HEWLETT
PACKARD

HARDBALL I/O SUBSYSTEM

EXTERNAL REFERENCE SPECIFICATION

Version 1.1

Printed: September 30, 1991

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Primary Objectives

The main purpose of this document is to provide essential information about the Hardball core I/O subsystem from an external viewpoint.

The main design objective for the I/O subsystem is to provide a competitive, high performance, low cost on-board I/O subsystem for Hardball.

1.2 Differences between Cobra/Coral I/O subsystem and Hardball I/O subsystem

1.2.1 Added Features

fast/wide SCSI

FDDI(optical)

stereo/CD quality audio

FIFO chip control(Shortstop ASIC replaces 245's and 646's and contains FDDI and fast/wide SCSI buffers for DMA)

1.2.2 Deleted Features

interval timer

serial 3 port

domain keyboard connector

test connectors

1.3 Feature Summary

The following is a list of the features on the I/O subsystem:

- Viper (memory interface)
- SCSI (DMA)
- Fast/Wide SCSI (DMA)
- Parallel Port (Bidirectional with ScanJet support, DMA)
- Two General Purpose RS-232 Ports
- LAN (DMA)

- FDDI(optional)
- HP-HIL
- Audio(CD/stereo quality)
- 512Kx8 ROM
- Real Time Clock (RTC) w/Lithium Battery Backup
- 8Kx8 EEPROM
- Front panel status LED register loading capability.(This is different then the way it was done on ASP)
- I/O subsystem status register is readable
- Sevice/Normal bootup switch
- Sevice/Normal switch and TOC front panel switches are readable.
- SPU ID (LAN ID is in EEPROM, fallback HP-HIL module)

2. I/O SUBSYSTEM OVERVIEW

2.1 Functional Organization

Refer to the "Hardball I/O Subsystem Data Path Block Diagram" in the Diagrams section of this document for the following discussion.

For this discussion, "IOSS" stands for "I/O Subsystem".

In the IOSS there are four buses, three local data busses and an address bus. FDDI and LAN sit on the iolf data bus; SCSI, fast/wide SCSI, audio, and the status register sit on the iodbig data bus; the rest of the subsystem sits on an 8 bit iod bus. Depending on the specific requirements of the block, they may, or may not, use some portion of the address bus.

The core IOSS is attached to the Standard Graphics Connection (SGC) bus. Data communication passes through an ASIC(Shortstop), which has two sets of 32 bit bi-directional tri-state buffers. Only one set of the 32-bit buffers will be driven at a time onto the SGC bus. I/O addresses pass through a 30 bit bi-directional tri-state register, making the IOSS capable of performing master DMA operations.

If necessary, byte addressing during DMA operations is done to get into word alignment or to finish off a transfer which does not end on a word boundary. The Cutoff DMA controller automatically handles word alignment.

"Cutoff" is the heart of the IOSS. Its main functions are IOSS address decoding, bus arbitration, interrupts and data flow control. Cutoff interfaces with Viper, the system I/O controller as well as with the IOSS devices.

Sub-word byte addressing access by the host is handled through the Viper I/O system controller interface with Cutoff. Viper is not actually considered part of the IOSS but is discussed because of its important control interaction with Cutoff.

2.1.1 Byte and Bit Ordering; Big Endian, Little Endian

PA-RISC instruction set bit and byte offset numbering:

0							7	8							15	16							23	24							31
0								1								2								3							

SGC and IOSS bit and byte offset numbering:

31							24	23							16	15							8	7							0
0								1								2								3							

Properly sequenced data byte word assembly and disassembly is handled by the IOSS hardware, however, it is the software driver's responsibility to properly translate the order of command sequences before passing them to the IOSS.

Other exceptions are found with respect to the SCSI register access. See "Effects of data byte swapping" in the "SCSI" section.

2.2 Viper Interface

The signals needed by the core-I/O to interface to Viper are all defined in the standard SGC bus specification except for the interrupt request signal. The interrupt is asserted by Cutoff on behalf of the devices inside the core-I/O subsystem. The interrupt will be deasserted after a CPU read to Cutoff's interrupt register.

3. STANDARD GRAPHICS CONNECTION (SGC)

Hardball Core I/O conforms to the VSC (Viper System Connect) specification.

3.1 Signal Definitions

For SGC Bus signal definitions, please refer to the Standard Graphics Connection Specification, Revision 1.1.

3.2 Transaction Types

There are no semaphores to I/O devices. None of the devices have local caches, so cache coherent reads and writes are not supported. It is possible that core I/O devices will read or write data with all combinations of contiguous valid bytes. Core I/O bus master devices may pipeline transactions. Core I/O does make use of burst mode.

3.3 Bus Arbitration

There are seven bus masters which use DMA on the core I/O board: LAN, parallel interface, SCSI, audio, fast/wide SCSI, Rapid Harmless DMA and FDDI. Cutoff will use the SGC bus request and bus grant signals on SGC to gain access to the bus for each device. If multiple bus requests happen simultaneously, Cutoff will arbitrate SGC for only one time and allow the above devices to master the bus according to their priority. However, if the bus grant signal is taken away from Cutoff in the middle of a DMA transfer, then Cutoff will arbitrate for the bus again for the next DMA transfer. In addition to the normal bus request and bus grant signals on SGC, Cutoff also utilizes the EISA refresh cycles. This means Cutoff will master the SGC bus while EISA cards are in the refresh cycles.

4. CUTOFF I/O CONTROLLER ASIC (ASP 2)

4.1 Introduction

"CUTOFF" is the controller ASIC chip for the I/O subsystem and interfaces with the "Viper" controller and the SGC bus. Besides control functions, it also contains registers for subsystem interrupts, a DMA channel for the Parallel Printer Interface scan path logic, some Parallel Printer Interface related logic including a 32 byte bidirectional FIFO, SGC bus address decoders, and other miscellaneous registers.

4.2 External Signal Definitions

?

4.3 Direct Memory Access (DMA)

Inside the NCR 53C700 SCSI controller, there is a bus master DMA device which is capable of moving data between disk and system memory at the rate of 27.7 Mbyte/sec. This assumes the NCR chip is running at 33 Mhz, has a burstsize of 24 bytes, and an arbitration overhead of 13 VSC clock cycles. The Intel 82596 LAN controller also has a built in high-performance DMA controller. Inside Cutoff, we will provide a 32 byte FIFO and a DMA channel for the Parallel Printer Interface to support the further bi-directional Parallel Printer Interface applications, such as Scan Jet.

4.4 Interrupt System

Inside Cutoff, similar to the interrupt structure inside PA-RISC, we provide an Interrupt Request Register (IRR) and an Interrupt Mask Register (IMR). An Interrupt Pending Register (IPR) is also provided. Each of these registers appear to be 32-bits and are accessed as such. However, Only the 15 least significant bits are implemented for each register. The remaining bits are not affected by writes and are always read as zeros.

The Interrupt Pending Register (IPR) is used to latch incoming interrupts and indicate them as pending. The external interrupts are synchronized in the IPR and an active edge on the synchronized signal causes the corresponding IPR bit to be set to 1. The IPR is a read/write register. Reads will return the status of all interrupts, pending or non-pending. Writes to this register are intended for diagnostic use only and will cause the entire register to be cleared. Since a write of the IPR clears all the IPR bits, any interrupt with an active edge during the write will not set the corresponding IPR bit. This leads to the possibility of a device interrupting during a IPR write and not creating a pending interrupt.

The Interrupt Mask Register (IMR) is a read/write register used to mask pending interrupts. A 1 in an IMR bit enables the corresponding pending interrupt to create an interrupt request. If the IMR bit is a zero, the corresponding pending interrupt can not cause an interrupt request.

The Interrupt Request Register (IRR) is a read only register that contains the status of all requesting interrupts. A 1 in an IRR bit indicates that the corresponding interrupt was pending and enabled by the IMR. When an IRR bit is set it will cause Cutoff to generate an interrupt request to Viper. Cutoff will continue to assert this interrupt line until it is acknowledged by a CPU read of the IRR.

A read of the IRR will also cause unmasked (IMR bit = 1) IPR bits to be cleared. Both the IRR and the unmasked IPR bits are cleared on the clock cycle following the read. From the IRR read, Software may find multiple interrupts pending. It is the responsibility of the software to set the priority on these interrupts and

to remember the lower priority interrupts which will not be served immediately. New interrupts may arrive before software finishes servicing these interrupts. Software should read the IRR again to see if there are any new higher priority interrupts after servicing an interrupt.

External devices must assert interrupts for just over two CPU clock cycles (one I/O subsystem cycle) in order for them to be synchronized and detected. The interrupt must also de-assert for at least two CPU clock cycles (one I/O subsystem cycle) in order for the next assertion to be recognized.

Listed below are the source of the defined interrupts. The registers are set by the transition to the defined active level.

4.4.1 Interrupt Registers Access and Bit Assignments

☞ The following registers should be accessed as words:

0xF080 0000 - Interrupt Request Register (Read Only)

0xF080 0004 - Interrupt Mask Register (Read/Write)

0xF080 0008 - Interrupt Pending Register (Read/Write)

☞ Bit 31 is the least significant bit and is on the right hand side of the word.

BIT	LEVEL	SOURCE
31	- LOW	NMI from EISA
30	- HIGH	8042 (rtc, hil, audio) General interrupts
29	- LOW	8042 (rtc, hil, audio) High priority interrupts
28	- LOW	Fast/Wide SCSI
27	- LOW	FDDI
26	- HIGH	WD16C552 SIO 1
25	- HIGH	WD16C552 SIO 2
24	- LOW	WD16C552 Parallel Printer Interface
23	- LOW	LAN-802.3
22	- LOW	SCSI
21	- LOW	EISA
20	- LOW	Graphics1
19	- LOW	Graphics2
18	- LOW	Audio
17	- LOW	Error

☞ In ASP, bit 28 was timer 1, bit 27 was timer 2, and bit 18 was serial port 3.

4.4.2 Interrupt Scenarios

Following, is a simple scenario of a device interrupting under three different conditions, along with a scenario that could generate spurious interrupts.

- Interrupt Masked:

When the device interrupts it will cause an active edge on the device's interrupt line. This will cause the IPR bit to be set to one which stays asserted until it is cleared by the CPU. However, this will not cause the IRR bit to be set as it will be masked to zero by the IMR.

● Interrupt Unmasked:

This condition is the same as a masked interrupt except the IRR bit will be set after the IPR is set because it will not be masked. This will cause the interrupt request line from Cutoff to Viper to be asserted. The CPU should then read the IRR register, causing the IPR bit to be cleared, and service the interrupt which will cause the device to de-assert its interrupt request.

● Masked then Unmasked by a Write to the IMR:

Same as masked until the IMR is written. When the IMR is written, causing the interrupt to be unmasked, the IRR bit will be set and the request line from Cutoff to Viper will be asserted.

● Spurious Interrupts:

The following sequence could create a spurious interrupt and should be handled by software.

1) A high priority and low priority interrupt is unmasked.

2) A high priority interrupt occurs.

3) Processor enters interrupt handler.

4) Interrupt handler reads IRR, clearing the high priority bit in IPR and IRR.

5) Low Priority interrupt occurs.

6) Interrupt handler, in processing of the high priority interrupt, ' decides to mask the low priority interrupt.

7) Interrupt handler exits.

8) CPU is reinterrupted by Cutoff because of low priority interrupt in IRR. (It was received before it was masked off).

~~9) Processor enters~~ interrupt ~~the handler~~.

10) Interrupt handler reads the IRR, clearing the low priority bit in IRR but not in the IPR because bit is masked. This could be considered a spurious interrupt.

11) Interrupt handler exits.

12) When a low priority interrupt is unmasked, an interrupt will occur and can be properly cleared by the next IRR read.

An option for software is to read the IRR after changing the IMR to look for interrupts that may have occurred between the last IRR read and the IMR write. The bit could be unmasked and read to clear it again and remasked and read until it no longer occurs.

4.5 Other Register and Byte Device Access

Refer to each section for details about whether the registers are word accessed or byte accessed.

There are a number of Cutoff specific registers. The first of these is the I/O Subsystem reset register (F082 F000), a write only byte access register, which resets Cutoff and the entire on-board I/O subsystem. This register initializes Cutoff and all of the on-board devices to their power-up values.

The second byte access register (F082 F020) is the Cutoff Status byte. This is a read only byte access register with the following information in each bit field.

D[7]	=	RH_DMA_MODE bit (from address F083 3020)
D[6:5]	=	NT[1:0] bits (from address F083 0004)
D[4]	=	Dsync bit (from address F082 F030)
D[3:0]	=	cutoff version. For the initial Cutoff release 1/92, this 4 bits has the value 0x03.

Note that D[7:4] have the information from other regiters which are write only. For more details on those registers, refer to the memory map for the appropiate sections.

The third byte access register (F080 F030) contains the DSYNC enable bit (bit 0, right hand bit) ~~for the SCSI subsystem~~. The default mode for this bit is enabled (DSYNC = 1). The purpose of this bit is to increase hardware performance for SCSI DMA transfers . This is a write only register. Although this is a write only register, its content could be read via the Cutoff Status register.

The fourth byte access register (F082 F040) is an error logging byte. This byte powers up to value 0x00 and is a read/write register. The purpose of this byte is to log the bus master information if an error occurs while Cutoff is the bus master on SGC. See the error handling for more details.

The fifth byte access register, LAN_CTRL_OEN (F082 F050), enables (un-tristates) either Cutoff's output-only FDDI pins, output-only ethernet pins (i82596 802.3 controller), or both. This allows external muxing of FDDI and 802.3 signals to a Slider board by tieing them together in a wired-AND fashion. Setting this register correctly is extremely important; improper settings could result in non-operation or worse, electrical shorts.

7							0
R	R	R	R	R	R	E	F

		value	
Bit	Description	0	1
E	Ethernet	disable	enable
F	FDDI	disable	enable
R	Reserved		

Firmware must follow the following algorithm

```
if (LAN controller signals muxed on board) # bit 14 of IOSS status register (F0800024)
then
    case NETWORK_ID of                # IOSS status register bits [13:12]
        802.3: enable ethernet signals;
        FDDI : enable FDDI signals;
    end_case;
else
```

```
    case NETWORK_ID of
        802.3: enable ethernet signals;
        FDDI : enable FDDI signals;
        both : enable both FDDI and ethernet;
    end_case;
end_if;
```

4.6 Power Up Reset

After power-on reset or on-board reset (write to 0xF082F000) the Cutoff is in the following state:

Address	Register Name	State
0xF080 0000	Interrupt request register	0x0000 (all requesting interrupts zeroed to prevent spurious interrupts at power-up).
0xF080 0004	Interrupt mask register	0x0000 (all interrupts masked)
0xF082 4802	Parallel Device Control	0bxx101000; rd/wr direction set to rd (when enabled). All other parallel port registers are cleared to 0, and interrupts from the parallel port module are disabled. The default mode is non DMA IBM PS/2 type configuration mode.
0xF081 8xxx	EEPROM enable bit	0b0xxxxxxx (EEPROM write access disabled, the MSB of byte 0 = 0)
0xF082 0000	DMA current addr. reg.	0x0000
0xF082 0001	DMA current count reg.	0xFFFF
0xF082 0008	DMA status reg.	0x00
0xF082 000A	DMA write single bit mask	0x04
0xF082 000B	DMA mode reg.	0x04
0xF082 000C	DMA clear byte pointer	byte pointer = 0
0xF082 000D	DMA master clear	not applicable
0xF082 000E	DMA clear mask reg.	mask = 1
0xF082 000F	DMA mask reg.	0x0E
0xF082 0010	DMA FIFO limit reg.	0x80
0xF082 0087	DMA current addr. low page	0x00
0xF082 0401	DMA high current count	0xFF
0xF082 040A	DMA interrupt log reg.	0x00
0xF082 0487	DMA current addr. high page	0x00
		DMA is disabled and all address pointers point to 0 and all count registers are 0. Interrupts are

		disabled from the DMA controller.
0xF082 280X	Serial port #2 Status	WD16C552 chip internal serial port related registers come up in the power up reset state.
0xF082 2804	Serial port #2 Status	0x04; Hardware flow control powers up disabled.
0xF082 380X	Serial port #1 Status	WD16C552 chip internal serial port related registers come up in the power up reset state.
0xF082 3804	Serial port #1 Status	0x04; Hardware flow control powers up disabled.
0xF080 F030	DSYNC enable bit	
0xF082 F050	LAN_CTRL_OEN	0x00; All LAN control signals power up disabled.

Note in addition the various controllers power up into defined states. This definition is available in the vendor documentation for the NCR53C700, the Intel 82596, the Western Digital WD16C552 (or National NS16550), and the HP documentation for the 8042 controller (A-1820-4784-2). Please consult these documents for the power-up state of these devices. Note in particular that the 8042 performs a self-test after reset of approximately 300 ms.

4.6.1 Initialization

Firmware/OS needs to initialize the following Cutoff registers:

- DSYNC: Byte write 0x00 to 0xF080F030.
- Interrupt Mask: Word write to 0xF0800004. At power up, all interrupts are disabled. Do NOT enable the interval timers.
- DMA FIFO Limit Register: Byte write to 0xF0820010. Power up sets this to 0x80, which causes Cutoff to transfer 8 words of parallel port data per VSC arbitration cycle. If system testing shows this to be unacceptable, then firmware/OS will have to set this to another value. See Bidirectional Parallel Printer Interface section "Fifo Limit Register."
- Parallel Port Timing Delay Counter: Byte write to 0xF0824806. Initially 0, the value for correct operation depends on which device is hooked up to the parallel port. The correct value = (10 x 10E6 x DELAYTIME) -1, where DELAYTIME is device dependent. See Bidirectional Parallel Printer Interface section "Timing Delay Counter."
- 8042 Reset Release: Byte write anything to 0xF0821C00. After power up, directed reset to the 8042 or directed reset to the I/O subsystem, the 8042 must be released from its reset state. See PDH section "8042 Reset."

Drivers must still set up devices such as LAN and SCSI for proper operation.

Refer to the sections on FDDI and Fast/Wide SCSI that are in this document. Also, refer to Vivace ERS for the audio initialization requirements.

Rapid Harmless DMA will initialize Viper into a known state without any Intel 596's available.

For Cutoff initialization, there is no 8042/Domain Keyboard Reset Release as appeared in ASP initialization.

4.7 IC Process Technology

Cutoff is being implemented in CMOS26B Standard Cell technology using the Hewlett-Packard "LogicArchitect IC Design System" for development.

4.8 Electrical and Thermal Specifications

4.8.1 DC Characteristics

- Voltage range: 4.5v. to 5.25v.

4.8.2 AC Characteristics

4.8.2.1 Clock Specifications

- Maximum clock speed: 33 MHZ (gclk).

4.8.3 Packaging Technology and Thermal Requirements

- 240 pin PQFP.
- Operating temperature range in degrees Centigrade: 0 to 85.

4.9 TESTING

4.9.1 Introduction

In order to achieve a high degree of fault coverage and to keep testing of all logic as simple as possible, it is often necessary to make certain accommodating modifications to the logic as described in the following section.

4.9.2 Testability Design Goals

4.9.2.1 On Chip Testability

There are two pins on Cutoff are designated for testability purposes. They are the ScanEnmode and trimode: pins. These two pins are decoded internally to provide the following test modes:

ScanEnmode	trimode	
0	0	: Normal operation
0	1	: Tristate mode
1	0	: Scan chain test mode
1	1	: Ptest mode

4.9.2.1.1 Normal operation

The ScanEnmode and the trimode are tied to ground via pull down resistors on the I/O board.

4.9.2.1.2 Tristate mode

(see on board testability section)

4.9.2.1.3 Scan chain test mode (pins free)

In this test mode, there are other 3 pins on Cutoff are multiplexed to verify the scan chain fuctions : NeepWecSPen, NeepOecScnI, and sclkselcScnO. (NeepWecSPen) is the serial phase enable pin for the scan chain. (NeepOecScnI) is scan-in data pin and (sclkselcScnO) is the scan-out data pin for the chain. (Note that in normal mode, these 3 pins have different functions.) In addition, All bidirectional pins are forced to be inputs in the scan mode.

Cutoff utilizes the Automatic Test pattern Generation tool (ATG) within the Logic Architect 4.0 to provide test vectors for the scan chain.

4.9.2.1.4 Ptest mode

This test mode is for testing the P transistors in the internal SRAM module.

The following guidelines are observed in the Cutoff testability design:

- For testability reasons, all internal registers which would not normally be readily controllable or observable should be in the scan chain. Because of this issue, scan path is particularly important in the control state machines.
- All flip-flops and latches should be able to reset or set by a nonsequential logic path accessible from a primary input (pin signals). This can be accomplished by providing a master clear or a parallel load capability. Also, simulation is made easier when all flip-flops can be initially set to a known state.
- There should be no race conditions which could cause the circuit to intermittently fail. This situation can occur in the design of sequential logic when the data arrives at the same time the clock signal arrives. In general, the gating of a clock to generate a clock signal is bad practice and inhibits taking advantage of automatic test generators.
- Counter circuits of more than about 12 bits frequently require test sequences that are too long for practical test generation. Providing a test count enable input signal which is common to all stages, allow all the stages to increment simultaneously, thus minimizing the number of test vectors required.
- The testability design goal of better than 97% fault coverage is required.
- Using asynchronous resets to implement system logic functions should be kept to a minimum. Asynchronous resets to implement initialization are beneficial and in fact are mandatory.

