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 (optional)
- 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)

HEWLETT-PACKARD
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 icodb data bus; the rest of the subsystem sits on an 8 bit iocl 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 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0 | 7 | 8 | 15 | 16 | 23 | 24 | 31 |
| 0 | 1 | 2 | 3 |

SGC and IOSS bit and byte offset numbering:

<table>
<thead>
<tr>
<th>31</th>
<th>24</th>
<th>23</th>
<th>16</th>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 not do 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 Harness DMA and PDDI. 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

4.1 Introduction

"CUTOFF" is the controller ASIC chip for the I/O subsystem and interfaces with the "Viper" controller and the S GC 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, S GC 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 Mbytes/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 clear the entire register to be cleared. Since a write to 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 zero, the corresponding pending interrupt cannot 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 unmasks (IMR bit = 1) IPR bits to be cleared. Both the IRR and the unmasks 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

HEWLETT-PACKARD

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 assertions 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

CR The following registers should be accessed as words:
0x008 0000 - Interrupt Request Register (Read Only)
0x008 0004 - Interrupt Mask Register (Read/Write)
0x008 0008 - Interrupt Pending Register (Read/Write)

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

<table>
<thead>
<tr>
<th>BIT</th>
<th>LEVEL</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>LOW</td>
<td>NMI from EISA</td>
</tr>
<tr>
<td>30</td>
<td>HIGH</td>
<td>8042 (tch, hil, audio) General interrupts</td>
</tr>
<tr>
<td>29</td>
<td>LOW</td>
<td>8042 (tch, hil, audio) High priority interrupts</td>
</tr>
<tr>
<td>28</td>
<td>LOW</td>
<td>Fast/Wide SCSI</td>
</tr>
<tr>
<td>27</td>
<td>LOW</td>
<td>FDD1</td>
</tr>
<tr>
<td>26</td>
<td>HIGH</td>
<td>WD16CS2 SIO 1</td>
</tr>
<tr>
<td>25</td>
<td>HIGH</td>
<td>WD16CS2 SIO 2</td>
</tr>
<tr>
<td>24</td>
<td>LOW</td>
<td>WD16CS2 Parallel Printer Interface</td>
</tr>
<tr>
<td>23</td>
<td>LOW</td>
<td>LAN-802.3</td>
</tr>
<tr>
<td>22</td>
<td>LOW</td>
<td>SCSi</td>
</tr>
<tr>
<td>21</td>
<td>LOW</td>
<td>EISA</td>
</tr>
<tr>
<td>20</td>
<td>LOW</td>
<td>Graphics1</td>
</tr>
<tr>
<td>19</td>
<td>LOW</td>
<td>Graphics2</td>
</tr>
<tr>
<td>18</td>
<td>LOW</td>
<td>Audio</td>
</tr>
<tr>
<td>17</td>
<td>LOW</td>
<td>Error</td>
</tr>
</tbody>
</table>

CR In ASP, bit 26 was timer 1, bit 27 was timer 2, and bit 18 was serial port.

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
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 (P082 P000), 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 (P082 P020) 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 F893 3020)
- D[6:5] = NTM/IO bit (from address F893 0004)
- D[4] = Dsync bit (from address P082 P030)
- 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 registers which are write only. For more details on those registers, refer to the memory map for the appropriate sections.

The third byte access register (P080 P030) 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 (P082 P040) 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.Gen (P082 P050), enables (un-tristates) Cutoff's output-only FDDI pins, output-only Ethernet pins (82596/8253 controller), or both. This allows external loading of FDDI and 8253 signals to a twisted pair board by tying 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.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>Ethernet</td>
<td>disable</td>
<td>enable</td>
</tr>
<tr>
<td>F</td>
<td>FDDI</td>
<td>disable</td>
<td>enable</td>
</tr>
<tr>
<td>R</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Firmware must follow the following algorithm:

if (LAN controller signals muxed on board)  # bit 14 of IOSS status register (P0800024)
then
   case NETWORK_ID of
      8253: enable Ethernet signals;
      FDDI: enable FDDI signals;
   end_case;
else
   Hewlett-Packard

HEWLETT-PACKARD
4.6 Power Up Reset

After power-on reset or on-board reset (write to 0x0F82F000) the Cutoff is in the following state:

<table>
<thead>
<tr>
<th>Address</th>
<th>Register Name</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0F80 0000</td>
<td>Interrupt request register</td>
<td>0x0000 (all requesting interrupts zeroed to prevent spurious interrupts at power-up).</td>
</tr>
<tr>
<td>0x0F80 0004</td>
<td>Interrupt mask register</td>
<td>0x0000 (all interrupts masked)</td>
</tr>
<tr>
<td>0x0F82 4802</td>
<td>Parallel Device Control</td>
<td>0xxx101000; 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.</td>
</tr>
</tbody>
</table>

0x0F81 8xxx  **HEEPROM enable bit**

<table>
<thead>
<tr>
<th>Address</th>
<th>Register Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0F82 0000</td>
<td>DMA current addr. reg.</td>
<td>0x0000 DMA is disabled and all addresses pointers point to 0 and all count registers are 0. Interrupts are disabled from the DMA controller.</td>
</tr>
<tr>
<td>0x0F82 0001</td>
<td>DMA current count reg.</td>
<td>0x0000 DMA FIFO Limit Register: Write to 0x0F820010. Power up sets this to 0x80, which causes Cutoff to not 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.”</td>
</tr>
<tr>
<td>0x0F82 0008</td>
<td>DMA status reg.</td>
<td>0x00 DMA is disabled and all addresses pointers point to 0 and all count registers are 0. Interrupts are disabled from the DMA controller.</td>
</tr>
<tr>
<td>0x0F82 000B</td>
<td>DMA write single bit mask</td>
<td>0x04</td>
</tr>
<tr>
<td>0x0F82 000C</td>
<td>DMA mode reg.</td>
<td>0x04 DMA is disabled and all addresses pointers point to 0 and all count registers are 0. Interrupts are disabled from the DMA controller.</td>
</tr>
<tr>
<td>0x0F82 000D</td>
<td>DMA clear byte pointer</td>
<td>byte pointer = 0 DMA is disabled and all addresses pointers point to 0 and all count registers are 0. Interrupts are disabled from the DMA controller.</td>
</tr>
<tr>
<td>0x0F82 000E</td>
<td>DMA clear mask reg.</td>
<td>mask = 1 DMA is disabled and all addresses pointers point to 0 and all count registers are 0. Interrupts are disabled from the DMA controller.</td>
</tr>
<tr>
<td>0x0F82 000F</td>
<td>DMA mask reg.</td>
<td>0x0E DMA is disabled and all addresses pointers point to 0 and all count registers are 0. Interrupts are disabled from the DMA controller.</td>
</tr>
<tr>
<td>0x0F82 0010</td>
<td>DMA FIFO limit reg.</td>
<td>0x80 DMA is disabled and all addresses pointers point to 0 and all count registers are 0. Interrupts are disabled from the DMA controller.</td>
</tr>
<tr>
<td>0x0F82 0087</td>
<td>DMA current addr. low page</td>
<td>0x00 DMA is disabled and all addresses pointers point to 0 and all count registers are 0. Interrupts are disabled from the DMA controller.</td>
</tr>
<tr>
<td>0x0F82 0401</td>
<td>DMA high current count</td>
<td>0x2F DMA is disabled and all addresses pointers point to 0 and all count registers are 0. Interrupts are disabled from the DMA controller.</td>
</tr>
<tr>
<td>0x0F82 040A</td>
<td>DMA interrupt log reg.</td>
<td>0x00 DMA is disabled and all addresses pointers point to 0 and all count registers are 0. Interrupts are disabled from the DMA controller.</td>
</tr>
<tr>
<td>0x0F82 0487</td>
<td>DMA current addr. high page</td>
<td>0x00 DMA is disabled and all addresses pointers point to 0 and all count registers are 0. Interrupts are disabled from the DMA controller.</td>
</tr>
</tbody>
</table>

Note in addition the various controllers power up into defined states. This definition is available in the vendor documentation for the NCR53C700, the latest 82596, the Western Digital WD16652 (or National NS16650), 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 30 ms.

