabled then status bits will accumulate until cleared by a read.

Since many different conditions can cause the interrupt, this means that it is possible for multiple conditions to be present when the status register is read. However, since the register is cleared by the read, only one interrupt will be presented. Therefore, the interrupt service routine must be capable of servicing all of the pending status bits.

2.3.9.2 Register Addressing

DMA Channel	Address Offset
0	+00028h (read only)
1	+01028h (read only)
2	+02028h (read only)
3	+03028h (read only)
4	+04028h (read only)
5	+05028h (read only)
6	+06028h (read only)
7	+07028h (read only)

2.3.9.3 Register Format

31 8	7	6	5	4	3	2	1	0
Reserved	TAB	MAB	LPR	LEX	Reserve		EOP	TC0

2.3.9.4 Bit Descriptions

- Bit 31 - 8. Reserved.
- Bit 7. Target Abort. This bit, when set, indicates a Target Abort has been detected while the channel was a master on the PCI Bus.
- Bit 6. Master Abort. This bit, when set, indicates a Master Abort occurred while the channel was a master on the PCI Bus.
- Bit 5. LPR. This bit, when set, indicates a Local Bus parity error has been detected when the DMA channel was reading data from the Local Bus. In this case, the channel is automatically stopped.
- Bit 4. LEX. This bit, when set, indicates a PERR# exception occurred. This happens when a Local Bus target drives the PERR# signal while VolantPCI is the Local Bus master. In this case, the VolantPCI DMA channel is automatically stopped.
- Bits 3-2. Reserved.
- Bit 1. EOP. This bit, when set, indicates the EOP signal was received during a receive DMA on this channel. This bit is set upon detecting an EOP regardless of the state of its corresponding interrupt enable bit.
- Bit 0. TC0. This bit, when set, indicates a terminal count (byte count changed to zero, see 2.3.4, “Transfer Count Register (TCR)” on page 28) condition has been reached for the DMA channel. This bit is set upon detecting a TC regardless of the state of its corresponding interrupt enable bit.

2.3.9.5 Reset Conditions

```
CHIP RESET:     0000 0000 0000 0000 0000 0000 0000 0000
DMA RESET:      0000 0000 0000 0000 0000 0000 0000 0000
```

2.3.10 Enhanced Status Pointer Register (ESP)

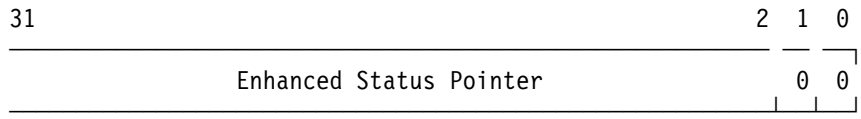
2.3.10.1 Description:

The ESP register contains the address where status will be written for a DMA channel if the chip is operating with Enhanced Status reporting enabled. This address is determined by VolantPCI during a chain event. After a CDB is read into the chip, the the value of the old CPR +1Ch is saved in this register. The ESP must always be 4-byte aligned (ie. bits 0,1 = 00).

2.3.10.2 Register Addressing

DMA Channel	Address Offset
0	+00038h (read/write)
1	+01038h (read/write)
2	+02038h (read/write)
3	+03038h (read/write)
4	+04038h (read/write)
5	+05038h (read/write)
6	+06038h (read/write)
7	+07038h (read/write)

2.3.10.3 Register Format



2.3.10.4 Bit Descriptions

- Bits 31-2. Enhanced Status Pointer
- Bits 1-0. Reserved. These bits always read '0'.

2.3.10.5 Reset Conditions

```
CHIP RESET:     UUUU UUUU UUUU UUUU UUUU UUUU UUUU UU00
DMA RESET:      SSSS SSSS SSSS SSSS SSSS SSSS SSSS SS00
```

2.4 Linked List Chaining (LLC)

The VolantPCI DMA controller supports a feature known as Linked List Chaining (LLC), sometimes called scatter/gather DMA. This feature allows the programmer to create, in memory, lists of different data buffer areas and buffer counts (CDB's) that can be automatically reloaded to the CDT after a terminal count or end-of-process condition for the present data buffer has occurred. The channel can optionally interrupt or not interrupt a local processor when any of these conditions has occurred. Also, the DMA channel can independently be stopped or not stopped at this point. In addition certain control information for the channel can be changed during the chaining operation.

2.4.1 Modes of List Chaining

VolantPCI supports two different list chaining modes configured by the ESM (Enhanced Status Mode) bit in the CFGR (see 6.1.1, “Configuration Register (CFGR)” on page 69) These two modes are: ***Compatibility Mode List Chaining*** and ***Enhanced Status Mode List Chaining***.

For both modes the hardware will always perform the three memory writes regardless of whether the DMA channel is transmit or receive, and regardless of the AIB OP settings in the CCR. If the option to do reads from the AIB is not chosen for a receive DMA channel, the two AIB_OP data words written will be indeterminate.

2.4.1.1 Compatibility Mode List Chaining

Within the CDT, a 32-bit address pointer (CPR) is maintained. This points to a CDB in memory. At the time of terminal count for a transmit channel or end-of-process detection for a receive channel, if linked list chaining is enabled, the hardware will do three memory writes followed by seven memory reads starting at this chain pointer address. The three writes store into memory: the present DMA transfer byte count and status in the first word, the AIB_OP1 data for the second word, and the AIB_OP2 data for the third word. (Note that for transmit channels, the AIB_OP data is always the same as the data last loaded into these registers either via list chain or direct register write.) The hardware then fetches seven new words of CDB information as detailed below. At the end of this operation, the CPR in the CDT will be the CPR value that was just read from the CDB in memory. This address is the starting address of the next CDB in the chain. This provides a mechanism to off-load the processor from processing DMA terminal count or end-of-process interrupts. In theory, this list of buffers could occupy any amount of free memory. This concept is detailed in Figure 2 on page 35.

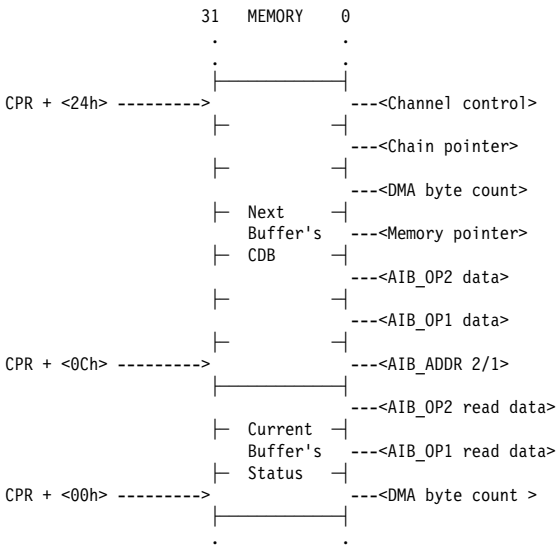


Figure 2. Linked List Chaining: Vero Compatibility Mode

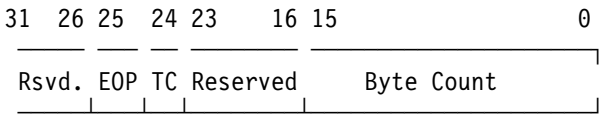
2.4.1.2 Enhanced Status Mode List Chaining

When VolantPCI is set for Enhanced Status Mode List Chaining, list chaining operates as described above in the Vero compatibility mode except that the 3 writes (byte count, termination status, and AIB OPs for receive channels and, byte count and termination status for transmit channels) that occur prior to the CDB read will be written into the CDB block area that is associated with the actual CDT rather than the next link's CDB block. This concept is detailed in Figure 3 on page 37. Note also, that the DMA channel termination status is now stored along with the byte count and that the status is stored at the memory addresses that follow the CDB as opposed to preceding it.

In Enhanced Status Mode List Chaining, status will be written only when the list chain condition is met. In other words if an error occurs or a 'stop condition' is reached before a 'list chain condition' the status for that CDB will not be written to memory. If the user would like status for the last CDB in a chain, they should use CPR=00000000h to stop the channel instead of CCR(0)='0', using a 'stop' condition or not using a 'chain' condition.

The advantages of using Enhanced Status Mode List Chaining are:

AIB OP Data and Byte Count are stored with associated CDB
A Dummy CDB is no longer required to store the last CDB's status
Termination Status (EOP and/or TC indication) is written along with the byte count in the format shown below (Status is also stored with the Byte Count in Compatibility Mode, in Vero these bits were reserved.)



2.4.2 Linked List Chaining/Stopping

The CCR provides many different options for linked list chaining and stopping DMA channels (see 2.3.1, “Channel Control Register (CCR)” on page 23). The chaining options and the stopping options must be looked at together to determine the action a DMA channel takes when these conditions are encountered during DMA channel execution.

If chaining is disabled, the DMA channel is stopped by the programmed option.
If chaining is enabled, the stop programming options are ignored unless the programmed stop option is different from the programmed chaining option.
If both a chain and stop option are reached simultaneously, the channel will chain.

The state of the CCR enable bit in a memory CDB determines whether a DMA channel remains enabled or disabled after the chaining operation occurs.

2.4.3 Adding CDB's to a Chain

The following method should be used for creating CDB chains:

1. Create the CDB in memory. One of the chain operations (CCR9:7) should be selected. This CDB's CPR should be 00000000h.
2. Write the CDB's address to the CPR of the previous CDB.
3. Read the CPR register in VolantPCI.
   - If its value is 00000000h then write the address of this CDB OR'ed with 00000001h to the CPR register.
   - otherwise you are done.

When it is known that the channel is stopped (i.e. first CDB, channel reset, etc.) software can do step 1 and then write the address of this CDB OR'ed with 00000001h to the CPR register.

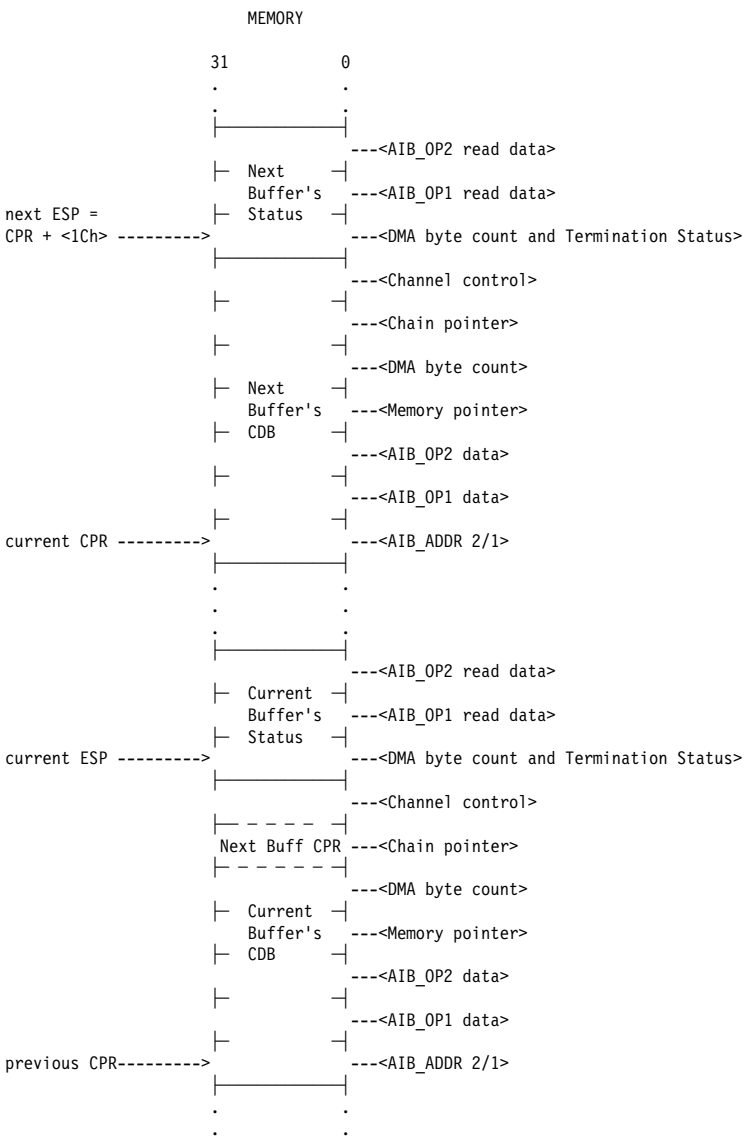


Figure 3. Linked List Chaining: VolantPCI Enhanced Status Mode List Chaining

2.5 DMA Miscellaneous Registers

2.5.1 AIB Bus Global DMA Command Register (GDCR)

2.5.1.1 Description:

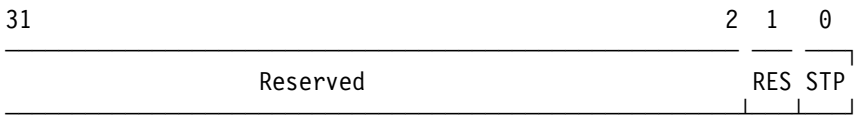
The GDCR allows two commands to be issued globally to all DMA channels at once for a given AIB bus. It allows all DMA channels to be 'stopped' with one command or all DMA channels to be reset with one command. The 'stopped' state of each channel is reflected in the EN/DIS bit of the CCR.

The command to re-start all channels should not be issued with the same write that releases all channels from a reset command.

2.5.1.2 Register Addressing

```
 Address Offset
-------------------------------------------------------------------------
 +08000h  (read/write)
```

2.5.1.3 Register Format



2.5.1.4 Bit Descriptions

- Bits 31-2. Reserved. Always write these bits to '0'.
- Bit 1. Reset all channels command. Setting this bit to '1' causes all the channels to be reset as shown by the Reset Command for each register. All channels are held in the reset state until the bit is set back to '0'.
- Bit 0. Stop all channels command. If written to a '1' all channels are stopped. If written to a '0' all channels are re-started.

  When a channel is 'stopped', all internally pending operations are performed before the channel stops servicing DMA requests.

2.5.1.5 Reset Conditions

```
CHIP RESET:     0000 0000 0000 0000 0000 0000 0000 0000
DMA RESET:      0000 0000 0000 0000 0000 0000 0000 00S0
```

2.5.2 DMA FIFO Residual Count registers (DFRC)

2.5.2.1 Description:

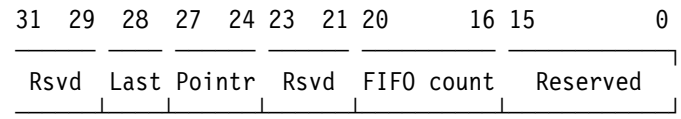
The DFRC register can be read when a DMA channel is stopped or known to be inactive to determine how many bytes of data are in the FIFO.

Note: This register is for debug purposes only. Its contents may not be compatible with future versions of this chip. This register will not be documented in the Hardware Technical Reference.

2.5.2.2 Register Addressing

DMA Channel	Address Offset
0	+0002Ch (read only)
1	+0102Ch (read only)
2	+0202Ch (read only)
3	+0302Ch (read only)
4	+0402Ch (read only)
5	+0502Ch (read only)
6	+0602Ch (read only)
7	+0702Ch (read only)

2.5.2.3 Register Format



2.5.2.4 Bit Descriptions

- Bits 31-29. Reserved. Always read as 0.
- Bits 28. This is the Last Buffer Indicator. When set, indicates that this is the last buffer needed to complete this DMA.
- Bits 27-24. Buffer Pointer indicates the next location in the buffer to be filled (Rx) or flushed (Tx).
- Bits 23-21. Reserved. Always read as 0.
- Bits 20-16. FIFO Count indicates the number of bytes left in the FIFO to flush (Tx) or number of bytes that may be filled (Rx).
- Bits 15-0. Reserved. Always read as 0.