4.9.2.2 On Board Testability

Cutoff also provides a test mode which facilitate board production testing. It is the "tristated" test modes.

In the tristated test mode, all of the outputs and bidirectional pins are tristated. This test mode helps board level testing to isolate the Cutoff chip while testing other parts on the board by ensuring Cutoff is not driving any of its outputs.

5. SHORTSTOP DATA PATH ASIC

5.1 Introduction

Shortstop is the data path ASIC for the I/O interface to SGC. It contains an interface similar to a set of 646's for both of the local data buses. It also offers byte steering for single-byte transactions. It contains two FIFO's for fast-wide SCSI and FDDI, to improve performance on master burst transactions. It contains no registers.

5.2 External Signal Definitions

D[31:0]	32	I/O	SGC data bus
IODBIG[31:0]	32	I/O	Data bus to SCSI, F-W SCSI, Status register, Audio, and 8-bit bus
IOLF[31:0]	32	I/O	Data bus to Lan and FDDI
NDINHIBIT	1	I	Active low inhibits CCLK[1] from clocking the buffers
CCLK[1]	1	I	Clock for the data buffers from SGC to I/O
NSGCDEN	1	I	Active low enables data in buffer mode
SGCDREGMOD	1	I	High: data path from I/O to SGC is registered Low: data path from I/O to SGC is transparent
IODREGMOD	1	I	High: data path from SGC to I/O is registered Low: data path from SGC to I/O is registered
IODCLK[3:0]	4	I	Clocks for the data buffers from I/O to SGC
DDIRNBA	1	I	Low: I/O data to SGC, high: SGC data to I/O
NSTEER[1:0]	2	I	Encoded for byte swapping - 11: no swapping 10: byte 1 is swapped to lower order byte 01: byte 2 is swapped to lower order byte 00: byte 3 is swapped to lower order byte
BUSSELECT	1	I	High: bus for transaction is IODBIG Low: bus for transaction is IOLF
FWSSGCDATAL	1	I	On write: Data enable from FWS FIFO onto SGC On read: Data enable from SGC into FWS FIFO
FWSLOCDATAL	1	I	On write: Data enable from IODBIG to FWS FIFO On read: Data enable from FWS FIFO to IODBIG
FWSWRL	1	I	Active low: transaction is a write to memory

FWSEMPTY	1	O	High: FWS FIFO is empty
RESETFWSL	1	I	Reset the FWS FIFO
FDDIIOCDATAL	1	I	On write: Data enable from IOLF into FDDI FIFO On read: Data enable from FDDI FIFO onto IOLF
FDDISGCDATAL	1	I	On write: Data enable from FDDI FIFO onto SGC On read: Data enable from SGC into FDDI FIFO
FDDIWRL	1	I	Active low: transaction is a write to memory
FDDICKSUML	1	I	On low edge: start to accumulate the checksum On high edge: clear the checksum
FDDICKSUMRD	1	I	On low edge: read the checksum into the FDDI buffers
RESETFDDIL	1	I	Reset the FDDI FIFO
READYL	1	I	SGC ready
CCLK[0]	1	I	SGC gclk
TRIMODE	1	I	Shortstop tristate mode
SCANENMODE	1	I	Shortstop scan path enable

```
Count for pin type "I" is       27
Count for pin type "O" is       1
Count for pin type "I/O" is     96
------------------------------------
Total signal pin count is       124
Maximum signal pins desired: 131 for a 160 pin package
```

5.3 IC Process Technology

Shortsop is being implemented in CMOS26B Standard Cell technology using the Hewlett-Packard "LogicArchitect IC Design System" for development.

5.4 Electrical and Thermal Specifications

5.4.1 DC Characteristics

- Voltage range: 4.5V to 5.25V.

5.4.2 AC Characteristics

5.4.2.1 Clock Specifications

- Maximum clock speed: 33 MHz (gclk)

5.4.3 Packaging Technology and Thermal Requirements

- 160 pin PLCC
- Operating temperature range in degrees Centigrade: 0 to 85.

5.5 TESTING

The testability strategy for Shortstop will be the same as for Cutoff. pins ScanEnmode and trimode will be used in the same way. The 3 other that are multiplexed for scan chain functions are FWSEMPTY (Scan SGCREGMOD (Scan In), IODREGMOD (Serial phase enable). Shortstop utilize the Automatic Test pattern Generation tool (ATG) within Architect.

6. BIDIRECTIONAL PARALLEL PRINTER INTERFACE

6.1 Introduction

The Bidirectional Parallel Printer Interface is an 8 bit parallel, synchronous interface commonly used for printers. The Hardball hardware implementation has bidirectional capabilities compatible with PS2 standards, also known to the world as Centronics(tm). The hardware is also capable of interfacing to the HP Scanjet parallel port, which requires a special bidirectional "BUSY" and NSTROBE handshake. The Apollo CP300 (alias Tektronics 4693D) raster copier (printer) is also supported.

6.2 Overview

The PS2 and AT compatible features are controlled through "Cutoff". The Western Digital WD16C552 chip is used at the parallel port as the device driver/receiver interface. Additional functionality including that required by the Scanjet and CP300 products are augmented by special hardware built into "Cutoff".

6.3 Features

The following is a list of the Parallel Printer Interface features:

- Hardware is capable of 380 Kbytes/sec. maximum data transfer rate when using standard setup, hold, and strobe pulse width times.
- Supports host DMA.
- A FIFO is an integral part of DMA. The FIFO supports 32 byte inbound or 32 byte outbound data transfers (only one direction at a time).
- Fully bidirectional.
- Meets the special handshaking requirements of the HP-9190A/9195A Scanjet products.
- Supports the Apollo CP300 (alias Tektronics 4693) raster copier (printer).
- 25 pin female DB25 connector (same as Vectra and IBM PS/2).
- NACK and BUSY handshakes.
- Pull-up resistors on all lines.

6.4 Register Set

The following are the register byte addresses and descriptions.

☞ NOTES:

- The bits defined below are numbered from the most significant bit 7 (on the left) to the least significant bit 0 (on the right).

MSB LSB
7 $\longrightarrow$ 0

6.5 Power Up Reset

The following conditions occur after a power up reset: ~~reset)~~:

- All the parallel printer interface state machine controllers in asp are reset
  $\uparrow$ Cutoff
- NSTB = 1
- NAFD (alias WRnRD, WR/nRd) = 1
- NINIT (alias NRESET, nRESET) = 0
- NSLIN (alias NSLCT_IN, NSLCTIN) = 0
- All other registers will be cleared.
- All Parallel Port Interrupts will be disabled.

6.5.1 Byte Address 0xF082 4000 (Directed Master Reset)

This has the same affect on the Parallel Printer Interface as a power up reset as described above except that the NSTB, NAFD, NINIT, and NSLIN lines in the Western Digital Chip are not affected. Software must also initialize those outputs by writing the appropriate bits at address 0xF082 4802 after a directed reset is invoked.

- Write

BIT	
7-0	Can be anything.

6.5.2 Byte Address 0xF082 4800 (Write/Read Data)

- Write

BIT	
7-0	Write data direct to parallel port according to handshake mode selected. There are two ways of setting the port to write mode: 1) Set the direction bit, biden, 0xF082 4804 bit 4 to 0, or 2) Set 0xF0824804 bit 4 to 1 and 0xF082 4802 bit 5 to 0. Use when in handshake Mode 0, 1, or 2 only

- Read

BIT	
7-0	Read data direct from parallel port. Biden (0xF082 4804 bit 4) must = 1 and 0xF082 4802 bit 5 must = 1.

Use when in handshake Mode 0 or 3 only

6.5.3 Byte Address 0xF082 4801 (Parallel Port Status)

- Write:

  Invalid. No effect.

- Read:

BIT	
7	Nbusy: returns 0 if the BUSY signal is asserted (high).
6	Nack: returns 0 if the NACK signal is asserted (low)
5	pe: returns 1 if the PE signal is asserted (high)
4	slct: returns 1 if the SLCT signal is asserted (high).
3	error: returns 0 if the NERR signal is asserted (low).
2	NINT: Shows the status of the INT2 (NACK interrupt) line @ the WD chip. This bit will be 0 if NACK had a 0 to 1 trailing edge trainsition. Reading this status register will set this bit to 1. WARNING: In this hardware implementation, the INT2 interrupt line from the WD chip is not connected for system interrupts. A complete set of interrupt choices, including an NACK interrupt, are available as shown in the "IE Control/Interrupt Status" sub-section at Byte Address 0xF082 4805. The state of this bit will have no effect on the interrupt mode chosen and visa-versa.
1	Returns 1.
0	Returns 1.

6.5.4 Byte Address 0xF082 4802 (Parallel Device Control)

- Write:

BIT	
7,6	Unassigned.
5	NwrRd: Direction control. Only significant if biden = 1 (see Byte Address 4, bit 4). If set, direction is "input from device". If clear, direction is "output to device".

4	IrqEnb: If set, enables NACK interrupt on trailing edge 0 to 1 transition affecting ONLY the INT2 line @ the WD chip. If clear, disables the NACK interrupt affecting ONLY the INT2 line @ the WD chip. WARNING: Refer to the warning given above regarding 0xF082 4801 bit 2.
3	Slin: If set, NSLIN = 1. If clear, NSLIN = 0.
2	Ninit: If set, NINIT = 1. If clear, NINIT = 0.
1	Autofd: If set, NAFD = 0. If clear, NAFD = 1. For Scanjet, if set, the WRnRd (nAFD) signal = 0 indicating to the ScanJet that the scanjet will drive the data bus (input mode). Otherwise, output mode.
0	Strobe: If set, NSTB = 0. If clear, NSTB = 1

- Read:

BIT	
2-0	Returns the corresponding signal value present at the parallel connector.
3	Always returns 0.
4	Returns value written.
7-5	Always returns 1.

6.5.5 Byte Address 0xF082 4804 (Mode Control)

Hardware supports both DMA and non-DMA transfers for automatic handshake modes 1, 2, 3, and 4. Mode 3 is the only mode which supports automatic handshake read operations from the peripheral.

CAUTION: Changing modes before a previous parallel data transaction is complete may cause unpredictable results. Software must always make sure previous transactions are complete before changing modes.

- Write:

BIT	
7-5	Handshake mode control

BIT7	BIT6	BIT5	
0	0	0	Mode 0 No automatic hardware handshake. Handshaking is under software control.

0	0	1	Mode 1 NACK pulse with not BUSY handshake. Handshake is complete if NACK pulse has completed and then BUSY=0. DMA or non-DMA transfers are allowed.
0	1	0	Mode 2 BUSY only handshake.
0	1	1	Mode 3 NSTB only handshake. Read from Scanjet NSTB input mode handshake. (BUSY and NSTB handshake directions are turned around by hardware).
1	0	0	Mode 4 Stream mode. Automatic NSTB generation with no NACK or BUSY handshakes. Hardware will make the data setup, strobe duration, and data hold times the same as determined by the delay value programmed via address F082 4806.
1	0	1	Same as decode "0 0 0" above.
1	1	0	Same as decode "0 0 0" above.
1	1	1	Same as decode "0 0 0" above.

4 Biden: When set, enables bidirectional mode capabilities. If this bit is set, software must also select the desired direction (see bit 5 of Byte Address 2). If this bit is clear, output only transfers are enabled regardless of the state of bit 5 of Byte Address 2.

If scanjet read "mode 3" is selected, this bit and bit 5 of Byte address 2 must also be set. Wait 5 usecs, then clear Wr/Nrd (NAFD) = 0. A transfer may then be invoked. This bit does not affect the WRnRD signal used by the Scanjet.

3 Force NSTB pulse. When 1, forces NSTB pulse to be low for the programmed delay period, i.e. NSTB 0 to 1 transition is not conditional on BUSY=1. Prevents deadlock condition for devices which only return BUSY=1 when NSTB makes a 0 to 1 transition. (such as the CP300 device in non-stream mode)

2 No affect.

1 No affect.

0 No affect.

- Read:

BIT	
7-0	Returns value written.

6.5.6 Byte Address 0xF082 4805 (IE Control/Interrupt Status)

Logic exists to prevent losing interrupts. The following example demonstrates the interrupt logic behavior:

- Two or more interrupts are pending and one or more bits of this interrupt enable mask register was set for one or more of the corresponding pending interrupts.
- As a result, the enabled Asp global parallel interrupt register bit got set thus causing the interrupt to be serviced by the host.
- The Host reads this interrupt status register, clears the Asp global parallel interrupt bit and then choses to clear only one of the pending interrupts here by writing this Interrupt Enable mask for that bit to zero but writes one(s) for the other pending interrupt bit(s).
- ANOTHER interrupt edge for the other pending AND enabled interrupt(s) here will be generated to the global parallel pending interrupt bit in Asp immediately following the write to this Interrupt Enable register.
- Also, if an interrupt here was pending but not enabled, and the host then enables this pending interrupt, an interrupt edge will be subsequently sent to the Asp global parallel interrupt register bit.

- Write

BIT	
7	DMA done IE. Enable interrupts for when DMA completes its block transfer. (Interrupt when the whole pipeline is empty.) Note: for mode 3 read operations: software must enable the NACK interrupt as well. Both interrupts must occur before the block transfer can be considered complete.
6	No affect.
5	Nbusy IE: Enable interrupts on BUSY trailing edge 1 to 0 transition.
4	Nack IE: Enable interrupts on NACK trailing edge 0 to 1 transition.
3	ackNbusy IE: Enable interrupts on NACK 0 to 1 transition was received and BUSY = 0 (not busy).
2	PE IE: Enable interrupts on any PE transition.

1	Select IE: Enable interrupts on any SLCT transition.
0	Error IE: Enable interrupts on any NERR transition.

- Read

BIT	
7	DMA done IR (0 to 1 transition). DMA has completed its block transfer. For mode 3 read operations: software must enable the NACK interrupt as well. Both interrupts must occur before the block transfer can be considered complete. This bit is cleared only by writing a zero into it.
6	Returns 0.
5	Nbusy IR: Busy 1 to 0 transition (not busy). This bit is cleared only by writing a zero into it.
4	Nack IR: NACK 0 to 1 transition occurred. This bit is cleared by writing a zero into it or by reading byte address 1.
3	ackNbusy IR: NACK 0 to 1 transition occurred and BUSY = 0. This bit is cleared only by writing a zero into it.
2	PE IR: The PE line has made any transition. This bit is cleared only by writing a zero into it.
1	Select IR: The SLCT line has made any transition. This bit is cleared only by writing a zero into it.
0	Error IR: The NERR line has made any transition. This bit is cleared only by writing a zero into it.

6.5.7 Byte Address 0xF082 4806 (Timing Delay Counter)

- Write

BIT	
7-0	Timing Delay Counter register. A counter divider which output is used by the controller state machine for establishing the minimum data setup and NSTB pulse times. Also establishes the data hold time when in mode 4 (stream mode). Note: The NSTB pulse will be low for the delay time programmed or the time it takes for BUSY to become true, whichever comes last.

This counter divider is needed so that the design can be used for varying peripheral/cable restrictions.

The formula for the value to load is:

VALUE TO LOAD =

((CPU FREQ. x DELAY TIME) - 22)/6

The standard "DELAY TIME" is typically equal to or greater than 1 x 10E-6 second, however, other values may be chosen depending on peripheral/cable restrictions.

Be sure to "round up" to the next higher integer value to ensure that the timing is not under the minimum delay time required. Some examples follow:

- calculated value = 4.6 ... round up to 5
- calculated value = 4.3 ... round up to 5
- calculated value = 4.0 ... use 4

- Read

BIT	
7-0	Returns value written.

6.5.8 Scanjet Activation Sequence Special Notes

The following is a rough outline of the Scanjet activation event sequence:

- Initialize the Scanjet: reset if necessary and send any desired configuration commands.
- Send the scan command.
- Set the mode register for mode 3.
- Prepare for DMA read from the Scanjet with DMA done interrupt enabled but don't enable DMA via DMA mask yet.
- Prepare direction for read from scanjet:
  - * Biden = 1 (0xf0824804 bit 4)
  - * NwrRd = 1 (0xf0824802 bit 5)
- Wait 5 microseconds.
- Make Nafd (WRnRd) = 0 (0xf0824802 bit 3). This enables the Scanjet to start scanning.
- Enable DMA transfer via the DMA mask.
- Wait for DMA done interrupt. (Nack pulse should occur after the

peripheral has sent the last byte but before the DMA done interrupt occurs. An NACK interrupt may be enabled if desired to detect this condition).

- CAUTION: ALWAYS return to the "write-to-peripheral" direction when you are done reading parallel data OR if a timeout occurs! This should always be the default direction to ensure that the parallel port data lines will not go to a "floating" logic state if someone unplugs the peripheral from the port. The WD16C552 is a CMOS IC. Floating inputs to a CMOS device may build up undue heat stress to the input buffers thus decreasing reliability.

6.5.9 Caveats when Accessing Parallel registers

When accessing byte address 0xF082 4800, 0xF082 4801, 0xF082 4802, or 0xF082 4803 (registers physically inside the WD16C552), and the mode selected is not mode 0, software must wait for previous data transactions to complete via interrupts to prevent the possible occurrence of a hardware lockup condition. Parallel registers at 0xF082 4804 or above (registers physically inside Asp), may be accessed any time without fear of causing a lockup condition.

In software, slave byte accesses of any WD16C552 register must meet a cycle time requirement of at least 250 ms. Reading any byte of the EEPROM before the WD16C552 chip access will guarantee the cycle time delay requirement is met.

CAUTION: It is possible for 2 consecutive WD16C552 slave transactions to "back-up" on the VSC bus. So, simply using a CPU clock based timer will not guarantee the cycle recovery time requirement between the two commands will be met.

6.6 Quick Summary Reference for Non-DMA Parallel Registers

BIT--------	7	6	5	4	3	2	1	0
BIT VALUE--	80	40	20	10	08	04	02	01
f082 4800	DATA							
f082 4801	Nbusy	Nack	pe	slct	Nerr	Nint	1	1
f082 4802	1	1	NwrRd	IrqEnb	Nslin	Ninit	Autofd	Strobe
f082 4803	Not Used (undefined)							
f082 4804	Mode[2]	Mode[1]	Mode[0]	Biden	fNstb	0	0	0
f082 4805	dmaInt	0	NbsyInt	NackInt	ackNbsy	peInt	slctInt	errInt
f082 4806	Timing Delay value							

6.7 Direct Memory Access (DMA)

Hardball's parallel port DMA controller attempts to emulate an EISA DMA controller by providing the same register map and counter behavior. To improve system performance Hardball's parallel port DMA provides a 32 byte FIFO. It should be relatively easy to create a printer/scanner driver that works for Hardball's internal parallel port and an add-on EISA parallel card.

6.7.1 Hardball DMA Register Map

DMA Register Map

ADDRESS	TYPE	SIZE(BYTES)	DESCRIPTION
F0820000	rw	1	DMA Current Address register
F0820001	rw	1	DMA Current Count register
F0820008	ro	1	DMA Status register
F082000A	wo	1	DMA Write single mask bit
F082000B	wo	1	DMA Mode register
F082000C	wo	1	DMA Clear byte pointer
F082000D	wo	1	DMA Master Clear
F082000E	wo	1	DMA Clear Mask register
F082000F	rw	1	DMA Mask register
F0820010	rw	1	DMA Fifo limit register
F0820087	rw	1	DMA Current Address low page register
F0820401	rw	1	DMA High Current Count
F082040A	rw	1	DMA Interrupt Pending register
F0820487	rw	1	DMA Current Address high Page register

6.7.2 How Parallel Port DMA Works

Hardball's Parallel Port DMA controller transfers data from/to memory to/from the parallel port without disturbing the CPU until the transfer sequence is complete. To start a sequence the DMA channel needs to have a beginning address and byte count placed into the proper registers. Given that the mode (read or write) is setup properly, DMA will start once the mask bit is reset. After the sequence is complete an interrupt will happen, the mask bit will be set and the address and count registers will be at there final value. This controller does not support chaining, so after each sequence the count and address registers need to be reinitialized to their starting values.

6.7.2.1 DMA Controller Problem

The DMA controller will not properly terminate a transfer sequence for some even byte counts greater than or equal to 64. Only a fraction of these even byte counts cause a problem, but a simple software work-around exists: always transfer odd byte counts. If an even count needs to be transferred subtract 1 from the count and transfer count-1, then transfer 1 byte.

This problem may be fixed in the future, the above software work-around will work on revised hardware.

