PCI Command Description

The PCI board is responsible for processing commands, re-directing controller commands, transferring pixel data to the host computer, and reporting data and replies from the controller. This document contains a description of the PCI registers and how they are used, a description of the interaction between application, device driver, and PCI board, and it describes the commands executed by both the 50 and 250 MHz PCI boards.

I. PCI REGISTERS

The device driver communicates with the PCI board through a series of registers located on the PCI board. The registers are "mapped" by the device driver to produce an equivalent set of virtual registers that the device driver can access as though it were communicating with the PCI board registers directly.

1) PCI CONFIGURATION REGISTERS

Every PCI board contains a set of 64 registers (DWORDS) used for configuration, initialization, and error handling. These registers are not manipulated directly by the user, rather they are typically used by the device driver to setup and initialize the hardware. It is the Base Address Registers that are PCI board dependent and are the registers of interest to users, as these are the driver mapped registers that are used for PCI-driver communications. The configuration space header is shown for documentation purposes only.

(i) Configuration Space Header

ADDRESS	***REGISTER***			
0x0000	Device Id			Vendor Id
0x0004	Status			Command
0x0008	Class Code			Revision Id
0x000C	BIST	Header Type	Latency Timer	Cache Line Size
0x0010	Base Address Registers			
0x0014				
0x0018				
0x001C				
0x0020				
0x0024				
0x0028	Cardbus Pointer			
0x002C	Subsytem Id		Subsystem Vendor Id	
0x0030	Expansion ROM Base Address			
0x0034	Reserved			
0x0038	Reserved			
0x003C	Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line

The Base Address Registers are shown in the table below. The first four are reserved for the DSP. The next five are used by the device driver. The Host Interface Control Register (HCTR) is used for setting command and data bit sizes. The Host Interface Satus Register (HSTR) is used for receiving replies and status information from the controller and PCI board. The Host Command Vector Register (HCVR) is used to send vector commands to the PCI board. The Reply Buffer is used to read reply data values from the

controller and PCI board. Finally, the Command Data register is used to send command parameters and other data values to the controller and PCI board. Each register is 16-bits and is discussed in the following sections.

Host/DSP Control and Status Registers	
Address	***Register***
0x0000	DSP Reserved
0x0004	DSP Reserved
0x0008	DSP Reserved
0x000C	DSP Reserved
0x0010	Host Interface Control Register (HCTR)
0x0014	Host Interface Status Register (HSTR)
0x0018	Host Command Vector Register (HCVR)
0x001C	Reply Buffer
0x0020	Command Data

a) HOST INTERFACE CONTROL REGISTER (HCTR)

Only four bits of this register are used. Bits 8, 9, 11 and 12 are used to set the PCI bus data size for commands and replies. Setting bit 8 to 1 and bit 9 to 0 converts 32-bit PCI commands into 24-bit DSP data. So the most significant byte (MSB) of the 32-bit word is lost for the PCI board. Setting bit 11 to 1 and bit 12 to 0 converts 24-bit DSP reply data into 32-bit PCI data. So the most significant byte (MSB) is filled with zeros for the host computer. These bits are normally set whenever a connection to the device driver is opened.

b) HOST INTERFACE STATUS REGISTER (HSTR)

This register communicates status information about the current state of the system. Two bits (1, 2) are used to determine if the PCI board has data available for reading or if the board is ready to except more commands. The DSP has an input/output FIFO that can hold a maximum of six 16-bit words at any given time. To prevent commands and parameters from being lost, due to a full FIFO, bit 1 is checked before sending a command or data, and bit 2 is checked before reading a reply data value, such as the current pixel count. If bit 1 is set, the input FIFO is empty and a command or data can be sent. If bit 2 is set, the output FIFO is not empty and a reply data value can be read. The bit summary is given below:

Check before sending a command or command parameter:

Bit 1 = 1 Host input FIFO is empty, can send command

0 Host input FIFO is not empty, cannot send command

Check before reading a reply data value:

Bit 2 = 1 DSP output FIFO is not empty, can read data

0 DSP output FIFO is empty, no data

Three bits (3, 4, 5) are used to communicate replies from both the PCI board and the controller. The default value is 0, which would mean a timeout has occurred if the bits are not changed to one of the other values before a specified amount of time has elapsed. A value of 1 means the last command completed successfully. A value of 2 means the last command has generated a reply data value, which is now avaiable for reading from the Reply Buffer. A value of 3 means an error has occurred and the last command has failed. A value of 4 means a system reset has occurred and may or may not be an error, depending on the command. A value of 5 means the controller is currently reading out an image and cannot be interrupted or process ascii commands. A value of 6 means the PCI board is currently busy processing a command and cannot be interrupted. The device driver will wait for one of the other reply values to be set before returning.