2.5.2.5 Reset Conditions

```
CHIP RESET:     000U UUUU 000U UUUU 0000 0000 0000 0000
DMA RESET:      000U UUUU 000U UUUU 0000 0000 0000 0000
```

2.5.3 DMA Buffer Data registers (DBD)

2.5.3.1 Description:

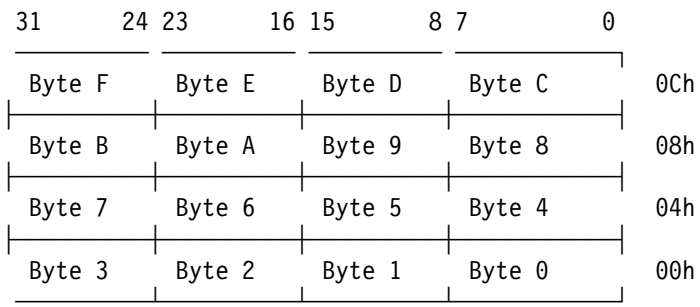
The DBD registers can be read/written when a DMA channel is stopped or known to be inactive to view, initialize or change the data in the buffer.

Note: These registers are for debug purposes only. Their contents may not be compatible with future versions of this chip. This register will not be documented in the Hardware Technical Reference.

2.5.3.2 Register Addressing

DMA Channel	Address Offsets				
0	+00100h	+00104h	+00108h	+0010Ch	(read/write)
1	+01100h	+01104h	+01108h	+0110Ch	(read/write)
2	+02100h	+02104h	+02108h	+0210Ch	(read/write)
3	+03100h	+03104h	+03108h	+0310Ch	(read/write)
4	+04100h	+04104h	+04108h	+0410Ch	(read/write)
5	+05100h	+05104h	+05108h	+0510Ch	(read/write)
6	+06100h	+06104h	+06108h	+0610Ch	(read/write)
7	+07100h	+07104h	+07108h	+0710Ch	(read/write)

2.5.3.3 Register Format



2.5.3.4 Bit Descriptions

- Each byte location corresponds to a data byte location in the internal data buffer. The validity of each byte depends on several factors including number of bytes transfered, Transfer count (TCR), Memory pointer (MPR) and whether the channel is Tx or Rx.

2.5.3.5 Reset Conditions

CHIP RESET:	UUUU	UUUU	UUUU	UUUU	UUUU	UUUU	UUUU	UUUU
DMA RESET:	UUUU	UUUU	UUUU	UUUU	UUUU	UUUU	UUUU	UUUU

3.0 VolantPCI Interrupts

3.1 General

VolantPCI provides an 8-bit interrupt vector register which can be read from the PCI Bus side of the chip. This vector indicates the highest priority interrupt currently pending. The values used for this register are the same as the vectors the Vero presented on its XINT bus.

The AIB side of VolantPCI can support up to 4 interrupts with up to two defined as vectored interrupt. In direct mode, the INT(3-0) signals correspond to specific local bus vector numbers. In vectored mode, the INT(1-0) and INTACK(1-0) signals are used in conjunction with an interrupt vector supplied by the interrupting device on the AIB bus. In addition to the vectored and direct interrupt support for each AIB bus, the AIB ERROR signal causes a high priority interrupt to occur.

3.2 Interrupt Sources

Interrupts are sourced from the AIB Bus in either Direct or Vectored mode, from the DMA Channels, from the AIB ERROR signal and from the PIO's. VolantPCI internally manages the prioritization of interrupt input signals and translates those inputs into specific interrupt vectors which are placed in the internal interrupt vector register.

Each of the DMA channels has several possible sources of interrupts. Of these, two are classified as 'normal termination' interrupts and the rest are classified as 'error' interrupts.

Each DMA channel has an interrupt status register (see 2.3.9, “DMA Interrupt Status Registers (DISR)” on page 33) associated with it that is read during the interrupt subroutine to determine the pending interrupt status bits for that channel. Reading the DISR clears all active status bits.

3.2.1 DMA Interrupts

Each DMA channel has a fixed interrupt vector assigned to it. These are shown in Table 13 on page 42.

3.2.2 AIB Interrupts

AIB sourced interrupts can be either direct or vectored. Vectored/Direct modes of interrupting are selected via the IIR (Interrupt Initialization Register, see 3.3.1, “Interrupt Initialization register (IIR)”)

3.2.2.1 Direct Mode

Table 13 on page 42 shows the corresponding interrupt vector for each of the direct mode input pins.

3.2.2.2 Vectored Mode

In vectored mode, the INT/INTACK signals provide the handshake for the interrupting device to deliver an 8-bit vector across the D(7-0) bus. There are two INT/INTACK pairs on the AIB bus. The vector received from the device will be readable in the IVR if it is the highest priority interrupt. Regardless of vector, INT0 is always higher priority than INT1. A vector of FF will be ignored by VolantPCI.

3.2.3 AIB ERROR Interrupt

The -AIB_ERROR signal provides a way for the AIB board to signal the processor with a high priority interrupt. It is level sensitive.

3.2.4 PIO Interrupts

Some of the PIO pins can be enabled to interrupt on the detection of a change in state of the pin. There is one interrupt vector per port.

3.2.5 Interrupt Priorities

The following table shows all VolantPCI interrupts and their corresponding vectors and priorities. The highest priority interrupt is at the top of the table.

Table 13. VolantPCI DMA Channel Vector Assignment

Interrupt Name	Interrupt Vector # (Decimal / hex)	Priority
AIB_ERROR	232 / E8h	highest
Vector mode INT(0)	(external)	
Vector mode INT(1)	(external)	
Direct mode INT(2)	120 / 78h	
Direct mode INT(3)	112 / 70h	
PIO port 0	111 / 6Fh	
PIO port 1	110 / 6Eh	
PIO port 2	109 / 6Dh	
PIO port 3	108 / 6Ch	
Direct mode INT(1)	83 / 53h	
Direct mode INT(0)	82 / 52h	
DMA Channel 0	71 / 47h	
DMA Channel 1	70 / 46h	
DMA Channel 2	69 / 45h	
DMA Channel 3	68 / 44h	
DMA Channel 4	67 / 43h	
DMA Channel 5	66 / 42h	
DMA Channel 6	65 / 41h	
DMA Channel 7	64 / 40h	lowest

3.3 Programmable Options

3.3.1 Interrupt Initialization register (IIR)

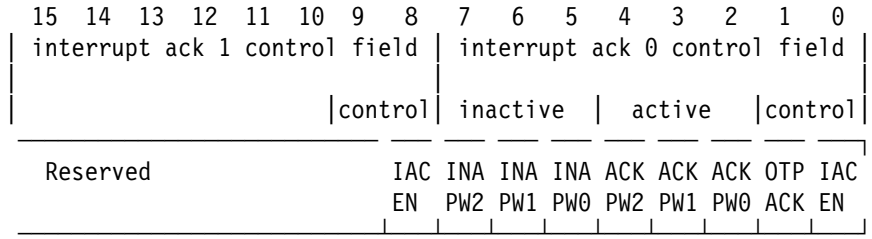
3.3.1.1 Description

The IIR register controls the initialization parameters for the VolantPCI chip interrupt controller logic. This register must be programmed before interrupts are generated from the AIB.

3.3.1.2 Register Addressing

Address Offset
+08010

3.3.1.3 Register Format



3.3.1.4 Bit Descriptions

- Bits 31-16. Reserved. Always write these to 0.
- Bits 15-8. Interrupt level 1 control field. **Note** that bits 15-9 do not exist and always read '0'. INT1 active, inactive and number of pulses are the same as defined for INT0.
- Bits 7-0. Interrupt level 0 control field.

Each 8 bit control field defines the interrupt controller operation for that interrupt level. All control fields are equivalent.

- IAC EN: Interrupt acknowledge enable.
  - 0 = Direct Interrupt Mode
  - 1 = Interrupt Acknowledge Mode

  When set to '0' the INT0 (or INT1) input functions as a direct interrupt input from the AIB. No interrupt acknowledge cycle will be run. In this mode the INT signals are level sensitive and must be held until software services it.

  When set to '1' an interrupt will cause an interrupt acknowledge cycle to be run on the AIB Bus. The vector during the interrupt acknowledge cycle will be presented to the processor. An EOI will be required to enable detection of the next interrupt. Vector FFh will be ignored by VolantPCI; no interrupt will be reported to the PCI Bus and no EOI will be needed.
- OTP ACK: One or two pulse interrupt acknowledge signal.
  - OTP ACK=0: one pulse.

– OTP ACK=1: two pulses.

Some devices use an 8259 type interrupt acknowledge cycle (two pulse). Others require external logic to eliminate the first pulse. This bit should reduce or eliminate the need for that external logic.

If two pulses are selected, an idle time is inserted between the two pulses for device recovery, the length of which is controlled by INA PW2-0.

- ACK PW2-0: Interrupt acknowledge cycle active pulse width.

```
ACK  ACK  ACK │ INTACK 'low' (active) Pulse Width
PW2  PW1  PW0 │ N  x  "Local Bus Clock Period in ns."
--------------------------------------------------
 0    0    0  │   N =  3
 0    0    1  │   N =  4
 0    1    0  │   N =  5
 0    1    1  │   N =  6
 1    0    0  │   N =  7
 1    0    1  │   N =  8
 1    1    0  │   N =  9
 1    1    1  │   N =  10
```

These three bits provide eight options for the interrupt acknowledge cycle active length. Some devices respond slower than others and therefore would require external logic to insert wait states into the interrupt acknowledge cycle. These bits should eliminate the need for this external logic.

- INA PW2-0: Interrupt acknowledge recovery (inactive) width. INA PW2 is hardwired to '0'.

```
INA  INA  INA │ INTACK 'high' (inactive) Pulse Width
PW2  PW1  PW0 │ N  x  "Local Bus Clock Period in ns."
--------------------------------------------------
 0    0    0  │   N =  1
 0    0    1  │   N =  2
 0    1    0  │   N =  3
 0    1    1  │   N =  4
 1    X    X  │   reserved
```

These three bits provide options for the interrupt acknowledge recovery length. Some devices are slower than others to release (float) data signals on the AIB Bus, and thus require a certain amount of bus idle time after the interrupt acknowledge cycle is completed.

If the OTP ACK bit ='1' (two pulses), both interrupts acknowledge pulses will be the length set by ACK PW2-0, with an inactive time between and after them determined by INA PW2-0.

3.3.1.5 Reset Conditions

```
CHIP RESET:     0000 0000 0000 0000 UUUU UUU0 UUUU UUU0
```

3.3.2 Interrupt Mask register (IMR)

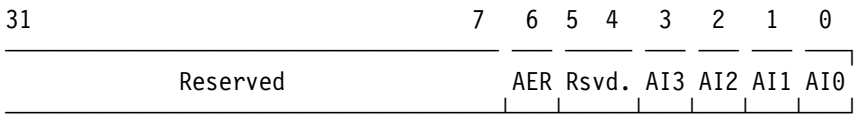
3.3.2.1 Description:

The IMR allows the four AIB interrupt inputs to be enabled and disabled from causing interrupts to the processor without programming the interrupting device directly. This allows for a point of synchronization between the processor and the interrupting device. Typically, devices contain their own interrupt enable functions. However, these are sometimes not usable on a real time basis since spurious interrupts can result if an interrupt is disabled within a device immediately after its interrupt output has been asserted. This register prevents the possibility of these spurious interrupts if the device takes no precautions to do this.

3.3.2.2 Register Addressing

Address Offset
+08014

3.3.2.3 Register Format



3.3.2.4 Bit Descriptions

- Bit 31-7. Reserved. Always write these as 0.
- Bit 6. AIB_ERROR input mask. This is an alternate access to the same bit in the EIMR. See 3.3.4, “AIB Error Interrupt Mask register (EIMR).”
- Bit 5-4. Reserved. Always write these as 0.
- Bits 3-0. AIB_INT3-0 mask bits.
  - 1 = AIB Interrupt Enabled
  - 0 = AIB Interrupt Disabled

  These bits can individually mask the 4 AIB interrupt inputs from generating an interrupt to the processor. The disabling of these bits is synchronized with the interrupt input signal to prevent spurious interrupts when disabling.

3.3.2.5 Reset Conditions

CHIP RESET: 0000 0000 0000 0000 0000 0000 0000 0000

3.3.3 Interrupt Status register (ISR)

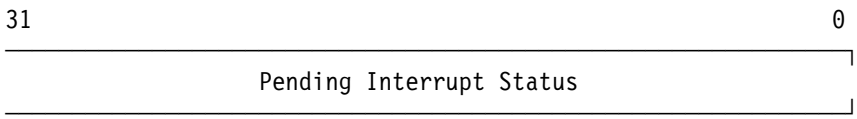
3.3.3.1 Description:

The ISR allows all VolantPCI interrupt input sources associated with the AIB bus to be read in a single status register.

3.3.3.2 Register Addressing

Address Offset
+08018

3.3.3.3 Register Format



3.3.3.4 Bit Descriptions

- Bits 31-24. Reserved.
- Bits 23-20. AIB_INT 3-0 status bits. (direct-mode only)
- Bit 19. AIB error input status bit.
- Bits 18-8. Reserved.
- Bits 7-0. VolantPCI DMA Channels 7-0.

3.3.3.5 Reset Conditions

Not applicable

3.3.4 AIB Error Interrupt Mask register (EIMR)

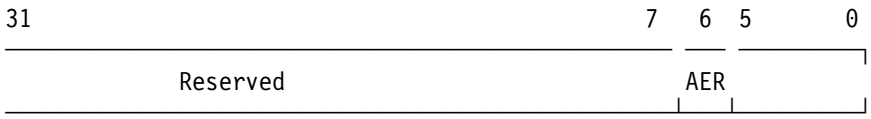
3.3.4.1 Description:

The EIMR allows the AIB Error input to be enabled and disabled from causing interrupts without programming the interrupting device directly.

3.3.4.2 Register Addressing

Address Offset
+0801C

3.3.4.3 Register Format



3.3.4.4 Bit Descriptions

- Bit 31-7. Reserved. Always write these as 0.
- Bits 6. AIB_ERROR input mask bit.
  - 1 = AIB Error Interrupt Enabled
  - 0 = AIB Error Interrupt Disabled

  This bit can mask the AIB_ERROR input signal from generating an interrupt to the processor. The disabling of this bit is synchronized with the interrupt input signal to prevent spurious interrupts when disabling. This bit is also accessible in the IMR (3.3.2, “Interrupt Mask register (IMR)” on page 45)
- Bit 5-0. Reserved. Always write these as 0.

3.3.4.5 Reset Conditions

CHIP RESET: 0000 0000 0000 0000 0000 0000 0000 0000

3.4 Interrupt Commands

End of Interrupt (EOI) commands must be issued for INT0 and INT1 when these signals are used in interrupt acknowledge mode. The EOI command is actually a write (of any data) or read to the corresponding interrupt's EOI register. When an interrupt occurs, that interrupt signal is disabled. The EOI command is used to inform VolantPCI that the interrupt source on the AIB bus has been cleared and to re-enable the receiver for a new interrupt. The EIO command will be 'retried' by VolantPCI until any pending AIB bus write operations from the PCI Bus has completed.

3.4.1 AIB INT0/1 End-of-Interrupt (EOI0/1) commands

3.4.1.1 Description:

The EOI0/1 commands are used to inform the interrupt controller that the processor has cleared the AIB INT0/1 source. These commands must be issued for all interrupting AIB devices that use the Interrupt Acknowledge option in 3.3.1, “Interrupt Initialization register (IIR)” on page 43. These commands should be issued after the command that goes directly to the AIB to clear the interrupting condition.

Since writes to the AIB bus are posted in VolantPCI, the hardware does not accept EOI commands while a posted AIB access is pending. This is done to prevent the EOI command from occuring before an interrupt clearing write completes on the AIB bus. This is done by using the PCI retry mechanism.

The "read EOI" is recommended over the "write EOI" because writes on the PCI Bus can be posted and therefore the timing of when the interrupt actually clears cannot be controlled.

3.4.1.2 Register Addressing

Address Offset	
+09000	EOI0
+0A000	EOI1

3.4.1.3 Command Format

- Bits 31-1. Reserved. Write data is "don't care". Read is always '0'.
- Bit 0. Write data is "don't care". When read as '1', this bit indicates the EOI was needed to allow further interupts. When '0', it indicates VolantPCI was already enabled for further interrupts. Any write or read to this register will clear this bit.

3.4.1.4 Reset Conditions

```
CHIP RESET:     0000 0000 0000 0000 0000 0000 0000 0000
```

3.4.2 Interrupt Vector Register (IVR)

3.4.2.1 Description:

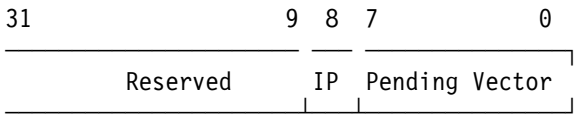
This register is used to indicate the highest priority interrupt currently pending in the VolantPCI chip. The vector numbers it returns are compatible with the interrupt vectors used with the Vero chip.