4.6.1 Initialization

Firmware/OS needs to initialize the following Cutoff registers:

- **DSYNC**: Byte write to 0x00 to 0x0F82F000. At power up, all interrupts are disabled. Do NOT enable the interval timers.
- **DMA FIFO Limit Register**: Byte write to 0x0F820010. Power up sets this to 0x80, which causes Cutoff to not 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 0x0F824806. 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)/3, where DELAYTIME is device dependent. See Bidirectional Parallel Printer Interface section “Timing Delay Counter.”
- **8042 Reset Release**: Byte write anything to 0x0F821C00. 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 PDI 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 CMOS66 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 (gsk).

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
```

HEWLETT-PACKARD

4.9.2.1 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

In this test mode, there are other 3 pins on Cutoff are multiplexed to verify the scan chain functions: NeepWootPens, NeepOccal, and scikscalScal (NeepWootPens is the serial phase enable pin for the scan chain, NeepOccal is scan-in data pin and scikscalScal 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 Pstest mode

This test mode is for testing the F 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 ready 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 signal). 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.

HEWLETT-PACKARD
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 clap 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 666'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 PDDSL to improve performance on master burst transactions. It contains no registers.

5.2 External Signal Definitions

<table>
<thead>
<tr>
<th>Signal</th>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>D[31:0]</td>
<td>32 I/O</td>
<td>SOC data bus</td>
</tr>
<tr>
<td>ICDBG[31:0]</td>
<td>32 I/O</td>
<td>Data bus to SCSI, F-W SCSI, Status register, Audio, and 8-bit bus</td>
</tr>
<tr>
<td>IOLF[31:0]</td>
<td>32 I/O</td>
<td>Data bus to Lan and PDDI</td>
</tr>
<tr>
<td>NOINHIBIT</td>
<td>1 I</td>
<td>Active low inhibits CCLK[1] from clocking the buffers</td>
</tr>
<tr>
<td>CCLK[1]</td>
<td>1 I</td>
<td>Clock for the data buffers from SOC to I/O</td>
</tr>
<tr>
<td>NSCDBN</td>
<td>1 I</td>
<td>Active low enables data in buffer mode</td>
</tr>
<tr>
<td>SOCDROMD</td>
<td>1 I</td>
<td>High: data path from I/O to SOC is registered Low: data path from I/O to SOC is transparent</td>
</tr>
<tr>
<td>ICROMD</td>
<td>1 I</td>
<td>High: data path from SOC to I/O is registered Low: data path from SOC to I/O is registered</td>
</tr>
<tr>
<td>ICCLK[3:0]</td>
<td>4 I</td>
<td>Clocks for the data buffers from I/O to SOC</td>
</tr>
<tr>
<td>DDINBA</td>
<td>1 I</td>
<td>Low: I/O data to SOC, high: SOC data to I/O</td>
</tr>
<tr>
<td>NTSTR[1:0]</td>
<td>2 I</td>
<td>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</td>
</tr>
<tr>
<td>BUSSELECT</td>
<td>1 I</td>
<td>High: bus for transaction is ICDBG Low: bus for transaction is IOLF</td>
</tr>
<tr>
<td>PMSOCDATAL</td>
<td>1 I</td>
<td>On write: Data enable from PMS FIFO onto SOC On read: Data enable from SOC into PMS FIFO</td>
</tr>
<tr>
<td>PMLCODATAL</td>
<td>1 I</td>
<td>On write: Data enable from ICDBG to PMS FIFO On read: Data enable from PMS FIFO to ICDBG</td>
</tr>
<tr>
<td>PMSWR1L</td>
<td>1 I</td>
<td>Active low: transaction is a write to memory</td>
</tr>
</tbody>
</table>
### 5.4.2 AC Characteristics

#### 5.4.2.1 Clock Specifications

- Maximum clock speed: 33 MHz (gcik)

### 5.4.3 Packaging Technology and Thermal Requirements

- 160 pin PLOCC
- 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 ScanInMode and TrimMode will be used in the same way. The 3 other that are multiplexed for scan chain functions are PASEMPTE (Scan In), SOCREMODE (Scan In), SOCREMOD (Serial phase enable). Shortstop will utilize the Automatic Test pattern Generation tool (ATG) within the Architect.

### 5.3 IC Process Technology

Shortstop 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.

---

<table>
<thead>
<tr>
<th>Pin</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASEMPTE</td>
<td>0</td>
<td>High: PAS FIFO is empty</td>
</tr>
<tr>
<td>RESETPSL</td>
<td>1</td>
<td>Reset the PAS FIFO</td>
</tr>
<tr>
<td>PDD1LOCENTAL</td>
<td>1</td>
<td>On write: Data enable from IOFL into PDDI FIFO On read: Data enable from PDDI FIFO onto IOFL</td>
</tr>
<tr>
<td>PDD1SOCCENTAL</td>
<td>1</td>
<td>On write: Data enable from PDDI FIFO onto SOCC On read: Data enable from SOCC into PDDI FIFO</td>
</tr>
<tr>
<td>PDD1WRI</td>
<td>1</td>
<td>Active low: transaction is a write to memory</td>
</tr>
<tr>
<td>PDD1CSSML</td>
<td>1</td>
<td>On low edge: start to accumulate the checksum On high edge: clear the checksum</td>
</tr>
<tr>
<td>PDD1CRSMOD</td>
<td>1</td>
<td>On low edge: read the checksum into the PDDI buffers</td>
</tr>
<tr>
<td>RESETPDD1L</td>
<td>1</td>
<td>Reset the PDDI FIFO</td>
</tr>
<tr>
<td>REACT</td>
<td>1</td>
<td>SOC ready</td>
</tr>
<tr>
<td>CRCLK</td>
<td>1</td>
<td>SOC gcik</td>
</tr>
<tr>
<td>TRIMODE</td>
<td>1</td>
<td>Shortstop tri-state mode</td>
</tr>
<tr>
<td>SCANMODE</td>
<td>1</td>
<td>Shortstop scan path enable</td>
</tr>
</tbody>
</table>

Count for pin type "I" is 27
Count for pin type "O" is 96

---

Total signal pin count is 124
Maximum signal pins desired: 131 for a 160 pin package

---

HEWLETT-PACKARD
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 widths.
- 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.

\[ \text{NOTES:} \]

HEWLETT-PACKARD

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

6.5 Power Up Reset

The following conditions occur after a power up reset: (ensures):

- All the parallel printer interface state machine controllers in ap are reset
- NSTB = 1
- NAFD (alias WRanRD, WR/anRD) = 1
- NINT (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, NINT, 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 0xF082 4804 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.

HEWLETT-PACKARD
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.
  
  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.

  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.

6.5.4 Byte Address 0xF082 4802 (Parallel Device Control)

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

HEWLETT-PACKARD
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 time programmed via address
F0E2 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.

Bidirectional: 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 usec,
then clear Wr/Nrd (NAPD) = 0. A transfer may then be
invoked. This bit does not affect the WRA/RD 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:
HEWLETT-PACKARD

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 gets 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 chooses
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.

- **BIT 6** No affect.

- **BIT 5** NBury IE: Enable interrupts on BUSY trailing edge 1 to 0
transition.

- **BIT 4** NACK IE: Enable interrupts on NACK trailing edge 0 to 1
transition.

- **BIT 3** ackNbury IE: Enable interrupts on NACK 0 to 1 transition