Bits 3, 4, 5

= 0	TIMEOUT	(Timeout)
= 1	DON	(Done)
= 2	RDR	(Read Reply Value)
= 3	ERR	(Error)
= 4	SYR	(System Reset)
= 5	READOUT	(Readout)
= 6	BUSY	(PCI Busy)

c) HOST COMMAND VECTOR REGISTER (HCVR)

This Host Command Vector Register (HCVR) is used to send commands to the PCI board only. Commands sent through this resister have assigned values and do not represent ascii characters, as do the controller commands. Vector commands generally represent commands that must be executed immediately, thus interrupting the PCI boards processing. The complete list of vector commands is given below.

II. NORMAL COMMANDS

These commands are the typical three character ascii commands sent in the normal fashion.

SBS – Set Byte Swapping

This instructs the PCI board whether or not to byte swap the image data before writing it to the host computer image buffer. This mode is set at Voodoo startup and after a PCI lod file has been downloaded. This command is for Sun Systems only. Replies with ‘DON’.

Usage: 0x000203 <’SBS’> <arg1>

- arg1 turns hardware byte swapping on/off

= 0 No hardware byte swapping
= 1 Yes hardware byte swapping

TBS – Test Byte Swapping

This command asks the PCI board if hardware byte swapping is supported by the boards firmware. If the PCI's firmware doesn't support byte swapping or the host computer is not a Sun System, then the command will return a value other than 'DON'.

Usage: 0x000202 <'TBS'>

III. VECTOR COMMANDS

These are commands that must be executed immediately. There are two types of vector commands, maskable and non-maskable. The non-maskable commands interrupt the DSP processor for immediate servicing. The maskable commands may or may not immediately interrupt the processor, this is entirely up to the writer of the firmware as to the commands importance. The complete list of these commands is defined in the PCI assembly file "*pciboot.asm*".

Vector commands are 16-bit integers that are sent to the PCI board through device driver functions, which write the command to the PCI's Host Command Vector Register (HCVR). Any vector command arguments are sent before the command and are sent through device driver functions, which write the arguments to the PCI's Command Data Register.

NON-MASKABLE COMMANDS

CLEAR INTERRUPT - 0x8073

This instructs the PCI board to clear any interrupts it may be sending. This command has no arguments. Replies with 'DON'.

READ PIXEL COUNT - 0x8075

This instructs the PCI board to write the current number of pixels, that have been transferred to the host computer, into its reply register. This command has no arguments. Replies with the current pixel count.

RESET PCI - 0x8077

This instructs the PCI board to reset the DSP program counter (PC) back to its initial start position, thus "restarting" the PCI firmware. This command has no arguments. Replies with 'DON'.

ABORT READOUT - 0x8079

The instructs the PCI board to stop transferring pixel data and put the controller back into idle mode. This command has no arguments. Replies with 'DON'.

BOOT EEPROM - 0x807B

This instructs the PCI board to ?.

READ NUMBER OF FRAMES READ - 0x807D
This instructs the PCI board to write the current number of frames, that have been transferred to the host computer, into the reply register. This command has no arguments. Replies with the number of frames written by the PCI board.

PCI DOWNLOAD - 0x802F
This instructs the PCI board to disable command replies and enter a special mode that allows a PCI lod file to be downloaded in a continuous stream without replies being sent. This prevents data values such as 'ERR' from being interpreted as an error reply. This command has no arguments.

MASKABLE COMMANDS

READ REPLY HEADER - 0x81
This instructs the PCI to ?.