Upon receiving a VolantPCI interrupt, the first level handler should read this location to determine which interrupt needs service.

3.4.2.2 Register Addressing

Address Offset
+0F000 (read)

3.4.2.3 Register Format



3.4.2.4 Bit Descriptions

- Bits 31-9. Reserved.
- Bit 8. Interrupt Pending.

  This bit, when '1', indicates that an interrupt is waiting to be serviced in VolantPCI (i.e. the INTA# pin is active).
- Bits 7-0. Pending Vector Number (read only)

  These bits will return the vector of the highest priority interrupt currently pending in VolantPCI.

3.4.2.5 Reset Conditions

CHIP RESET: 0000 0000 0000 0000 0000 0000 0000 0000

4.0 VolantPCI AIB Bus Interface

4.1 General

The VolantPCI module provides for a separate I/O bus (AIB Bus) that is split from the PCI Bus. This allows the DMA controller to only access the Local Bus when one of its 16 byte data buffers is empty (transmit) or full (receive). In this manner, only low speed non-bursted data movement occurs on the AIB bus, and only bursted data movement occurs on the Local Bus. This describes the nature of most data movement (DMA controlled) that occurs between the AIB bus and base adapter.

The AIB Bus supports target devices. Target devices can be directly accessed by the processor for initialization and status operations, and will typically use the VolantPCI DMA controller to move data from the device to memory. The AIB Bus supports a 256k byte address space within the PCI Bus address space which is mapped to 64k for addressing target devices on the AIB. The AIB addresses match the PCI addresses for easy conversion. Only byte accesses to 4-byte aligned PCI addresses are supported.

The following address map describes the mapping of each AIB bus onto the PCI Bus address space.

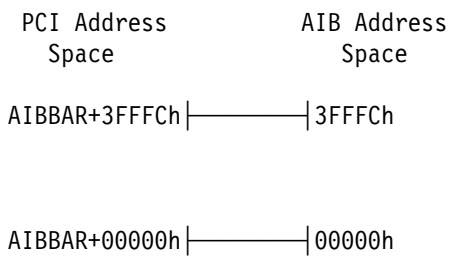


Figure 4. PCI Bus Address to AIB Address Map

Within the 256kB AIB bus region, four separate regions (chip selects) can be defined with each having a programmable number of wait states.

4.1.1 AIB Arbiter

Contained within the VolantPCI chip is the arbitration logic that determines which requesting master gains control of the AIB Bus to run its cycle. There are three requesting masters for the AIB Bus.

1. Local Bus master cycle to AIB
2. VolantPCI DMA channel cycle to AIB including AIB OPs
3. Interrupt Acknowledge cycle to AIB

The prioritization scheme is the order shown above. The Local Bus master will always be given highest priority. If another cycle is currently active on the AIB the Local Bus master will be granted the next cycle. Therefore, the maximum latency for the Local Bus master to begin its cycle is the maximum cycle time of a DMA channel cycle or interrupt acknowledge cycle or two AIBOP cycles.

The DMA channels also arbitrate amongst themselves. A DMA channel whose input -DREQ signal is held active will continue to be granted the AIB bus ahead of other DMA channels and interrupt acknowledge requests until its request is sampled inactive at the end of a DMA cycle, or until its DMA channel buffer inside VolantPCI becomes full (for a receive channel) or empty (for a transmit channel). As stated above, a Local Bus master request for the

AIB will interrupt this process and be granted immediately. Service priority among these groups is fixed with channel 0 being the highest priority and channel 7 being the lowest priority. Arbitration is in a round-robin fashion (CH0-CH1-CH2...CH7-CH0-etc).

The interrupt acknowledge cycle request is the lowest priority in the arbitration scheme. In order for this cycle request to be granted, both Local Bus master and DMA cycle requests must be inactive for at least one clock cycle. This can lead to extended servicing latencies based upon such things as the size of AIB device DMA fifos and AIB device DMA cycle time. If a more predictable interrupt response is needed (perhaps due to a heavy DMA environment) direct interrupt inputs from the AIB should be used.

4.2 AIB Initialization registers

Prior to accessing devices on the AIB buses, the VolantPCI AIB initialization registers must be programmed with the proper AIB bus setup information. The AIB bus initialization registers create a flexible AIB interface that allow devices to be connected to the AIB bus with minimal additional circuitry.

4.2.1 Chip Select Definition registers (CSD0-3)

4.2.1.1 Description:

Each of the chip select signals can be programmed for a base starting address on any 8KB boundary within the 256KB address space of the AIB bus. Each of the chip select signals can be programmed to decode a range of addresses from 8KB to 64KB. Also, each of the chip select signals can be programmed to run a variable cycle based on the local bus clock frequency in increments of the clock period. In addition, both active and inactive (restore) portions of the cycle can be varied. This should be suitable to accommodate most devices. There is a separate CSD register for each chip select signal.

4.2.1.2 Register Addressing

Chip Select	Address Offset
0	+0B000
1	+0B004
2	+0B008
3	+0B00C

4.2.1.3 Register Format

31 – 22	21	20	19	18	17	16
Reserved	A17	A16	A15	A14	A13	RSV

15	14	13	12	11	10	9	8 – 6	5	4	3	2	1	0
RSV	AC2	AC1	AC0	RSV	IC1	IC0	Reserved	RG1	RG0	X	DS1	DS0	ENA

4.2.1.4 CSD Register Bit Descriptions

- Bits 31-22. Reserved. Always write these to 0.
- Bits 21-17. Chip select base address. These bits determine the base address of the chip select address range. Starting addresses can be on any 8KB boundary (based on the chip select range, see bits 4 and 5) of the 256kB AIB address space. These bits can be thought of as replacing bits 17-13 of the physical PCI Bus address. The actual bits replaced is dependent on the bits 5-4.
- Bits 16-15. Reserved. Always write to 0.
- Bits 14-12. Active cycle time. These bits determine the active cycle time for a particular chip select. Active time is defined as the time when the -A_CS signal is low (0 volts).

AC2	AC1	AC0	CS 'low' (active) Pulse Width N x "Local Bus Clock Period in ns."
0	0	0	N = 3
0	0	1	N = 4
0	1	0	N = 5
0	1	1	N = 6
1	0	0	N = 7
1	0	1	N = 8
1	1	0	N = 9
1	1	1	N = 10

- Bits 11. Reserved. Always write to 0.
- Bits 10-9. Inactive cycle time. These bits determine the inactive cycle time for a particular chip select. Inactive time is defined as the time when the -A_CS signal is high (5 volts).

IC1	IC0	CS 'high' (inactive) Pulse Width N x "Local Bus Clock Period in ns."
0	0	N = 2
0	1	N = 3
1	0	N = 4
1	1	N = 5

- Bits 8-6. Reserved. Always write these to 0.
- Bits 5-4. Chip select address range bits. These bits are binary encoded to provide the range of addresses that are decoded for each chip select signal. There are four possible address ranges: 8K, 16K, 32K, 64k. There is a restriction that the base starting address selected in bits 20-17 must be on a boundary that is equal to the decode range selected by bits 6-4. For instance, if a decode range of 8K is selected the base starting address can be any 8K boundary.

RG1	RG0	Chip Select Decode Range
0	0	8K
0	1	16K
1	0	32K
1	1	64K

- Bits 3-1. Reserved. Always write to 0. Bit 2-1 were the device size bits in Vero. Since only 8-bit devices are supported for VolantPCI these bits are hard-wired to '00'.

DS1	DS0	Device size
0	0	8-bit
0	1	Reserved
1	0	Reserved
1	1	Reserved

- Bit 0. Chip select enable. This bit when set to '0' disables this chip select and when set to '1' enables the chip select.

4.2.1.5 Reset Conditions

```
CHIP RESET:     0000 0000 00UU UUU0 UUUU 0UU0 00UU 0000
```

4.2.2 DMA Acknowledge Pulse Width Registers (DAPW)

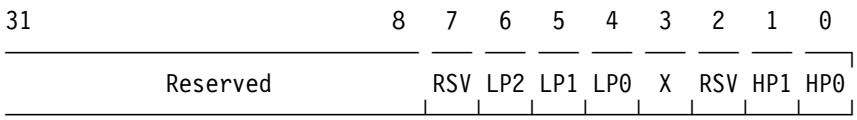
4.2.2.1 Description:

The DAPW register is used to program the pulse width of the DMA acknowledge (DACK) signal that the VolantPCI chip sends to the DMA requesting device. There is one DAPW register for Rx channels, one for Tx channels. This register allows the DACK cycle to be optimized relative to specific DMA devices.

4.2.2.2 Register Addressing

```
 Address Offset
-----------------
    +0B030    -- receive channels
    +0B038    -- transmit channels
```

4.2.2.3 Register Format



4.2.2.4 Bit Descriptions

- Bits 31-7. Reserved.
- Bits 6-4. DACK 'low' pulse width. These bits are binary encoded to provide the options for setting the duration that the DACK signal will be 'low' (0 volts).

LP2	LP1	LP0	DACK 'low' (active) Pulse Width N x "Local Bus Clock Period in ns."
0	0	0	N = 2
0	0	1	N = 3
0	1	0	N = 4
0	1	1	N = 5
1	0	0	N = 6
1	0	1	N = 7
1	1	0	N = 8
1	1	1	N = 9

- Bits 3-2. Reserved. Always program this bit to a '0'.
- Bits 1-0. DACK 'high' pulse width. These bits are binary encoded to provide the options for setting the duration that the DACK signal will be 'high'.

HP1	HP0	DACK 'high' (inactive) Pulse Width N x "Local Bus Clock Period in ns."
0	0	N = 1
0	1	N = 2
1	0	N = 3
1	1	N = 4

4.2.2.5 Reset Conditions

```
CHIP RESET:     0000 0000 0000 0000 0000 0000 0UUU 00UU
```

4.2.3 AIB Bus Configuration Register (ACR)

4.2.3.1 Description:

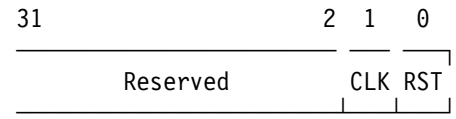
The ACR provides program control of the AIB Reset signal, and the AIB CLK signal. On power up, the AIB CLK driver is enabled. The clock driver should be disabled if the clock is not required on the AIB.

Note: The Vero chip supported AIB bus parity and a checking disable bit in the ACR. This feature is not supported in VolantPCI

4.2.3.2 Register Addressing

Address Offset
+0C000

4.2.3.3 Register Format



4.2.3.4 Bit Descriptions

- Bits 31-2. Reserved. Always write these to 0.
- Bit 1. AIB Bus Clock Enable. This enables/disables the A_CLK output which is a buffered version of the CLK input. Note that its reset state is **enabled**.
  - 0 = Enable the AIB CLK Driver
  - 1 = Disable the AIB CLK Driver
- Bit 0. AIB Bus Reset.
  - 1 = AIB Bus Reset Active
  - 0 = AIB Bus Reset Inactive

  Note: The AIB Bus reset also is active when the RST# signal is active on the PCI Bus.

4.2.3.5 Reset Conditions

CHIP RESET: 0000 0000 0000 0000 0000 0000 0000 0000

5.0 PCI Bus Interface

5.1 PCI Bus Operation

VolantPCI is both a PCI bus master and a PCI bus target.

5.2 PCI Bus target

Only non-burst accesses are supported. Any attempt to burst multiple words will result in VolantPCI disconnecting the cycle.

5.2.1 AIB bus access

Writes to the AIB bus from masters on the PCI bus may be posted (stored in a register) within the VolantPCI chip. The posted write data will be forwarded to the AIB bus a soon as the bus is free. An EOI command will be 'retried' until any pending AIB write has completed. (This ensures a write to the AIB device to clear the pending interrupt condition has completed.) For more information on the EOI command see 3.4, “Interrupt Commands” on page 48.

Reads from the AIB bus by masters on the PCI bus will be delayed using PCI Bus retry until the data can be fetched from the AIB.

5.2.2 Internal Registers Accesses

Writes to internal registers may be posted. Accesses to interrupt clearing registers will cause the interrupt being cleared to clear within three clocks of the acceptance of the access. All registers between offset +0000h and +BFFFh will be accessed in a delayed manner using write posting and PCI delayed reads.

5.3 PCI Bus Master

VolantPCI is a bursting PCI bus master capable of supporting zero wait state PCI Bus cycles.

VolantPCI becomes a PCI bus master for DMA data transfers, posting status and DMA list chain operations. For data transfers, VolantPCI will transfer a maximum of one full internal buffer under a single PCI bus ownership period. Therefore, a maximum of 16 bytes will be transferred for one VolantPCI request/grant on the local bus. All byte enable signals will be asserted on 'read' operations regardless of the number of bytes required. For status, a burst write of three words will be performed. For DMA list chain operations, a burst read of seven words will be performed.

5.3.1 Internal Arbitration for the PCI Bus

All DMA channels arbitrate for ownership of the PCI Bus internal to the VolantPCI chip in a round-robin fashion similar to the DREQ priorities.

5.3.2 Multiple VolantPCI Chips on the PCI Bus

The PCI Bus handles multiple copies of the same chip on the same bus as part of its architecture.

When using a defined PCI Bus expansion board (such as the PMC) there is a limit of one chip attached to the bus. This limit was set to allow the base card designers to have a consistent model of the loading of the daughter card. Because of this limit there is generally only one REQ# and GNT# signal on the connector. VolantPCI contains logic to allow multiple VolantPCI chips to share a single REQ#/GNT# pair. This will generally result in a violation of the expansion board's loading limits. Given a known motherboard, however, this violation may not be a problem. This can result in cost savings, space savings and performance gains (vs. PCI Bus bridge).

Figure 5 on page 60 shows how two VolantPCI chips would be connected to share a single request/grant signal pair.

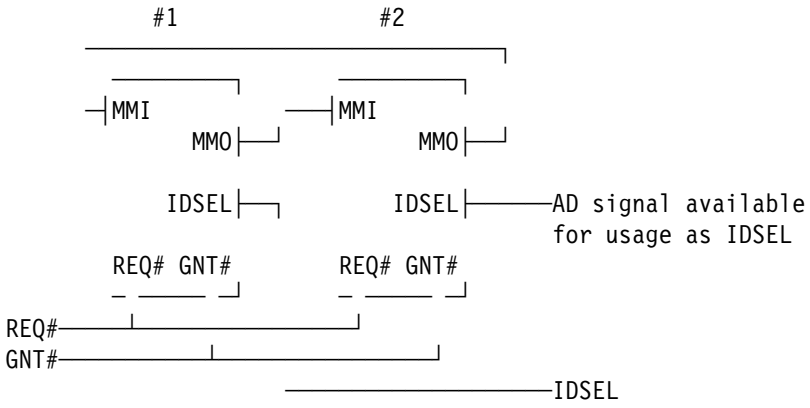


Figure 5. Connecting two VolantPCI chips using Multi-Master

In Figure 5 on page 60, all other PCI signals would be connected normally (i.e. both devices AD(1) signals would be connected to PCI bus AD(1)). The REQ# signals would together be connected to the request line. Similarly for the GNT# signal. The IDSEL of the second chip needs to be connected to an AD signal that can be used for IDSEL selection (usually AD31-AD16) that is not used for the IDSEL for any other device (including the first chip).

The board designer needs to be aware of the following considerations:

- The PCI Bus must not be parked to chips sharing a request/grant pair.
- The added capacitance on the PCI Bus must be looked at.
- The added capacitance on the REQ# and GNT# lines must be looked at.
- The connection of the IDSEL line may need to be resistively coupled.

5.4 PCI Bus Configuration Registers

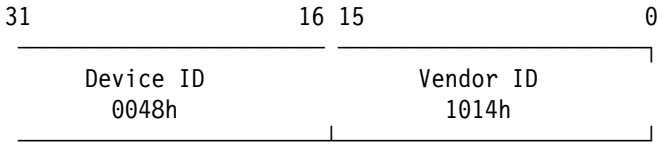
Note: According to PCI specification, care should be taken to deal correctly with registers that have reserved bits. On reads, software must use appropriate masks to extract defined bits, and ***may not rely on reserved bits being any particular value***. On writes, software must ensure that the values of the reserved bit positions are preserved, by first reading the values of these bits and writing these values back when writing the new values of other bit positions.