was received and BUSY = 0 (not busy).

- **BIT 2** PE IE: Enable interrupts on any PE transition.
Select IE: Enable interrupts on any SLCT transition.

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.

Returns 0.

Nbusy IR: Busy 1 to 0 transition (not busy). This bit is cleared only by writing a zero into it.

ACK IR: NACK 0 to 1 transition occurred. This bit is cleared only by writing a zero into it or by reading byte address 1.

ackNbusy IR: NACK 0 to 1 transition occurred and BUSY = 0. This bit is cleared only by writing a zero into it.

PE IR: The PE line has made any transition. This bit is cleared only by writing a zero into it.

Select IR: The SLCT line has made any transition. This bit is cleared only by writing a zero into it.

Error IR: The NERR line has made any transition. This bit is cleared only by writing a zero into it.

• 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:

• Bid: = 1 (0x0824804 bit 4)

• NwRd = 1 (0x0824802 bit 5)

• Wait 5 microseconds.

• Make Nabf (WRead) = 0 (0x0824802 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

HEWLETT-PACKARD
6.5.9 Caveats when Accessing Parallel registers

When accessing byte address 0xFO00 4800, 0xFO02 4801, 0xFO02 4802, or 0xFO02 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 0xFO02 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 nsec. 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

<table>
<thead>
<tr>
<th>BIT</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIT VALUE</td>
<td>80</td>
<td>40</td>
<td>20</td>
<td>10</td>
<td>08</td>
<td>04</td>
<td>02</td>
<td>01</td>
</tr>
<tr>
<td>0802 4800</td>
<td>DATA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0802 4801</td>
<td>Nbusy</td>
<td>Nack</td>
<td>pe</td>
<td>sict</td>
<td>Nerr</td>
<td>Nint</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0802 4802</td>
<td>1</td>
<td>1</td>
<td>Nerr</td>
<td>IrfEnb</td>
<td>Nint</td>
<td>Nint</td>
<td>AutoFd</td>
<td>Strobe</td>
</tr>
<tr>
<td>0802 4803</td>
<td>Not Used (undefined)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0802 4805</td>
<td>demint</td>
<td>0</td>
<td>Nbusy</td>
<td>Nack</td>
<td>latch</td>
<td>pulst</td>
<td>sict</td>
<td>errint</td>
</tr>
<tr>
<td>0802 4806</td>
<td>Timing Delay value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>TYPE</th>
<th>SIZE (BYTES)</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>F020000</td>
<td>rw</td>
<td>1</td>
<td>DMA Current Address register</td>
</tr>
<tr>
<td>F020001</td>
<td>rw</td>
<td>1</td>
<td>DMA Current Count register</td>
</tr>
<tr>
<td>F020008</td>
<td>rw</td>
<td>1</td>
<td>DMA Status register</td>
</tr>
<tr>
<td>F0200A0</td>
<td>wo</td>
<td>1</td>
<td>DMA Write single mask bit</td>
</tr>
<tr>
<td>F0200B0</td>
<td>wo</td>
<td>1</td>
<td>DMA Mode register</td>
</tr>
<tr>
<td>F0200C0</td>
<td>wo</td>
<td>1</td>
<td>DMA Clear byte pointer</td>
</tr>
<tr>
<td>F0200D0</td>
<td>wo</td>
<td>1</td>
<td>DMA Master Clear</td>
</tr>
<tr>
<td>F0200E0</td>
<td>wo</td>
<td>1</td>
<td>DMA Clear Mask register</td>
</tr>
<tr>
<td>F0200F0</td>
<td>rw</td>
<td>1</td>
<td>DMA Mask register</td>
</tr>
<tr>
<td>F020100</td>
<td>rw</td>
<td>1</td>
<td>DMA FIFO limit register</td>
</tr>
<tr>
<td>F02017F</td>
<td>rw</td>
<td>1</td>
<td>DMA Current Address low page register</td>
</tr>
<tr>
<td>F020401</td>
<td>rw</td>
<td>1</td>
<td>DMA High Current Count</td>
</tr>
<tr>
<td>F02040A</td>
<td>rw</td>
<td>1</td>
<td>DMA Interrupt Pending register</td>
</tr>
<tr>
<td>F020477</td>
<td>rw</td>
<td>1</td>
<td>DMA Current Address high Page register</td>
</tr>
</tbody>
</table>

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 set. After the sequence is complete an interrupt will happen, the mask bit will be set and the address and count registers will be at their final values. 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 workaround 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.

HEWLETT-PACKARD

6.7.2.2 DMA Control Bits

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

- `dma_mode_register` F02 000B[1:0]
- `dma_mask_bit` F02 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 affect 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, F02 0008[0] is set to 1.

HEWLETT-PACKARD
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:
- F802 0000(0) is cleared to 0 (DMA in progress)
- F802 0000(0) is set to 1 (DMA mask bit)
- F802 040A(0) is set to 1 (DMA interrupt bit)
- F802 4005(7) is set to 1 (parallel DMA terminal count)
- if F802 0000(7) is clear, F802 4005(7) is set and an external interrupt request is forwarded to Aap to Viper

Also, the parallel port handshake interrupt bit (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 back in the evolution of the PC-AT the 8-bit Intel 8237 DMA controller was popular. All accesses to the aap 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.

6.7.4 Detailed Register Descriptions

6.7.4.1 Current Address Register

- F802 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 F802 000C (Clear_BYTE_Pointer), or a write to F802 000D (Master_Clear).

- F802 0087 (Current_Address_Low_Page) rw init: 0
  8 bit read/write

- F802 0487 (Current_Address_High_Page) rw init: 0
  8 bit read/write

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

-- Additional details on DMA functionality and register descriptions --
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 8277 compatible segment (address 0000) combined 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 big read/write to the low page segment (address 0087) for Address register bits [22:16].

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

6.7.4.2 Byte Count Register

P082 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 P082 000C (Clear_Byte_Pointer), or a write to P082 000D (Master_Clear).

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

P0820401 (Current_Counter_High_Byte) rw init:FF

DMA transfer length is specified by a 24 bit register:

<table>
<thead>
<tr>
<th>High Count</th>
<th>Current Count</th>
</tr>
</thead>
</table>

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 00000000) after reset.

The Byte Count register consists of two parts, the 16-bit 8277 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 8 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

P082 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 bit is 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

P082 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 this bit.

HEWLETT-PACKARD
6.7.4.5 DMA Mode register

FOR2 000B (Mode_Register)  \( \text{rw init: 01} \)

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 0: Demand Mode DMA transfer.

State after reset is 0011: DMA is in write mode.

Examples:
A write of XXXXXXX000 enables data
A write of XXXXXXX010 sets transfer type to write (Hardball to peripheral)
A write of XXXXXXX100 sets transfer type to read (peripheral to Hardball)
A write of XXXXXXX110 disables data
A write of XXXXXXX111 has no effect
A write of XXXXXXX111 has no effect
bit 7-4 hardwired to zero

6.7.4.6 DMA Interrupt Logging Register

FOR2 040A (Interrupt_Log_Bits)  \( \text{rw init: 0} \)

bit 0 - Indicate if there is a pending interrupt on Parallel Printer Interface DMA channel. This bit is cleared after reset.