READ REPLY VALUE - 0x83
This instructs the PCI board to write a command reply data value into the reply register, which will then be read and passed by the device driver to the user application. This command is only issued if the HTF reply status bits 3, 4, 5 of the Host Interface Status Register (HSTR) are set to a value of 2, which means "read reply" ('RDR'). This command has no arguments.

CLEAR REPLY FLAGS - 0x85
This instructs the PCI board to set the reply flags, which are the HTF bits 3, 4, 5 of the Host Interface Status Register (HSTR), to zero, which is the default "no reply" value. To ensure a valid response, this command should always be sent before any command that expects a reply. This command has no arguments. Replies with 'DON'.

RESET CONTROLLER - 0x87
This instructs the PCI board to instruct the controller to reset itself. This command has no arguments. Replies with 'SYR'.

INITIALIZE IMAGE ADDRESS - 0x91
This command writes the physical address of the image buffer to the PCI board. The address is written in two 16-bit parts, the first argument for this command is the lower 16-bits and the second argument is the upper 16-bits.

WRITE COMMAND - 0xB1
This instructs the PCI board to read and execute all parameters that have been written to the Command Data Register. This is the command that results in the execution of the normal ascii commands. This command has up to six arguments, which are the normal ascii commands and their arguments. See the "*Controller Commands*" document for a description of the ascii controller commands.

IV. ACCESSING THE PCI REGISTERS

The PCI board resisters are accessed by an application through a device driver function call, ioctl on Linux and Unix , and DeviceIoControl on Windows. Below are the list of commands needed for an application to access four of the five PCI registers. The Reply Buffer register cannot be directly accessed by an application.

Unix and Linux

Usage: int ioctl(*int file descriptor, int command, int *arg*)

Where *file descriptor* is the integer returned from the open() function. *Command* is one of the commands described below and defined in astropci_ioctl.h. *arg* is a variable used to send parameters and receive values associated with the execution of the specifed command.

To access the HCVR:
A vector command is sent by first sending any optional data values, followed by the command.

```
ioctl(pci_fd, ASTROPCI_HCVR_DATA, &data);        [optional data value]
…
ioctl(pci_fd, ASTROPCI_SET_HCVR, &command);
```

To access the HCTR:
There are two functions to access the HCTR, a get and a set function. The new value for the set function will be the new HCTR register value.

```
ioctl(pci_fd, ASTROPCI_GET_HCTR, &oldValue);
ioctl(pci_fd, ASTROPCI_SET_HCTR, &newValue);
```

To access the HSTR:
There is a single function to read the HSTR.

```
ioctl(pci_fd, ASTROPCI_GET_HSTR, &hstrValue);
```

Where the parameters are defined as:

ASTROPCI_GET_HCTR	0x1
ASTROPCI_GET_HSTR	0x4
ASTROPCI_HCVR_DATA	0x10
ASTROPCI_SET_HCTR	0x11
ASTROPCI_SET_HCVR	0x12

Windows 2000

In the first DeviceIoControl call, *inBuffer* contains the optional data value and *outBuffer* contains any returned data value. In the second DeviceIoControl call, *inBuffer* contains a valid vector command value and *outBuffer* contains any returned value.

Usage: BOOL DeviceIoControl

```
(
HANDLE hDevice,        // handle to device of interest
DWORD dwIoControlCode, // control code of operation to perform
LPVOID lpInBuffer,     // pointer to buffer to supply input data
```

```
    DWORD nInBufferSize,   // size, in bytes, of input buffer
    LPVOID lpOutBuffer,    // pointer to buffer to receive output data
    DWORD nOutBufferSize,  // size, in bytes, of output buffer
    LPDWORD lpBytesReturned, // pointer to variable to receive byte count
    LPOVERLAPPED lpOverlapped // pointer to struct for synchronous operation
    );
```

To access the HCVR:
A vector command is sent by first sending any optional data values, followed by the command.

```
DeviceIoControl((HANDLE)pci_fd, ASTROPCI_HCVR_DATA, &inBuffer,
                sizeof(inBuffer),  &outBuffer, sizeof(outBuffer),
                &bytesReturned, NULL));
…
DeviceIoControl((HANDLE)pci_fd, ASTROPCI_SET_HCVR, &inBuffer,
                sizeof(inBuffer), &outBuffer, sizeof(outBuffer),
                &bytesReturned, NULL));
```