5.4.1 Device/Vendor ID (DEVID)

Description. The Device/Vendor ID Register identifies IBM as the manufacturer of VolantPCI. This register is read only from the PCI Bus.

Register Format

(PCI Config Offset = 00 h) 32-bit read only



Bit Descriptions

- Bits 31-16: Device ID. This read only field identifies the device as VolantPCI. Its value is X'0048'.
- Bits 15-0: Vendor ID. This read only field identifies the manufacturer as IBM. Its value is X'1014'.

Reset Conditions

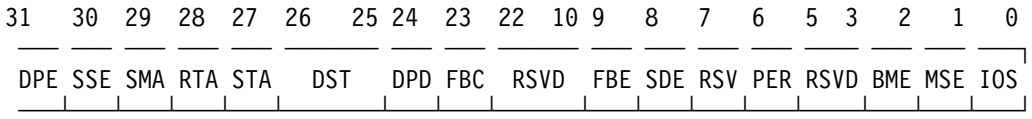
RST#: 0000 0000 0100 1000 0001 0000 0001 0100

5.4.2 Host Status/Command Register (HSCR)

Description. The Host Status/Command Register provides the control and status of the PCI interface in VolantPCI. Bits in the status field can be reset from '1' to '0' by writing a '1' to the corresponding bit position.

Register Format

(PCI Config Offset = 04 h) 32-bit rd/wr



Bit Descriptions

- Bit 31: Detected Parity Error. This bit is set to '1' when VolantPCI detects a PCI Bus parity error, regardless of the state of the Parity Error Response Bit (Bit 6). A system configuration write of '1' to this bit will reset it to '0'.
- Bit 30: Signalled System Error. This bit is set to '1' whenever VolantPCI signals SERR#. A system configuration write of '1' to this bit will reset it to '0'.
- Bit 29: Signalled Master Abort. This bit is set to '1' whenever VolantPCI terminates a PCI Bus master cycle with a master abort. A system configuration write of '1' to this bit will reset it to '0'.
- Bit 28: Received Target Abort. This bit is set to '1' whenever VolantPCI receives a target abort as a PCI Bus initiator. A system configuration write of '1' to this bit will reset it to '0'.
- Bit 27: Signalled Target Abort Bit. VolantPCI never signals a target abort, therefore this bit always reads '0'.
- Bit 26-25: DEVSEL Timing Bits. This read-only field indicates the slowest DEVSEL timing of VolantPCI on the PCI bus. DEVSEL timing is set at medium, or '01'b.
- Bit 24: Data Parity Detected. VolantPCI sets this bit to '1' when all of the following conditions are met: 1) VolantPCI asserted or observed -PERR active; 2) VolantPCI is the PCI Bus Master when the error occurs, and 3) the Parity Error Response bit (Bit 6) is set to '1'. A system configuration write of '1' to this bit will reset it to zero.
- Bit 23: Fast Back-to-Back Capable. VolantPCI supports this function, therefore, this bit always reads back a '1'.
- Bits 22-10: Reserved. Always read back as '0'.
- Bit 9: Fast Back-to-Back Enable. This capability is not implemented in VolantPCI, therefore, this bit always reads back a zero.
- Bit 8: -SERR Enable. This bit, when set, enables VolantPCI to drive SERR# on the PCI Bus.
- Bit 7: Wait Cycle Control. VolantPCI does not implement address/data stepping, therefore this bit is always '0'.
- Bit 6: Parity Error Response. When set to '1', VolantPCI is enabled to check parity during system PCI Bus Master and target transactions. When reset to '0', parity errors are ignored. Note that the Detected Parity Error bit (Bit 31) is set independent of the state of this bit.
- Bit 5: VGA Palette Snoop. This function is not implemented, therefore, this bit always reads zero.
- Bit 4: Memory Write and Invalidate. This function is not implemented. This bit always reads zero.
- Bit 3: Special Cycles. This function is not implemented, therefore, this bit always reads zero.
- Bit 2: Bus Master Enable. When set to '1'. VolantPCI is enabled to become a PCI Bus Master.
- Bit 1: Memory Space Enable. When set to '1', VolantPCI is enabled to respond to PCI Bus memory space accesses.
- Bit 0: I/O Space Enable. VolantPCI does not use I/O space, therefore, this bit is always '0'.

Reset Conditions

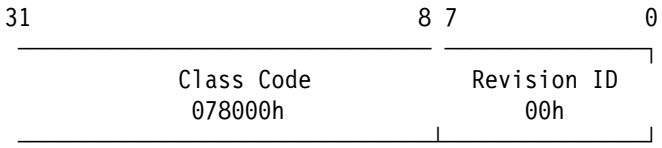
RST#: 0000 0010 1000 0000 0000 0000 0000 0000

5.4.3 Class Code/Revision ID Register (CCRID)

Description. The Class Code/Revision ID Register defines the general function and implementation level of the VolantPCI chip. This register is read-only.

Register Format

(PCI Config Offset = 08 h) 32-bit read only



Bit Descriptions

- Bits 31-8: Class Code. This field specifies the general function of the VolantPCI chip. The value is 078000 h, representing a Base Class=Communication Controller, SubClass=other, Programming Interface=not defined.
- Bits 7-0: Revision ID. This field specifies the revision level of the VolantPCI chip.

Reset Conditions

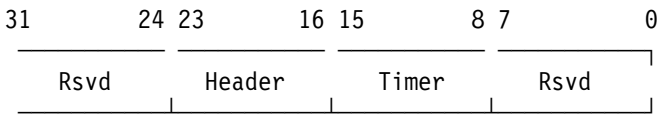
RST# : 0000 0111 1000 0000 0000 0000 0000 0000

5.4.4 Host Miscellaneous Functions Register (HMFR)

Description. This register is used to set the PCI bus latency timer in the VolantPCI chip.

Register Format

(PCI Config Offset = 0C h) 32-bit rd/wr



Bit Descriptions

- Bits 31-24: Reserved. This field is reserved for Built-In Self Test (BIST) support. Always read as zeros.
- Bits 23-16: This read-only field specifies the configuration space layout. This field reads back all zeroes.
- Bits 15-8: This read/write field specifies the latency timer value in units of PCI bus clocks. Only the upper five bits of this field are programmable, the other bits read back all zeroes. This field is reset to all zeroes by -RST.
- Bits 7-0: Reserved. This field is reserved for setting the cache line size. This field reads back all zeroes.

Reset Conditions

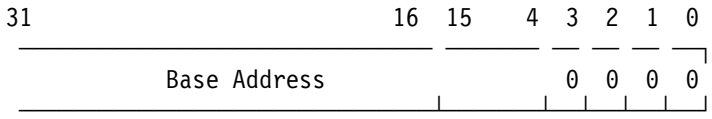
RST#: 0000 0000 0000 0000 0000 0000 0000 0000

5.4.5 Register Base Address Register (REGBAR)

Description. This register defines the 64kB window location of the VolantPCI registers. In Vero, these registers were located at 1FF80000h.

Register Format

(PCI Config Offset = 10 h) 32-bit rd/wr



Bit Descriptions

- Bits 31-16: Base Address. This read/write field specifies the starting PCI bus address for the VolantPCI register space.
- Bits 15-4: These bits always read zero. They are used by the system configuration routine to determine the size of the window requested.
- Bit 3-1: These bits always reads '0'.
- Bit 0: Window Type. '0' indicates that this window is located in memory address space (vs. I/O space).

Reset Conditions

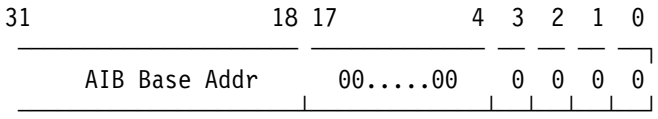
RST# : uuuu uuuu uuuu uuuu 0000 0000 0000 0000

5.4.6 AIB Base Address Register (AIBBAR)

Description. This register defines the 256kB window location in VolantPCI which maps to the AIB bus address space. In Vero, this window was at location 1FF00000h. (**Note:** Vero defined a 512kB window to the AIB.)

Register Format

(PCI Config Offset = 14 h) 32-bit rd/wr



Bit Descriptions

- Bits 31-18: AIB Base Address. This read/write field specifies the starting PCI bus memory address.
- Bits 17-4: These bits always read zero. They are used by the system configuration routine to determine the size of the window requested.
- Bit 3-1: These bits always reads '0'.
- Bit 0: Window Type. This bit is always '0' indicating a memory window.

Reset Conditions

Power Up Reset: uuuu uuuu uuuu uu00 0000 0000 0000 0000

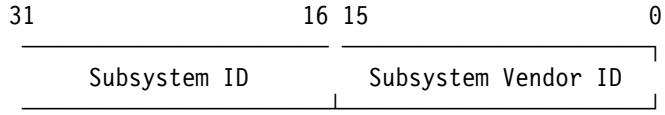
5.4.7 Subsystem ID (SSID)

Description. The Subsystem ID Register identifies the manufacturer and function of the subsystem using VolantPCI. The Subsystem Vendor ID is assigned by the PCI Special Interest Group. This register is read only. The value of this register is loaded from a serial ROM. If there is no serial ROM present, the value will remain at its reset value of '0000 0000 h', indicating the Subsystem ID function is not implemented.

This register is used by the device driver to identify what type of VolantPCI based adapter the card is. It is possible to have multiple VolantPCI based cards installed in a system, each requiring a different device driver.

Register Format

(PCI Config Offset = 2C h) 32-bit read only



Bit Descriptions

- Bits 31-16: Subsystem ID. This read only field identifies the board level implementation using the VolantPCI chip. Its value powers up to 0000h and is loaded from a serial ROM.
- Bits 15-0: Subsystem Vendor ID. This read only field identifies the manufacturer of the subsystem. Its value powers up to 0000h and is loaded from a serial ROM.

Reset Conditions

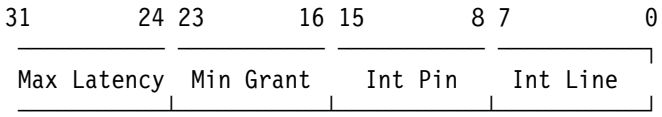
RST# : 0000 0000 0000 0000 0000 0000 0000 0000
(Note: loaded from serial ROM after RST#)

5.4.8 Latency/Grant/Interrupt Register (LGIR)

Description. The Latency/Grant/Interrupt Register is used to specify the maximum latency, minimum grant time, and interrupt information of the VolantPCI chip PCI Bus.

Register Format

(PCI Config Offset = 3Ch) 32-bit rd/wr



Bit Descriptions

- Bits 31-24: Maximum Latency. This read-only field defaults to 0x00 but can be loaded from SEPROM.
- Bits 23-16: Minimum Grant. This read-only field defaults to 0x00 but the least significant 2 bits can be loaded from SEPROM.
- Bits 15-8: Interrupt Pin. This read-only field defaults to X'01', indicating connection to -INTA on the PCI Bus.
- Bits 7-0. Interrupt Line. This read/write field is used for PCI interrupt priority and vector information. Values in this field are system architecture specific.

Reset Conditions

RST#: 0000 0000 0000 0000 0000 0001 0000 0000
(Note: loaded from serial ROM after RST#)

6.0 VolantPCI Miscellaneous Registers

6.1.1 Configuration Register (CFGR)

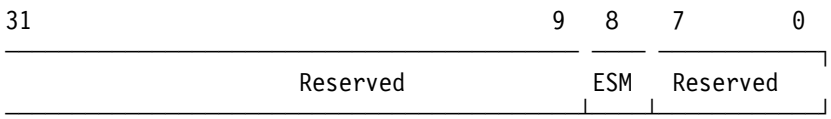
6.1.1.1 Description:

The CFGR is used to enable/disable various features in the VolantPCI chip.

6.1.1.2 Register Addressing

Address Offset
+0D008

6.1.1.3 Register Format



6.1.1.4 Bit Descriptions

- Bits 31-9. Reserved. Always write to 0.
- Bit 8. Enhanced Status Mode enable. When '1', VolantPCI will run in Enhanced Status Mode List Chaining. When '0', it will run in a mode which is highly compatible with Vero.
- Bits 7-0. Reserved. Always write to 0.

6.1.1.5 Reset Conditions

RST#:	0000 0000 0000 0000 0000 0000 0000 0000

6.1.2 LED Enable Register (LER)

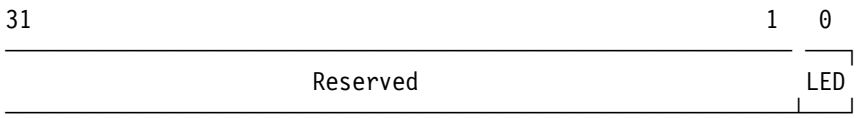
6.1.2.1 Description:

The LER is used to allow an external LED to be enabled and disabled with a write operation by the processor. It could also be used as a general purpose output.

6.1.2.2 Register Addressing

Address Offset
+0D004

6.1.2.3 Register Format



6.1.2.4 Bit Descriptions

- Bits 31-1. Reserved. Always write to 0.
- Bit 0. LED enable. (4mA Drive)
  - 1 = -LED_EN Signal at "low" voltage
  - 0 = -LED_EN Signal at "high" voltage

6.1.2.5 Reset Conditions

RST#: 0000 0000 0000 0000 0000 0000 0000 0000

6.1.3 Clock Timer Register (CTR)

6.1.3.1 Description:

This register can be used to determine the clock speed of the PCI bus. Two counters are run simultaneously, one using the PCI clock and the other using the REFCLK input. When either one reaches its maximum count, both counters stop. While running, the REFCLK counter will read 0 and the PCI clock counter will increment. If the pci count is read as the same value twice by software then the PCI clock speed can be determined using this formula:

If (PCI count) > (Reference count) then

$$\text{PCI clock frequency} = \frac{(\text{REFCLK frequency}) * (\text{PCI count})}{(\text{reference count} - 2)}$$

else if (PCI count) < (Reference count) then

$$\text{PCI clock frequency} = \frac{(\text{REFCLK frequency}) * (\text{PCI count} - 2)}{\text{reference count}}$$

else

PCI clock frequency = REFCLK frequency

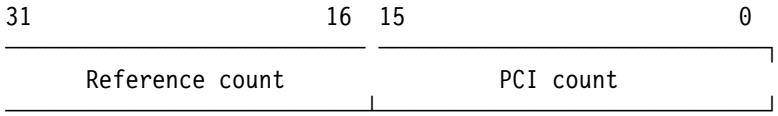
The accuracy will be within 0.5% when one clock is not more than 4 times faster than the other.

The register will start counting after the RST# signal is released. It can be cleared and restarted by writing this register with any value (byte enables are ignored).

6.1.3.2 Register Addressing

Address Offset
+0D00C

6.1.3.3 Register Format



6.1.3.4 Bit Descriptions

- Bits 31-16. Represents the number of REFCLK cycles counted. This value is only valid when PCI count is stopped.
- Bits 15-0. Represents the number of PCI clock cycles counted. When this value is read as the same value twice by software, both it and the Reference count are valid.

6.1.3.5 Reset Conditions

RST#: UUUU UUUU UUUU UUUU UUUU UUUU UUUU UUUU

6.2 Programmable I/O Control Registers

There are four programmable I/O (PIO) ports consisting of eight physical pins each. Of these eight pins, four are 'output-only' and four are programmable as either input or output. All have programmable polarity selection. Inputs can be programmed to interrupt on a change of state to the input.

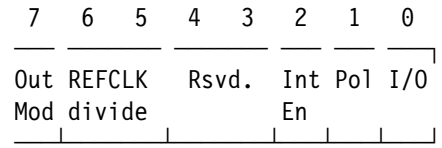
6.2.1 PIO Configuration Registers (PIOCFG)

The PIOCFG registers set the configuration for each PIO pin. Each PIOCFG is a byte register located in the VolantPCI register space. Multiple PIOCFG's are located in a single 32-bit word. They may be accessed individually (by byte) or in any combination of bytes. The general naming convention is PIOCFGx_y, where x is the port number and y specifies which pin of the port.