HEWLETT-PACKARD

6.7.4.7 DMA Mask Register

FOR2 000F (Mask_Register)  \( \text{rw init: 000011} \)

Bit 1 is the dma_mask_bit.
Writing 1 to bit 0 enables (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 read 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

FOR2 0010 (FIFO_Limit_Register)  \( \text{rw init: 1000000} \)

[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 limit 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 limit a memory write transaction will be mastered on the SGC bus.
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 width 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 width 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

POR2 0000C (Clear_byte_Pointer) w/o 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 registers. 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

POR2 0000D (Master_Clear) w/o init: no value
A write to this address clears the byte_pointer and 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

POR2 0000E (Clear_Mask_Register) w/o 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.
which is capable of emulating all of the different peripherals supported by the design.

HEWLETT-PACKARD
6.10 I/O Connector

<table>
<thead>
<tr>
<th>HOST SIDE</th>
<th>Hosting</th>
<th>DB-25 PIN</th>
<th>PERIPHERAL</th>
<th>ALIAS NAMES</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST SIZE</td>
<td>SIDE</td>
<td>PIN #</td>
<td>PIN #</td>
<td></td>
</tr>
<tr>
<td>BUSY</td>
<td>I/O</td>
<td>10</td>
<td>10</td>
<td>ACD-0, NACK, NACKNLG</td>
</tr>
<tr>
<td>DS0</td>
<td>I/O</td>
<td>11</td>
<td>11</td>
<td>BUSY-1, Busy</td>
</tr>
<tr>
<td>DS1</td>
<td>I/O</td>
<td>12</td>
<td>12</td>
<td>IR/SO, ERROR, paper ERROR, PError</td>
</tr>
<tr>
<td>DS2</td>
<td>I/O</td>
<td>13</td>
<td>13</td>
<td>SLCT-1, SELECT, SelOut</td>
</tr>
<tr>
<td>DS3</td>
<td>O</td>
<td>14</td>
<td>14</td>
<td>WRD, WR/WRD, N/AUTO_FEED, X, N/AutoFd</td>
</tr>
<tr>
<td>DS4</td>
<td>I</td>
<td>15</td>
<td>15</td>
<td>NFault, NFault, NFault</td>
</tr>
<tr>
<td>DS5</td>
<td>O</td>
<td>16</td>
<td>16</td>
<td>NRESET, NRESET, nil</td>
</tr>
<tr>
<td>DS6</td>
<td>O</td>
<td>17</td>
<td>36</td>
<td>NSLCTIN, NSLCT_IN, nSelectIn, nSelIn</td>
</tr>
<tr>
<td>GROUND</td>
<td>-</td>
<td>18-25</td>
<td>19-29</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(twisted pair ground returns see the following group for breakout)</td>
</tr>
</tbody>
</table>

| GROUND    | -       | 19       |             | (twisted pair ground return for DS0) |
| GROUND    | -       | 20-27    |             | (twisted pair ground return for DS0-D7) |
| GROUND    | -       | 28       |             | (twisted pair ground return for BUSY) |
| GROUND    | -       | 29       |             | (twisted pair ground return for NACK) |
|           | -       | 15       |             | 0 VDC |
|           | -       | 16       |             | CHASSIS GND |
|           | -       | 17       |             | +5 VDC |
|           | -       | 18       |             | SIGNAL GND |
|           | -       | 30       |             | AUXOUT1, AUXOUT2 |
|           | -       | 31       |             | AUXOUT1, AUXOUT2 |

6.11 Electrical Specifications

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 R8232C 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/ VcomRA/AT pinsout
- EIA R8232C 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
- 19.2K baud inbound w/no data loss w/software XON/XOFF flow control only
- 230.4K baud 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, 16 byte programmable FIFO interrupt level
- Programmable loop-back control for testing
- Supported as console w/HP TERM 0, ANSI, ASCII terminals

HEWLETT-PACKARD

7.4 Register Set

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

- Serial channel #1: 0x0F80 3800
- Serial channel #2: 0x0F82 3800

See the Western Digital "WD16C252 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 WD16C252 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 (PCR0=0) or in the FIFO Mode (PCR0=1, PCR3=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 (PCR0=1) when the PCR3=1 and the trigger level or the timeout

HEWLETT-PACKARD
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 an input FIFO or holding register data over-run 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 signifying 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: 0x0F0E 3904
- Serial channel #2: 0x0F0E 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: 0x0F0E 3000
- Serial channel #2: 0x0F0E 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, NSTD, NAFD, NRST, and NSLIN to be initialized to the reset state!*

0x0F0E 1000 Writing anything to this address causes reset "hold" (also resets 8042).
0x0F0E 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 0x0F0E 1000 to set and hold the reset condition.
- wait more than 100 microseconds then write to address 0x0F0E 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" data sheets.
7.7.1 Cavets

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:

<table>
<thead>
<tr>
<th>DTE</th>
<th>HOST</th>
<th>HOST</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST SIDE</td>
<td>SIDE</td>
<td>DB-9</td>
</tr>
<tr>
<td>BUS SIGNALS</td>
<td>DIR</td>
<td>PIN #</td>
</tr>
<tr>
<td>CD</td>
<td>I</td>
<td>1</td>
</tr>
<tr>
<td>RXD</td>
<td>I</td>
<td>2</td>
</tr>
<tr>
<td>TxD</td>
<td>O</td>
<td>3</td>
</tr>
<tr>
<td>DRX</td>
<td>O</td>
<td>4</td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>DSR</td>
<td>I</td>
<td>6</td>
</tr>
<tr>
<td>RTS</td>
<td>O</td>
<td>7</td>
</tr>
<tr>
<td>CTS</td>
<td>I</td>
<td>8</td>
</tr>
<tr>
<td>RI</td>
<td>I</td>
<td>9</td>
</tr>
</tbody>
</table>

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 de facto 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 Handball/CRJAL 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 Handball/CRJAL. 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 its 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 0XFD825000 is a write only register. A write to this register will cause a direct reset to the SCSI subsystem.

Registers 0XFD825100 to 0XFD82513F are implemented inside the NCR53C700. See the NCR53C700 data sheet for the definitions of each field within each register.

Bit 11:9 of Register 0XFD800024 also contains the Snakes System ID. Register 0XFD800024 contains the Rev level of the I/O subsystem. Drivers can use the information in the above two registers to optimize performance.
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 0x0F082510 to 0x0F082513F, 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 + 04" should be supplied, and etc. In this case, the "SCSI_slot_base_add" = 0x0F08251.

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.

<table>
<thead>
<tr>
<th>address with offset</th>
<th>data field on SGC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 4, 8, and C</td>
<td>D31 - D24</td>
</tr>
<tr>
<td>1, 5, 9, and D</td>
<td>D23 - D16</td>
</tr>
<tr>
<td>2, 6, A, and E</td>
<td>D15 - D8</td>
</tr>
<tr>
<td>3, 7, B, and F</td>
<td>D7 - D0</td>
</tr>
</tbody>
</table>