6.7.2.2 DMA Control Bits

The control of Hardball's DMA is governed by 3 bits.

dma_mode_register	F082 000B{1:0}
dma_mask_bit	F082 000F{0}
byte_pointer	

The mode_register bits control whether the controller is in read or write mode. The resetting the mask bit starts a DMA sequence. The mask bit is automatically set at the end of the sequence. The byte_pointer determines which half of the 16-bit address/count registers is accessed when setting up the channel; this does not effect the DMA channel while DMA is in progress.

6.7.2.3 Bus Errors

If Hardball's Parallel Port DMA controller gets a bus error while mastering a transaction it will release the bus as soon as possible. The Current Address and Count registers will keep the values they had when the error happened; this should help in the debug process. All other state will be reset and whatever data was in the fifo at the time will be lost.

6.7.2.4 Typical DMA Sequence

This is what needs to be done to start a DMA sequence. Remember all the writes are byte writes to DMA control registers in I/O space.

```
;
; Assemble 32 bit physical address for base of transfer
;
Write Clear_Byte_Pointer
Write Current_Address   (low eight bits)
Write Current_Address   (high eight bits)
Write Page_Low
Write Page_High
;
; Assemble 24 bit transfer length
;
Write Clear_Byte_Pointer
Write Current_Count   (low eight bits)
Write Current_Count   (high eight bits)
Write High_Count
;
; Set DMA direction
;
Write Mode_Register
;
; Clear DMA mask bit
;
Write Mask_Register
```

When the DMA mask bit is cleared, F082 0008{0} is set to 1.

If the parallel port DMA enable bit is set, DMA transfer starts.
If the parallel port DMA enable bit is clear, the DMA transfer waits for the parallel port DMA enable bit to be set before starting.

Once the DMA transfer is complete:

```
        F082 0008{0} is cleared to 0    (DMA in progress)
        F082 040A{0} is set to 1        (DMA mask bit)
        F082 040A{0} is set to 1        (DMA interrupt bit)
        F082 4805{7} is set to 1        (parallel DMA terminal count)
        if F080 0008{7} is clear, F080 4008{7} is set and an
                external interrupt request is forwarded from Asp to Viper
```

Also, the parallel port handshake interrupt bits (~BUSY, Nack 0->1, etc) may be set as appropriate for the current handshake.

6.7.3 EISA Compatibility

Hardball's DMA controller used the EISA specification as a guide for it's original definition. This DMA controller acts like DMA channel zero of an EISA system. Mostly this adds a few things that appear wasteful for directed reads and writes to control ports. One example of this is the way a byte pointer is used to access the most significant byte of the address and count registers: the byte pointer allows 8 bit reads and writes to access a 16-bit register from one byte address. EISA is set up this way because way back in the evolution of the PC-AT the 8-bit Intel 8237 dma controller was popular. All accesses to the asp dma controller MUST be done with BYTE READS and WRITES, all other accesses (WORD etc.) will do nothing.

Here is a summary of known differences between the EISA specification and Hardball's internal DMA controller:

- The FIFO

  Hardball's DMA channel has a 32-byte FIFO integrated with the DMA controller. A FIFO limit register is placed onto the same 4K page as the EISA like registers of the DMA controller. So the register address map is not exactly the same as that of an EISA system.

- No Chaining

  In order to support DMA buffer chaining and autoinitialization, each EISA DMA channel has a 32-bit write-only Base Address register that is programmed with the base address for DMA transfer. It does not decrement or increment.

  In order to simplify the design, we make no attempt to support the DMA buffer chaining and autoinitialization for Parallel Printer Interface DMA due to its low data transfer rate. (maximum at ~ 400Kbytes/sec). Thus , inside Cutoff, we fold the Base Address register and the Current Address register into a single Address register.

  The write-only Base Address register and the read-only Current Address register are accessed through the same address; that is, when you write to that register, you write to the base address register and when you read from that address, the value you get is Current Address register. The same holds for the Current Count registers.

- No Auto-reset of Address/Count registers

  Writing to the less significant Address/Count register bytes automatically clears the high ones on an EISA system. Hardball's DMA controller doesn't do this; it's address and count registers are straight read and write registers- no tricks.

- Read/Write mode

  For Hardball's parallel port Read Mode is defined as reading from the parallel port and writing to memory. Write mode is defined as writing to the parallel port and reading from memory. In an EISA DMA controller Write mode means write to memory and Read mode means read from memory.

6.7.4 Detailed Register Descriptions

6.7.4.1 Current Address Register

F082 0000 (Current_Address) rw init: 0
This is a 16 bit register.
The byte pointer indicates which byte is accessed on a read or write. On the first access after reset, the byte pointer indicates to access the low byte. On subsequent accesses, the byte pointer indicates to access the high byte. The byte pointer is reset by power on, a write to F082 000C (Clear_Byte_Pointer), or a write to F082 000D (Master_Clear).

F082 0087 (Current_Address_Low_Page) rw init: 0
8 bit read/write

F082 0487 (Current_Address_High_Page) rw init: 0
8 bit read/write

DMAs current address is a 32 bit physical address constructed as follows:

```
+------------+-----------+-----------.-----------+
| High Page  |  Low Page  |      Current Address      |
+------------+------------+------------.------------+
```

Each EISA DMA channel also has a 32-bit read-only Current Address register. The DMA controller automatically increments or decrements the address after each transfer and the intermediate values of the address are stored in the Current Address register during the transfer. This register is cleared (set to 0x00000000) after reset.

In EISA, the way to access the Base or Current Address register is very cumbersome. The address register includes a 16 bit 8237 compatible segment (address 0000) combines with the low page segment (address 0087) and the high page segment (address 0487) to provide a 32-bit EISA DMA address. Listed below are the procedure to access the Address register:

a) CPU performs a write to the Clear Byte Pointer register. (000C)

b) CPU performs an 8 bit read/write to the least significant byte (bit 7-0) of the register 0000.

c) CPU performs an 8 bit read/write to the most significant byte (bit 15-8) of the register 0000.

d) CPU performs a bit read/write to the low page segment (address 0087) for Address register bits [23:16].

e) CPU performs a bit read/write to the high page segment (address 0487) for Address register bit [31:24].

6.7.4.2 Byte Count Register

F082 0001 (Current_Count) rw init: FFFF

This is a 16 bit register.
The byte pointer indicates which byte is accessed on a read or write. On the first access after reset, the byte pointer indicates to access the low byte. On subsequent accesses, the byte pointer indicates to access the high byte. The byte pointer is reset by power on, a write to F082 000C (Clear_Byte_Pointer), or a write to F082 000D (Master_Clear).

Note that this register counts down to -1, so (Current_Count + 1) bytes are transferred during a dma transfer.

F0820401 (Current_Count_High_Byte) rw init:FF
DMA transfer length is specified by a 24 bit register:

```
+------------+------------.------------+
| High Count |      Current Count      |
+------------+------------.------------+
```

As a DMA transfer progresses, the transfer length is decremented by the number of bytes transferred so far, and the physical address

increments by the number of bytes transferred so far.

For the same reason as in the case of the Base and Current Address register, we fold the Base and Current Word Count register into a single Byte Count register. This register is cleared (set to 0x000000) after reset.

The Byte Count register consists of two parts, the 16-bit 8237 compatible segment, and the 8-bit high byte count segment. The two segments are mapped at different I/O address and must be programmed separately. Listed below are the procedure to access the Byte Count register:

a) CPU performs a write to the Clear Byte Pointer register. (000C)

b) CPU performs an 8 bit read/write to the least significant byte (bit 7-0) of the register 0001.

c) CPU performs an 8 bit read/write to the most significant byte (bit 15-8) of the register 0001.

d) CPU performs an bit read/write to the high byte count segment (address 0401) for Byte Count register bit 23-16.

6.7.4.3 DMA Status register

F082 0008 (Status_Register) ro init: 0
bit 0 is cleared whenever the terminal count is reached.
bit 4 is set whenever the DMA channel is requesting servicing
the other bits are hardwired zero.
Simulator notes: bit 4 can be wired to zero for simulation.

The DMA Status register contains status information about the DMA channels that may be read by the CPU. The information includes which channels have reached a terminal count and which channels have pending DMA requests.
In Cutoff, we only support one DMA channel, thus only the status bits corresponding to channel 0 will be reported.

bit 0 - This bit is set every time terminal count is reached. This bit is cleared upon power-on and on each Status Read.

bit 4 - This bet are set whenever the DMA channel are requesting servicing. This bit will be zero when the requesting unit is reset, in this case the parallel port.

bits 1, 2, 3, 5, 6, and 7 are hardwired to zero.

6.7.4.4 DMA Write single mask bit

F082 000A (Write_Single_Mask) wo init: 0

This register may be used to set or clear any mask register bit.

bit[1:0] - must both be zero to write to write to this bit.

bit[2] - 0: Clear DMA mask bit
1: Set DMA mask bit (STATE AFTER RESET IS BIT 2 IS SET)

Examples:
A write of XXXXX000 clears the dma_mask_bit
A write of XXXXX100 sets the dma_mask_bit and starts dma transaction
A write of XXXXXX10 has no effect
A write of XXXXXX01 has no effect
A write of XXXXXX11 has no effect

6.7.4.5 DMA Mode register

F082 000B (Mode_Register) wo init: 01
bit[1:0] - must both be zero to write to write to these bits.
For example, 00001000 sets read transfer mode
00001011 changes nothing

bit[3:2] - Data Transfer Type
00 No transfer dma Disabled
01 Write transfer
10 Read transfer
11 No transfer dma Disabled

bit 4 - Cutoff hardwired to 0: Disable Auto-initialization

bit 5 - Cutoff hardwired to 0: Address increment select

bit[7:6] - Cutoff hardwired to 00: Demand Mode DMA transfer.

State after reset is 0x01: DMA is in write mode.

Examples:
A write of XXXX0000 disables dma
A write of XXXX0100 sets transfer type to write (Hardball to peripheral)
A write of XXXX1000 sets transfer type to read (peripheral to Hardball)
A write of XXXX1100 disables dma
A write of XXXXXX01 has no effect
A write of XXXXXX10 has no effect
A write of XXXXXX11 has no effect
bits 7-4 hardwired to zero

6.7.4.6 DMA Interrupt Logging Register

F082 040A (Interrupt_Log_Bit) rw init: 0

bit 0 - Indicates if there is a pending interrupt on Parallel Printer Interface DMA channel. This bit is cleared after reset.
bit[7:1] - always return zeros.

Writing 0xX0 or 0xXC will clear the pending interrupt. Writing anything else does nothing. This decodes dma channel zero as being the channel that gets its interrupt cleared.

read:

F082 0401{0} is 1 when interrupt is pending on DMA
F082 0401{0} is 0 when no interrupt is pending on DMA

write:
A write of XXXX1100 clears the pending interrupt
A write of XXXX0001 clears the pending interrupt
All other values have no effect

IMPORTANT NOTE:

This bit only logs the fact that a DMAs terminal count has changed; it does not generate an interrupt nor does it effect the state of anything. This bit is simply a log bit that redundantly logs interrupt information that is available elsewhere. THIS SINGLE BIT REGISTER CAN BE IGNORED.

Note:

In EISA this is also the chaining mode register which enables or disables DMA buffer chaining. Since DMA chaining will not be supported on Cutoff, writing to this register can only clear interrupts.

6.7.4.7 DMA Mask Register

F082 000F (Mask_Register) rw init: 00001110
Bit 1 is the dma_mask_bit.
Writing 1 to bit 0 enable (starts) the pending dma transaction.

This register is similar to DMA write single mask bit since we only have one DMA channel on Cutoff.

bit 0 - 0: Clear DMA mask bit
1: Set DMA mask bit (STATE AFTER RESET IS BIT 0 IS SET)
bit[3:1] always reads set because that's what EISA would do if only dma channel zero were being used.
bit[7:4] are reserved and always wired low.

6.7.4.8 Fifo Limit Register

F082 0010 (FIFO_Limit_Register) rw init: 10000000
{3:0} granularity of word transfers to peripheral
{7:4} granularity of word transfers to memory

NOTE: THIS REGISTER IS NOT OF THE EISA MODEL. EISA HAS NO FIFO.

This register sets the SGC bus request fence in the FIFO. On a write transfer if the number of words in the FIFO is less than or equal to the wlimit a memory read transaction will be mastered on the SGC bus. On a read transaction if the number of words is greater than or equal to the rlimit a memory write transaction will be mastered on the SGC bus. The dma controller doesn't release the bus until the FIFO is emptied or refilled.

bit[3:0] FIFO wlimit in words. Note: Only values 0,1,2,3,4,5,6,7 are valid; all other values are illegal and will do something bad. Set to 0 after RESET.

bit[7:4] FIFO rlimit in words. Note: Only values 0,1,2,3,4,5,6,7,8 are valid; all other values are illegal and will do something bad. Writing a 0 to this field will generate a flush to SGC. Set to 8 after RESET.

6.7.4.9 DMA Clear byte pointer

F082 000C (Clear_byte_Pointer) wo init: byte pointer = 0.

The Clear Byte Pointer command clears the internal latch used to address the upper or lower byte of the 16-bit address and word count register. The latch is also cleared at power-on and by DMA controller Master Clear command. For details, please reference to the EISA spec 3.1.8. The value written doesn't matter all that needs to happen is a byte write to this address.

6.7.4.10 DMA Master Clear

F082 000D (Master_Clear) wo init: no value
A write to this address clears the byte_pointer and sets the dma_mask_bit

The Master Clear instruction clears the command, Status, and Request registers, sets the Mask register to disable DMA requests and executes a Clear Byte Pointer command. The value written doesn't matter all that needs to happen is a byte write to this address.

6.7.4.11 Clear Mask register

F082 000E (Clear_Mask_Register) wo init: mask register = 1.
A write to this address sets the dma_mask_bit

The clear mask register command enables the DMA channel by clearing the mask bit. The value written doesn't matter all that needs to happen is a byte write to this address.

6.8 Testing

Testing the Parallel Printer Interface can be accomplished by attaching a Centronics Test Hood Box, E.T. ~~30733~~ which is capable of emulating all of the different peripherals supported by the design.

37310

6.9 Timing Examples

```
OUTPUT (Host writes):

             ___________  ______________  __
DATA         ___________|______________|__   Host Drives
             _______________       ________  Host Drives
NSTROBE                     |____|

             __________        ________
BUSY                   |______|        |___   Peripheral Drives

             _______   ____________   ___
NACK                |__|            |__|      Peripheral Drives

                       ^End of Last xfer (NACK pulse finished and Not BUSY)
                        ^Start of current xfer
                        ^DATA setup time will be the same as NSTROBE pulse width time
                             ^NSTROBE standard min. width 1us, handshakes with BUSY
                                      ^End of Last xfer
                                       ^Start of next xfer
======================================================================
INPUT (Scanjet protocol; Host changes direction then Scanjet writes to Host):
    (H) Host drives.
    (P) Peripheral drives.
    (N) Neither side is driving low.
    (B) Both sides drive low.

            ______________________
Wr/Nrd      (H)                   |_________________________
(Nafd)
            _______________ ______ _________________________
DATA        (H)____________|(N)   |_(P)_____________________

            (H) ______________________________          _______
NSTROBE     ___|               (N)      (P)      |______|

            _________                           __________
BUSY        (P)      |______(B)________________|          |_

            ______    ____________________________________ ... _  _
NACK        (P)   |__|                              *(P)     |__|

                  ^End of Last byte xfered (NACK pulse complete and Not BUSY)
                        ^Host sets up Scanjet read MODE 3 and BIDEN = 1
                         (then wait 5 microseconds. min.)
                              ^Host signals direction change to Scanjet
                               ^Start of Scanjet sending data to Host
                                ^Enable DMA transfer via DMA mask
                                                 *No NACK is sent by Scanjet
                                                  until after the last byte
                                                  is xfered to Host
```

6.10 I/O Connector

HOST SIDE BUS SIGNALS	HOST SIDE DIR	HOST DB-25 PIN #	36 PIN CONN PERIPHERAL PIN #	ALIAS NAMES
NSTB	I/O	1	1	DSTB-0, NSTROBE, nSTROBE
DB0	I/O	2	2	D0, DATA0, Data 1
DB1	I/O	3	3	D1, DATA1, Data 2
DB2	I/O	4	4	D2, DATA2, Data 3
DB3	I/O	5	5	D3, DATA3, Data 4
DB4	I/O	6	6	D4, DATA4, Data 5
DB5	I/O	7	7	D5, DATA5, Data 6
DB6	I/O	8	8	D6, DATA6, Data 7
DB7	I/O	9	9	D7, DATA7, Data 8
NACK	I	10	10	ACK-0, nACK, NACKNLG
BUSY	I/O	11	11	BUSY-1, Busy
PE	I	12	12	IR/SO, ERROR, paper ERROR, PError
SLCT	I	13	13	SLCT-1, SELECT, SelOut
NAFD	O	14	14	WRnRD, WR/nRD, NAUTO_FEED_XT, NAutoFd
NERR	I	15	32	NFAULT, FAULT-0, nFAULT, nFault
NINIT	O	16	31	NRESET, nRESET, nInit
NSLIN	O	17	36	NSLCTIN, NSLCT_IN, nSelectin, nSelIn
GROUND	-	18-25	19-29	(twisted pair ground returns see the following group for breakout)
GROUND	-	-	19	(twisted pair ground return for NSTB)
GROUND	-	-	20-27	(twisted pair ground return for D0-D7).
GROUND	-	-	28	(twisted pair ground return for BUSY)???
GROUND	-	-	29	(twisted pair ground return for NACK)???
	-	-	15	
	-	-	16	0 VDC
	-	-	17	CHASSIS GND
	-	-	18	+5 VDC
	-	-	30	SIGNAL GND
	-	-	33	AUXOUT1, Auxout1
	-	-	34	
	-	-	35	AUXOUT2, Auxout2

6.11 Electrical Specifications

See the Western Digital WD16C552 data sheet for details.

6.12 Power Requirements

See the Western Digital WD16C552 data sheet for details.

7. SERIAL CHANNEL COMMUNICATIONS

7.1 Introduction

There are two serial ports in the I/O subsystem each being fully compatible with the National NS16550A chip.

The serial ports are type RS232C which is a serial communications interface standard commonly used for modems, terminals, printers, and various other relatively low speed peripheral equipment.

7.2 Overview

Host communication to the serial ports and the status and control registers is done through "Cutoff".

7.3 Features

- Two ports implemented with two 9 pin male DB9 connectors w/Vectra/AT pinout
- EIA RS232C asynchronous type "D" supporting full CCITT V.24 modem control
- 5, 6, 7, 8 bits/char
- Odd, even, none, one, zero parity
- All cardinal baud rates: 110, 300, 1200, 2400, 9600, 19.2K, 38.4K
- 19.2Kbaud inbound w/no data loss w/software XON/XOFF flow control only
- 230.4Kbaud inbound w/no data loss w/hardware flow control using the RTS line in conjunction with software XON/XOFF flow control.
- Additional baud rates: 50, 75, 150, 600, 4800, 7200, 57.6K, 115.2K, 230.4K. Long cables, or short cables with baud rates above 57.6K (or maybe 115.2K), will require an external RS232 to RS422 converter box.
- Less than 0.003% skew error at all listed baud rate selections w/7.3728 MHz baud rate clock
- CTS (CB), DSR (CC), RI (CE), DCD (CF) modem status interrupts
- Scratch pad register available
- 1 start bit, and 1, 1.5, 2 stop bits
- 16 byte inbound and 16 byte outbound FIFO buffers
- 1, 4, 8, 14 byte programmable FIFO interrupt level
- Programmable loop-back control for testing
- Supported as console w/HP TERM 0, ANSI, ASCII terminals

- Each Asynchronous Communications Element is fully compatible with the National NS16550A

7.4 Register Set

The base address for accessing the serial interface registers is as follows:

- Serial channel #1: 0xF082 380X
- Serial channel #2: 0xF082 280X

See the Western Digital "WD16C552 Dual Enhanced Asynchronous Communications Elements (ACE) with Parallel Port" or National Semiconductor "NS16550A Universal Asynchronous Receiver/Transmitter with FIFOs" data sheets for register set for the standard byte address access and bit information.

For the special hardware flow control registers available for channel #1 and channel #2 only, see the Hardware Flow Control section.

7.5 Hardware Flow Control

Channels #1 and #2 only !

7.5.1 Introduction

Hardware flow control is accomplished by controlling the RTS line to the peripheral, thus preventing data overrun errors in the input FIFO or input holding register. This feature is intended for use with high speed serial devices which are capable of quickly suspending serial data transfers to the host when the RTS line is dropped by the host interface controller hardware.

RTS hardware flow control helps protect against input FIFO or input register overrun conditions but does not protect against a memory buffer overflow. XON/XOFF software flow control is still required to prevent memory buffer overflows.

Caution: Some modems will drop carrier if RTS is deasserted.

7.5.2 Theory

The hardware flow control is based upon the behavior of the RXRDY line coming from the WD16C552 serial controller. If the hardware flow control feature is enabled, and the RXRDY line is active, hardware will drop the RTS line to the peripheral. When the RXRDY line becomes inactive, hardware will reassert RTS. If hardware flow control is not enabled, RXRDY will have no affect on the data flow. Also, the RTS control register bit must be true to allow RTS flow control to work properly.