6.2.1.1 Register Addresses

PIO Port	Pin 0	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7
0	+D100h	+D101h	+D102h	+D103h	+D104h	+D105h	+D106h	+D107h
1	+D110h	+D111h	+D112h	+D113h	+D114h	+D115h	+D116h	+D117h
2	+D120h	+D121h	+D122h	+D123h	+D124h	+D125h	+D126h	+D127h
3	+D130h	+D131h	+D132h	+D133h	+D134h	+D135h	+D136h	+D137h

6.2.1.2 Register Format



6.2.1.3 Bit Descriptions

- Bit 7. **Output mode** (pin 4 ONLY)
  - 0 = normal mode (output is based on value written in PIOSTAT)
  - 1 = clock mode (output is divided version of REFCLK, see bits 5-6)
- Bit 6-5. **Clock mode divider** (pin 4 ONLY)
  - 00 = in clock mode, output is REFCLK ÷ 2
  - 01 = in clock mode, output is REFCLK ÷ 4
  - 10 = in clock mode, output is REFCLK ÷ 6
  - 11 = in clock mode, output is REFCLK ÷ 8
- Bit 4-3. **Reserved**
- Bit 2. **PIO Interrupt Enable**
  - 0 = interrupt is disabled
  - 1 = interrupt is generated when a state change to the pin is detected

  Note: This bit is only available on pins 0-3.
- Bit 1. **PIO Polarity**
  - 0 = input or output is not inverted
  - 1 = input or output is inverted
- Bit 0. **PIO I/O**

- 0 = this pin functions as an input
- 1 = this pin functions as an output

Note: Only pins 0-3 can function as inputs. For pins 4-7, '0' means the output is tri-stated.

Reset Conditions

```
RST#:      0000 0000 0000 0000 0000 0000 0000 0000
```

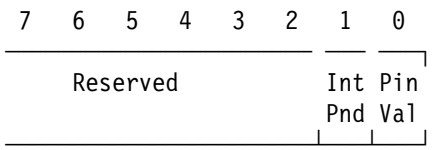
6.2.2 PIO Status Registers (PIOSTAT)

The PIOSTAT registers contain status information for each PIO and also allow the state of the output to be set. Each PIOSTAT is a byte register located in the VolantPCI register space. Multiple PIOSTAT's are located in a single 32-bit word. They may be accessed individually (by byte) or in any combination of bytes. The general naming convention is PIOSTATx_y, where x is the port number and y specifies which pin of the port.

6.2.2.1 Register Addresses

PIO Port	Pin 0	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7
0	+D200h	+D201h	+D202h	+D203h	+D204h	+D205h	+D206h	+D207h
1	+D210h	+D211h	+D212h	+D213h	+D214h	+D215h	+D216h	+D217h
2	+D220h	+D221h	+D222h	+D223h	+D224h	+D225h	+D226h	+D227h
3	+D230h	+D231h	+D232h	+D233h	+D234h	+D235h	+D236h	+D237h

6.2.2.2 Register Format



6.2.2.3 Bit Descriptions

- Bit 7-2. **Reserved**
- Bit 1. **PIO Interrupt Pending**
  - 0 = no interrupt pending
  - 1 = interrupt pending (clears on read)

  Note: This bit is only available on pins 0-3.
- Bit 0. **PIO pin value**

  This bit allows the state of the PIO pin to be read or written. For pins 0-3, the actual value of the pin is indicated (adjusted for polarity). For pin 4-7, the value comes directly from the register.

Reset Conditions

```
RST#:      0000 0000 0000 0000 0000 0000 0000 000U (pins 0-3)
RST#:      0000 0000 0000 0000 0000 0000 0000 0000 (pins 4-7)
```

6.2.3 Serial EPROM Register (SER)

This register is used to access the Serial EPROM. Usage generally is as follows:

1. Write SER with 0x01300000 to enable write.
2. Read SER until Busy bit (Bit 24) is clear.
3. Write SER with start bit, command, address and data (for write commands). This can be a single 32-bit write.
4. Read SER until Busy bit (Bit 24) is clear.
5. If detect bit is clear (Bit 25) then no SEPROM was detected and command did not complete. Data will be garbage.
6. If detect bit is set then command is complete. For reads, the data is in Bits 15-0.

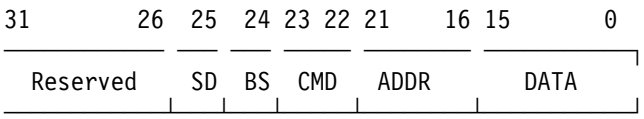
The SEPROM should be programmed as follows:

Address	Data
0	reserved
1	SSID low 16 bits
2	SSID high 16 bits
3	LGIR 31:24, b'000000', LGIR 17:16
4-7	reserved
>=8	user area

6.2.3.1 Register Address

Address Offset
+0F100h

6.2.3.2 Register Format



6.2.3.3 Bit Descriptions

- **Bits 31-26:** Reserved.
- **Bit 25:** Serial EPROM Detect. This read-only bit indicates whether VolantPCI detected ('1') or did not detect ('0') a serial EPROM connected. This bit is only valid when Busy/Start (bit 24) is clear ('0').
- **Bit 24:** Busy/Start. This read/write bit serves two functions. When written to a '1', VolantPCI will start the specified transaction. The bit will read back '1' while VolantPCI is busy performing the transaction. During this time, writes to the SER are discarded. The bit reads '0' when the transaction is complete and another transaction can be initiated.
- **Bits 23-22:** Serial EPROM Command. These bits contain the serial EPROM command. The following are the commands for the SGS ST93C06:
  - 10 - Read
  - 01 - Write
  - 11 - Erase Location
  - 00 - Command dependent on A5-4 as:
    - 00 - Erase/Write disable
    - 11 - Erase/Write enable
    - 01 - Write all with data
    - 10 - Erase all
- **Bits 21-16:** Serial EPROM Address. These bits contain the serial EPROM address A5-0.

- **Bits 15-0:** Serial EPROM Data. These bits contain the serial EPROM data. Data for writes should be written here. Data returned from reads will be valid here when the Busy bit (bit 24) is clear ('0').

```
RST# :       0000 00UU UUUU UUUU UUUU UUUU UUUU UUUU
```

Appendix A. VolantPCI Pin Name/Number Cross Reference

#	Name	#	Name	#	Name	#	Name
001	nc	051	AD(13)	101	vdd	151	PIO0(5)
002	vss	052	nc	102	-AINTACK(1)	152	PIO0(6)
003	AD(31)	053	vss	103	-AINTACK(0)	153	PIO0(7)
004	AD(30)	054	AD(12)	104	-AEOPTC	154	PIO1(0)
005	AD(29)	055	AD(11)	105	-ACS(0)	155	PIO1(1)
006	vss	056	AD(10)	106	-ACS(1)	156	PIO1(2)
007	vdd	057	vss	107	-ACS(2)	157	PIO1(3)
008	AD(28)	058	vdd	108	-ACS(3)	158	PIO1(4)
009	AD(27)	059	AD(9)	109	vss	159	PIO1(5)
010	AD(26)	060	AD(8)	110	A_D(0)	160	PIO1(6)
011	vss	061	CBE(0)#	111	A_D(1)	161	PIO1(7)
012	AD(25)	062	AD(7)	112	A_D(2)	162	PIO2(0)
013	AD(24)	063	AD(6)	113	A_D(3)	163	PIO2(1)
014	CBE(3)#	064	AD(5)	114	vss	164	PIO2(2)
015	vss	065	vss	115	vdd	165	PIO2(3)
016	vdd	066	vdd	116	A_D(4)	166	vss
017	IDSEL	067	AD(4)	117	A_D(5)	167	vdd
018	AD(23)	068	AD(3)	118	A_D(6)	168	PIO2(4)
019	AD(22)	069	AD(2)	119	A_D(7)	169	PIO2(5)
020	vss	070	AD(1)	120	-AWR	170	PIO2(6)
021	AD(21)	071	AD(0)	121	-ARD	171	PIO2(7)
022	AD(20)	072	vss	122	A_A(2)	172	PIO3(0)
023	AD(19)	073	vdd	123	A_A(3)	173	PIO3(1)
024	vss	074	MMI	124	A_A(4)	174	PIO3(2)
025	vdd	075	MMO	125	A_A(5)	175	PIO3(3)
026	AD(18)	076	SECS	126	A_A(6)	176	PIO3(4)
027	AD(17)	077	SECLK	127	A_A(7)	177	PIO3(5)
028	AD(16)	078	SED	128	A_A(8)	178	PIO3(6)
029	vssc	079	-ADREQ(7)	129	A_A(9)	179	PIO3(7)
030	vddc	080	-ADACK(7)	130	vss	180	-LED_EN
031	vss	081	-ADREQ(6)	131	vdd	181	-ARESET
032	CBE(2)#	082	-ADACK(6)	132	A_A(10)	182	PTSTOUT
033	FRAME#	083	vdd	133	A_A(11)	183	vssc
034	IRDY#	084	vss	134	A_A(12)	184	vddc
035	vss	085	-ADREQ(5)	135	A_A(13)	185	IDDTST
036	vdd	086	-ADACK(5)	136	A_A(14)	186	ScanTstEn#
037	CLK	087	-ADREQ(4)	137	A_A(15)	187	ScanMuxSel#
038	vss	088	-ADACK(4)	138	A_A(16)	188	vss
039	TRDY#	089	vssc	139	A_A(17)	189	ACLK
040	DEVSEL#	090	vddc	140	vddc	190	vss
041	STOP#	091	-ADREQ(3)	141	vssc	191	vdd
042	vss	092	-ADACK(3)	142	REFCLK	192	JTAG-TD0
043	PERR#	093	-ADREQ(2)	143	vss	193	JTAG-MOD
044	SERR#	094	-ADACK(2)	144	PIO0(0)	194	JTAG-RST
045	PAR	095	vss	145	PIO0(1)	195	JTAG-TCK
046	vss	096	-ADREQ(1)	146	PIO0(2)	196	JTAG-TDI
047	vdd	097	-ADACK(1)	147	PIO0(3)	197	vss
048	CBE(1)#	098	-ADREQ(0)	148	vss	198	-AERROR
049	AD(15)	099	-ADACK(0)	149	vdd	199	-AINT(0)
050	AD(14)	100	vss	150	PIO0(4)	200	-AINT(1)

#	Name
201	-AINT(2)
202	-AINT(3)
203	INTA#
204	vdd
205	vss
206	RST#
207	GNT#
208	REQ#

Name	#	Name	#	Name	#	Name	#
ACLK	189	-AINTACK(1)	102	PIO1(2)	156	vss	020
-ACS(0)	105	-ARD	121	PIO1(3)	157	vss	024
-ACS(1)	106	-ARESET	181	PIO1(4)	158	vss	031
-ACS(2)	107	-AWR	120	PIO1(5)	159	vss	035
-ACS(3)	108	A_A(10)	132	PIO1(6)	160	vss	038
AD(0)	071	A_A(11)	133	PIO1(7)	161	vss	042
AD(1)	070	A_A(12)	134	PIO2(0)	162	vss	046
AD(10)	056	A_A(13)	135	PIO2(1)	163	vss	053
AD(11)	055	A_A(14)	136	PIO2(2)	164	vss	057
AD(12)	054	A_A(15)	137	PIO2(3)	165	vss	065
AD(13)	051	A_A(16)	138	PIO2(4)	168	vss	072
AD(14)	050	A_A(17)	139	PIO2(5)	169	vss	084
AD(15)	049	A_A(2)	122	PIO2(6)	170	vss	095
AD(16)	028	A_A(3)	123	PIO2(7)	171	vss	100
AD(17)	027	A_A(4)	124	PIO3(0)	172	vss	109
AD(18)	026	A_A(5)	125	PIO3(1)	173	vss	114
AD(19)	023	A_A(6)	126	PIO3(2)	174	vss	130
AD(2)	069	A_A(7)	127	PIO3(3)	175	vss	143
AD(20)	022	A_A(8)	128	PIO3(4)	176	vss	148
AD(21)	021	A_A(9)	129	PIO3(5)	177	vss	166
AD(22)	019	A_D(0)	110	PIO3(6)	178	vss	188
AD(23)	018	A_D(1)	111	PIO3(7)	179	vss	190
AD(24)	013	A_D(2)	112	PTSTOUT	182	vss	197
AD(25)	012	A_D(3)	113	REFCLK	142	vss	205
AD(26)	010	A_D(4)	116	REQ#	208	vssc	029
AD(27)	009	A_D(5)	117	RST#	206	vssc	089
AD(28)	008	A_D(6)	118	SECLK	077	vssc	141
AD(29)	005	A_D(7)	119	SECS	076	vssc	183
AD(3)	068	CBE(0)#	061	SED	078		
AD(30)	004	CBE(1)#	048	SERR#	044		
AD(31)	003	CBE(2)#	032	STOP#	041		
AD(4)	067	CBE(3)#	014	ScanMuxSel#	187		
AD(5)	064	CLK	037	ScanTstEn#	186		
AD(6)	063	DEVSEL#	040	TRDY#	039		
AD(7)	062	FRAME#	033	nc	001		
AD(8)	060	GNT#	207	nc	052		
AD(9)	059	IDDTST	185	vdd	007		
-ADACK(0)	099	IDSEL	017	vdd	016		
-ADACK(1)	097	INTA#	203	vdd	025		
-ADACK(2)	094	IRDY#	034	vdd	036		
-ADACK(3)	092	JTAG-MOD	193	vdd	047		
-ADACK(4)	088	JTAG-RST	194	vdd	058		
-ADACK(5)	086	JTAG-TCK	195	vdd	066		
-ADACK(6)	082	JTAG-TD0	192	vdd	073		
-ADACK(7)	080	JTAG-TDI	196	vdd	083		
-ADREQ(0)	098	-LED_EN	180	vdd	101		
-ADREQ(1)	096	MMI	074	vdd	115		
-ADREQ(2)	093	MMO	075	vdd	131		
-ADREQ(3)	091	PAR	045	vdd	149		
-ADREQ(4)	087	PERR#	043	vdd	167		
-ADREQ(5)	085	PIO0(0)	144	vdd	191		
-ADREQ(6)	081	PIO0(1)	145	vdd	204		
-ADREQ(7)	079	PIO0(2)	146	vddc	030		
-AEOPTC	104	PIO0(3)	147	vddc	090		
-AERROR	198	PIO0(4)	150	vddc	140		
-AINT(0)	199	PIO0(5)	151	vddc	184		
-AINT(1)	200	PIO0(6)	152	vss	002		
-AINT(2)	201	PIO0(7)	153	vss	006		
-AINT(3)	202	PIO1(0)	154	vss	011		
-AINTACK(0)	103	PIO1(1)	155	vss	015		

Appendix B. PIO Functional Diagrams

The following diagrams are simplified, functional level drawings of the PIO circuitry.