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_slot_base_add + 028"), then the data pattern 0x78563412 should be supplied from the host on the SGC bus. The SCSI registers with more than 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-0 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 1.3.7 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/XX High-Performance 32-Bit Local Area Network Coprocessor (Order Number: 290213)
- 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 (AUT). 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 0xF80E0008. 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 0xFOE25004. Valid data should be placed on the highest/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 0xFOE1 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.
9.7.2 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 CPU 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# signal, which by definition is asserted on power up. Code must wait for 10 system clocks and 5 transmit clocks (30 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 the port 03F8H. A Channel Attention following a hardware reset will cause the 82596 to access the SCB, 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 SCB, 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 (30 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 in 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 CPU 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 errors 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 handbook 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

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

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 hardware inactive. These two measures in tandem with the burst 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_CNTR 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.
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 512Kx8 EEPROM 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-volatile 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 = 0xx xxxx, read data = 1xx xxxx). Once the automatic programming cycle is complete, I/O-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.

HEWLETT-PACKARD

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 to 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 write of code that also wants to program the EEPROM.
- Write 10 microseconds minimum (to satisfy the pre-write delay timer 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 [19:20] (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/cycle between writing successive bytes within a page.
- Wait/poll for the write (or writes) to complete which typically takes 5 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

HEWLETT-PACKARD
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 8255 slave microprocessor which acts as a "server" for these devices. Access to the devices is done through the 8042 command/data protocol under "CutFff" 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 - 0xP08 1800 data I/O
  - 428003 - 0xP08 1801 status/control
  - 478005 - 0xP08 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:

- 0xP08 1000 Writing anything to this address causes reset "hold" (also resets the Domain keyboard SIO #3 [no components will be loaded for this port]).
- 0xP08 1000 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:**

Hewlett-Packard
10.6 Front Panel for Trailways, Strider, and Hardball

Trailways and Strider are included for reference.

Front Panel when units are in vertical position:

<table>
<thead>
<tr>
<th>Trailways (Top)</th>
<th>Strider (Top)</th>
<th>Hardball (Top)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. 5 V On (Grn)*</td>
<td>1. 5 V On (Grn)*</td>
<td>1. 5 V On (Grn)*</td>
</tr>
<tr>
<td>2. BR Diag/Lan Xmit*</td>
<td>2. BR Diag/Lan Xmit*</td>
<td>2. BR Diag/Lan Xmit*(LSB)</td>
</tr>
<tr>
<td>3. BR Diag/Lan Recv*</td>
<td>3. BR Diag/Lan Recv*</td>
<td>3. BR Diag/Lan Recv*</td>
</tr>
<tr>
<td>5. BR Diag/Heartbt*</td>
<td>5. BR Diag/Heartbt*</td>
<td>5. BR Diag/Heartbt*</td>
</tr>
<tr>
<td>6. BR Diag</td>
<td>6. BR Diag</td>
<td>6. BR Diag</td>
</tr>
<tr>
<td>7. BR Diag</td>
<td>7. BR Diag</td>
<td>7. BR Diag</td>
</tr>
<tr>
<td>8. BR Diag</td>
<td>8. BR Diag</td>
<td>8. BR Diag</td>
</tr>
<tr>
<td>9. BR Diag</td>
<td>9. BR Diag</td>
<td>9. BR Diag(MSB)</td>
</tr>
<tr>
<td>10. Service On (Grn)</td>
<td>10. Service On (Grn)</td>
<td>10. Service On (Grn)</td>
</tr>
</tbody>
</table>

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 "Boostron 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

HEWLETT-PACKARD
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

5/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:

- **Bit**
- **[2:0]** SCSI ID. (see table below )

<table>
<thead>
<tr>
<th>[2:0]</th>
<th>SCSI ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>111</td>
<td>7</td>
</tr>
</tbody>
</table>

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 hard decoded to 01

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

HEWLETT-PACKARD
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 addition 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

HEWLETT-PACKARD
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
- Burst mode 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 outfld 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)
11.6.2  iodbg[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 overhead, 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 iodbg[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 0x000000A, 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 its loss of the termpower or other proper causes. The driver should wait a minimum of 100 microseconds after an lost of termpower before reading the FWSCSI-TERMPOWER bit to allow sufficient time for the circuit to reset itself.

Unlike the single ended SCSI subsystem, there is no SCSI ID bit for the FWSCSI in the status register 0x0000004. 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 contents of the general purpose register (GPREG) to the SCSI ID register (SCID)

Register 0x0F800000 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 fdiow in Shortstop.

Register 0x0F800004 is the NT register (number of transfers). It is a write only register and is implemented in Cutoff. Bit[10] of the NT register should have the same value as the BL1 and BL0 bits in the DMODE register inside the NCR53C720 (address = 0x0F80013B) 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 = 0X000202)

Registers 0x0F80100 to 0x0F8015C 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.

HEWLETT-PACKARD
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 FWSCI subsytem, the "FWSCSI, slot_base_add" = 0x000301.

<table>
<thead>
<tr>
<th>Address with offset</th>
<th>Data field on SGC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 4, 8, and C</td>
<td>D31 - D25</td>
</tr>
<tr>
<td>1, 5, 9, and D</td>
<td>D23 - D16</td>
</tr>
<tr>
<td>2, 6, A, and E</td>
<td>D15 - D8</td>
</tr>
<tr>
<td>3, 7, B, and F</td>
<td>D7 - D0</td>
</tr>
</tbody>
</table>

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 0x003D0124 and then extract the 24 bits belongs to the DBC register.

11.9 Initializing the registers related to the FWSCI subsystem