7.5.2.1 RXRDY Behavior

- RXRDY MODE 0: When in the SCC Mode (FCR0=0) or in the FIFO Mode (FCR0)=1, (FCR3=0) and there is at least 1 character in the RCVR FIFO or RCVR holding register, RXRDY will be active. Once it is activated, RXRDY will go inactive when there are no more characters in the FIFO or holding register.
- RXRDY MODE 1: In the FIFO Mode (FCR0=1) when the FCR3=1 and the trigger level or the timeout

has been reached, RXRDY will go active. Once it is activated, it will go inactive when there are no more characters in the FIFO or holding register.

Refer to the National Semiconductor data book for the NS16550 for more details about setting up different modes of operation for this signal.

7.5.3 Host Interface/Peripheral Interaction

Once the RTS line has been dropped, the peripheral should then suspend transmitting data to the host as quickly as possible to prevent a input FIFO or holding register data overrun condition. Just how quickly data transmission must be suspended will depend mainly upon five factors:

1) The host interface controller input FIFO trigger level setting.
2) The speed at which the CPU can service the FIFO interrupt.
3) The number of bytes that will be transmitted by the peripheral after RTS has been dropped by the host interface hardware.
4) The rate at which the FIFO will still be filled with the residual data coming from the peripheral.
5) The rate at which the CPU can empty the FIFO data.

When the FIFO becomes empty, RTS will be reasserted by the hardware signaling the peripheral to resume transmitting data.

7.6 Enabling Hardware RTS Flow Control

The address for accessing the following serial interface register is as follows:

- Serial channel #1: 0xF082 3804
- Serial channel #2: 0xF082 2804

7.6.1 Byte Address 4

- Write:

BIT	
7-3	See WD data sheets.
2	normRTS: When set, allows normal non hardware RTS flow control. When clear, enables RTS hardware flow control.
1-0	See WD data sheets.

- Read:

BIT	
2	Unassigned.

- Write/Read:

BIT	
7-3,1-0	See the Western Digital "WD16C552 Dual Enhanced Asynchronous Communications Elements (ACE) with Parallel Port" or National Semiconductor "NS16550A Universal Asynchronous Receiver/Transmitter with FIFOs" data sheets for register set access and bit information.

7.7 Power Up Reset

The address for a directed hard reset of the serial interface is as follows:

- Serial channel #1: 0xF082 3000
- Serial channel #2: 0xF082 2000

☞ Note: Writing any data to EITHER of these above memory locations causes hardware to pull on the master reset pin of the WD16C552 chip which in turn will cause BOTH RS232 channels, the WD parallel port outputs, NSTB, NAFD, NINIT, and NSLIN to be initialized to the reset state!

0xF082 1000 Writing anything to this address causes reset "hold" (also resets 8042).

0xF082 1C00 Writing anything to this address causes reset "release" (also releases reset of 8042).

☞ Important Software notes:

After a power up or directed reset, bootup firmware MUST release the reset hold condition. There is no minimum wait time requirement, but access of the 8042 must not occur until more than 400 milliseconds after releasing reset.

For Directed resets to the 8042 software should proceed as follows:

- Write to address 0xF082 1000 to set and hold the reset condition.
- wait more than 100 microseconds then write to address 0xF082 1C00 to release the reset condition.
- Software/firmware must wait more than 400 milliseconds before accessing the 8042 after a directed reset has occurred.
- I/O subsystem resets are treated the same as a directed reset.

Specific channel specific directed soft resets may be accomplished by writing to the appropriate register as described in the "WD16C552 Dual Enhanced Asynchronous Communications Elements (ACE) with Parallel Port" or National Semiconductor "NS16550A Universal Asynchronous Receiver/Transmitter with FIFOs" documents.

7.7.1 Caveats

See the cycle time requirements stated in section 6.5.8.

7.8 Testing

Testing may be accomplished by programming the ACE for the loopback mode of operation. In this mode data which gets written to the serial port will be automatically returned (looped back) thus making it possible to test the port without actually having a peripheral attached.

For more thorough testing, an external cable wired to interface a DTE to DTE may be used for an external loopback between serial port #1 and #2. Software may then transfer data both directions between the two ports.

7.9 I/O Connector

Channel #1 and #2:

DTE	HOST	HOST	
HOST SIDE	SIDE	DB-9	
BUS SIGNALS	DIR	PIN #	ALIAS NAMES
CD	I	1	CF, DCD
RXD	I	2	BB, RX DATA
TXD	O	3	BA, TX DATA
DTR	O	4	CD
GND	-	5	AB
DSR	I	6	CC
RTS	O	7	CA
CTS	I	8	CB
RI	I	9	CE

7.10 Electrical Specifications

For detailed information on the RS232C standard, see the document "Interface Between Data Terminal Equipment and Data Communication Equipment "Employing Serial Binary Data Interchange" and/or "EIA Standard RS-232-C".

8. SCSI

8.1 Introduction

SCSI (Small Computer System Interface) is a system level interface bus used to connect disc drives, tape drives, and other I/O devices to a computer system. Numerous workstations today support this bus standard, as SCSI is becoming the defacto disc interface standard. SCSI-1 consists of 8 bits of data at up to 4MB/sec of synchronous transfer rate. SCSI-2, however, allows for up to 10MB/sec transfer rate with 8 bits and up to 40MB/sec with an extended 32 bit data bus.

8.2 Overview

The Hardball/CORAL I/O subsystem will support SCSI-2 specification and HPCS (HP Common SCSI) command set. However, it will only support the 8 bit data bus configuration and will not support neither the 16 nor 32 bit data bus configuration.

The NCR53C700 intelligent SCSI controller chip is used in Hardball/CORAL. It can transfer data at the maximum rate of 6.25 MB/sec on the SCSI bus. On the host bus side, It has an on-chip 32 bit DMA engine which interfacing to the MUSTANG host processor, and a "script processor," which fetches it's own commands and performs SCSI transactions with minimal host processor intervention. For a detailed description, refer to the NCR53C700 data manual and programmers guide rev 2.5.

Besides the NCR53C700 chip, part of the Cutoff I/O controller chip is used to implement the control logics which interface between the NCR53C700 and the SGC bus.

8.3 Register Set

Following is the expanded SCSI subsystem register map looking from the host side to the SGC bus. Note that the bytes are already swapped from the "Little Endian" convention on the I/O subsystem bus to the "Big Endian" convention on the SGC bus.

Register 0XF0800024 is a read only register. Bit 2-0 are the SCSI initiator ID. Software should read these three bits to set the SCSI subsystem's ID as an initiator. (i. e. Bit 2-0 = 000 means the initiator address is 0) Bit 3 is the status bit for the termination power on the SCSI connector. a "0" on this bit means power is lost. Software could use this bit to determine if an interrupt from the SCSI subsystem is due to the lost of termination power or other proper causes.

Register 0XF0825000 is a write only register. A write to this register will cause a direct reset to the SCSI subsystem.

Registers 0XF0825100 to 0XF082513F are implemented inside the NCR53C700. See the NCR53C700 data sheet for the definitions of each field within each register.

Bit 11:9 of Register 0XF0800024 also contains the Snakes System ID. Register 0XF080F024 contains the Rev level of the IO subsystem.. Drivers can use the information in the above two registers to optimize performance.

Address (Hex)	Byte 0	Byte 1	Byte 2	Byte 3
. .	. .			
F0800024	Status (R)			
. .	. .			
F0825000	SCSI reset (W)			
F0825004	RESERVED			
. .	. .			
F08250FF	RESERVED			
F0825100	SCNTL0 (R/W)	SCNTL1 (R/W)	SDID (R/W)	SIEN (R/W)
F0825104	SCID (R/W)	SXFER (R/W)	SODL (R/W)	SOCL (R/W)
F0825108	SFBR (R)	SIDL (R)	SBDL (R)	SBCL (R)
F082510C	DSTAT (R)	SSTAT0 (R)	SSTAT1 (R)	SSTAT2 (R)
F0825110	RESERVED			
F0825114	CTEST0 (R)	CTEST1 (R)	CTEST2 (R)	CTEST3 (R)
F0825118	CTEST4 (R/W)	CTEST5 (R/W)	CTEST6 (R/W)	CTEST7 (R/W)
F082511C	TEMP-B0 (R/W)	TEMP-B1 (R/W)	TEMP-B2 (R/W)	TEMP-B3 (R/W)
F0825120	DFIFO (R/W)	ISTAT (R/W)	RESERVED	
F0825124	DBC-B0 (R/W)	DBC-B1 (R/W)	DBC-B2 (R/W)	DCMD (R/W)
F0825128	DNAD-B0 (R/W)	DNAD-B1 (R/W)	DNAD-B2 (R/W)	DNAD-B3 (R/W)
F082512C	DSP-B0 (R/W)	DSP-B1 (R/W)	DSP-B2 (R/W)	DSP-B3 (R/W)
F0825130	DSPS-B0 (R/W)	DSPS-B1 (R/W)	DSPS-B2 (R/W)	DSPS-B3 (R/W)
F0825134	DMODE (R/W)	RESERVED		
F0825138	RESERVED	DIEN (R/W)	DWT (R/W)	DCNTL (R/W)
F082513C	RESERVED			
. .	. .			
F08251FF	RESERVED			
. .	. .			
F082F024	IO VCB			
. .	. .			

8.3.1 Effects of data byte swapping

Swapping the data bus from the "Little Endian" convention on the I/O Subsystem to the "Big Endian" convention on the SGC bus means the following to the SCSI driver.

Definitions of each field within each register stay the same. For registers with address range from 0XF0825100 to 0XF082513F, refer to the NCR 53C700 data sheet - Rev 2.4 for more details.

Addressing to the registers set remains the same. e. g. to access the SCNTL0 register, the address "SCSI_slot_base_add + 00" should be supplied, to access the SCNTL1 register, the address

"SCSI_slot_base_add + 01(byte_offset)" should be supplied, and etc ... In this case, the "SCSI_slot_base_add" = 0XF08251.

For byte wide registers, the SCSI driver needs not to do extra work since the byte swapping is already done in hardware. The following table illustrates the locations of data field for registers with different address offsets.

address with offsets	data field on SGC
0, 4, 8, and C	D31 - D24
1, 5, 9, and D	D23 - D16
2, 6, A, and E	D15 - D8
3, 7, B, and F	D7 - D0

For registers which have more than one byte, it is the SCSI driver's responsibility to assemble the bytes correctly in a write transaction and disassemble the bytes correctly in a read transaction. For example, if the data 0x12345678 is intended to write to the DNAD register (address = "SCSI_base_add + 028"), then the data pattern 0x78563412 should be supplied from the host on the SGC bus. The SCSI registers with more one byte are listed below:

TEMP (4 bytes)
DBC (3 bytes)
DNAD (4 bytes)
DSP (4 bytes)
DSPS (4 bytes)

In the above swapped SCSI register map, the registers with more than one byte are broken down into bytes and a suffix is added to their name to indicate the byte number. (e. g. DBC-B0 stands for byte 0 of the DBC register) Remember that in the "Little Endian" convention, the higher number byte is the more significant byte.

9. LOCAL AREA NETWORK (LAN)

9.1 Document Version

This is version 1.37 of the LAN section.

9.2 Introduction

Hardball implements a local area network (LAN) to the 802.3/Ethernet standard. Ethernet is a 10 Mbit/sec. packet-switched serial interface employing Carrier Sense Multiple Access/Collision Detection (CSMA/CD).

9.3 References

- Intel 82596DX/SX High-Performance 32-Bit Local Area Network Coprocessor (Order Number 290219)
- Intel 82596 User's Manual (Order Number: 296443-001)
- Intel 82596 Data Sheet Supplement November 1989 (see Rob Snyder)
- Intel Microcommunications Applications, vol. 1 & 2 (Order Number: 231658)

9.4 Nomenclature and Conventions

Note that an Intel "word" is 16 bits, and a PA-RISC "word" is 32 bits. From left to right, Intel numbers its bits from 31 to 0, whereas PA-RISC numbers bits from 0 to 31.

9.5 Overview

The Hardball built-in LAN is divided into two main blocks: the backplane interface to SGC and the frontplane interface to the network cable.

The frontplane consists of the Intel 82596DX - 82C501AD chip set, plus a transceiver chip and associated circuitry. The 82596DX is an intelligent, high-performance 32-bit LAN controller that interfaces between the host system and the 82C501AD. The 82C501AD is an Ethernet Serial Interface that generates the 10 MHz transmit clock for the LAN controller, performs Manchester encoding/decoding of the transmitted/received frames, and provides the electrical interface to the Ethernet transceiver cable (AUI). Programmers need only be concerned with the 82596DX.

To see how the LAN controller fits into the core I/O subsystem, please refer to the block diagram at the end of this document. The 82596DX has a four channel DMA controller which allows it to communicate directly with the main memory via the SGC interface. The four channels are: CU (transmit header), TXD (transmit data), RU (receive header) and RXD (receive data). Following is a brief description of the shared memory model, repeated without permission from the Intel 82596 User's Manual, section 2.4.3. The reader is strongly encouraged to consult this book for information about the chip:

> "To off-load the CPU the 82596 implements a shared memory communication system with the host CPU. The 82596 and CPU do not communicate directly, but rather through a shared system memory "mailbox." This allows the CPU to place commands in the mailbox, activate Channel Attention to notify the 82596 of delivery, and return to its other processing chores. The 82596 checks its mailbox in response, retrieves and interprets the commands, and executes them without further CPU interaction. After the 82596 completes its tasks it places the results in system memory and updates the mailbox. Then the 82596 Uses its interrupt line to notify the CPU of the presence of return mail...."

For normal DMA operations, Cutoff arbitrates for SGC on the 82596DX's behalf, manages the address valid/ready handshake, and synchronizes the address and data buses via the transceivers.

9.6 Transaction Types

Apart from normal communication via shared memory, a limited number of operations directly to the chip are available:

9.6.1 Channel Attention

To issue a Channel Attention to the 82596DX, do a word write to the LAN Channel Attention Register, at 0xF0826008. No data is associated with this operation.

9.6.2 Port Access

The 82596DX's "CPU Port" provides 4 functions:

- alternate System Configuration Pointer (SCP) address
- Dump command
- software reset
- self-test

The port is memory mapped. It is accessed with two consecutive 32-bit word writes to 0xF0826004. Valid data should be placed on the rightmost/least significant/highest address 2 bytes. The other two bytes are undefined. First write the "Upper Command Word" and then the "Lower Command Word". See the 82596 User's Manual, pp. 5-18:5-20 for instructions on how to encode the data.

9.7 Reset

The 82596 has a hardware reset, CPU Port reset (see Port Access section above), and a software reset. In addition, a word write to the I/O Subsystem Reset register at 0xF082 F000 will cause a hardware reset of the whole core I/O board, LAN included.

9.7.1 Consequences

Resetting the chip does NOT trigger self-test.

9.7.1.1 Hardware Reset

The hardware reset causes the chip to immediately cease all activity; the CU and RU become IDLE and clear all internal requests.

9.7.1.2 CPU Port Reset

The CPU Port reset causes the chip to immediately cease all activity and execute a software reset.

9.7.1.3 Software Reset

The CU performs the following on recognition of a software reset:

- Terminates DMA activity.
- Writes zeros to the SCB Command word.
- Triggers a hardware reset.

9.7.2 Protocol

9.7.2.1 Hardware Reset

After power up, the 82596 requires a hardware reset. Cutoff will do this automatically as a result of the SGC RESET.L signal, which by definition is asserted on power up. *Code* must wait for 10 system clocks and 5 transmit clocks (20 processor clocks + 0.5 microseconds for 10 Mbit/sec LAN) before doing a Channel Attention after a hardware reset. The LAN subsystem hardware will *not* check for the proper interval. **Code can cause a hardware reset by writing to 0xF0826000.** A Channel Attention following a hardware reset will cause the 82596 to access the SCP, which is located at 0x00FFFFF4 (or at an alternative address selected via the CPU Port). After Channel Attention the 82596 will read the sysbus byte and begin the initialization process.

9.7.2.2 CPU Port Reset

For information on CPU Port operations, see Port Access section above. A Channel Attention following a CPU Port reset will cause the 82596 to access the SCP, which is located at 0x00FFFFF4 (or at an alternative address selected via the CPU Port). After Channel Attention the 82596 will read the sysbus byte and begin the initialization process. The CPU must wait for 10 system clocks and 5 transmit clocks (20 processor clocks + 0.5 microseconds for 10 Mbit/sec LAN) before issuing another Channel Attention to the 82596. The LAN subsystem hardware will *not* check for the proper interval.

9.7.2.3 Software Reset

A software reset is available through (Intel numbered) bit 7 of the control command word int the SCB. It can be used after the 82596 has been initialized and has the ISCP and SCP addresses.

9.8 Interrupts

Cutoff provides a uniform interface for all I/O interrupts, including LAN (see interrupt section of this document). Interrupts are generated by one or more of the following events (from page 3-46 of the User's Manual):

- Execution of a Command Block with its *I* bit set (*CX* interrupt).
- Reception of a frame (*RU* interrupt).
- The CU becoming not active (*CNA* interrupt).
- The RU becoming not ready (*RNR* interrupt).

For more information about what these actually mean, consult section 5.3 of the User's Manual.

9.9 Programming Considerations

Note: Several errata have been discovered in the Intel chip and documentation. Some are mentioned below, but firmware, driver and software writers are strongly encouraged to consult the errata sheet.

9.9.1 Endian Mode

The 82596 will be set to operate in big endian mode to be compatible with PA-RISC. Programmers should take care to consult the big endian sections of the various references.

9.9.2 Bus Size

Core I/O provides a 32 bit data bus to the LAN controller.

9.9.3 System Configuration Pointer

The SCP defaults to 0x00FFFFF4. If this is unacceptable given memory configuration and other system parameters, firmware/software must write an acceptable value to the LAN Port Select (see above) between reset and first Channel Attention. The reader is reminded that alternate SCP addresses must be divisible by 16. In addition, *the SCP must reside in system memory.* The Intel book suggests putting the SCP in ROM, but in Hardball, the ROM is on the core I/O board, and on SGC, core I/O can only talk to the host. All data structures must reside in main memory.

Please make sure that any bits marked "x" in the SCP description are set to 0. Intel informs us that the chip will not work otherwise.

As always, see the manual for more information.

9.9.4 Sysbus Byte

The sysbus byte must be located at byte offset 7. Following is a discussion of the sysbus configuration byte. Please refer to section 5.4 of the manual.

9.9.4.1 Mode

Presumably, the 82596DX will be used in the "linear" mode. Set the bits accordingly.

9.9.4.2 Bus Throttle and Arbitration

In brief, this is how arbitration works: When the LAN needs the bus, it pulls its HOLD line. Based on the core I/O priority scheme, Cutoff arbitrates for SGC on the LAN's behalf, then grants it the bus by asserting

HLDA. The 82596 then has the bus for as long as it wants, which in general should be less than 5 us.

By default, we will not use the bus throttle features. To do this, **set the TRG bit of the sysbus byte to 1** for external bus release triggering. The external trigger is hardwired inactive. These two measures in tandem with the Cutoff arbitration mechanism allow the 596 to have the bus for as long as it needs to finish all pending work. However, if during system testing we find that the 596 hangs onto the bus to the extent that it degrades system performance, it would be desirable to have an easy (i.e. run-time) way of 1) enabling the internal bus throttle trigger by setting the TRG bit to 0, and 2) configuring the 596 throttle registers.

9.9.4.3 Locked Cycles

The LOCK bit of the sysbus byte should be set to 0 to enable "semaphore" operations on UPDATE_ERR_CNTRS and RCV_RBD_PREFETCH.

9.9.4.4 Interrupt Pin Polarity

The INT bit of the sysbus byte should be set to 1 to force the INT pin to be **active low.**

9.9.4.5 Channel Switch Algorithm

Intel informs us that the CSW bit of the sysbus byte must be set to 1 for correct chip operation.

9.9.5 Performance Considerations

9.9.5.1 FIFO Vector

Although the 82596DX has large FIFOs, 128 bytes on receive and 64 on transmit, certain system configurations, especially in Coral, may impose long bus access latencies on the LAN. The FIFO has a programmable threshold. If the FIFO *vector* (note this is *not* the same as the threshold *value*) is set too high, the chip will frequently request the bus, thus causing inefficiency due to arbitration overhead. Refer to section 7.3, *Setting the FIFO Thresholds,* in the User's Manual. If it is set too low, the FIFO may overrun or underrun before getting the bus. The moral is that we need to do lots of system performance simulations to make a reasonable recommendation, and this parameter may have to be tuned after we examine real systems. In pathological cases, it may also be necessary to give core I/O priority on SGC via Viper diagnose commands. See the separate writeup on bus latency for more information.

9.9.5.2 Memory Organization

In general, simplified, linear memory structures aligned on 4 byte boundaries, and larger size for data buffers, will tend to help LAN hardware performance, but at a cost to efficient memory utilization.