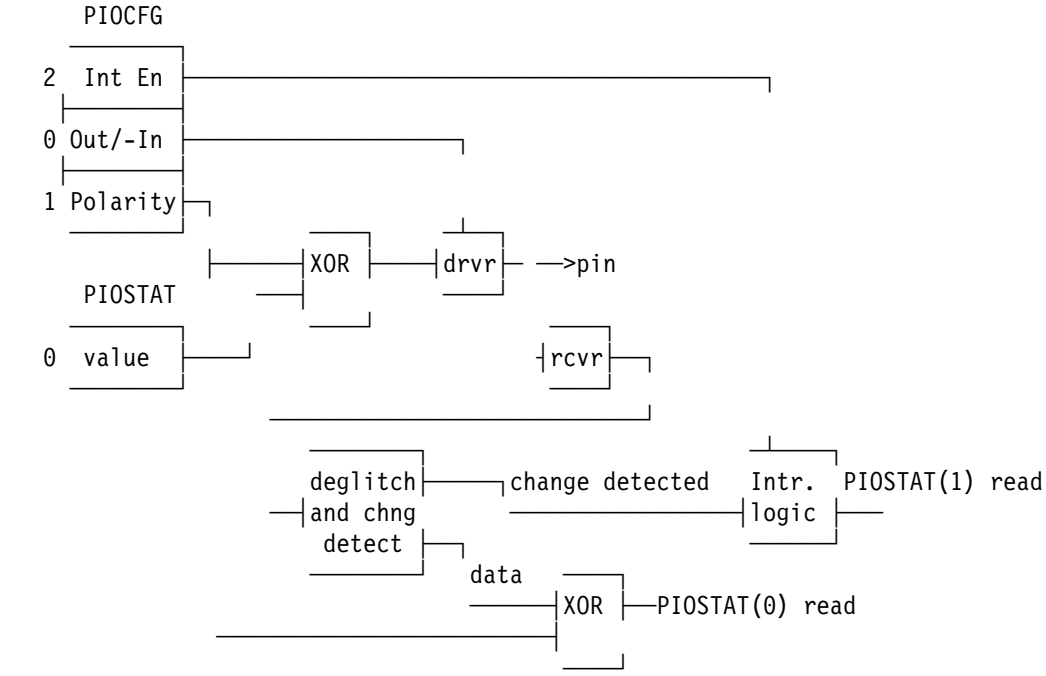


Figure 6. Input/Output PIO

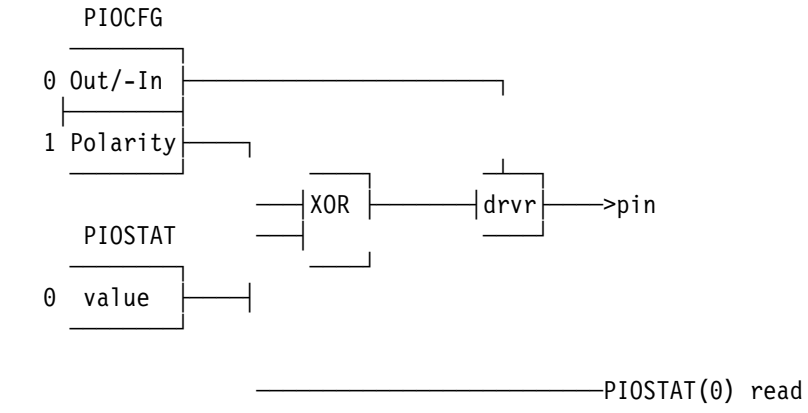


Figure 7. Output-Only PIO

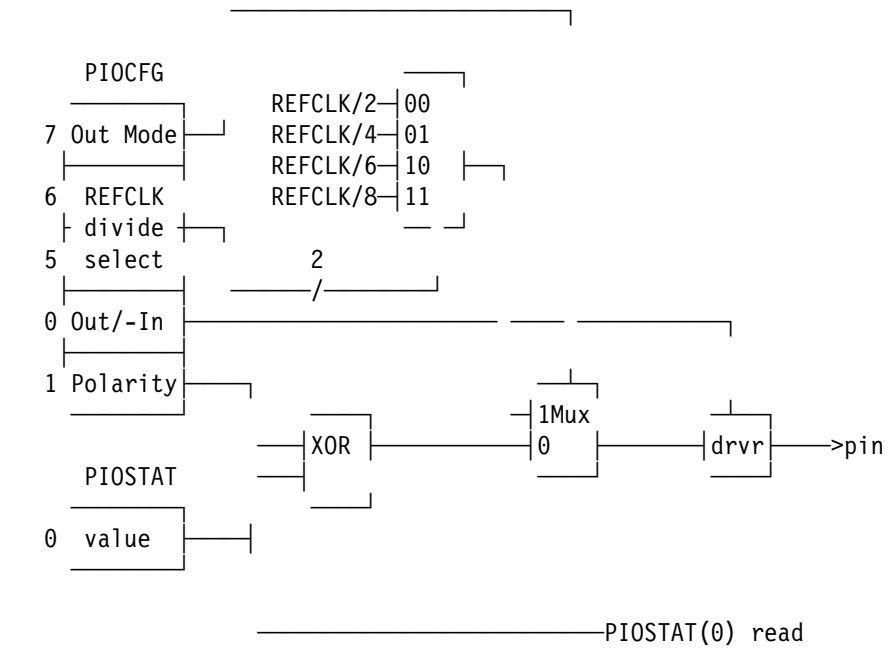


Figure 8. Output-Only w/clock PIO

Appendix C. VolantPCI Electrical Specifications

Note: ALL INFORMATION PROVIDED IN THE FOLLOWING SECTIONS IS PRELIMINARY.

C.1 Absolute Maximum Ratings

Table 14. VolantPCI Maximum Ratings (referenced to Vss)

Parameter	**Maximum Rating**
Storage Temperature	-40°C to +125°C
Case Temperature Under Bias	-40°C to +110°C
Supply Voltage	-0.3v to +3.9v
Voltage on ScanMuxSel, ScanTestEn	-1.0v to 3.6v
Voltage on Other Pins	-1.0v to 6.5v

Note: Except for ScanMuxSel and ScanTestEn, all VolantPCI inputs are "5 volt tolerant".

C.2 Operating Conditions

Table 15. VolantPCI Operating Conditions

Parameter	**Min**	**Max**	**Units**
Supply Voltage	3	3.6	V
Case Temperature Under Bias	0	75	°C
Input Clock Freq CLK	0	33	MHz
Input Clock Freq REFCLK	0	25	MHz
Supply Current	-	150mA	mA

C.3 Recommended Connections

Consult the PCI Bus specification V2.1 for detailed information regarding the PCI bus signals.

C.3.1 Decoupling

The following is recommended **minimum** VDD to VSS decoupling of the VolantPCI chip:

- 1 - 1000pF Capacitor near the clock input pins for CLK and REFCLK.
- 5 - 0.01µF Capacitors. One cap. near each side of the chip (two on PCI side).
- 2 - 22µF Capacitor for bulk decoupling. One cap on each side of the chip.

C.4 Specifications for the PCI Bus Interface

VolantPCI meets PCI Local Bus Specification, Rev. 2.1 requirements for both 3.3v and 5v signalling environments. VolantPCI's PCI inputs are "5 volt tolerant".

More detailed information can be supplied upon request.

C.5 DC Specifications

C.5.1 DC Specifications for the AIB Bus and Misc. Signals

DC parameters include miscellaneous signals such as JTAG and Serial ROM

Table 16. VolantPCI AIB Bus and Misc. Signals

Symbol	Description	Min	Max	Units	Notes
Vil	Input Low Voltage	Vss-0.5	0.8	v	
Vih	Input High Voltage	2.0	VDD+0.3	v	
Vol	Output Low Voltage	-	0.40	v	Iol= 4.0 mA
Voh	Output High Voltage	2.4	Vdd	v	Ioh= -4.0 mA
Ci	Input and Bidirectional Pin Capacitance	3	8	pF	
Co	Output Pin Capacitance	3	8	pF	

C.6 AC Timing Specifications

The following sections provide the AC timing specifications for the VolantPCI chip.

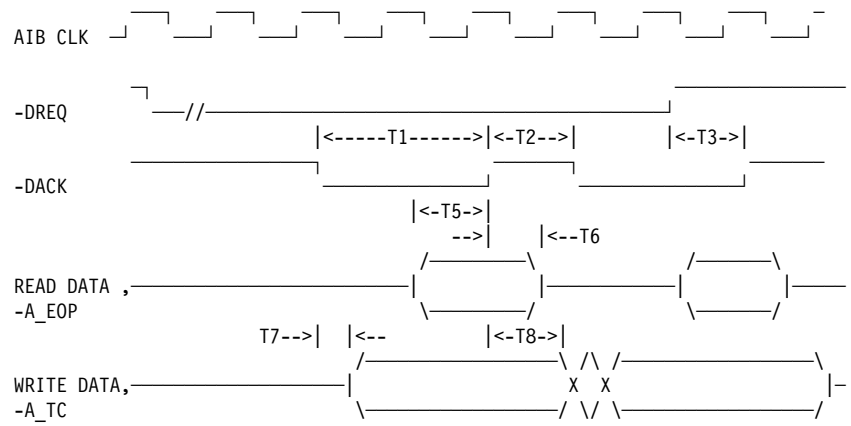
C.6.1 PCI Bus Timings

VolantPCI meets the PCI Bus timings for a 33MHz device as specified in the PCI Local Bus Specification, Rev. 2.1. The user is referred to this document for more information.

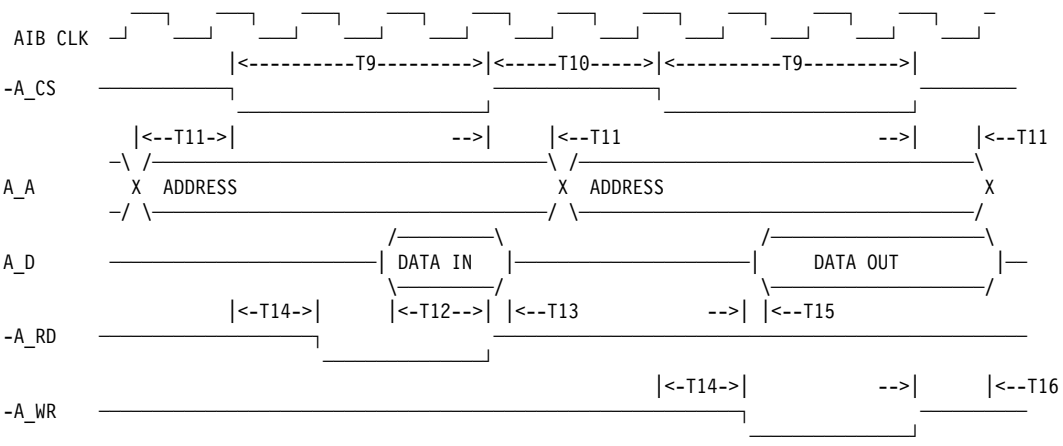
C.6.2 AIB Bus Timings

The following timing diagrams show AIB Bus timings for the various types of cycles that run on the AIB interface. All timings are specified with 50pf load capacitance.

C.6.2.1 DMA Cycles



C.6.2.2 AIB Target Cycles



C.6.2.3 AIB Interrupt Acknowledge Cycle

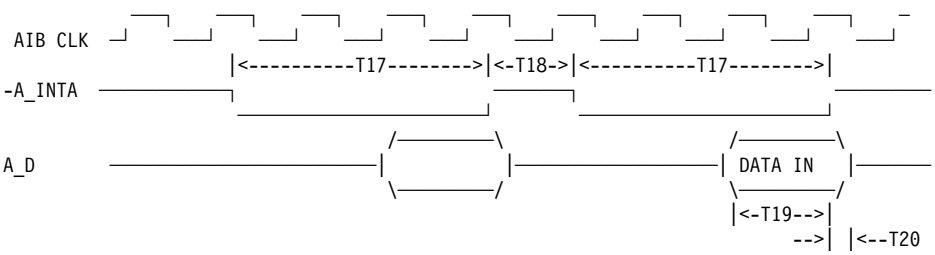


Table 17. Timings

	Description	Min	Max	Units	Notes
T1	-DACK active pulse width	da*clk-5	da*clk+5	ns	1,2
T2	-DACK inactive pulse width	di*clk-5	di*clk+5	ns	1,2
T3	-DREQ inactive to -DACK inactive	30	-	ns	3
T5	read data setup to -DACK inactive	20	-	ns	
T5	-A_EOP setup to -DACK inactive	25	-	ns	
T6	input hold time from -DACK inactive	0	di*clk	ns	1,2,4
T7	-DACK active to outputs valid	0	10	ns	
T8	outputs hold from -DACK inactive	clk-7	clk	ns	1
T9	-A_CS active pulse width	ca*clk-5	ca*clk-5	ns	1,5
T10	-A_CS inactive pulse width	ci*clk-5	ci*clk-5	ns	1,5
T11	A_A valid to -A_CS active	clk-5	clk+5	ns	1
T12	A_D setup to -A_RD inactive	20	-	ns	
T13	A_D hold from -A_RD inactive	0	ci*clk	ns	1,4,5
T14	-A_CS active to -A_RD or -A_WR active	clk-5	clk+5	ns	1
T15	A_D valid from -A_WR active	-	5	ns	
T16	A_D hold from -A_WR inactive	clk-10	-	ns	1
T17	-A_INTA active pulse width	ia*clk-5	ia*clk-5	ns	1,6
T18	-A_INTA inactive pulse width	ii*clk-5	ii*clk-5	ns	1,6
T19	A_D setup to -A_INTA inactive	20	-	ns	
T20	A_D hold from -A_INTA inactive	0	ii*clk	ns	1,4,6

Notes:

1) 'clk' represents the period of the input CLK.
2) 'da' and 'di' represent the programmed active and inactive DACK pulse widths, respectively.
3) This timing must be met to ensure VolantPCI does not begin another DACK cycle for this DREQ.
4) The max timing ensures there is no driver conflict between this read and a subsequent write by VolantPCI.
5) 'ca' and 'ci' represent the programmed active and inactive Chip Select pulse widths, respectively.
6) 'ia' and 'ii' represent the programmed active and inactive Interrupt Acknowledge pulse widths, respectively.

C.6.3 VolantPCI Serial EPROM Interface Timing

This section documents the Serial EPROM Interface Timing.

Note: Although the serial EPROMs provide separate Data In (DI) and Data Out (DO) pins, VolantPCI uses a single bi-directional pin, that is tri-stated for data in. A pull-up holds the value at '1'. This makes it possible to use a single pin for data without driver conflicts, and allows VolantPCI to automatically detect the presence (or absence) of a SEPROM by checking the dummy bit that is returned by the SEPROM for each read access.

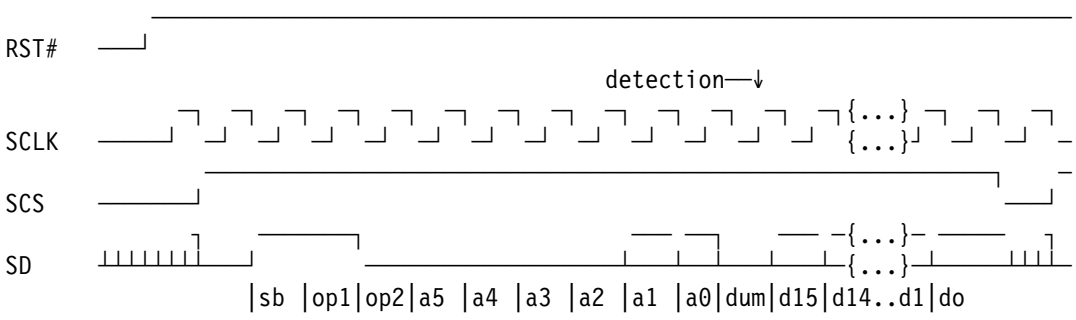


Figure 9. SEPROM read timing

C.6.3.1 SEPROM Auto-Initialization Description

- When RST# is asserted: SCLK stays LOW, SCS stays LOW, SD will float HIGH (it is pulled-up).
- About 20 clocks after RST# is deasserted, SCLK starts (see Figure 9).
- SCS goes HIGH. SD is driven LOW for one SCLK for compatibility with SGS-Thomson parts.
- SD is driven HIGH. This is the 'start' bit (sb).
- The next two SCLKs contain the opcode, driven by VolantPCI which is always b'10' (read command).
- The next six SCLKs are the address bits, driven by VolantPCI. The first address is b'000001'.
- The next bit is the 'dummy' bit which is a '0' output by the SEPROM. If the '0' is detected then the next 16 cycles are data read cycles. If the '0' is not detected, then it is assumed no SEPROM is installed, SCLK goes LOW, SCS goes low and the state machine stops. The SER will indicate no SEPROM was found.
- After the sixteenth data bit is read SCS goes low for one SCLK.
- VolantPCI performs four read operations. After the fourth read, SCLK goes LOW, SCS goes low.

C.6.3.2 Other SEPROM Operations

Refer to the the SER register (6.2.3, “Serial EPROM Register (SER)” on page 74) and the specific SEPROM's documentation for more information on SEPROM programming.