In order for the FWSCI 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 internally connect SLACK/ to TA/ in slave mode. Address for the DCNTL register is 0x030038

* 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 (0x0F0311B) should NEVER be set. Cache 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 [15] 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 = 0x030004

"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 FWSCI fifo in Shortstop is NOT empty. Refer to the interrupt register inside Cutoff for more details.
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 - 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 (Am97C390) 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 Shortarp IC (fili) 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 Shortarp chip will receive/send the data based on the internal DMA 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 Shortarp 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:

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

<table>
<thead>
<tr>
<th>Address (HEX)</th>
<th>R/W</th>
<th>Word 0</th>
<th>Word 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>F081000</td>
<td>W</td>
<td>CMDREG1</td>
<td>NA</td>
</tr>
<tr>
<td>F081000</td>
<td>R</td>
<td>STIU</td>
<td>NA</td>
</tr>
<tr>
<td>F081004</td>
<td>W</td>
<td>CMDREG2</td>
<td>NA</td>
</tr>
<tr>
<td>F081004</td>
<td>R</td>
<td>STIL</td>
<td>NA</td>
</tr>
<tr>
<td>F081008</td>
<td>R</td>
<td>STIU</td>
<td>NA</td>
</tr>
<tr>
<td>F08100C</td>
<td>R</td>
<td>STIL</td>
<td>NA</td>
</tr>
<tr>
<td>F081010</td>
<td>R/W</td>
<td>BMREG1</td>
<td>NA</td>
</tr>
<tr>
<td>F081012</td>
<td>R/W</td>
<td>BMREG2</td>
<td>NA</td>
</tr>
<tr>
<td>F081018</td>
<td>R/W</td>
<td>BMREG3</td>
<td>NA</td>
</tr>
<tr>
<td>F08101C</td>
<td>R/W</td>
<td>BMREG4</td>
<td>NA</td>
</tr>
<tr>
<td>F081020</td>
<td>R/W</td>
<td>LISR</td>
<td>NA</td>
</tr>
<tr>
<td>F081024</td>
<td>R/W</td>
<td>LISR</td>
<td>NA</td>
</tr>
<tr>
<td>F081028</td>
<td>R/W</td>
<td>LISR</td>
<td>NA</td>
</tr>
<tr>
<td>F08102C</td>
<td>R/W</td>
<td>LISR</td>
<td>NA</td>
</tr>
<tr>
<td>F081030</td>
<td>R/W</td>
<td>SAGP</td>
<td>NA</td>
</tr>
<tr>
<td>F081034</td>
<td>R/W</td>
<td>SAGP</td>
<td>NA</td>
</tr>
<tr>
<td>F081038</td>
<td>R/W</td>
<td>SAGP</td>
<td>NA</td>
</tr>
<tr>
<td>F08103C</td>
<td>R/W</td>
<td>LACG</td>
<td>NA</td>
</tr>
<tr>
<td>F081040</td>
<td>R/W</td>
<td>LACG</td>
<td>NA</td>
</tr>
<tr>
<td>F081044</td>
<td>R</td>
<td>STMCHN</td>
<td>NA</td>
</tr>
<tr>
<td>F081048</td>
<td>R</td>
<td>MIR1</td>
<td>NA</td>
</tr>
<tr>
<td>F08104C</td>
<td>R</td>
<td>MIR0</td>
<td>NA</td>
</tr>
<tr>
<td>F081050</td>
<td>R/W</td>
<td>TMAX</td>
<td>NA</td>
</tr>
<tr>
<td>F081054</td>
<td>R/W</td>
<td>TMAX</td>
<td>NA</td>
</tr>
<tr>
<td>F081058</td>
<td>R/W</td>
<td>TVX</td>
<td>NA</td>
</tr>
<tr>
<td>F08105C</td>
<td>R/W</td>
<td>TVX</td>
<td>NA</td>
</tr>
<tr>
<td>F081060</td>
<td>R</td>
<td>TNEQ</td>
<td>NA</td>
</tr>
<tr>
<td>F081064</td>
<td>R</td>
<td>TMRS</td>
<td>NA</td>
</tr>
<tr>
<td>F081068</td>
<td>R/W</td>
<td>TRIO</td>
<td>NA</td>
</tr>
<tr>
<td>F08106C</td>
<td>R/W</td>
<td>TRIO</td>
<td>NA</td>
</tr>
</tbody>
</table>

**TABLE 13-1. AMD Formac Memory Map**
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.

<table>
<thead>
<tr>
<th>Address (HEX)</th>
<th>R/W</th>
<th>Word 0</th>
<th>Word 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>F0831020</td>
<td>R/W</td>
<td>PLC_CNTRL_A</td>
<td>NA</td>
</tr>
<tr>
<td>F0831024</td>
<td>R/W</td>
<td>PLC_CNTRL_B</td>
<td>NA</td>
</tr>
<tr>
<td>F0831028</td>
<td>R/W</td>
<td>INTR_MASK</td>
<td>NA</td>
</tr>
<tr>
<td>F083102C</td>
<td>R/W</td>
<td>XMTT_VECTOR</td>
<td>NA</td>
</tr>
<tr>
<td>F0831110</td>
<td>R/W</td>
<td>VECTOR_LENGTH</td>
<td>NA</td>
</tr>
<tr>
<td>F0831114</td>
<td>R/W</td>
<td>LE_THRESHOLD</td>
<td>NA</td>
</tr>
<tr>
<td>F0831118</td>
<td>R/W</td>
<td>C_MIN</td>
<td>NA</td>
</tr>
<tr>
<td>F083111C</td>
<td>R/W</td>
<td>TL_MIN</td>
<td>NA</td>
</tr>
<tr>
<td>F0831120</td>
<td>R/W</td>
<td>TB_MIN</td>
<td>NA</td>
</tr>
<tr>
<td>F0831124</td>
<td>R/W</td>
<td>T_OUT</td>
<td>NA</td>
</tr>
<tr>
<td>F0831128</td>
<td>R/W</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>F083112C</td>
<td>R/W</td>
<td>LC LENGTH</td>
<td>NA</td>
</tr>
<tr>
<td>F083112D</td>
<td>R/W</td>
<td>T_SCRUB</td>
<td>NA</td>
</tr>
<tr>
<td>F0831134</td>
<td>R/W</td>
<td>NS_MAX</td>
<td>NA</td>
</tr>
<tr>
<td>F0831138</td>
<td>W</td>
<td>TPC_LOAD_VALUE</td>
<td>NA</td>
</tr>
<tr>
<td>F083113C</td>
<td>W</td>
<td>TNL_LOAD_VALUE</td>
<td>NA</td>
</tr>
<tr>
<td>F0831140</td>
<td>R</td>
<td>PLC_STATUS_A</td>
<td>NA</td>
</tr>
<tr>
<td>F0831144</td>
<td>R</td>
<td>PLC_STATUS_B</td>
<td>NA</td>
</tr>
<tr>
<td>F0831148</td>
<td>R</td>
<td>TPC</td>
<td>NA</td>
</tr>
<tr>
<td>F083114C</td>
<td>R</td>
<td>TNE</td>
<td>NA</td>
</tr>
<tr>
<td>F0831150</td>
<td>R</td>
<td>CLK_DIV</td>
<td>NA</td>
</tr>
<tr>
<td>F0831154</td>
<td>R</td>
<td>BIST_SIGNATURE</td>
<td>NA</td>
</tr>
<tr>
<td>F0831158</td>
<td>R</td>
<td>RCV_VECTOR</td>
<td>NA</td>
</tr>
<tr>
<td>F083115C</td>
<td>R</td>
<td>INTR_EVENT</td>
<td>NA</td>
</tr>
<tr>
<td>F0831160</td>
<td>R</td>
<td>VIOL_STM_CTR</td>
<td>NA</td>
</tr>
<tr>
<td>F0831164</td>
<td>R</td>
<td>MIN_IDLE_CTR</td>
<td>NA</td>
</tr>
<tr>
<td>F0831168</td>
<td>R</td>
<td>LINK_ERR_CTR</td>
<td>NA</td>
</tr>
</tbody>
</table>

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 bit wide (D0-D7) the byte passing logic will have to be used.
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.

<table>
<thead>
<tr>
<th>D15</th>
<th>D14</th>
<th>D13</th>
<th>D12</th>
<th>D11</th>
<th>D10</th>
<th>D9</th>
<th>D8</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
</table>

TABLE 13-6. FDDI_DMA_ADDR1

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 = xxxxxxx000000000001).

<table>
<thead>
<tr>
<th>D15</th>
<th>D14</th>
<th>D13</th>
<th>D12</th>
<th>D11</th>
<th>D10</th>
<th>D9</th>
<th>D8</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>WC11</td>
<td>WC10</td>
<td>WC9</td>
<td>WC8</td>
<td>WC7</td>
<td>WC6</td>
<td>WC5</td>
<td>WC4</td>
<td>WC3</td>
<td>WC2</td>
<td>WC1</td>
<td>WC0</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>D15</th>
<th>D14</th>
<th>D13</th>
<th>D12</th>
<th>D11</th>
<th>D10</th>
<th>D9</th>
<th>D8</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
</table>

TABLE 13-8. FDDI_DMA_ADDR2

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 = xxxxxxx000000000001).

<table>
<thead>
<tr>
<th>D15</th>
<th>D14</th>
<th>D13</th>
<th>D12</th>
<th>D11</th>
<th>D10</th>
<th>D9</th>
<th>D8</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>WC11</td>
<td>WC10</td>
<td>WC9</td>
<td>WC8</td>
<td>WC7</td>
<td>WC6</td>
<td>WC5</td>
<td>WC4</td>
<td>WC3</td>
<td>WC2</td>
<td>WC1</td>
<td>WC0</td>
</tr>
</tbody>
</table>

TABLE 13-9. 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_SEMA01, 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_SEMA01 to 01 if the DMA resource was available.

HEWLETT-PACKARD
13.3.3.1.6 GLOBAL READ DATA FOR xIC THRU x28

All reads of the DMA registers in this address range 0xF803131C through 0xF8031328 will also provide the FDDI_DMA_DEMA0 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:

- Bit 0 HSREQ0. These bits have similar definitions as the HSREQ[0-1] in the AMD FDDI Formac Plus. The HSREQ0 and HSREQ0 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:

<table>
<thead>
<tr>
<th>CHSREQ</th>
<th>HSREQ</th>
<th>HSREQ</th>
<th>HSREQ</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>None</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Read Request: Receive Queue</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Write Request: Sync. Queue</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Write Request: Async. Queue</td>
</tr>
</tbody>
</table>

* Not Supported.

- Bit 2 CSUM_ACCUM. This bit when set will cause the Shortstop IC to accumulate a 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.

<table>
<thead>
<tr>
<th>FDDI_DMA_SEMA0</th>
<th>FDDI_DMA_SEMA1</th>
<th>Current Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>None</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Transmit to FDDI in progress</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Receive from FDDI in progress</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>None</td>
</tr>
</tbody>
</table>

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_CMP. This bit will be set (1) when both DMA word count (FDDI_DMA_WCL/FDDI_DMA_WCR) registers equal 0.

- Bit 1 TAG_SEEN. On a receive from the network the AMD Formac Plus will assert a signal BDAG 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 MAC_INTR. This bit is generated when the AMD Formac Plus nINT1 signal becomes active. This bit will have positive logic associated with it (ie. when the nINT1 asserts this bit will go to 1, and will reset to 0).

- Bit 1 MAC_INTR. This bit is generated when the AMD Formac Plus nINT2 signal becomes active. Again, set to 1 when nINT2 is active and 0 when nINT2 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_CMP (later) and the RDATA interrupt bits set it will know that a packet has partially been received.

- Bit 4 DMA_CMP. This will become set when the DMA word count registers (FDDI_DMA_WCL/FDDI_DMA_WCR) 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.

HEWLETT-PACKARD
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 its 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.

<table>
<thead>
<tr>
<th>Address (HEX)</th>
<th>R/W</th>
<th>Word 0</th>
<th>Word 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>F0EB1400</td>
<td>R</td>
<td>REC_DATA,CSUM_LW</td>
<td>REC_DATA,CSUM_HW</td>
</tr>
</tbody>
</table>

**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_WCI to the desired count.
  - If the count is greater than 4K page the second FDDI DMA will have to be programmed.
  - Program the FDDI_DMA_CNTL to CHSREQ, 1L, CSUM, ACCUM = 0, CLR_SECOND_DMA_REGS to 1 if < 4K and 0 if > 4K, and keep the FDDI_DMA_SEMA1 (=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.

Version 1.1 HARDBALL I/O ERS Page 80

- 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 has completed the driver must release the FDDI_DMA_SEMA by writing 00 to these bits in the FDDI_DMA_CNTL 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_WCI 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 is 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_SEMA1) 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 # 0 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 programmed as well as reasserting FDDI_REC_DMA_RDY (write 0), keep the FDDI_DMA_SEMA, set CSUM, ACCUM, and program CHSREQ, 1L 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.

HEWLETT-PACKARD
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 MULTICAST_EN. The multicast address ranges are:

<table>
<thead>
<tr>
<th>Multi-cast Address</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>?? ?? ?? ?? ?? ?? XX</td>
<td>SMT multi-cast address block</td>
</tr>
<tr>
<td>00 00 00 00 00 0X</td>
<td>HP proprietary multi-cast address</td>
</tr>
<tr>
<td>00 00 00 00 00 0X</td>
<td>HP proprietary multi-cast address</td>
</tr>
<tr>
<td>?? ?? ?? ?? ?? ?? ??</td>
<td>(unknown)</td>
</tr>
</tbody>
</table>

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 check sums
   - 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 data 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

HEWLETT-PACKARD

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 means 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 Darystone MIPS of the various implementations, let's express the overhead per packet in "Darystone 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.

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

\[
8200 \text{ DI/pkt} \times 1 \text{ us} / 57 \text{ DI} = 144 \text{ us/pkt (720)}
\]

We can derive similar numbers for a 730 or a PCXT-T (assuming a MHz to MIPS conversion for PCXT of 1.5):

\[
8200 \text{ DI/pkt} \times 1 \text{ us} / 76 \text{ DI} = 108 \text{ us/pkt (730)}
\]

\[
8200 \text{ DI/pkt} \times 1 \text{ us} / 92 \text{ DI} = 89 \text{ us/pkt (PCXT)}
\]

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

\[
87 \text{ cycles/32 bytes x 4096 bytes/pkt x 1 us/50 cycle} = 223 \text{ us/pkt (720)}
\]

Assuming the same cache miss penalty on the 730:

\[
87 \text{ cycle/32 bytes x 4096 bytes/pkt x 1 us/66 cycle} = 169 \text{ us/pkt (730)}
\]

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

\[
111 \text{ cycles/32 bytes x 4096 bytes/pkt x 1 us/80 cycle} = 178 \text{ us/pkt (PCXT)}
\]

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

Again, let us 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:

\[
87 \text{ cycle/32 bytes x 4096 bytes/2494 instr} = 4.4 \text{ CPl}
\]

DMA interference

An FDDI transmit corresponds to an SOC "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.

Hewlett-Packard

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:

\[
30 \text{ clks/64 bytes x us/25 clks x 4160 bytes/pkt} = 78 \text{ us/pkt (720)}
\]

\[
30 \text{ clks/64 bytes x us/33 clks x 4160 bytes/pkt} = 59 \text{ us/pkt (730)}
\]

\[
30 \text{ clks/64 bytes x us/26.67 clks x 4160 bytes/pkt} = 73 \text{ us/pkt (PCXT)}
\]

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:

\[
\text{pkt/(144 + 223)us x 4096 bytes/pkt x MB/(1024)2 bytes x us/1E-6 sec} = 10.6 \text{ MB/sec (720)}
\]

\[
\text{pkt/(108 + 169)us x 4096 bytes/pkt x MB/(1024)2 bytes x us/1E-6 sec} = 14.1 \text{ MB/sec (730)}
\]

\[
\text{pkt/(89 + 178)us x 4096 bytes/pkt x MB/(1024)2 bytes x us/1E-6 sec} = 14.6 \text{ MB/sec (PCXT)}
\]

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

\[
\text{pkt/(144 + 223 + 78)us x 4096 bytes/pkt x MB/(1024)2 bytes x us/1E-6 sec} = 8.8 \text{ MB/sec (720)}
\]

\[
\text{pkt/(108 + 169 + 59)us x 4096 bytes/pkt x MB/(1024)2 bytes x us/1E-6 sec} = 11.6 \text{ MB/sec (730)}
\]

\[
\text{pkt/(89 + 178 + 73)us x 4096 bytes/pkt x MB/(1024)2 bytes x us/1E-6 sec} = 11.5 \text{ MB/sec (PCXT)}
\]

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 PCXT-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.

Hewlett-Packard
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 line 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 line rate using a processor with one-third the Därystone MIPS, clearly Cobra FDDI should have no trouble receiving SNAP packets at full line 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 IR71. 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.