9.10 Station Nodal Address

A permanent copy of the LAN Station Nodal Address is kept in location 0 of the non-volatile RAM (EEPROM). Consult the NVRAM section of this document for location and access information.

9.11 Ethernet

The hardware design is 802.3 compliant and Ethernet rev 2 compatible. There is no need for a jumper.

9.12 Front Panel LEDs

The 68040/Apollo machines indicate LAN activity on the front panel LEDs. There are no LEDs inherently associated with the LAN hardware. **It is the responsibility of the firmware/driver/IODC/software to deal with this. If you need LEDs for some reason, access them through the front panel interface. See PDH section for further information.**

10. PROCESSOR DEPENDENT HARDWARE (PDH)

10.1 Introduction

For definition purposes, the PDH includes boot ROM, EEPROM non-volatile memory, status switches, and status LEDs, as well as the 8042 slave subsystem which consists of the RTC, audio generator, and HP-HIL.

10.2 System Boot-up Procedure

Upon boot-up, the host will read data from the boot ROM starting at address 0xF0000004. The first word in the PDC IO space is the HPMC vector address, 0xF0000004.

The boot ROM is byte-wide and must be assembled into word format for host execution. The I/O subsystem controller will do the proper byte assembly.

10.3 Boot ROM

There is one 512Kx 8 EPROM available.

10.4 Non-Volatile Memory (EEPROM)

The I/O subsystem has an 8K x 8 EEPROM which may be used for storing system configuration status and any other alterable, non-volatile information.

The manufacturer of the EEPROM guarantees reliability of at least 10 years with a maximum total number of write cycles of 10,000 for any given byte.

10.4.1 Enabling Writes to the EEPROM

In order to write to the EEPROM, the EEPROM write enable control bit must be set by writing bit 7 to a 1 at any byte address in the range 0xF0810000 to 0xF081FFFF. Clearing the bit 7 disables writing to the EEPROM.

☞ Software should protect the EEPROM from unauthorized writes.

10.4.1.1 Writing Data to the EEPROM (non-volitile memory)

The Xicor EEPROM features DATA bar polling as a method to indicate to the host system that the byte(s) written during an automatic internal page write cycle (or programming cycle) has completed. DATA bar polling allows a simple software bit test operation to determine the status of the X2864B, eliminating additional interrupts inputs or external hardware. During the internal programming cycle, any attempt to read the LAST BYTE written will produce the complement of that data on the MS bit 7 (ie write data = 0xxx xxxx, read data = 1xxx xxxx). Once the automatic programming cycle is complete, I/0-7 will reflect true data. The automatic internal programming cycle begins if more than 100 microseconds elapse between successive bytes written into a page. Attempts to write any bytes once the automatic internal programming cycle begins will not succeed.

The power up to read operation minimum delay is 0 ms.

The power up to write operation minimum delay is 4 ms.

Software may write the EEPROM according to the following algorithm:

- Enable writes to the EEPROM as previously described.
- Interrupts should be turned off. Software needs to avoid the problem of another interrupting piece of code that also wants to program the EEPROM.
- Wait 10 microseconds minimum (to satisfy the pre-write delay time from any other previous polling operations which may have occurred).
- Write any number of bytes between 1 and 32 which are to be loaded into a page as determined by address bits [19:26] (big endian, PA-RISC) or [12:5] (little endian). The bytes within a page may be written in any order. (The page address is latched on the latest byte written). Software must wait ☞1 microsecond☜ between writing successive bytes within a page.
- Wait/poll for the write (or writes) to complete which typically takes 3 ms and a maximum of 10 ms. Software may use the data polling technique as described earlier do determine if the automatic write cycle is complete for a page.

  Software should not attempt to write a different page unless the EEPROM internal automatic page write cycle is complete.
- If, after any byte is written, more than 100 microseconds elapse causing the automatic internal programming cycle to begin, all subsequent bytes will fail to write. This condition should be rare. However, to remedy this possibility, software should always verify all the bytes which were written, repeating the write sequence for any bytes which did not write successfully. ONLY the bytes which did not write successfully should be rewritten in order to maximize the useful life time of the EEPROM.
- Continue the sequence for any other pages which need to be written.
- Turn interrupts back on.
- Return to normal software routines.
- There are no software constraints for reading bytes from the EEPROM once the programming is complete.

10.4.2 EEPROM Memory Map

Base address for EEPROM = 0xF081 XXXX

```
Offset          Byte 0
------          ------
 0000           Byte 0 of LAN nodal station address
 0001           Byte 1 of LAN nodal station address
 0002           Byte 2 of LAN nodal station address
 0003           Byte 3 of LAN nodal station address
 0004           Byte 4 of LAN nodal station address
 0005           Byte 5 of LAN nodal station address
 0006           Byte 6 of LAN nodal station address

 0007           Reserved for hardware configuration information
   |
 03FF

 0400           3K bytes reserved for EISA configuration information
   |
 0FFF

 1000           3K minus 32 byte reserved for firmware usage
   |
 1BDF

 1BE0           1K Reserved for diagnostic usage
   |
 1BFF

 1C00           1K Reserved for EISA configuration information
   |
 1FFF
```

```
EEPROM write enable:
==================

 8000           EEPROM access enable
   |
 8FFF
```

A byte write of 1XXXXXXX binary into the address offset range 8000-8FFF enables write access to the EEPROM. The value of this bit can be seen by reading any address from the address range 8000-8FFF. The most significant bit is the enable bit.

```
      Any Byte Address in 8000 to 8FFF
      -------------------------------
      Enable bit = Exxxxxxx
```

E = 0, writes are not enabled into EEPROM (this is default state, power-up and other system resets should clear this bit, firmware should leave this bit in the 0 state prior to exiting to ISL or the OS)

E = 1, writes into EEPROM are enabled, allowing the contents to be changed.

10.5 8042 Subsystem (RTC, Timers, Audio Generator, HP-HIL)

The 8042 Subsystem is composed of several I/O devices: battery backed Real Time Clock (RTC), system timers, user timers, audio generator, and HP Human Interface Loop (HP-HIL). These devices are controlled via an Intel 8042 slave microprocessor which acts a "server" for these devices. Access to the devices is done through the 8042 command/data protocol under "Cutoff" control.

10.5.1 External Reference Documentation

See the document "System Device Controller Microprocessor Firmware Theory of Operation for Part Number 1820-4784 Revision B", drawing number A-1820-4784-2 for complete 8042 external reference details. The document was written assuming a 68000 host processor in the series 200/300 products, however, since Serpent is leveraging the same subsystem, the document would also apply to the Serpent implementation with the following exceptions:

address space (in HEX) have been remapped according to the following table:

428001 → 0xF082 1800 data I/O

428003 → 0xF082 1801 status/control

478005 → 0xF080 0000 system device interrupt register

10.5.2 8042 Firmware Documentation

Software and "burn" file references for the 8042 can be found in document number A-1820-4784-3, "Software Loading Instructions for System Device Processor 1820-4784 Revision B".

10.5.2.1 8042 Reset

Reset control addresses:

0xF082 1000 Writing anything to this address causes reset "hold" (also resets the Domain keyboard SIO #3 [no components will be loaded for this port]).

0xF082 1C00 Writing anything to this address causes reset "release" (also releases reset of the Domain keyboard SIO #3 [no components will be loaded for this port]).

A power on reset also causes a reset HOLD condition.

Software must allow a minimum of 400 ms after reset is asserted before accessing anything in the 8042 subsystem.

During the 400 ms reset time, the 8042 performs a selftest.

☞ Important Software notes:

After a power up or directed reset, bootup firmware MUST release the reset hold condition. There is no software minimum wait time requirement releasing the reset, but access of the 8042 must not occur until more than 400 milliseconds after releasing reset.

For Directed resets to the 8042 or Serial channel #3, software should proceed as follows [this port is not supported; no components will be loaded]:

- Write to address 0xF082 1000 to set and hold the reset condition.
- wait more than 100 microseconds then write to address 0xF082 1C00 to release the reset condition.
- Software/firmware must wait more than 400 milliseconds before accessing the 8042 after a directed reset has occurred.
- I/O subsystem resets are treated the same as a directed reset.

10.5.3 8042 Register Map

The register map for the 8042 subsystem is described in the document "System Device Controller Microprocessor Firmware Theory of Operation for Part Number 1820-4784 Revision B", drawing number A-1820-4784-2.

10.5.4 Caveats for 8042 Accesses

Software should make certain HIL auto-polling is disabled before sending HIL access commands. If the HIL loop devices are busy responding to auto-polling commands from the 8042, other commands may cause a command conflict and thus command execution failure.

Normally, software interactions with the 8042 are "paced" via the 8042 general interrupt mechanism. One exception occurs when software reads the 8042 status register, immediately following a RESET of the 8042, in order to clear the "I passed self-test" 8042 interrupt. The 8042 internal program takes some amount of time to clear this interrupt bit. If software issues a command to the 8042 or reads the 8042 status before the 8042 is done with its interrupt clearing algorithm, the command may be missed and may prevent the interrupt bit from getting cleared as well. There is no chance of this happening when instructions are being fetched from PDC or IODC ROM because a word access from ROM takes about 100 CPU clock cycles, allowing more than enough time for the 8042 to clear its interrupt bit. However, if software is executing from cache, it would be possible for another 8042 access command to come too quickly thus causing the problem mentioned above.

If the latter is the case, doing a dummy read of the EEPROM before issuing an 8042 command is a good way to guarantee the needed delay time is satisfied.

10.6 Front Panel for Trailways, Strider, and Hardball

Trailways and Strider are included for reference.

Front Panel when units are in vertical position:

Trailways (Top)	Strider (Top)	Hardball (Top)
		1. +5 V On (Grn)*
1. +5 V On (Grn)*		2. BR Diag/Lan Xmit*(LSB)
2. BR Diag/Lan Xmit*	1. BR Diag/Lan Xmit*	3. BR Diag/Lan Recv*
3. BR Diag/Lan Recv*	2. BR Diag/Lan Recv*	4. BR Diag/Disc Act*
4. BR Diag/Disc Act*	3. BR Diag/Disc Act*	5. BR Diag/Heartbt*
5. BR Diag/Heartbt*	4. BR Diag/Heartbt*	6. BR Diag
6. BR Diag	5. BR Diag	7. BR Diag
7. BR Diag	6. BR Diag	8. BR Diag
8. BR Diag	7. BR Diag	9. BR Diag(MSB)
9. BR Diag	8. BR Diag	10. Service On (Grn)
10. Service ON (Grn)	9. Service On (Grn)	
Service Toggle (rear)	10. Service Pushbutton	11. Service Pushbutton
Reset Pushbutton (rear)	11. Reset Pushbutton	12. Reset Pushbutton
DC On/Off (on "anchor")	DC On/Off (upper left)	DC On/Off (upper right)

Notes:

- When "laid down" on a desk, Strider rotates CCW, Hardball CW.
- DC On/Off is a latching pushbutton switch on all.
- BR Diag: Abbreviation for "Bootrom Diagnostics".
- LEDs marked with asterisks (*) are always visible. Other LEDs are behind the door.
- All LEDs are Yellow unless otherwise indicated.
- Strider's "+5 V On" LED is located near the On/Off switch (at the top-left of the unit, when the unit is in the vertical position).
- The first Trailways units will not implement the LEDs as shown above, because the parts have already been designed. However, subsequent units will implement the LEDs as indicated.
- The "service" momentary contact pushbutton switch is used to set a

"service mode" flip-flop in Strider and Trailways. This flip-flop is cleared at power-up (by RESET.L). Provisions are made for manufacturing to jumper this and force power-up in service mode. Trailways simply has a slide switch to select Service or Normal modes. The "Service On" LED reflects the mode the machine is in.

10.6.1 Hardball Implementation

See VSC Spec for signal pinouts, terminations, drive levels, etc.

10.6.2 +5 On

LED tied to +5 output.

10.6.3 BR/DIAG

All eight BR/Diag LED's are driven serially by PDH under direct software (PDC) control. LSB and MSB are indicated. Data on LEDDATA driven into LSB and shifted "down" (when looking at above diagram). Sense is TBD. Data is clocked by LEDSTB.L. PDC must take this line from 1 to 0 to 1 for each shift while writing the same data both times. In software, then, this is simply two writable lines that don't glitch. Software should be able to write these control lines without concern for inter-write wait times.

10.6.4 Service

Machine mode is readable by PDC in a bit that indicates state (SERV.NORM). In Hardball, the storage flip-flop is on the Connector Board. (see VSC) High (true) indicates Service mode.

10.6.5 Reset

In Hardball, this switch maps to "Transfer of Control", a non-maskable interrupt that goes through Viper to the CPU. When the button is pressed, TOC.L is momentarily grounded.

PDC must debounce (as in PCXN).

NOTE: The processor should provide a pullup on this line, per VSC.

10.6.6 DC On/Off

In the On state, POO is grounded by the switch. In the off state, it is pulled up by the power supply.

10.7 Cutoff Status LEDs Control and Status Register Access

The design objective is to have as few wires as possible connecting to the front panel status LEDs, and switches.

For the LED control, 8 bit chunks of data will be serially shifted automatically into a register located on the front panel. An eight bit register on the core I/O board is required to accomplish this function.

The O.S. must maintain a data image shared by SCSI, LAN and FDDI drivers that reflects the current state

of the front panel. The O.S. should also be responsible for periodic updates of the front panel LEDs to reflect the disk and LAN activity.

The Service/Normal switch is located on the front panel and may be read by the host via the I/O subsystem status register. Other system status information may accessed as indicated in a subsection below.

10.7.1 Status LED Control Register Map

Byte address 0xF080 0020

S/W loads the byte and hardware automatically shifts the 8 bits out to the front panel, with the most significant bit first.

Bit 0 is least significant bit and bit 7 is the most significant bit.

The register does not have read capability.

10.7.2 Status Register for the I/O Subsystem

WORD address 0xF080 0024

For the following table, bit 0 is on the right and is least significant:

- Read

BIT
[2:0] SCSI ID. (see table below)

[2:0]	SCSI ID
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

3 SCSI termination power. Power lost = 0.

4 LAN AUI fuse status. Fuse blown = 0, fuse intact = 1.
Note: This is only valid when the AUI port is selected.

[6:5] LAN jumper status
hardcoded to 01

7 SCSI select. 1 indicates 2X speed clock is selected; 0 indicates 1X speed clock is selected.

8 Front panel Service/Normal toggle switch value.
1 indicates Normal mode. 0 indicates service mode.

[11:9] SPU ID from board jumpers.
000 = Cobra
001 = Coral
010 = Bushmaster
011 = Hardball
100 = Scorpio
Other values = reserved for future machines.

[13:12] Network ID
00 = LAN-802.3
01 = undefined
10 = undefined
11 = FDDI

14 Fast/Wide SCSI termination power. Power lost = 0.

15 FDDI speed select. 1 indicates 1X speed clock is selected; 0 indicates .5X speed clock is selected.

16 LAN controller signals muxed:
0 = no
1 = yes

[31:17] Undefined.

☞ Differences from ASP to Cutoff status register: LAN jumper status is hardcoded to 01 which was the AUI port on Cobra; on Cobra, the user could select between AUI and ThinLAN. Added bits for FDDI speed select, network ID, and Fast/Wide SCSI termpower.

11. FAST WIDE SCSI

11.1 Introduction

FAST WIDE SCSI (FWSCSI) is an extension to the existing 8 bit single ended SCSI specification. In this ERS, FAST means signals on the SCSI bus are differentially terminated and WIDE means there are 16 data bits on the SCSI bus. On the Outfield system board, FWSCSI is a separate disk interface (in addition to the existing 8 bit single ended SCSI interface) and is designed for connecting high speed disk drives.

11.2 Overview

The FWSCSI interface on the Outfield system board consists of the following components:

11.2.1 NCR53C720

The NCR53C720 is a SCSI controller chip. It has a 32 data bit DMA interface on the host bus side which transfers data at a maximum rate of 97 Mbyte/sec. On the SCSI bus side, it has a 16 bit differential data bus with data rate of 20 Mbyte/sec. For more details and features regarding to the NCR 53C720, see the reference list.

The NCR53C720 is operating at 33 Mhz (BCLK) at the DMA interface and 40 Mhz (SCLK) at the SCSI interface. To take advantage of the cache burst mode, bus mode 2 which is the Motorola 68040 interface of the controller is used on the board. This also implies the big endian convention is used.

11.2.2 FWSCSI control block inside the Cutoff ASIC

The FWSCSI control block inside the Cutoff ASIC interfaces between the NCR53C720, SGC bus, and the shortstop ASIC. In addtion to the control signals, this block also handles the translation of the size lines from the NCR53C720 to byte enables on SGC.

11.2.3 FWSCSI fifo inside the Shortstop ASIC

The FWSCSI fifo inside the Shortstop ASIC is used to buffer data to/from the NCR53C720 such that SGC bus bandwidth is utilized efficiently. For instance, if the NCR53C720 is doing a DMA write to memory, data is first stored to the Shortstop; then the same data is written to memory through SGC at the maximum data transfer rate by using the burst mode while the NCR53C720 is continuing to gather data from the SCSI bus.

11.2.4 Miscellaneous discrete components

The Miscellaneous discrete components relating to the FWSCSI interface are the Termpower circuit, the 543 address buffer, and the 33 Mhz clock generation circuit.

11.3 References

• NCR53C720 SCSI I/O Processor

- NCR53C720 Programmer's Guide

11.4 Supported Transactions on SGC

The FWSCSI interface supports (as slave) or initiates (as master) the following transaction on SGC.

11.4.1 As a slave on the bus

- Byte read
- Half-word read
- Word read
- Byte write
- Half-word write
- Word write

11.4.2 As a master on the bus

- Byte read
- Half-word read
- Word read
- Burst mode read
- Byte write
- Half-word write
- Word write
- Burst mode write

11.5 Supported Transactions on NCR53C720

At the DMA interface, the NCR53C720 operates in bus mode 2. The transactions used on the outfield board are:

11.5.1 As a slave on the bus

- Byte read
- Word read (16 bits)
- Long word read (32 bits)
- Byte write
- Word write (16 bits)
- Long word write (32 bits)

11.5.2 As a master on the bus

- Non-Cache-Line Burst byte read
- Non-Cache-Line Burst word read (16 bits)
- Non-Cache-Line Burst long word read (32 bits)
- Cache-Line Burst read (4 long words)
- Non-Cache-Line Burst byte write
- Non-Cache-Line Burst word write (16 bits)

- Non-Cache-Line Burst long word write (32 bits)
- Cache-Line Burst write (4 long words)

11.5.3 Self-access mode

Self-access mode is a type of transaction in which the NCR53C720 addresses itself when it is a bus master (instead of memory). Self-access occurs when the HP-UX driver executes the register to memory move instruction (a specific case of the memory to memory move script)

The outfield board supports the self-access mode but requires the driver to set the TT[1] bit in the CTEST0 register in the NCR53C720 via script before executing the register to memory move instruction and clear the TT1 bit afterward.

11.6 Performance

This section gives some tentative low level performance numbers such as FWSCSI's data transfer rate on SGC and on iodbig[31:0] when the NCR53C720 is a bus master. (refer to the board block diagram)

Assuming the NCR53C720 is set with burst size equal to 8, which means the controller can do a maximum of 8 words transfer per bus arbitration. These numbers could be changed accordingly if the burst size equal to 16.

11.6.1 SGC

11.6.1.1 Burst Data Rate

The FWSCSI subsystem should be able to provide one word of data every SGC clock cycle. It would take 9 cycles to transfer 8 words per arbitration. Including 2 cycles (?) for arbitration overhead, the transfer rate should be 8 words per 11 cycles for DMA write and read. Note that this number excluded the number of wait states from Viper.

With Viper wait states included, Viper can transfer 2 words every 3 cycles in burst mode. For reads, Viper needs another 210 ns for the initial memory access. This means that for the FWSCSI subsytem with Viper as the memory controller, it takes 15 cycles to transfer 8 words in DMA write and 22 cycles in DMA read. (assuming 30 ns cycles) Translating into MBytes/sec, DMA write is approximately 64.6 Mbytes/sec and DMA read is 44 MBytes/sec.

11.6.1.2 Non Burst Data Rate

For the FWSCSI, It takes 4 cycles to do a non burst transaction on SGC. This occurs when the NCR53C720 is fetching SCSI Scripts from memory. Normally there are 2 transactions per bus arbitration. the arbitration overhead in this case is 4 cycles because the data is not buffered in Shortstop. Shortstop is acting like a transparent data buffer and data is going directly from the NCR53C720 to SGC or visa versa. Therefore, the transfer rate is 2 words per 12 cycles. Again, this number excluded the number of wait states from Viper. This seems to be slow but Script DMA should be about 7 percent of all the DMA from the NCR53C720. (from measurement)

11.6.2 iodbig[31:0]

11.6.2.1 Cache-Line Burst Rate

The NCR53C720 takes 5 cycles to transfer 4 long words. Including 2 cycles for arbitration over head, this means the transfer rate is 8 words per 12 cycles.

11.6.2.2 Non Cache-Line Burst Rate

The Non Cache-Line Burst rate on the iodbig[31:] should same as the SGC Non Burst data rate which is 2 words per 12 cycles.