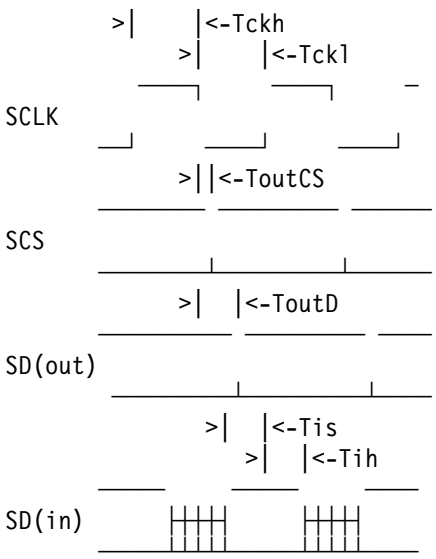


Figure 10. SCLK, SCS, SD timing

Table 18. VolantPCI Serial EPROM Interface Timings

Symbol	Description	Min (ns)	Max (ns)	Notes
Tckh	Clock High	17*n-10	17*n+10	1,2
Tckl	Clock Low	17*n-10	17*n+10	1,2
ToutCS	SCLK low to SCS output	-20	20	
ToutD	SCLK low to SD output	n-20	n+20	1,2
Tis	input setup	20		
Tih	input hold	0		

Notes:

1. n = PCI clock period
2. SCLK(freq) = PCI Clock(freq) ÷ 34

Appendix D. VolantPCI Test Information

Using multiplexed and dedicated test pins and a JTAG Tap Controller, VolantPCI provides the following testability features for both on and off-card component testing:

- Full Chip Internal Scan Testing
- Internal Ram Isolation
- Driver Tri-State
- Boundary Scan

Although some of these features are used only by the chip manufacturer, they are shown here for documentation purposes and general information.

D.1 JTAG TAP Controller Features

VolantPCI provides a JTAG compliant tap controller and JTAG tap interface. The chip has full JTAG boundary scan implementation. The JTAG ring has 268 total elements in the chain. The following table lists the JTAG Interface Pins.

Table 19. VolantPCI JTAG Interface

JTAG PIN	**VolantPCI Pin #**
JTAG-TCK	195
JTAG-MOD	193
JTAG-RST#	194
JTAG-TDO	192
JTAG-TDI	196

The following table lists the supported JTAG instructions and their respective instruction codes.

Table 20. VolantPCI JTAG Interface

JTAG Instruction	**Code**
EXTEST	000
SAMPLE/PRELOAD	001
CLAMP	011
HIGHZ	100
RamTest (internal Ram)	101
BYPASS	111

D.1.1 Boundary Scan

The following is the BSDL description of VolantPCI.

```
--note: signals with names of the type xxxx# or -xxxx
--      have been changed to n_xxxx in this file
--      also the PIO pins have added "High" or "Low" suffixes
entity vpcichip is

generic (PHYSICAL_PIN_MAP : string := "LSI_PACKAGE");

port (
        n_aERROR        : in          bit         ;
        n_aINT          : in          bit_vector  (0 to 3);
        N_IntA          : inout       bit         ;
        N_RST           : in          bit         ;
        n_GNT           : in          bit         ;
        n_REQ           : inout       bit         ;
        IDSEL           : in          bit         ;
        n_FRAME         : inout       bit         ;
        n_IRDY          : inout       bit         ;
        CLK             : in          bit         ;
        n_TRDY          : inout       bit         ;
        n_DEVSEL        : inout       bit         ;
        n_STOP          : inout       bit         ;
        n_PERR          : inout       bit         ;
        n_SERR          : inout       bit         ;
        PAR             : inout       bit         ;
        n_CBE           : inout       bit_vector  (0 to 3);
        AD              : inout       bit_vector  (0 to 31);
        MMI             : in          bit         ;
        MMO             : buffer      bit         ;
        SECS            : buffer      bit         ;
        SECLK           : buffer      bit         ;
        SED             : inout       bit         ;
        n_aDREQ         : in          bit_vector  (0 to 7);
        n_aDACK         : buffer      bit_vector  (0 to 7);
        n_aINTACK       : buffer      bit_vector  (0 to 1);
        n_aEOPTC        : inout       bit         ;
        n_aCS           : buffer      bit_vector  (0 to 3);
        a_D             : inout       bit_vector  (0 to 7);
        n_aWR           : buffer      bit         ;
        n_aRD           : buffer      bit         ;
        a_A             : buffer      bit_vector  (2 to 17);
        REFCLK          : in          bit         ;
        PIO0Low         : inout       bit_vector  (0 to 3);
        PIO0High        : out         bit_vector  (4 to 7);
        PIO1Low         : inout       bit_vector  (0 to 3);
        PIO1High        : out         bit_vector  (4 to 7);
        PIO2Low         : inout       bit_vector  (0 to 3);
        PIO2High        : out         bit_vector  (4 to 7);
        PIO3Low         : inout       bit_vector  (0 to 3);
        PIO3High        : out         bit_vector  (4 to 7);
        n_LED_EN        : buffer      bit         ;
        n_aRESET        : buffer      bit         ;
        PTSTOut         : out         bit         ;
        IDDTST          : in          bit         ;
        n_ScanTestEn    : in          bit         ;
        n_ScanMuxSel    : in          bit         ;
        aCLK            : out         bit         ;
        vss             : linkage     bit_vector  (0 to 31);
        vdd             : linkage     bit_vector  (0 to 19);
        jtag_tdo        : out         bit         ;
        jtag_mod        : in          bit         ;
        jtag_rst        : in          bit         ;
        jtag_tck        : in          bit         ;
        jtag_tdi        : in          bit
        ) ;

use STD_1149_1_1990.all;

attribute PIN_MAP of vpcichip : entity is PHYSICAL_PIN_MAP ;

constant pr36 : PIN_MAP_STRING :=

        "jtag_tdi : 196, jtag_tck : 195, jtag_rst : 194, "  &
        "jtag_mod : 193, jtag_tdo : 192, vdd : ( 7, 16, 25, 30, 36, 47, 58, 66, 73, "  &
        "83, 90, 101, 115, 131, 140, 149, 167, 184, 191, 204 )),"  &
        "vss : ( 2, 6, 11, 15, 20, 24, 29, 31, 35, "  &
        "38, 42, 46, 53, 57, 65, 72, 84, 89, 95, "  &
        "100, 109, 114, 130, 141, 143, 148, 166, 183, 188, "  &
        "190, 197, 205 ),aCLK : 189, n_ScanMuxSel : 187, "  &
        "n_ScanTestEn : 186, IDDTST : 185, PTSTOut : 182, "  &
        "n_aRESET : 181, n_LED_EN : 180, PIO3High : ( 176, 177, 178, 179 ),"  &
        "PIO3Low : ( 172, 173, 174, 175 ),PIO2High : ( 168, 169, 170, 171 ),PIO2Low : ( 162, 163, 164, 165 ),"  &
        "PIO1High : ( 158, 159, 160, 161 ),PIO1Low : ( 154, 155, 156, 157 ),PIO0High : ( 150, 151, 152, 153 ),"  &
        "PIO0Low : ( 144, 145, 146, 147 ),REFCLK : 142, a_A : ( 122, 123, 124, 125, 126, 127, 128, 129, 132, "  &
```

```
        "133, 134, 135, 136, 137, 138, 139 )," &
        "n_aRD : 121, n_aWR : 120, a_D : ( 110, 111, 112, 113, 116, 117, 118, 119 )," &
        "n_aCS : ( 105, 106, 107, 108 ),n_aEOPTC : 104, n_aINTACK : ( 103, 102 )," &
        "n_aDACK : ( 99, 97, 94, 92, 88, 86, 82, 80 ),n_aDREQ : ( 98, 96, 93, 91, 87, 85, 81, 79 ),SED : 78, " &
        "SECLK : 77, SECS : 76, MMO : 75, " &
        "MMI : 74, AD : ( 71, 70, 69, 68, 67, 64, 63, 62, 60, " &
        "59, 56, 55, 54, 51, 50, 49, 28, 27, 26, " &
        "23, 22, 21, 19, 18, 13, 12, 10, 9, 8, " &
        "5, 4, 3 ),n_CBE : ( 61, 48, 32, 14 )," &
        "PAR : 45, n_SERR : 44, n_PERR : 43, " &
        "n_STOP : 41, n_DEVSEL : 40, n_TRDY : 39, " &
        "CLK : 37, n_IRDY : 34, n_FRAME : 33, " &
        "IDSEL : 17, n_REQ : 208, n_GNT : 207, " &
        "N_RST : 206, N_IntA : 203, n_aINT : ( 199, 200, 201, 202 )," &
        "n_aERROR : 198 ";

attribute TAP_SCAN_IN of jtag_tdi : signal is true;

attribute TAP_SCAN_OUT of jtag_tdo : signal is true;

attribute TAP_SCAN_MODE of jtag_mod : signal is true;

attribute TAP_SCAN_RESET of jtag_rst : signal is true;

attribute TAP_SCAN_CLOCK of jtag_tck : signal is ( 1.200000e+07, BOTH );


attribute INSTRUCTION_LENGTH of vpcichip : entity is 3;

attribute INSTRUCTION_OPCODE of vpcichip : entity is

        "SAMPLE (001)," &
        "BYPASS (111)," &
        "EXTEST (000)," &
        "highz (100)," &
        "clamp (011)," &
        "ramtst (101)" ;

attribute INSTRUCTION_CAPTURE of vpcichip : entity is "001";

attribute INSTRUCTION_DISABLE of vpcichip : entity is "HIGHZ";

attribute INSTRUCTION_GUARD   of vpcichip : entity is "CLAMP";

attribute INSTRUCTION_PRIVATE of vpcichip : entity is "ramtst";



attribute REGISTER_ACCESS of vpcichip : entity is

"BOUNDARY (SAMPLE, EXTEST)," &
"BYPASS (CLAMP, HIGHZ, BYPASS)" ;

attribute BOUNDARY_CELLS of vpcichip : entity is "BC_2, BC_4, BC_1";
attribute BOUNDARY_LENGTH of vpcichip : entity is 268;

attribute BOUNDARY_REGISTER of vpcichip : entity is

        -- num   cell  port  function safe ffccell disval rslt“

        "0 ( BC_1, *, controlr, 1 ) ," &
        "1 ( BC_1, *, controlr, 1 ) ," &
        "2 ( BC_1, *, controlr, 1 ) ," &
        "3 ( BC_1, *, controlr, 1 ) ," &
        "4 ( BC_1, *, controlr, 1 ) ," &
        "5 ( BC_1, *, controlr, 1 ) ," &
        "6 ( BC_1, *, controlr, 1 ) ," &
        "7 ( BC_1, *, controlr, 1 ) ," &
        "8 ( BC_1, *, controlr, 1 ) ," &
        "9 ( BC_1, *, controlr, 1 ) ," &
        "10 ( BC_1, *, controlr, 1 ) ," &
        "11 ( BC_1, *, controlr, 1 ) ," &
        "12 ( BC_1, *, controlr, 1 ) ," &
        "13 ( BC_1, *, controlr, 1 ) ," &
        "14 ( BC_1, *, controlr, 1 ) ," &
        "15 ( BC_1, *, controlr, 1 ) ," &
        "16 ( BC_1, *, controlr, 1 ) ," &
        "17 ( BC_1, *, controlr, 1 ) ," &
        "18 ( BC_1, *, controlr, 1 ) ," &
        "19 ( BC_1, *, controlr, 1 ) ," &
        "20 ( BC_1, *, controlr, 1 ) ," &
        "21 ( BC_1, *, controlr, 1 ) ," &
        "22 ( BC_1, *, controlr, 1 ) ," &
```

```
"23 ( BC_1, *, controlr, 1 ) ,"  &
"24 ( BC_1, *, controlr, 1 ) ,"  &
"25 ( BC_1, *, controlr, 1 ) ,"  &
"26 ( BC_1, *, controlr, 1 ) ,"  &
"27 ( BC_1, *, controlr, 1 ) ,"  &
"28 ( BC_1, *, controlr, 1 ) ,"  &
"29 ( BC_1, *, controlr, 1 ) ,"  &
"30 ( BC_1, *, controlr, 1 ) ,"  &
"31 ( BC_1, *, controlr, 1 ) ,"  &
"32 ( BC_1, *, controlr, 1 ) ,"  &
"33 ( BC_1, *, controlr, 1 ) ,"  &
"34 ( BC_1, *, controlr, 1 ) ,"  &
"35 ( BC_1, *, controlr, 1 ) ,"  &
"36 ( BC_1, *, controlr, 1 ) ,"  &
"37 ( BC_1, *, controlr, 1 ) ,"  &
"38 ( BC_1, *, controlr, 1 ) ,"  &
"39 ( BC_1, *, controlr, 1 ) ,"  &
"40 ( BC_1, *, controlr, 1 ) ,"  &
"41 ( BC_1, *, controlr, 1 ) ,"  &
"42 ( BC_1, *, controlr, 1 ) ,"  &
"43 ( BC_1, *, controlr, 1 ) ,"  &
"44 ( BC_1, *, controlr, 1 ) ,"  &
"45 ( BC_1, *, controlr, 1 ) ,"  &
"46 ( BC_1, *, controlr, 1 ) ,"  &
"47 ( BC_1, *, controlr, 1 ) ,"  &
"48 ( BC_1, *, controlr, 1 ) ,"  &
"49 ( BC_4, n_aERROR, clock, X ) ,"  &
"50 ( BC_4, n_aINT(0), clock, X ) ,"  &
"51 ( BC_4, n_aINT(1), clock, X ) ,"  &
"52 ( BC_4, n_aINT(2), clock, X ) ,"  &
"53 ( BC_4, n_aINT(3), clock, X ) ,"  &
"54 ( BC_1, N_IntA, output3, X , 47, 1, Z),"  &
"55 ( BC_2, N_IntA, input, X ) ,"  &
"56 ( BC_4, N_RST, clock, X ) ,"  &
"57 ( BC_4, n_GNT, clock, X ) ,"  &
"58 ( BC_1, n_REQ, output3, X , 44, 1, Z),"  &
"59 ( BC_2, n_REQ, input, X ) ,"  &
"60 ( BC_1, AD(31), output3, X , 4, 1, Z),"  &
"61 ( BC_2, AD(31), input, X ) ,"  &
"62 ( BC_1, AD(30), output3, X , 4, 1, Z),"  &
"63 ( BC_2, AD(30), input, X ) ,"  &
"64 ( BC_1, AD(29), output3, X , 4, 1, Z),"  &
"65 ( BC_2, AD(29), input, X ) ,"  &
"66 ( BC_1, AD(28), output3, X , 4, 1, Z),"  &
"67 ( BC_2, AD(28), input, X ) ,"  &
"68 ( BC_1, AD(27), output3, X , 4, 1, Z),"  &
"69 ( BC_2, AD(27), input, X ) ,"  &
"70 ( BC_1, AD(26), output3, X , 4, 1, Z),"  &
"71 ( BC_2, AD(26), input, X ) ,"  &
"72 ( BC_1, AD(25), output3, X , 4, 1, Z),"  &
"73 ( BC_2, AD(25), input, X ) ,"  &
"74 ( BC_1, AD(24), output3, X , 4, 1, Z),"  &
"75 ( BC_2, AD(24), input, X ) ,"  &
"76 ( BC_1, n_CBE(3), output3, X , 5, 1, Z),"  &
"77 ( BC_2, n_CBE(3), input, X ) ,"  &
"78 ( BC_4, IDSEL, clock, X ) ,"  &
"79 ( BC_1, AD(23), output3, X , 3, 1, Z),"  &
"80 ( BC_2, AD(23), input, X ) ,"  &
"81 ( BC_1, AD(22), output3, X , 3, 1, Z),"  &
"82 ( BC_2, AD(22), input, X ) ,"  &
"83 ( BC_1, AD(21), output3, X , 3, 1, Z),"  &
"84 ( BC_2, AD(21), input, X ) ,"  &
"85 ( BC_1, AD(20), output3, X , 3, 1, Z),"  &
"86 ( BC_2, AD(20), input, X ) ,"  &
"87 ( BC_1, AD(19), output3, X , 3, 1, Z),"  &
"88 ( BC_2, AD(19), input, X ) ,"  &
"89 ( BC_1, AD(18), output3, X , 3, 1, Z),"  &
"90 ( BC_2, AD(18), input, X ) ,"  &
"91 ( BC_1, AD(17), output3, X , 3, 1, Z),"  &
"92 ( BC_2, AD(17), input, X ) ,"  &
"93 ( BC_1, AD(16), output3, X , 3, 1, Z),"  &
"94 ( BC_2, AD(16), input, X ) ,"  &
"95 ( BC_1, n_CBE(2), output3, X , 5, 1, Z),"  &
"96 ( BC_2, n_CBE(2), input, X ) ,"  &
"97 ( BC_1, n_FRAME, output3, X , 41, 1, Z),"  &
"98 ( BC_2, n_FRAME, input, X ) ,"  &
"99 ( BC_1, n_IRDY, output3, X , 42, 1, Z),"  &
"100 ( BC_2, n_IRDY, input, X ) ,"  &
"101 ( BC_4, CLK, clock, X ) ,"  &
"102 ( BC_1, n_TRDY, output3, X , 43, 1, Z),"  &
"103 ( BC_2, n_TRDY, input, X ) ,"  &
"104 ( BC_1, n_DEVSEL, output3, X , 43, 1, Z),"  &
"105 ( BC_2, n_DEVSEL, input, X ) ,"  &
```

```
"106 ( BC_1, n_STOP, output3, X , 43, 1, Z),"  &
"107 ( BC_2, n_STOP, input, X ) ,"  &
"108 ( BC_1, n_PERR, output3, X , 45, 1, Z),"  &
"109 ( BC_2, n_PERR, input, X ) ,"  &
"110 ( BC_1, n_SERR, output3, X , 46, 1, Z),"  &
"111 ( BC_2, n_SERR, input, X ) ,"  &
"112 ( BC_1, PAR, output3, X , 40, 1, Z),"  &
"113 ( BC_2, PAR, input, X ) ,"  &
"114 ( BC_1, n_CBE(1), output3, X , 5, 1, Z),"  &
"115 ( BC_2, n_CBE(1), input, X ) ,"  &
"116 ( BC_1, AD(15), output3, X , 2, 1, Z),"  &
"117 ( BC_2, AD(15), input, X ) ,"  &
"118 ( BC_1, AD(14), output3, X , 2, 1, Z),"  &
"119 ( BC_2, AD(14), input, X ) ,"  &
"120 ( BC_1, AD(13), output3, X , 2, 1, Z),"  &
"121 ( BC_2, AD(13), input, X ) ,"  &
"122 ( BC_1, AD(12), output3, X , 2, 1, Z),"  &
"123 ( BC_2, AD(12), input, X ) ,"  &
"124 ( BC_1, AD(11), output3, X , 2, 1, Z),"  &
"125 ( BC_2, AD(11), input, X ) ,"  &
"126 ( BC_1, AD(10), output3, X , 2, 1, Z),"  &
"127 ( BC_2, AD(10), input, X ) ,"  &
"128 ( BC_1, AD(9), output3, X , 2, 1, Z),"  &
"129 ( BC_2, AD(9), input, X ) ,"  &
"130 ( BC_1, AD(8), output3, X , 2, 1, Z),"  &
"131 ( BC_2, AD(8), input, X ) ,"  &
"132 ( BC_1, n_CBE(0), output3, X , 5, 1, Z),"  &
"133 ( BC_2, n_CBE(0), input, X ) ,"  &
"134 ( BC_1, AD(7), output3, X , 1, 1, Z),"  &
"135 ( BC_2, AD(7), input, X ) ,"  &
"136 ( BC_1, AD(6), output3, X , 1, 1, Z),"  &
"137 ( BC_2, AD(6), input, X ) ,"  &
"138 ( BC_1, AD(5), output3, X , 1, 1, Z),"  &
"139 ( BC_2, AD(5), input, X ) ,"  &
"140 ( BC_1, AD(4), output3, X , 1, 1, Z),"  &
"141 ( BC_2, AD(4), input, X ) ,"  &
"142 ( BC_1, AD(3), output3, X , 1, 1, Z),"  &
"143 ( BC_2, AD(3), input, X ) ,"  &
"144 ( BC_1, AD(2), output3, X , 1, 1, Z),"  &
"145 ( BC_2, AD(2), input, X ) ,"  &
"146 ( BC_1, AD(1), output3, X , 1, 1, Z),"  &
"147 ( BC_2, AD(1), input, X ) ,"  &
"148 ( BC_1, AD(0), output3, X , 1, 1, Z),"  &
"149 ( BC_2, AD(0), input, X ) ,"  &
"150 ( BC_4, MMI, clock, X ) ,"  &
"151 ( BC_1, MMO, output2, X ) ,"  &
"152 ( BC_1, SECS, output2, X ) ,"  &
"153 ( BC_1, SECLK, output2, X ) ,"  &
"154 ( BC_1, SED, output3, X , 48, 1, Z),"  &
"155 ( BC_2, SED, input, X ) ,"  &
"156 ( BC_4, n_aDREQ(7), clock, X ) ,"  &
"157 ( BC_1, n_aDACK(7), output2, X ) ,"  &
"158 ( BC_4, n_aDREQ(6), clock, X ) ,"  &
"159 ( BC_1, n_aDACK(6), output2, X ) ,"  &
"160 ( BC_4, n_aDREQ(5), clock, X ) ,"  &
"161 ( BC_1, n_aDACK(5), output2, X ) ,"  &
"162 ( BC_4, n_aDREQ(4), clock, X ) ,"  &
"163 ( BC_1, n_aDACK(4), output2, X ) ,"  &
"164 ( BC_4, n_aDREQ(3), clock, X ) ,"  &
"165 ( BC_1, n_aDACK(3), output2, X ) ,"  &
"166 ( BC_4, n_aDREQ(2), clock, X ) ,"  &
"167 ( BC_1, n_aDACK(2), output2, X ) ,"  &
"168 ( BC_4, n_aDREQ(1), clock, X ) ,"  &
"169 ( BC_1, n_aDACK(1), output2, X ) ,"  &
"170 ( BC_4, n_aDREQ(0), clock, X ) ,"  &
"171 ( BC_1, n_aDACK(0), output2, X ) ,"  &
"172 ( BC_1, n_aINTACK(1), output2, X ) ,"  &
"173 ( BC_1, n_aINTACK(0), output2, X ) ,"  &
"174 ( BC_1, n_aEOPTC, output3, X , 39, 1, Z),"  &
"175 ( BC_2, n_aEOPTC, input, X ) ,"  &
"176 ( BC_1, n_aCS(0), output2, X ) ,"  &
"177 ( BC_1, n_aCS(1), output2, X ) ,"  &
"178 ( BC_1, n_aCS(2), output2, X ) ,"  &
"179 ( BC_1, n_aCS(3), output2, X ) ,"  &
"180 ( BC_1, a_D(0), output3, X , 0, 1, Z),"  &
"181 ( BC_2, a_D(0), input, X ) ,"  &
"182 ( BC_1, a_D(1), output3, X , 0, 1, Z),"  &
"183 ( BC_2, a_D(1), input, X ) ,"  &
"184 ( BC_1, a_D(2), output3, X , 0, 1, Z),"  &
"185 ( BC_2, a_D(2), input, X ) ,"  &
"186 ( BC_1, a_D(3), output3, X , 0, 1, Z),"  &
"187 ( BC_2, a_D(3), input, X ) ,"  &
"188 ( BC_1, a_D(4), output3, X , 0, 1, Z),"  &
```

```
        "189 ( BC_2, a_D(4), input, X ) ,"  &
        "190 ( BC_1, a_D(5), output3, X , 0, 1, Z),"  &
        "191 ( BC_2, a_D(5), input, X ) ,"  &
        "192 ( BC_1, a_D(6), output3, X , 0, 1, Z),"  &
        "193 ( BC_2, a_D(6), input, X ) ,"  &
        "194 ( BC_1, a_D(7), output3, X , 0, 1, Z),"  &
        "195 ( BC_2, a_D(7), input, X ) ,"  &
        "196 ( BC_1, n_aWR, output2, X ) ,"  &
        "197 ( BC_1, n_aRD, output2, X ) ,"  &
        "198 ( BC_1, a_A(2), output2, X ) ,"  &
        "199 ( BC_1, a_A(3), output2, X ) ,"  &
        "200 ( BC_1, a_A(4), output2, X ) ,"  &
        "201 ( BC_1, a_A(5), output2, X ) ,"  &
        "202 ( BC_1, a_A(6), output2, X ) ,"  &
        "203 ( BC_1, a_A(7), output2, X ) ,"  &
        "204 ( BC_1, a_A(8), output2, X ) ,"  &
        "205 ( BC_1, a_A(9), output2, X ) ,"  &
        "206 ( BC_1, a_A(10), output2, X ) ,"  &
        "207 ( BC_1, a_A(11), output2, X ) ,"  &
        "208 ( BC_1, a_A(12), output2, X ) ,"  &
        "209 ( BC_1, a_A(13), output2, X ) ,"  &
        "210 ( BC_1, a_A(14), output2, X ) ,"  &
        "211 ( BC_1, a_A(15), output2, X ) ,"  &
        "212 ( BC_1, a_A(16), output2, X ) ,"  &
        "213 ( BC_1, a_A(17), output2, X ) ,"  &
        "214 ( BC_4, REFCLK, clock, X ) ,"  &
        "215 ( BC_1, PIO0Low(0), output3, X , 10, 1, Z),"  &
        "216 ( BC_2, PIO0Low(0), input, X ) ,"  &
        "217 ( BC_1, PIO0Low(1), output3, X , 11, 1, Z),"  &
        "218 ( BC_2, PIO0Low(1), input, X ) ,"  &
        "219 ( BC_1, PIO0Low(2), output3, X , 12, 1, Z),"  &
        "220 ( BC_2, PIO0Low(2), input, X ) ,"  &
        "221 ( BC_1, PIO0Low(3), output3, X , 13, 1, Z),"  &
        "222 ( BC_2, PIO0Low(3), input, X ) ,"  &
        "223 ( BC_1, PIO0High(4), output3, X , 6, 1, Z),"  &
        "224 ( BC_1, PIO0High(5), output3, X , 7, 1, Z),"  &
        "225 ( BC_1, PIO0High(6), output3, X , 8, 1, Z),"  &
        "226 ( BC_1, PIO0High(7), output3, X , 9, 1, Z),"  &
        "227 ( BC_1, PIO1Low(0), output3, X , 18, 1, Z),"  &
        "228 ( BC_2, PIO1Low(0), input, X ) ,"  &
        "229 ( BC_1, PIO1Low(1), output3, X , 19, 1, Z),"  &
        "230 ( BC_2, PIO1Low(1), input, X ) ,"  &
        "231 ( BC_1, PIO1Low(2), output3, X , 20, 1, Z),"  &
        "232 ( BC_2, PIO1Low(2), input, X ) ,"  &
        "233 ( BC_1, PIO1Low(3), output3, X , 21, 1, Z),"  &
        "234 ( BC_2, PIO1Low(3), input, X ) ,"  &
        "235 ( BC_1, PIO1High(4), output3, X , 14, 1, Z),"  &
        "236 ( BC_1, PIO1High(5), output3, X , 15, 1, Z),"  &
        "237 ( BC_1, PIO1High(6), output3, X , 16, 1, Z),"  &
        "238 ( BC_1, PIO1High(7), output3, X , 17, 1, Z),"  &
        "239 ( BC_1, PIO2Low(0), output3, X , 26, 1, Z),"  &
        "240 ( BC_2, PIO2Low(0), input, X ) ,"  &
        "241 ( BC_1, PIO2Low(1), output3, X , 27, 1, Z),"  &
        "242 ( BC_2, PIO2Low(1), input, X ) ,"  &
        "243 ( BC_1, PIO2Low(2), output3, X , 28, 1, Z),"  &
        "244 ( BC_2, PIO2Low(2), input, X ) ,"  &
        "245 ( BC_1, PIO2Low(3), output3, X , 29, 1, Z),"  &
        "246 ( BC_2, PIO2Low(3), input, X ) ,"  &
        "247 ( BC_1, PIO2High(4), output3, X , 22, 1, Z),"  &
        "248 ( BC_1, PIO2High(5), output3, X , 23, 1, Z),"  &
        "249 ( BC_1, PIO2High(6), output3, X , 24, 1, Z),"  &
        "250 ( BC_1, PIO2High(7), output3, X , 25, 1, Z),"  &
        "251 ( BC_1, PIO3Low(0), output3, X , 34, 1, Z),"  &
        "252 ( BC_2, PIO3Low(0), input, X ) ,"  &
        "253 ( BC_1, PIO3Low(1), output3, X , 35, 1, Z),"  &
        "254 ( BC_2, PIO3Low(1), input, X ) ,"  &
        "255 ( BC_1, PIO3Low(2), output3, X , 36, 1, Z),"  &
        "256 ( BC_2, PIO3Low(2), input, X ) ,"  &
        "257 ( BC_1, PIO3Low(3), output3, X , 37, 1, Z),"  &
        "258 ( BC_2, PIO3Low(3), input, X ) ,"  &
        "259 ( BC_1, PIO3High(4), output3, X , 30, 1, Z),"  &
        "260 ( BC_1, PIO3High(5), output3, X , 31, 1, Z),"  &
        "261 ( BC_1, PIO3High(6), output3, X , 32, 1, Z),"  &
        "262 ( BC_1, PIO3High(7), output3, X , 33, 1, Z),"  &
        "263 ( BC_1, n_LED_EN, output2, X ) ,"  &
        "264 ( BC_1, n_aRESET, output2, X ) ,"  &
        "265 ( BC_4, n_ScanTestEn, clock, X ) ,"  &
        "266 ( BC_4, n_ScanMuxSel, clock, X ) ,"  &
        "267 ( BC_1, aCLK, output3, X , 38, 1, Z)";

end vpcichip;
```