To access the HCTR:
There are two functions to access the HCTR, a get and a set function. The inBuffer for the set function will be the new HCTR register value.

```
DeviceIoControl((HANDLE)pci_fd, ASTROPCI_GET_HCTR, NULL,
                0,  &outBuffer, sizeof(outBuffer),  &bytesReturned, NULL));
DeviceIoControl((HANDLE)pci_fd, ASTROPCI_SET_HCTR, &inBuffer,
                sizeof(inBuffer),  &outBuffer, sizeof(outBuffer),
                &bytesReturned, NULL));
```

To access the HSTR:
There is a single function to read the HSTR.

```
DeviceIoControl((HANDLE)pci_fd, ASTROPCI_GET_HSTR, NULL,
                0,  &outBuffer, sizeof(outBuffer), &bytesReturned, NULL));
```

Where the parameters are defined as:

```
// Device Type - arbitrary # in range: 32768 to 65535.
ASTROPCI_DEVICE                 33000

ASTROPCI_GET_HCTR \
        CTL_CODE(ASTROPCI_DEVICE, 0x801, METHOD_BUFFERED,
                        FILE_ANY_ACCESS)

ASTROPCI_GET_HSTR \
        CTL_CODE(ASTROPCI_DEVICE, 0x804, METHOD_BUFFERED,
                        FILE_ANY_ACCESS)

ASTROPCI_HCVR_DATA \
        CTL_CODE(ASTROPCI_DEVICE, 0x810, METHOD_BUFFERED,
                        FILE_ANY_ACCESS)

ASTROPCI_SET_HCTR \
        CTL_CODE(ASTROPCI_DEVICE, 0x811, METHOD_BUFFERED,
```

```
                FILE_ANY_ACCESS)

ASTROPCI_SET_HCVR \
        CTL_CODE(ASTROPCI_DEVICE, 0x812, METHOD_BUFFERED,
                FILE_ANY_ACCESS)
```

V. DOWNLOADING A PCI LOD FILE

Because the PCI board firmware contains the reply definitions, it requires a special method for downloading lod files to prevent false errors from being reported. First, bits 8 and 9 of the HCTR must be cleared to allow 32-bit values to be written without loss of bytes. The 32-bit values are broken up into two 16-bit values by the DSP. Next, the device driver command ASTROPCI_PCI_DOWNLOAD (0x13) is called. This command merely sends the PCI_DOWNLOAD vector command to the PCI board, but does not look for a reply, as there isn't one. The PCI board will now be looking for a magic number which indicates that the contents of an lod file will soon be streaming in. The magic number is 0x00555AAA, and is written to the Command Data register via ASTROPCI_HCVR_DATA. Finally, the contents of the lod file are written using the Command Data register. The first two words written should be the total word count and the start address, as read from the lod file. Only data blocks starting with "_DATA P" are sent, all others are discarded. After the data has been written, the DSP input/output data sizes are reset by setting bits 8 and 11 to 1 and clearing bits 9 and 12 in the HCTR. And lastly, the device driver command ASTROPCI_DOWNLOAD_WAIT (0x14) is called. This command will wait for the PCI board to change the HSTR reply flags from BUSY to (hopefully) DON. To verify that the download was successful, several test data link (TDL) commands should be sent to the PCI board. The download sequence is summarized below.

1. Clear bits 8 and 9 of the HCTR using the ioctl commands, ASTROPCI_GET_HCTR and ASTROPCI_SET_HCTR.
2. Send the vector command PCI_DOWNLOAD to the PCI board by calling ASTROPCI_PCI_DOWNLOAD.
3. Send the magic number 0x00555AAA by writing it to the Command Data register via the ioctl command ASTROPCI_HCVR_DATA.
4. Read the total word count from the lod file and write it to the Command Data register.
5. Read the start address from the lod file and write it to the Command Data register.
6. Send all data contained within "_DATA P" blocks via the Command Data register.
7. Set bits 8 and 11 to 1, and clear bits 9 and 12 in the HCTR.
8. Call the device driver ioctl command ASTROPCI_DOWNLOAD_WAIT to wait for the PCI board to complete its processing.
9. Send a few test data link (TDL) commands to verify the download.'