11.7 Register Set

Next page is the expanded FWSCSI subsystem register map looking from the host side to the SGC bus.

In register 0XF0800024, bit 5 is the FWSCSI-TERMPOWER bit and is related to the FWSCSI subsystem. this bit represents the state of the termination power on the FWSCSI 68 pin connector. a "0" on this bit means power is lost. Similar to the single ended SCSI subsystem, the HP-UX driver could use this bit to detect if an interrupt from the FWSCSI is due to te lost of the termpower or other proper causes. The driver should wait a minimum of 100 microseconds after a lost of termpower before reading the FWSCSI-TERMPOWER bit to allow sufficient time for the circuit to reset itself.

Unlike the single ended SCSI sybsystem, there is NO SCSI ID bit for the FWSCSI in the status regiser 0XF0800024. The NCR53C720 automatically loads SCSI ID 0x07 (highest priority on the 16 data bits bus) into the general purpose register after a reset to the chip. Software needs to execute a register to register move script to move the content of the general purpose register (GPREG) to the SCSI ID register (SCID)

Register 0XF0830000 is a write only register and is implemented in Cutoff. A write to this register causes a hard reset to the NCR53C720 and the FWSCSI fifo in Shortstop.

Register 0XF0830004 is the NT register (number of transfers). It is a write only register and is implemented in Cutoff. Bit[1:0] of the NT register should have the same value as the BL1 and BL0 bis in the DMODE register insdie the NCR53C720 (address = 0xf083013B) Software should set the NT and the DMODE registers first before executing any SCSI scripts. Although this is a write only register, its value could be read via the Cutoff Status register (address = 0Xf082f020)

Registers 0XF0830100 to 0XF083015C are implemented inside the NCR53C720. See the NCR53C720 data sheet for the definitions of each field within each register. Note that the address used is in the big endian convention.

Address (Hex)	Byte 0	Byte 1	Byte 2	Byte 3
. .	. .			
F0800024	Status (R)			
. .	. .			
F0830000	SCSI reset (W)			
F0830004	NT (W)			
F0830008	RESERVED			
. .	. .			
F08300FF	RESERVED			
F0830100	SCNTL3 (R/W)	SCNTL2 (R/W)	SCNTL1 (R/W)	SCNTL0 (R/W)
F0830104	GPREG (R/W)	SDID (R/W)	SXFER (R/W)	SCID (R/W)
F0830108	SBCL (R/W)	SSID (R)	SOCL (R/W)	SFBR (R/W)
F083010C	SSTAT2 (R)	SSTAT1 (R)	SSTAT0 (R)	DSTAT (R)
F0830110	DSA (R/W)			
F0830114	RESERVED			ISTAT (R/W)
F0830118	CTEST3 (R)	CTEST2 (R)	CTEST1 (R)	CTEST0 (R/W)
F083011C	TEMP (R/W)			
F0830120	CTEST6 (R/W)	CTEST5 (R/W)	CTEST4 (R/W)	DFIFO (R/W)
F0830124	DCMD (R/W)	DBC (R/W)		
F0830128	DNAD (R/W)			
F083012C	DSP (R/W)			
F0830130	DSPS (R/W)			
F0830134	SCRATCHA (R/W)			
F0830138	DCNTL (R/W)	DWT (R/W)	DIEN (R/W)	DMODE (R/W)
F083013C	ADDER (R)			
F0830140	SIST1 (R)	SIST0 (R)	SIEN1 (R/W)	SIEN0 (R/W)
F0830144	RESERVED		SWIDE (R)	SLPAR (R/W)
F0830148	RESERVED		STIME1 (R/W)	STIME0 (R/W)
F083014C	STEST3 (R/W)	STEST2 (R/W)	STEST1 (R)	STEST0 (R)
F0830150	RESERVED		SIDL (R)	
F0830154	RESERVED		SODL (R/W)	
F0830158	RESERVED		SBDL (R)	
F083015C	SCRATCHB (R/W)			
F0830160	RESERVED			
. .	. .			
F08301FF	RESERVED			
. .	. .			
F082F024	IO VCB			
. .	. .			

11.8 Addressing the registers inside the NCR53C720

To access the SCNTL3 register, the address "FWSCSI_slot_base_add + 00" should be supplied, to access the SCNTL2 register, the address "FWSCSI_slot_base_add + 01 (byte_offset)" should be supplied, and etc ... For the FWSCSI subsystem, the "FWSCSI_slot_base_add" = 0XF08301.

address with offsets	data field on SGC
0, 4, 8, and C	D31 - D24
1, 5, 9, and D	D23 - D16
2, 6, A, and E	D15 - D8
3, 7, B, and F	D7 - D0

The registers can be accessed in 8 bits, 16 bits, or 32 bits. To read the DBC register which is 24 bits, software should read the entire 32 bit at address 0XF0830124 and then extract the 24 bits belongs to the DBC register.

11.9 Initializing the registers related to the FWSCSI subsystem

In order for the FWSCSI hardware to work properly, it is very important to initialize the following registers :

- Important! this MUST be the FIRST slave access after reset. Set the Enable Acknowledge (EA) bit in the DCNTL register to internallaly connect SLACK/ to TA/ in slave mode.
  Address for the DCNTL register is 0xf0830138

- Execute a register to a register move script to move the content of the GPREG to SCID to set up the SCSI ID in the 53C720. The SCSI ID is set to 0x07 in the GPREG register after a reset.

- the CDIS bit in CTEST0 (0xfo83011B) should NEVER be set.
  Cahce burst should be always enabled.

- prior to executing the register to memory move instruction, software need to set the TT[1] bit via script and clear the TT[1] bit after executing the instruction. Refer to the Self-access mode section for details.

- Set the Number of Transfer (NT) register inside Cutoff IN ADDITION to setting the BL1 and BL0 bits in the DMODE register The value of the [1:0] bits in the NT register and the BL1, BL0 bits should have the same values. They should be either 4, 8, or 16. Address for the NT bits register = 0xf0830004

".H 2 "Interrupts"

Interrupts from the NCR53C720 are passed through Cutoff before being send to SGC. Since data are first stored in Shortstop for cache burst writes and reads, interrupts are postponed if the FWSCSI fifo in Shortstop is NOT empty. Refer to the interrupt register inside Cutoff for more details.

12. AUDIO

12.0.1 Refer to Vivace ERS.

13. FDDI

This document will make references to the following documents:

AMD - "The SUPERNET(tm) 2 Family for FDDI", 1991 Data Book referencing the MAC controller and PHY interface chips.

Hewlett Packard - Standard Graphics Connection Specification (A-5960-1585-1)

Hewlett Packard - Cobra/Coral I/O Subsystem ERS version 1.41

Hewlett Packard - ASP Theory of operation version 1.0

13.1 INTRODUCTION

The FDDI (Fibre Distributed Data Interface) is intended for use in a high performance general purpose multi-station network and is designed for efficient operation with a peak data rate of 100 Mbit/s. It uses a token ring architecture with optical fibre as the transmission medium. FDDI provides for hundreds of stations operating over an extent of tens of kilometers.

13.2 OVERVIEW

The Hardball I/O subsystem will support the FDDI protocol by implementing an interface from the AMD Formac Plus chip (Am79C830) to the internal SGC. The AMD provides local buffer support and a single port to access the data for the host CPU this data port will provide data into the Shortstop IC (fifo) at the following rate:

The transfers to and from the AMD buffer memory will be controlled by logic in the Cutoff chip. This operation does not require any address generation by Cutoff as the AMD handles the buffer memory access. The Shortstop chip will receive/send the data based on the internal IDMA state machine control logic.

The transfers to and from system memory (via SGC) will be controlled by the external EDMA state machine and control logic. This EDMA will need address pointers, counters, interrupts, status, and control in order to move the data between Shortstop and system memory. All of these registers are defined below.

The FDDI interface located on the slider card will have interface ID bits for the CPU to determine which of the different slider cards has been inserted into the core I/O card. Refer to the Status Register section of this ERS for more information (section 10.7.3).

13.3 Memory Map

Below is a map of the registers for the FDDI subsystem, refer to the AMD FDDI chip set for specific information on the registers within these ranges.

0xF083 1000	- FDDI MAC, PHY, and DMA Registers	4k bytes
to		
0xF083 1FFF		

13.3.1 Memory Map for the AMD Formac Plus

The data bus into the Formac Plus is only 16 bits wide. Therefore rather than attempting to unpack into and pack bytes out of the controller onto SGC we will only use the low word (D0-D15) of the SGC when accessing the controller. This causes the memory map to spread a little but we seem to have the room.

Address (HEX)	R/W	Word 0	Word 1
F0831000	W	CMDREG1	NA
F0831000	R	ST1U	NA
F0831004	W	CMDREG2	NA
F0831004	R	ST1L	NA
F0831008	R	ST2U	NA
F083100C	R	ST2L	NA
F0831010	R/W	IMSK1U	NA
F0831014	R/W	IMSK1L	NA
F0831018	R/W	IMSK2U	NA
F083101C	R/W	IMSK2L	NA
F0831020	R/W	SAID	NA
F0831024	R/W	LAIM	NA
F0831028	R/W	LAIC	NA
F083102C	R/W	LAIL	NA
F0831030	R/W	SAGP	NA
F0831034	R/W	LAGM	NA
F0831038	R/W	LAGC	NA
F083103C	R/W	LAGL	NA
F0831040	R/W	MDREG1	NA
F0831044	R	STMCHN	NA
F0831048	R	MIR1	NA
F083104C	R	MIR0	NA
F0831050	R/W	TMAX	NA
F0831054	R/W	TVX	NA
F0831058	R/W	TRT	NA
F083105C	R/W	THT	NA
F0831060	R	TNEG	NA
F0831064	R	TMRS	NA
F0831068	R/W	TREQ	NA
F083106C	R/W	TREQ1	NA

TABLE 13-1. AMD Formac Memory Map

Address (HEX)	R/W	Word 0	Word 1
F0831070	R/W	PRI0	NA
F0831074	R/W	PRI1	NA
F0831078	R/W	PRI2	NA
F083107C	R/W	TSYNC	NA
F0831080	R/W	MDREG2	NA
F0831084	R/W	FRNTHR	NA
F0831088	R/W	EACB	NA
F083108C	R/W	EARV	NA
F0831090	R/W	EAS	NA
F0831094	R/W	EAA0	NA
F0831098	R/W	EAA1	NA
F083109C	R/W	EAA2	NA
F08310A0	R/W	SACL	NA
F08310A4	R/W	SABC	NA
F08310A8	R/W	WPXSF	NA
F08310AC	R/W	RPXSF	NA
F08310B0	R/W	NA	NA
F08310B4	R/W	RPR	NA
F08310B8	R/W	WPR	NA
F08310BC	R/W	SWPR	NA
F08310C0	R/W	WPXS	NA
F08310C4	R/W	WPXA0	NA
F08310C8	R/W	WPXA1	NA
F08310CC	R/W	WPXA2	NA
F08310D0	R/W	SWPXS	NA
F08310D4	R/W	SWPAX0	NA
F08310D8	R/W	SWPAX1	NA
F08310DC	R/W	SWPAX2	NA
F08310E0	R/W	RPXS	NA
F08310E4	R/W	RPXA0	NA
F08310E8	R/W	RPXA1	NA
F08310EC	R/W	RPXA2	NA
F08310F0	R/W	MARR	NA
F08310F4	R/W	MARW	NA
F08310F8	R/W	MDRU	NA
F08310FC	R/W	MDRL	NA
F0831100	R/W	TMSYNC	NA
F0831104	R/W	FCNTR	NA
F0831108	R/W	LCNTR	NA
F083110C	R/W	ECNTR	NA

TABLE 13-2. AMD Formac Memory Map(cont.)

13.3.2 Memory Map for the AMD PLC

The data bus into the AMD PLC is only 16 bits wide. Therefore rather than attempting to unpack into and pack bytes out of the controller onto SGC we will only use the low word (D0-D15) of the SGC when accessing the controller.

Address (HEX)	R/W	Word 0	Word 1
F0831200	R/W	PLC_CNTRL_A	NA
F0831204	R/W	PLC_CNTRL_B	NA
F0831208	R/W	INTR_MASK	NA
F083120C	R/W	XMIT_VECTOR	NA
F0831210	R/W	VECTOR_LENGTH	NA
F0831214	R/W	LE_THRESHOLD	NA
F0831218	R/W	C_MIN	NA
F083121C	R/W	TL_MIN	NA
F0831220	R/W	TB_MIN	NA
F0831224	R/W	T_OUT	NA
F0831228	/	NA	NA
F083122C	R/W	LC_LENGTH	NA
F0831230	R/W	T_SCRUB	NA
F0831234	R/W	NS_MAX	NA
F0831238	W	TPC_LOAD_VALUE	NA
F083123C	W	TNE_LOAD_VALUE	NA
F0831240	R	PLC_STATUS_A	NA
F0831244	R	PLC_STATUS_B	NA
F0831248	R	TPC	NA
F083124C	R	TNE	NA
F0831250	R	CLK_DIV	NA
F0831254	R	BIST_SIGNATURE	NA
F0831258	R	RCV_VECTOR	NA
F083125C	R	INTR_EVENT	NA
F0831260	R	VIOL_STM_CTR	NA
F0831264	R	MIN_IDLE_CTR	NA
F0831268	R	LINK_ERR_CTR	NA

TABLE 13-3. AMD PLC Memory Map

13.3.3 FDDI DMA Register Memory Map

The registers in the FDDI DMA section are being defined as 32 bit wide registers. Since the data path into Cutoff is only 8 bits wide (D0-7) the byte steering logic will have to be used.

Address (HEX)	R/W	Data
F0831300	R/W	FDDI_DMA_ADDR1
F0831304	R/W	FDDI_DMA_WC1
F0831308	R/W	FDDI_DMA_ADDR2
F083131C	R/W	FDDI_DMA_WC2
F0831320	R	FDDI_XMIT_REQ
F0831324	R/W	FDDI_DMA_CNTRL
F0831328	R/W	FDDI_DMA_STATUS
F083132C	R/W	FDDI_DMA_INTR
F0831330	R/W	FDDI_DMA_INTR_MSK
F0831334	R/W	FDDI_SUBS_RST

TABLE 13-4. FDDI DMA Register Memory Map

13.3.3.1 DMA Register Definition

There will be a single DMA controller in Cutoff. It will be shared between reads and writes. This DMA controller will have two linked register sets. When the DMA completes on the first register set, the second set will automatically be started if the transfer size is not zero.

Each register set will have a host address and a transfer size (word count). In addition, there will be direction and priority queue information (to be passed to the F+) shared by both register sets.

Since the DMA path is half-duplex, and the driver will normally want to leave it programmed for inbound, the transmit/receive switch requires a semaphore, similar to the SR71.

The hardware performs a crude checksum on inbound data. The checksum will likely span both register sets. All data that crosses the bus will be included in the checksum; the card does not interpret headers. The checksum value can be read from I/O space. The read has a side effect of clearing the register.

13.3.3.1.1 FDDI_DMA_ADDR1

This is the 32 bit address in the first DMA address/count pair.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	X	X

TABLE 13-5. FDDI_DMA_ADDR1

D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19	D18	D17	D16
A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16

TABLE 13-6. FDDI_DMA_ADDR1(cont)

13.3.3.1.2 FDDI_DMA_WC1

This is the word count for the first DMA address/count pair. The LSB of the DMA count is in D0 (ie. to move one word program FDDI_DMA_WC1 = xxxx000000000001).

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	WC11	WC10	WC9	WC8	WC7	WC6	WC5	WC4	WC3	WC2	WC1	WC0

TABLE 13-7. FDDI_DMA_WC1

13.3.3.1.3 FDDI_DMA_ADDR2

This is the 32 bit address in the second DMA address/count pair.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	X	X

TABLE 13-8. FDDI_DMA_ADDR2

D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19	D18	D17	D16
A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16

TABLE 13-9. FDDI_DMA_ADDR2(cont)

13.3.3.1.4 FDDI_DMA_WC2

This is the word count for the second DMA address/count pair. The LSB of the DMA count is in D0 (ie. to move one word program FDDI_DMA_WC1 = xxxx000000000001).

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	WC11	WC10	WC9	WC8	WC7	WC6	WC5	WC4	WC3	WC2	WC1	WC0

TABLE 13-10. FDDI_DMA_WC2

13.3.3.1.5 FDDI_XMIT_REQ

When the driver has built a outbound packet it will then read this register. This read will cause the internal logic to acquire the DMA resource if available. And if available the read operation will also clear the FDDI_REC_DMA_RDY bit.

- Bit 13,14 FDDI_DMA_SEMA0,1, the table below will indicate to the driver if the resource has been obtained. This will never read 00 as the act of reading this register will change the FDDI_DMA_SEMA0,1 to 01 if the DMA resource was available.

FDDI_DMA_SEMA0	FDDI_DMA_SEMA1	Current Function			
0	0	None			
0	1	Transmit to FDDI in progress			
1	0	Receive from FDDI in progress			
1	1	None			

TABLE 13-11. FDDI_DMA_SEMA[1,0] definition

13.3.3.1.6 GLOBAL READ DATA FOR x1C THRU x28

All reads of the DMA registers in this address range 0xF083131C through 0xF0831328 will also provide the FDDI_DMA_DEMA0,1 and FDDI_REC_DMA_RDY in bit locations d13,14,15 respectively.

13.3.3.1.7 FDDI_DMA_CNTRL

This register will contain the bits to control the FDDI DMA state machine. These bits are the following:

- bits 0,1 CHSREQ, These bits have similar definitions as the HSREQ[2-0] in the AMD FDDI Formac Plus. The HSREQ0 and HSREQ1 going from Cutoff to the Formac are tied together. This will also be the last event prior to the DMA beginning, prior to this the address and count registers should be set. This results in the following function:

CHSREQ1	CHSREQ0	HSREQ2	HSREQ1	HSREQ0	FUNCTION
0	0	0	0	0	None
0	1	0	1	1	Read Request: Receive Queue
1	0	1	0	0	Write Request: Sync. Queue *
1	1	1	1	1	Write Request: Async. Queue

TABLE 13-12. CHSREQ[2:0] definition

* Not Supported.

- Bit 2 CSUM_ACCUM, This bit when set will cause the Shortstop IC to accumulate a data checksum when the data is sampled into the IC. This function is only available on data coming in from the network and into host memory.
- Bit 3 CLR_SECOND_DMA_REGS, This bit will (prior to the DMA beginning) clear the second set of DMA registers (saves 3 SGC accesses on packets less than a page in length. When set to 1 the clear will occur.
- Bit 4,5 FDDI_DMA_SEMA, These bits will indicate to the different processes who owns the DMA resource.

FDDI_DMA_SEMA0	FDDI_DMA_SEMA1	Current Function			
0	0	None			
0	1	Transmit to FDDI in progress			
1	0	Receive from FDDI in progress			
1	1	None			

TABLE 13-13. FDDI_DMA_SEMA[1,0] definition

- Bit 6 FDI_REC_DMA_RDY, This bit indicates to the internal state machines that the DMA address and count registers have been preprogrammed for a receive when RDATA becomes active. This also can provide a secondary function of indicating to an ISR which part of a packet had just been moved from AMD buffer memory into host memory.

Register bits 7-12 are not defined at this point.

13.3.3.1.8 FDDI_DMA_STATUS

This register will provide indications of the state of the FDDI DMA controller as well as the AMD Formac Plus if possible.

- Bit 0 DMA_COMPL, This bit will be set (1) when both DMA word count (FDDI_DMA_WC1,FDDI_DMA_WC2) registers equal 0.
- Bit 1 TAG_SEEN, On a receive from the network the AMD Formac Plus will assert a signal BDTAG when the data stream goes from data to frame status, thus indicating the end of the receive packet. On a transmit this bit must be generated by the Cutoff or Shortstop H/W? This bit will be automatically cleared when the checksum is read from the Shortstop.

Register bits 2-12 are not defined at this point.

13.3.3.1.9 FDDI_INTR

This register will provide reason for the last interrupt to the CPU.

- Bit 0 MAC1_INTR, This bit is generated when the AMD Formac Plus nMINTR1 signal becomes active. This bit will have positive logic associated with it (ie. when the nMINTR1 asserts this bit will go to 1, and will reset to 0).
- Bit 1 MAC2_INTR, This bit is generated when the AMD Formac Plus nMINTR2 signal becomes active. Again, set to 1 when nMINTR2 is active and 0 when nMINTR2 is inactive and reset.
- Bit 2 PLC_INTR, This bit is generated when the Physical Layer Controller interrupt signal is asserted, nINTPLC. Again, set to 1 when nINTPLC is active
 and 0 when nINTPLC is inactive and reset.
- Bit 3 RDATA, This is the RDATA signal coming from the Formac Plus. Upon receive the DMA controller will be preset to receive some of the data into a buffer. When the DMA controller completes this operation it will cause a system interrupt. At this point the driver may interrogate this register to determine the cause and if it sees the DMA_COMP (later) and the RDATA interrupt bits set it will know that a packet has partially been received.
- Bit 4 DMA_COMP, This will become set when the DMA word count registers (FDDI_DMA_WC1,FDDI_DMA_WC2) equal 0 after a DMA had been requested. This bit will automatically clear after being read and will not become active until after a DMA request.