D.2 Scan Testing

VolantPCI utilizes a full-scan testing methodology with 4 independent scan chains. The scan chain inputs and outputs are depicted in the following table.

Table 21. VolantPCI Scan Chains

Chain #	Scan Input	Pin #	Scan Output	Pin #	# ff's in Chain
1	A_DREQ(7)	pin 79	SCS	pin 76	908
2	A_INT(3)	pin 202	-LEDEN	pin 180	879
3	-AIB_ERROR	pin 198	-A_RESET	pin 181	21
4	MMI	pin 74	MMO	pin 75	235

Note: Scan chain #3 is clocked by REFCLK. The others are clocked by CLK.

The following additional pins are used to perform Scan Testing.

ScanTstEn# (pin 186) is used to enable scan testing of the chip. This signal should be pulled up to Vdd during normal chip operation.

ScanMuxSel# (pin 187) is used to shift the serial patterns through the scan chains. This signal should be pulled up to Vdd during normal chip operation.

D.3 RAM Isolation Testing

Through the JTAG RamTest instruction, VolantPCI's internal RAM can be isolated directly to primary I/Os for testing. The following table shows the relationship between the RAM Test signals (ie. Address, Data, ..) and the pin names.

Table 22. RAM Testing

RAM Test Signal	VolantPCI Pin Name
RamAddress(5:0)	A_D(5:0)
RamWrStrobe	A_D(6)
RamWrData(31:0)	AD(31:0)
RamRdData(31:16)	A_A(17:2)
RamRdData(15:8)	-A_DACK(7:0)
RamRdData(7)	-A_WR
RamRdData(6)	-A_RD
RamRdData(5:4)	-A_INTACK(1:0)
RamRdData(3:0)	-A_CS(3:0)

D.4 Driver Tri-State

In addition to using the JTAG "highZ" command, VolantPCI's drivers can be placed into Tri-State mode by driving the IDDTST pin 'high'. This will tri-state all output drivers except SCS, -LEDEN, -A_RESET, MMO and PTSTOUT.

Appendix E. Volant Errata

1. The RST# signal when active does not tri-state all of VolantPCI's PCI Bus signals.

   The work around for this is to connect the PCI Bus RST# to both the VolantPCI RST# pin and an inverted version of the signal to the IDDTST pin. This will tri-state all VolantPCI outputs while RST# is active (except for -A_RESET, -LED_EN, MMO and SCS). The board designer may need to add pull-ups to some signals to force them to desired values during reset (i.e. -A_CS3:0, -A_RD, -A_WR, -A_INTACK, etc.).

2. Flushing DMA receive fifo causes TC status bit in the DISR to get set.

   The work around for this is to read the DISR and discard its value after flushing the DMA receive fifo.