Register bits 5-12 are not defined at this point.

13.3.3.1.10 FDDI_INTR_MASK

This register will provide the driver the ability to mask the 5 interrupts defined above. A '0' in the register will mask interrupts and a '1' will allow them to interrupt the host. At reset this register will have interrupts masked '0'.

The bit definition is :

- Bit 0 MAC1_INTR_MSK
- Bit 1 MAC2_INTR_MSK
- Bit 2 PLC_INTR_MSK
- Bit 3 RDATA_MSK
- Bit 4 DMA_COMP_MSK
- Bit 5 MULTI_CAST_EN, This bit will enable the external (to the AMD Formac Plus) multicast detection logic. It was placed here because it, like the interrupt mask bits is typically a static bit.

13.3.3.1.11 FDDI_SUBS_RST

A write to this address will cause the entire FDDI subsystem to be reset to it's initial state.

13.3.4 RECEIVE CHECKSUM

This defines a memory location that the system will read and in turn the Cutoff IC will prompt the Shortstop IC to present the accumulated checksum on the SGC.

Address (HEX)	R/W	Word 0	Word 1
F0831400	R	REC_DATA_CSUM_LW	REC_DATA_CSUM_HW

TABLE 13-14. Receive data Checksum from Shortstop

13.4 EXAMPLE DMA TRANSFER (TRANSMIT)

This will present a proposed use of the FDDI DMA controller in order to move a FDDI packet from host memory into AMD Formac Plus buffer memory.

- Assemble packet in host memory space with header, data, and checksum.
- Read the FDDI_XMIT_REQ register and determine if the semaphore has been granted. If semaphore has not been granted the driver must wait until it can be obtained.
- Program the FDDI_DMA_ADDR1_LW and FDDI_DMA_ADDR1_HW to the desired address and the FDDI_DMA_WC1 to the desired count.
    - If the count is greater than a 4K page the second FDDI DMA will have to be programed.
- Program the FDDI_DMA_CNTRL to CHSREQ0,1 = 11, CSUM_ACCUM = 0, CLR_SECOND_DMA_REGS to 1 if < 4K and 0 if >4k, and keep the FDDI_DMA_SEMA0,1 (=01).
- The write to CHSREQ will automatically cause the DMA process to begin.
- The DMA will now proceed until either an error or a completed DMA will cause an interrupt. When the ISR interrogates the cutoff interrupt register a good transfer will show the FDDI_DMA_COMP and MAC1_INTR bits set. From here the driver must also process the ST1 registers in the AMD Formac Plus. If at this point the PLC_INTR is set it must be investigated.

- Also, at this point the RDATA signal could be active now the driver must realize that a transmit is in process (via FDDI_DMA_SEMA bits) and ignore it.
- When the routine is has completed the driver must release the FDDI_DMA_SEMA by writing 00 to these bits in the FDDI_DMA_CNTRL register.

13.5 TRANSITION FROM TRANSMIT TO RECEIVE

This process will proceed from where the transmit process just ended.

- Now if the RDATA is set the receive cannot begin as the DMA address and count have not been set up.
- The driver will now program the FDDI_DMA_ADDR1_lW,FDDI_DMA_ADDR1_HW, and the FDDI_DMA_WC1 to receive the beginning of the packet that is either pending or is anticipated to be received soon.
- When the driver has set the address and count it must then set the FDDI_REC_ DMA_RDY bit and the CLR_SECOND_DMA_REGS bit (assuming the portion of the packet to be transferred automatically is less than a 4k page). This will indicate to the internal receive state machine that if RDATA becomes active and the FDDI_DMA_SEMA0,1 are available the transfer to host memory may begin.

13.6 COMPLETION OF THE RECEIVE

This process will begin when the RDATA signal from the AMD Formac Plus becomes active and the system has been preloaded to receive data.

- When RDATA becomes active the FDDI DMA state machine will attempt to obtain the FDDI DMA semaphore only after the FDDI_REC_DMA_RDY bit is allowing the automatic receive.
- If the semaphore in unavailable or the FDDI_REC_DMA_RDY is not set the RDATA will wait.
- If the semaphore is available and the FDDI_REC_DMA_RDY is set the internal state machine will set these bits (FDDI_DMA_SEMA0,1) to lock out any transmit and set the CHSREQ bits to 10 which begins the DMA into host memory.

  *** NOTE the FDDI_REC_DMA_RDY bit must remain set at this point also. ***
- When the CPU receives the interrupt it must look at the FDDI_INTR register.
    - IF RDATA = 1 and FDDI_DMA_SEMA0,1 # 10 the interrupt was from some other source and must be investigated.
    - If RDATA = 1 and FDDI_DMA_COMP = 1 as well as the FDDI_DMA_SEMA0,1 = 10 and FDDI_REC_DMA_RDY = 1 this is an indication that the header has been received into host memory and the processing of that data must occur.
- After the processing of the data the address and count register must be programed as well as resetting FDDI_REC_DMA_RDY (write 0), keep the FDDI_DMA SEMA, set CSUM_ACCUM, and program CHSREQ0,1 to 10. The setting of the CHSREQ bits will begin the remainder of the data transfer (DMA).
- Now the CPU will receive another interrupt and will have to decide what it is.
    - If DMA_COMP = 1 and FDDI_DMA_SEMA=10 and FDDI_REC_DMA_RDY=0 the DMA has completed. The reason the FDDI_DMA_REC_RDY was tested is that RDATA may already be active again and the driver must know if the interrupt was from the header receive or the data receive.
    - Any other bits in the FDDI_INTR must be investigated.

13.7 CAM REGISTERS

The FDDI subsection will support four hard-wired multi-cast group address ranges for external destination address match. These are enabled as a group from the FDDI_INTR_MASK register bit MULTI_CAST_EN. The multicast address ranges are:

Multi-cast Address	comment
?? ?? ?? ?? ?? XX	SMT multi-cast address block
09 00 09 00 00 0X	HP proprietary multi-cast address
09 00 2B 00 00 0X	HP proprietary multi-cast address
?? ?? ?? ?? ?? ??	(unknown)

TABLE 13-15. Additional Multi-cast Addresses

These addresses are located inside a PAL that will be located on the slider card. The cutoff will supply the enable signal only to this PAL.

13.8 PERFORMANCE

Dependencies: ----------

1. The NIKE Transport interface must be implemented in HP-UX for the Series 700, and Cobra FDDI must be treated as a NIKE device. In particular, Cobra FDDI needs the following NIKE features:

   Page remapping on inbound packets
   User-to-system copy skipped on outbound packets
   Transport must allow the data link layer to handle checksums
   Lightweight (real mode) interrupts

2. The Cobra FDDI driver must implement an optimized (assembly language) routine to copy and checksum the mbuf chain containing the outbound packet into a single, properly aligned dma buffer.

3. The Cobra FDDI driver will assist the hardware in ensuring that the inbound user data is aligned on a page boundary. This is accomplished by having the driver handle one lightweight interrupt per inbound packet to perform a header/data split.

4. On inbound packets, the hardware performs only a crude checksum of the data. The driver must fix up the checksum calculated by the hardware.

Method of Analysis --------------

For the sake of analysis, it is useful to divide network overhead into 4 main parts:

1. Processing overhead
2. Copy cost
3. Checksum cost
4. DMA interference

Processing overhead is per-packet. When measured in us/KB, this gets better with packet size. Copy and checksum costs are generally per-byte, and don't vary with packet size.

DMA interference is the wild-card. This is the lost CPU time when cache operations to memory are not serviced immediately due to interference with card DMA. It messes up attempts to measure any of the other three directly. Very often, measurements of the other three have a certain amount of DMA interference factored in.

If we calculate the overhead represented by each part in terms of us/pkt, we can sum these to get the total overhead per packet. With some unit conversion, this number can be turned into MB/sec throughput.

Transmit -------

The Cobra built-in FDDI will look like a NIKE card to the Transport Layer. This means that Transport will pass pointers to the user data, without copying or checksumming it, to the data link layer. On X-15, these functions were literally executed by the hardware. On Cobra FDDI, the copy and checksum will still be done by the host processor, but there will be only one copy, and the checksum will be rolled into the loop.

Thus, the transmit overhead will consist of the normal processing overhead, plus a single copy/pack/checksum loop.

Transmit Overhead per packet: ----------------------

Measured in us/KB, the processing overhead will vary with both the CPU power and the packet size. To allow conversion between different architectures, we need a number that can be more readily converted. Since we know the packet sizes and the Dhrystone MIPS of the various implementations, let's express the overhead per packet in "Dhrystone Instructions/pkt".

Since we're interested in TCP performance, what we need is an estimate of DMIPS/packet for TCP. Since the X-15 card does no data copying, we can get a handle on this from looking at the performance of X-15. Since X-15 uses a high percentage of the Silverfox bus, DMA interference may make the X-15 numbers look a little high. At least we can take it as an upper bound.

X-15 achieved 7.5 MB/sec on TCP and 10 MB/sec on UDP. Assuming the Silverfox host is 14 DMIPS (14E6 DI/sec), we get:

TCP throughput : ~ 7.5 MB/sec
7.5 MB/sec x 1024 KB/MB x pkt/4.5 KB x sec/14E6 DI = 122E-6 pkt/DI
= 8200 DI/pkt

UDP throughput : ~ 10 MB/sec
10 MB/sec x 1024 KB/MB x pkt/4.5 KB x sec/14E6 DI = 162E-6 pkt/DI
= 6150 DI/pkt

The UDP number is interesting as a sanity check. instruction counts for UDP show 8000 instructions/pkt, of which 3116 are processing, 2494 are user-sys copy, and 2390 are checksum.

Taking the value of 3116 inst/pkt for processing, the 6150 DI/pkt derived above would result in a CPI of approximately 2 for the networking code (i.e., the "MIPs derating factor" is 0.5). This is, in fact, the value people usually use. So, 8200 DI/pkt is probably a pretty good value.

Using a value of 57 DI/us for a 720 (57 Dhrystone MIPS):

```
8200 DI/pkt x 1 us/57 DI              = 144 us/pkt (720)
```

We can derive similar numbers for a 730 or a PCX-T (assuming a MHz to MIPS conversion for PCXT of 1.15):

```
8200 DI/pkt x 1 us/76 DI              = 108 us/pkt (730)
8200 DI/pkt x 1 us/92 DI              =  89 us/pkt (PCXT)
```

Copy/pack/checksum ---------------

To estimate the overhead involved in this loop, consider what is required to copy/pack/checksum a cache line.

First consider the cache operations. The first load and the first store will cause misses. Assuming that the lines to be overwritten are dirty, this will cause two copy_out/copy_in operations ("dirty misses"). At some point the code will also explicitly flush the store line. However, this isn't really a third copy_out, since it results in a "clean" line. The next time a miss occurs to that hashed address, a copy_out will be avoided. Hence, there is a net of two dirty misses.

So, we have two dirty misses, plus the actual copy/pack/checksum code execution.

On a 720, each dirty miss costs approximately 21 cycles, resulting in a copy/pack/checksum cost of 21*2 + 45 = 87 cycles to process 32 bytes:

```
87 cyc/32 bytes x 4096 bytes/pkt x 1 us/50 cyc      = 223 us/pkt (720)
```

Assuming the same cache miss penalty on the 730:

```
87 cyc/32 bytes x 4096 bytes/pkt x 1 us/66 cyc      = 169 us/pkt (730)
```

estimate the cache miss penalty on PCX-T as 33 clocks:

```
111 cyc/32 bytes x 4096 bytes/pkt x 1 us/80 cyc     = 178 us/pkt (PCXT)
```

Note that the PCX-T number may be somewhat pessimistic, since the cache hint ability of PCX-T should cache miss on the store from a copy_out/copy_in to just a copy_out.

Again, lets use UDP numbers for a sanity check. The checksum in this loop is essentially free (squeezed into dead cycles), so we can use UDP copy instruction count of 2494 directly:

```
87 cyc/32 bytes x 4000 bytes/2494 instr  =  4.4 CPI
```

DMA Interference --------------

An FDDI transmit corresponds to an SGC "guest" READ of host memory. There will be a latency to first access of ~6 clocks, then Viper will insert one wait for every two words transferred.

Thus, a 16 word burst takes 6 + 8 * 3 = 30 SGC clocks.

If the CPU were to stall instantly as soon as FDDI DMA acquired the bus, the effective CPU overhead represented by DMA would be:

```
30 clks/64 bytes x us/25 clks x 4160 bytes/pkt       = 78 us/pkt (720)

30 clks/64 bytes x us/33 clks x 4160 bytes/pkt       = 59 us/pkt (730)

30 clks/64 bytes x us/26.67 clks x 4160 bytes/pkt    = 73 us/pkt (PCXT)
```

Note: Since the DMA will transfer the entire packet, including the header, there are 4096 + 64 = 4160 bytes/pkt.

These number represent an upper bound, since the CPU will not always stall immediately. If the CPU were to execute an average of 10 cycles before stalling, these numbers would drop by one third.

Note that these numbers represent SGC transfer rates of 51 to 65 MB/sec.

Transmit Performance - Adding it all up ------------------------------

Without DMA interference, we can directly calculate the maximum throughput on transmit by taking the inverse of the us/pkt number and doing some unit conversion:

```
pkt/(144 + 223)us x 4096 bytes/pkt x MB/(1024)^2 bytes
 x us/1E-6 sec                                  = 10.6 MB/sec (720)

pkt/(108 + 169)us x 4096 bytes/pkt x MB/(1024)^2 bytes
 x us/1E-6 sec                                  = 14.1 MB/sec (730)

pkt/(89 + 178)us x 4096 bytes/pkt x MB/(1024)^2 bytes
 x us/1E-6 sec                                  = 14.6 MB/sec (PCXT)
```

If we plug in the worst-case overhead due to DMA interference, we get:

```
pkt/(144 + 223 + 78)us x 4096 bytes/pkt x MB/(1024)^2 bytes
 x us/1E-6 sec                                  = 8.8 MB/sec (720)

pkt/(108 + 169 + 59)us x 4096 bytes/pkt x MB/(1024)^2 bytes
 x us/1E-6 sec                                  = 11.6 MB/sec (730)

pkt/(89 + 178 + 73)us x 4096 bytes/pkt x MB/(1024)^2 bytes
 x us/1E-6 sec                                  = 11.5 MB/sec (PCXT)
```

Note that this is DATA throughput. Although the bandwidth of the physical link is 12.5 MB/sec., max data throughput is lower due to headers, link protocol overhead, etc. "Link rate" for FDDI is usually defined as 10-11 MB/sec., so we can expect to transfer at link rate on a 730 or PCX-T host, and very near link rate on a 720.

Receive ------

Receive performance could be analyzed the same way as transmit, but it's not really necessary.

Although the processing overhead on receive is slightly higher than on transmit, the complete elimination of data copies for SNAP packet reception virtually guarantees that any of the hosts will be able to receive SNAP packets at full link rate. The only overhead is the per-packet protocol overhead, plus one additional "lightweight" interrupt per packet.

This is confirmed by experience with the X-15 card. Although the X-15 card does the header/data split in hardware, it turns out that, due to an optimization for the NIO bus, the X-15 code actually takes one additional "lightweight" interrupt per packet as well. Therefore, the overhead on receive should be nearly identical between X-15 and the Cobra FDDI.

Since X-15 is able to receive at full link rate using a processor with one-third the Dhrystone MIPS, clearly Cobra FDDI should have no trouble receiving SNAP packets at full link rate.

Reception of non-SNAP packets will require one data copy. This is equivalent to transmit, except that receive processing overhead is slightly higher. Therefore, non-SNAP packet performance will probably be slightly lower than the transmit performance cited above.

Cobra FDDI
==========

Overview: ------

There will be a single DMA controller in Cutoff. It will be shared between reads and writes. This DMA controller will have two linked register sets. When the DMA completes on the first register set, the second set will automatically be started if the transfer size is not zero.

Each register set will have a host address and a transfer size (word count). In addition, there will be direction and priority queue information (to be passed to the F+) shared by both register sets. The path to Cutoff is only 8 bits wide, so the simplest implementation would be to map these registers as one or more byte registers. However, this is not optimal from either a software or performance point of view. A cost/benefit analysis is still needed.

Since the DMA path is half-duplex, and the driver will normally want to leave it programmed for inbound, the transmit/receive switch requires a semaphore, similar to the SR71. Details have not yet been worked out.

The hardware performs a crude checksum on inbound data. The checksum will likely span both register sets. All data that crosses the bus will be included in the checksum; the card does not interpret headers. The checksum value can be read from I/O space. The read has a side effect of clearing the register.

Inbound: ------

When not transmitting, the driver will leave the DMA controller programmed to transfer enough bytes of the next packet to ensure that the header (and possibly some data) gets transferred to memory. The F+ will be programmed to offset the packet by 3 bytes. This will cause the data in SNAP packets to be long-word aligned.

When the F+ indicates that a receive packet is available, the DMA controller in Cutoff will transfer this first block of the packet to host memory, and give a "DMA complete" interrupt.

This will be handled as a "lightweight" interrupt. The ISR will examine the packet header to determine whether the packet is a SNAP packet and where the data starts.

If it is a SNAP packet, the data will already be long-word aligned (because the F+ was programmed with the 3 byte offset). The ISR will copy any data bytes necessary to a new buffer (accumulating the checksum along the way), then program the DMA controller to transfer the rest of the packet to this buffer, and return. In the case of an NFS packet, the ISR will program the DMA controller to do consecutive transfers to two buffers with the size of the buffers chosen to put the rest of the header in the first buffer, and the data in another buffer. The packet processing will occur under "heavyweight" interrupt processing.

In this way, header/data splitting can be accomplished on SNAP packets at the cost of a single additional lightweight interrupt.

If the packet was not SNAP, the ISR will have to program the DMA controller to transfer the rest of the packet into one or two buffers as appropriate, then will have to copy the data to re-align it during the "heavyweight" interrupt processing.

The driver may choose to transfer more than the minimum number of bytes on the first DMA in order to be able to handle small packets with a single interrupt. This is a tradeoff between the gain on small packets and the extra copy time on large packets. Since the byte count is programmable, this is entirely up to the driver.

Inbound DMA will be terminated by expiration of the programmed word count OR by the tag indication from the F+. Either cause will generate an interrupt, and the ISR will be able to distinguish one from the other by looking at the FDDI interrupt register.

Outbound: ------

Since the Cutoff DMA controller does not do any byte packing, the driver must pack the entire packet into one or more buffers before DMA'ing it to the card. Note that all buffers except the last must begin and end on 4-byte boundaries. All checksumming is done by the driver.

Also, the driver is responsible for creating and appending the descriptor value at the end of the packet. The Cutoff DMA controller will automatically assert tag on the last two words of the transfer. Note that this means that packets may NEVER SPAN MORE THAN TWO BUFFERS, since the Cutoff DMA controller will interpret expiration of the word count as the end of the packet.

Having built the packet, the driver must acquire the semaphore, then program the DMA controller to move the packet to the F+ buffer memory. Since there are two register sets, this can be programmed at once, even if the transfer exceeds one 4K page. The DMA controller will interrupt on completion of this DMA, at which time the driver may choose to send another packet, or program the DMA controller for inbound again.

The first register set will actually allow transfers up to 2048 words (8K bytes). This will reduce the overhead on large packets if the OS environment allows the driver to request physically contiguous pages.

Cutoff FDDI DMA Registers: ----------------

FDDI_DMA_ADDR_1	30 bits	FDDI_DMA_WC_1	12 bits
FDDI_DMA_ADDR_2	30 bits	FDDI_DMA_WC_2	11 bits
FDDI_DMA_CONTROL	xx bits		
HSREQ[0-2]	3 bits		
FDDI_DMA_SEMAPHORE			
FDDI_DMA_STATUS	xx bits		
DMA_COMPLETE	1 bit		
INBOUND_TAG_SEEN	1 bit		

FDDI_DMA_CHECKSUM 32 bits

FDDI_INTERRUPT 6 bits FDDI_INTERRUPT_MASK 6 bits

Notes:

The first WC register is 1 bit wider than the second to allow OS's capable of allocating physically contiguous pages to transfer a full packet without using the second register set.

Both WC registers are 1 bit wider (12 bits rather than 11 for WC_1 and 11 bits rather than 10 for WC_2) than necessary allow the max value of 1024 words to be programmed directly, without funny issues of what zero represents.

The semaphore and control bits are a little fuzzy still. These may be combined.

Status bits only refer to FDDI DMA status. These might actually be combined with other FDDI status bits in another register.

The checksum register actually resides in the Shortstop chip. A read of this register will cause Cutoff to tell Shortstop to dump this register onto SGC. Cutoff will take care of the timing.

There will probably be only a single fddi interrupt bit in the ASP interrupt register. To determine the actual source of an FDDI interrupt, the ISR will have to examine the FDDI_INTERRUPT register.

14. Rapid Harmless DMA

This function is embedded in Cutoff and performs 10 subword DMA writes at full SGC bandwidth. Its primary purpose is to fix a problem with Viper, described in the DTS report PCXaa00105.

Address	Size	R/W	Description
0xF0833000	byte	W	Rapid harmless DMA address byte 0 (high)
0xF0833004	byte	W	Rapid harmless DMA address byte 1
0xF0833008	byte	W	Rapid harmless DMA address byte 2
0xF083300C	byte	W	Rapid harmless DMA address byte 3 (low)
0xF0833010	byte	W	Rapid harmless DMA execute - any value will do
0xF0833020	byte	R/W	Rapid harmless DMA done

Programming Model:

RH_DMA_ADDR[0:3]:
Address for DMA writes. The least 2 significant bits of RH_DMA_ADDR[3] (i.e. the byte offset) are ignored. The address must be initialized by firmware before writing the execute register. The initial value after reset is NOT guaranteed. All 10 writes occur to this address.

RH_DMA_EXECUTE:
Causes Cutoff to request the SGC bus and perform 10 subword writes to RH_DMA_ADDR at full SGC bandwidth when granted the bus.

RH_DMA_DONE:
Only the least significant bit is valid. Initial value: 0

	0	1
Read	not done	done
Write	reset to 0	no effect

The RH_DMA does NOT interrupt on DMA done. The RH_DMA_DONE register is set to 1 after the last SGC transaction but before Cutoff gives up the bus. After writing RH_DMA_EXECUTE, the processor should poll the RH_DMA_DONE register until it returns a value of 1. The register should be reset to 0 before the next RH_DMA_EXECUTE. **WEIRDNESS ALERT: To write the RH_DMA_DONE register, write the RH_DMA_DONE address (0xF0833020). To read the register, examine bit 7 (most significant) of the Cutoff Status Register (0xF082F020).**

DATA:
The value of data written by the DMA transfer is not guaranteed.

Fixing PCXaa00105:

To implement the Viper fix:

1. Set RH_DMA_ADDR[0:3]
2. Set the following bits in Viper's MEMORY_CONTROL register:
   a. REFRESH[0:11] = 0xFF7
   b. SCOL = 0b1
   c. PCWAIT = 0b1
   d. RCDELAY = 0b11
3. Write RH_DMA_EXECUTE
4. Poll RH_DMA_DONE until it returns 1
5. Write a value of 0 to RH_DMA_DONE
6. When RhDMA is done, restore Viper registers.

15. ERROR HANDLING

Cutoff has no special features for error handling. Cutoff itself cannot detect any special error conditions other than the assertion of SGC/VSC bus error. The LAN/SCSI controllers can detect errors from their various interfaces. These errors are reported via interrupts.

Cutoff performs no special functions when a VSC bus error is asserted if it is a slave on the VSC bus. If Cutoff is the bus master on the VSC, refer to the following section.

15.1 Error Logging Register

If an error occurs while Cutoff is the bus master on VSC, Cutoff logs the bus master information in the error logging register (0xF082F040) by setting either bit 0, 1, or 2. Possible bus masters in Cutoff are SCSI, LAN, the DMA controller for the parallel port device, AUDIO, FAST/WIDE SCSI, and FDDI. Following is the format of the register :

Bit 0: equal to 1 if an error occured while SCSI was the bus master.

Bit 1: equal to 1 if an error occured while LAN was the bus master.

Bit 2: equal to 1 if an error occured while DMA of parallel port device was the bus master.

Bit 3: equal to 1 if an error occured while Fast/Wide SCSI was the bus master.

Bit 4: equal to 1 if an error occured while FDDI was the bus master.

Bit 5: equal to 1 if an error occured while AUDIO was the bus master.

Bit 6: equal to 1 if an error occured while Rapid Harmless DMA was the bus master.

Bit 7: reserved.

For example, if bit 0 set to 1, then the bus master capability of the SCSI subsystem is disabled. Software should reset this bit before attempting to do DMA transfer between the SCSI device and memory.

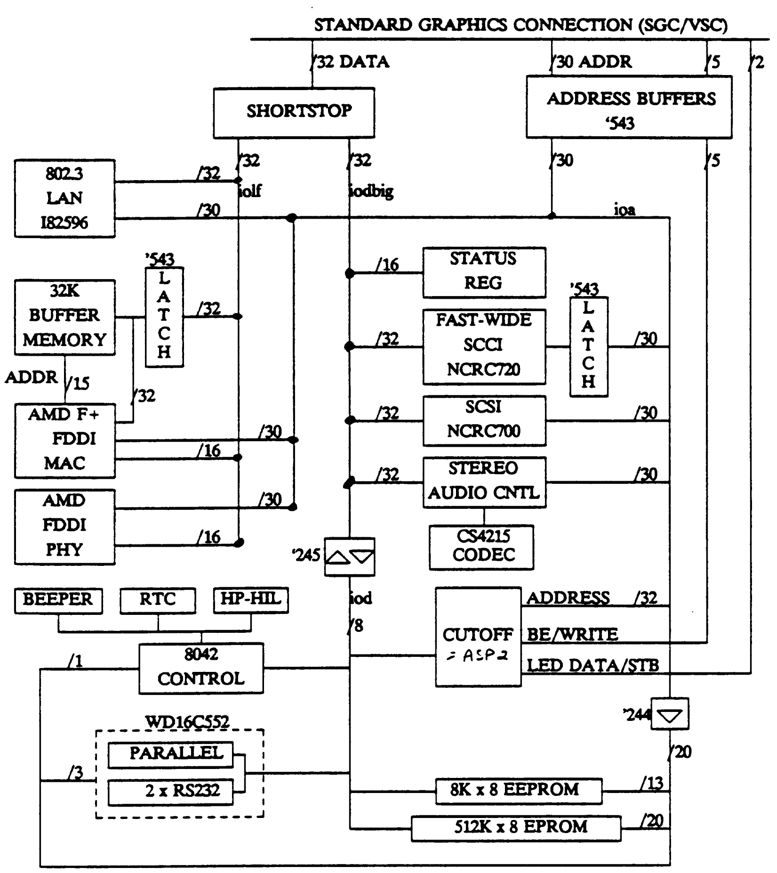
☞ For Cutoff, we have added bits 3,4, and 5 to ASP's error logging register.

16. VERIFICATION

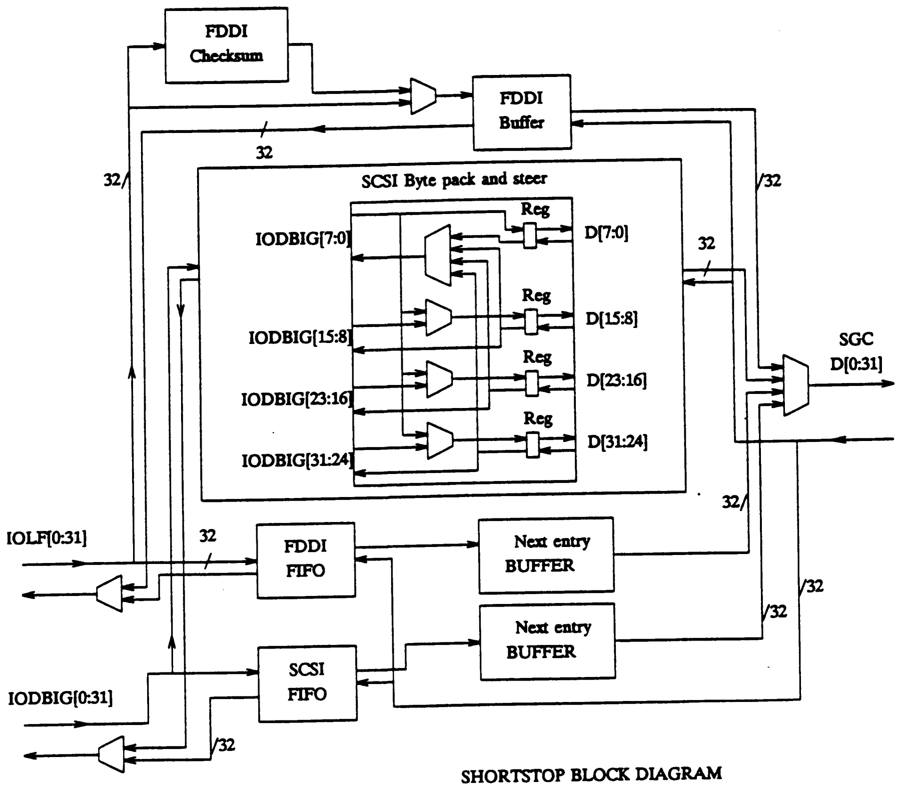
For details of the I/O subsystem design verification, refer to the document "Outfield Verification Plan".

17. Performance Measurement Page

In order to facilitate external hardware performance measurement devices, a no-op page (0xF0832xxx) was added to the memory map. All reads and writes of any size to any address in this page will cause a normal SGC cycle to complete, i.e. Cutoff will assert READYL to terminate the transaction. Writes end up in a bit bucket and reads return indeterminate data. Note that the data bus is still driven by the I/O subsystem during these transactions. The number of cycles Cutoff takes to complete the transaction will be 5-10.

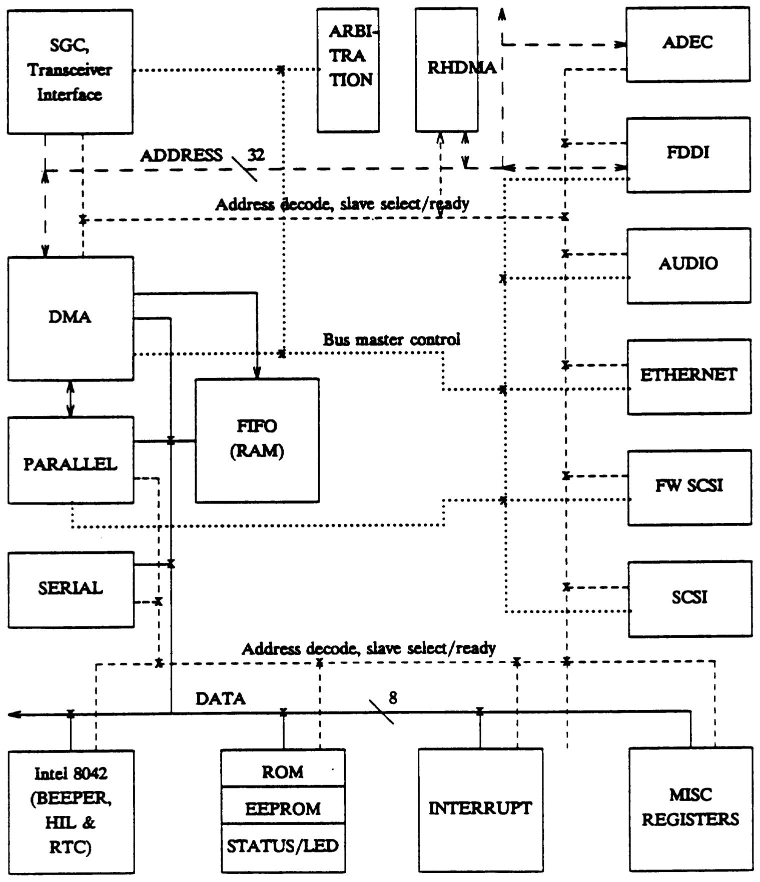


HARDBALL SUBSYSTEM I/O BLOCK DIAGRAM
DATE: 9-11-91



SHORTSTOP BLOCK DIAGRAM
AUGUST 28,1991

Figure 17-1. CUTOFF BLOCK DIAGRAM
CUTOFF BLOCK DIAGRAM



18. APPENDIX A

18.1 Memory Space

```
0xF000 0000 -   ROM Space (byte assembly; word access capable) 4M bytes
          |
0xF03F FFFF
---------------------------------------------------------------------
0xF040 0000 -   Unassigned                                 4M bytes
          |
0xF07F FFFF
---------------------------------------------------------------------
0xF080 0000 -   Interrupt/Status/LED                       64K bytes
          |
0xF080 FFFF

        0xF080 0000    r     Interrupt Request Register (Read Only)
        0xF080 0004    rw    Interrupt Mask Register (Read/Write)
        0xF080 0008    rw    Interrupt Pending Register (Read Only)
        0xF080 0020    w     Front Panel Status LED control Register
        0xF080 0024    r     I/O Subsystem Status Register
---------------------------------------------------------------------
0xF081 0000 -   EEPROM (byte access only)                  64K bytes
          |
0xF081 FFFF

        0xF081 0000    rw    EEPROM reserved for Cobra/Coral configuration
                  |
        0xF081 03FF             -     -     -     -

        0xF081 0400    rw    EEPROM reserved for EISA configuration
                  |
        0xF081 0FFF             -     -     -     -

        0xF081 1000    rw    EEPROM reserved for firmware usage
                  |
        0xF081 1BDF             -     -     -     -

        0xF081 1BE0    rw    EEPROM reserved for diagnostic usage
                  |
        0xF081 1BFF             -     -     -     -

        0xF081 1C00    rw    EEPROM reserved for EISA configuration
                  |
        0xF081 1FFF             -     -     -     -

        0xF081 2000          Currently not used (exceeds 8K bytes of EEPROM)
                  |
        0xF081 7FFF             -     -     -     -

        0xF081 8000    rw    EEPROM Enable bit access (MS bit 7)
                  |
        0xF081 8FFF    rw           -     -     -     -
----------------------------------------------------------------------
0xF082 0000 -   DMA Register                              4K bytes
     |
0xF082 0FFF

        0xF082 0000    rw     DMA ch-0 Base & Current Address register
        0xF082 0087    rw     DMA Ch0 Low page register
        0xF082 0487    rw     DMA Ch-0 High Page register
        0xF082 0001    rw     DMA ch-0 Base & Current Count register
        0xF082 0401    rw     DMA Ch-0 High Base & Current Count
        0xF082 0008    r      DMA (0-3) Status register
        0xF082 000A    w      DMA (0-3) Write single mask bit
        0xF082 000B    w      DMA (0-3) Mode register
        0xF082 040A    rw     DMA Interrupt Pending register
        0xF082 000C    w      DMA (0-3) Clear byte pointer
        0xF082 000D    w      DMA (0-3) Master Clear
        0xF082 000E    w      DMA (0-3) Clear Mask register
        0xF082 000F    rw     DMA (0-3) Mask Register
        0xF082 0010    rw     DMA FIFO Limit Register

ASP will acknowledge the following addresses, but no action will be taken.

        0xF082 00DE    rw     DMA(0-3) Chaining Mode register
        0xF082 040B    w      DMA(0-3) Extended Mode register
        0xF082 0008    w      DMA (0-3) Command register
        0xF082 0009    w      DMA (0-3) Request register
        0xF082 04D4    r      DMA Chaining Mode Status register
        0xF082 04E0    rw     DMA CH0 Stop register bits <7:0>
        0xF082 04E1    rw     DMA CH0 Stop register bits <15:8>
        0xF082 04E2    rw     DMA CH0 Stop register bits <23:16>
----------------------------------------------------------------------
0xF082 1000 -   8042 (controls the /RTC/HP-HIL/Sound)    4K bytes
     |
0xF082 1FFF

        0xF082 1000     w      8042 Hold Reset (also holds serial #3 reset)
        0xF082 1800     wr     8042 data I/O (for the RTC/HP-HIL/Sound)
        0xF082 1801     wr     8042 status/control (for the RTC/HP-HIL/Sound)
        0xF082 1C00     w      8042 release Reset (also releases serial #3 reset)
----------------------------------------------------------------------
0xF082 2000 -   RS232 #2                                  4K bytes
     |
0xF082 2FFF

        0xF082 2000     w      RS232 Master Reset (both channels: #1 & #2)
        0xF082 280X     wr     RS232 #2 Access
  (0xF082 2804 bit 2    w      RS232 #2 RTS flow control Enable bit custom
                                    implementation; all other bits are as
                                    defined in the data sheets)
----------------------------------------------------------------------
0xF082 3000 -   RS232 #1                                  4K bytes
     |
0xF082 3FFF
```

```
        0xF082 3000     w      RS232 Master Reset (both channels: #1 & #2)
        0xF082 380X     wr     RS232 #1 Access
  (0xF082 3804 bit 2    w      RS232 #1 RTS flow control Enable bit custom
                                    implementation; all other bits are as
                                    defined in the data sheets)
----------------------------------------------------------------------
0xF082 4000 -   Parallel Printer Interface                4K bytes
     |
0xF082 4FFF

        0xF082 4000     w      Parallel Printer Interface Reset
        0xF082 4800     rw     Write/Read Data
        0xF082 4801     r      Read Status
        0xF082 4802     rw     Device Control
        0xF082 4803            (Not used)
        0xF082 4804     wr     Mode Control
        0xF082 4805     wr     IE Control/Interrupt Status
        0xF082 4806     wr     Timing Delay Counter
----------------------------------------------------------------------
0xF082 5000 -   SCSI                                      4K bytes
     |
0xF082 5FFF

        0xF082 5000     w      SCSI Reset
        0xF082 51XX     rw     SCSI Chip Registers
----------------------------------------------------------------------
0xF082 6000 -   LAN                                       4K bytes
     |
0xF082 6FFF

        0xF082 6000     w      LAN Reset
        0xF082 6004     w      LAN Port Select
        0xF082 6008     w      LAN Channel Attention
----------------------------------------------------------------------
0xF082 7000 -   DMA
     |
0xF082 7FFF

        0xF082 7000     w      DMA Reset
----------------------------------------------------------------------
0xF082 8000 -   Not supported
     |
0xF082 9FFF
----------------------------------------------------------------------
0xF082 F000 -   Cutoff                                    4K bytes
     |
0xF082 FFFF

        0xF082 F000     w      I/O Subsystem Reset
        0xF082 F020     r      Cutoff  version control byte
        0xF082 F030     w      SCSI "DSYNC" byte (0b0000001 after reset)
        0xF082 F040     rw     Error logging byte
        0xF082 F050     w      LAN control output enables.
----------------------------------------------------------------------
```

```
0xF083 0000 - Fast wide SCSI                              4K bytes
        0xF083 0000    w       Fast wide SCSI reset
        0xF083 0004    w       Number of transfers (NT)
        0xF083 0008            Reserved
              |
        0xF083 00FF
        0xF083 0100            NCR53C720 registers
              |
        0xF083 015C
        0xF083 0160            Reserved
              |
        0xF083 0FFF
0xF083 0FFF
----------------------------------------------------------------------
0xF083 1000 - FDDI                                        4K bytes
                          WRITE  -  READ
        --------------------------------------------------------------
        0xF083 1000       CMDREG1 -  ST1U         MAC(AMD) begin
        0xF083 1004       CMDREG2 -  ST1L         (data is 16 bits wide)
        0xF083 1008                  ST2U
        0xF083 100C                  ST2L
        0xF083 1010       IMSK1U  -  IMSK1U
        0xF083 1014       IMSK1L  -  IMSK1L
        0xF083 1018       IMSK1U
        0xF083 101C       IMSK1U
              |              |          |
        0xF083 110C       BCNTR   -  BCNTR        MAC (end)

        ------

        0xF083 1200   PLC_CNTRL_A - PLC_CNTRL_A  PHY(AMD) (begin)
        0xF083 1204   PLC_CNTRL_B - PLC_CNTRL_B  (data is 16 bits wide)
              |              |          |
        0xF083 1268                - LINK_ERR_CTR PHY (end)

        ------

        0xF083 1300    rw      FDDI_DMA_ADDR_1
        0xF083 1304    rw      FDDI_DMA_WC_1
        0xF083 1308    rw      FDDI_DMA_ADDR_2
        0xF083 131C    rw      FDDI_DMA_WC_2
        0xF083 1320    r       FDDI_XMIT_REQ
        0xF083 1324    rw      FDDI_DMA_CNTRL
        0xF083 1328    rw      FDDI_DMA_STATUS
        0xF083 132C    rw      FDDI_INTR
        0xF083 1330    rw      FDDI_INTR_MASK
        0xF083 1334    rw      FDDI_SUBS_RESET
0xF083 1FFF
----------------------------------------------------------------------
0xF083 2000 - No-op (performance measurement)             4K bytes
      |                rw
0xF083 2FFF
----------------------------------------------------------------------
0xF083 3000 - PCXaa00105 Viper reset bug fix              4K bytes
```

```
        0xF083 3000    w     Rapid harmless DMA address byte 0 (high)
        0xF083 3004    w     Rapid harmless DMA address byte 1
        0xF083 3008    w     Rapid harmless DMA address byte 2
        0xF083 300C    w     Rapid harmless DMA address byte 3 (low)
        0xF083 3010    w     Rapid harmless DMA execute
        0xF083 3020    w     Rapid harmless DMA done (readable at 0xF082F020)
0xF083 3FFF
----------------------------------------------------------------------
0xF100 0000 - Audio (Jupiter)                             4K bytes
      |                rw
0xF100 0FFF
----------------------------------------------------------------------
```