HEWLETT-PACKARD

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:

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDDI_DMA_ADDR_1</td>
<td>30</td>
</tr>
<tr>
<td>FDDI_DMA_ADDR_2</td>
<td>30</td>
</tr>
<tr>
<td>FDDI_DMA_CONTROL</td>
<td>3</td>
</tr>
<tr>
<td>FDDI_DMA_SEMAPHORE</td>
<td>1</td>
</tr>
<tr>
<td>FDDI_DMA_STATUS</td>
<td>1</td>
</tr>
</tbody>
</table>

HEWLETT-PACKARD
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.

<table>
<thead>
<tr>
<th>Address</th>
<th>Size</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xF0830000 byte W Rapid harmless DMA address byte 0 (high)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xF0830004 byte W Rapid harmless DMA address byte 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xF0830008 byte W Rapid harmless DMA address byte 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xF083000C byte W Rapid harmless DMA address byte 3 (low)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xF0830010 byte W Rapid harmless DMA execute - any value will do</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xF0830010 byte R/W Rapid harmless DMA done</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>not done</td>
<td>done</td>
</tr>
<tr>
<td>Write</td>
<td>reset to 0</td>
<td>no effect</td>
</tr>
</tbody>
</table>

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 (0xF0830020). 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 PCXaa0105:

To implement the Viper fix:

1. Set RH_DMA_ADDR[23]
2. Set the following bits in Viper's MEMORY_CONTROL register:
   a. refresh[0:11] = 0xFF7
   b. SCOL = 0b1
   c. PCWAIT = 0b1
   d. RCDELAY = 0b1
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 ReadDMA 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 (0xF0RIPDM0) 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 occurred while SCSI was the bus master.
Bit 1: equal to 1 if an error occurred while LAN was the bus master.
Bit 2: equal to 1 if an error occurred while DMA of parallel port device was the bus master.
Bit 3: equal to 1 if an error occurred while Fast/Wide SCSI was the bus master.
Bit 4: equal to 1 if an error occurred while FDDI was the bus master.
Bit 5: equal to 1 if an error occurred while AUDIO was the bus master.
Bit 6: equal to 1 if an error occurred while Rapid/HighSpeed 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 "Onsite Verification Plan".

17. Performance Measurement Page

In order to facilitate external hardware performance measurement devices, a so-op page (0x00000000) 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 READY 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
18. APPENDIX A

18.1 Memory Space

0x0000 0000 - ROM Space (byte assembly; word access capable) 36 bytes
0x001F FFFF
0x0040 0000 - Unassigned 36 bytes
0x007F FFFF
0x0080 0000 - Interrupt/Status/LED 64K bytes
0x0080 FFFF
    0x0080 0000 r Interrupt Request Register (Read Only)
    0x0080 0004 rw Interrupt Mask Register (Read/Write)
    0x0080 0008 rw Interrupt Pending Register (Read Only)
    0x0080 0020 w Front Panel Status LED control Register
    0x0080 0024 r I/O Subsystem Status Register

0x0081 0000 - EEPROM (byte access only) 64K bytes
0x0081 FFFF
    0x0081 0000 rw EEPROM reserved for Cobra/Coral configuration
    0x0081 03FF
    0x0081 0400 rw EEPROM reserved for EISA configuration
    0x0081 0FFF
    0x0081 1000 rw EEPROM reserved for firmware usage
    0x0081 10FF
    0x0081 1100 rw EEPROM reserved for diagnostic usage
    0x0081 11FF
    0x0081 1200 rw EEPROM reserved for EISA configuration
    0x0081 14FF
    0x0081 2000 Currently not used (exceeds 8K bytes of EEPROM)
    0x0081 2FFF
    0x0081 3000 rw EEPROM Enable bit access (MS bit 7)

HEWLETT-PACKARD
0x0F02 0000 - DMA Register

0x0F02 0000 rw DMA ch-0 Base & Current Address register
0x0F02 0001 rw DMA ch-0 Low page register
0x0F02 0002 rw DMA ch-0 High Page register
0x0F02 0003 rw DMA ch-0 Base & Current Count register
0x0F02 0004 rw DMA ch-0 High Base & Current Count
0x0F02 0005 r DMA (0-3) Status register
0x0F02 0006 r DMA (0-3) Mode register
0x0F02 0007 r DMA (0-3) Clear byte pointer
0x0F02 0008 r DMA (0-3) Mask register
0x0F02 0009 rw DMA (0-3) Master Clear
0x0F02 000A rw DMA (0-3) Clear Mask register
0x0F02 000B rw DMA (0-3) Mask Register
0x0F02 0010 rw DMA I/O Limit Register

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

0x0F02 0024 rw DMA (0-3) Chaining Mode register
0x0F02 0025 w DMA (0-3) Extended Mode register
0x0F02 0026 w DMA (0-3) Command register
0x0F02 0027 w DMA (0-3) Request register
0x0F02 0028 r DMA Chaining Mode Status register
0x0F02 0029 r DMA (0-3) Stop register bits <7:0>
0x0F02 002A r DMA (0-3) Stop register bits <5:0>

0x0F02 1000 - 8042 (controls the /RTC/HP-HIL/Sound) 4K bytes

0x0F02 1FFF

0x0F02 1000 w 8042 Hold Reset (also holds serial #3 reset)
0x0F02 1001 w 8042 data 1/0 (for the RTC/HP-HIL/Sound)
0x0F02 1002 w 8042 status/control (for the RTC/HP-HIL/Sound)
0x0F02 1003 w 8042 release Reset (also releases serial #3 reset)

0x0F02 2000 - RS232 #2

0x0F02 2000 w RS232 Master Reset (both channels: #1 & #2)
0x0F02 2001 r RS232 #2 Access
(0x0F02 2804 bit 2 w RS232 #2 RTS flow control Enable bit custom implementation; all other bits are as defined in the data sheets)

0x0F02 3000 - RS232 #1

0x0F02 3FFF

HEWLETT-PACKARD

Version 1.1 HARDBALL I/O ERS Page 96

0x0F02 4000 - Parallel Printer Interface

0x0F02 4000 w Parallel Printer Interface Reset
0x0F02 4001 r Read Status
0x0F02 4002 rw Device Control
(Not used)
0x0F02 4003
0x0F02 4004 wt Mode Control
0x0F02 4005 wt IE Control/Interrupt Status
0x0F02 4006 wt Timing Delay Counter

0x0F02 5000 - SCSI

0x0F02 5FFF

0x0F02 5000 w SCSI Reset
0x0F02 5001 w SCSI Chip Registers

0x0F02 6000 - LAN

0x0F02 6000 w LAN Reset
0x0F02 6004 w LAN Port Select
0x0F02 6008 w LAN Channel Attention

0x0F02 7000 - DMA

0x0F02 7FFF

0x0F02 7000 w DMA Reset

0x0F02 8000 - Not supported

0x0F02 9FFF

0x0F02 F000 - Cutoff

0x0F02 F000 w I/O Subsystem Reset
0x0F02 F010 r Cutoff version control byte
0x0F02 F030 w SCSI, "DEV1" byte (0b00000001 after reset)
0x0F02 F040 r Error logging byte
0x0F02 F050 w LAN control output enables.

HEWLETT-PACKARD
0xF083 0000 - Fast write on SCSI
0xF083 0000 w Fast write SCSI reset
0xF083 0004 w Number of transfers (NT)
0xF083 0008 Reserved
0xF083 00FF Reserved
0xF083 0100 NCR53C720 registers
0xF083 015C Reserved
0xF083 0160 Reserved
0xF083 0FFF
0xF083 1000 - FDI
0xF083 1000 4K bytes
WRITE READ
0xF083 1000 OMREGI - STIU MAC(AMO) begin
0xF083 1004 OMREGI - STIL (data is 16 bits wide)
0xF083 1008 OMREGI - STIL
0xF083 100C OMREGI - IMUX
0xF083 1010 IMUX - IMUX
0xF083 1014 IMUX - IMUX
0xF083 1018 IMUX - IMUX
0xF083 101C IMUX - IMUX
0xF083 110C IMUX - IMUX MAC (end)
0xF083 1200 PLC_CTRL_A - PLC_CTRL_A PHY/(AMO) (begin)
0xF083 1204 PLC_CTRL_B - PLC_CTRL_B (data is 16 bits wide)
0xF083 1268 - LINK_ERR_CTRL PHY (end)
0xF083 1300 rw FDI_DMA_ADDR_1
0xF083 1304 rw FDI_DMA_ADDR_2
0xF083 1308 rw FDI_DMA_ADDR_3
0xF083 1310 rw FDI_DMA_ADDR_4
0xF083 1320 r FDI_DMA_REG
0xF083 1324 rw FDI_DMA_CNTL
0xF083 1328 rw FDI_DMA_STATUS
0xF083 1330 rw FDI_INTR
0xF083 1334 rw FDI_INTR_MAG
0xF083 1FFF
0xF083 2000 - No-op (performance measuremen) 4K bytes
0xF083 2FFF
0xF083 3000 - Exa00105 Viper reset bug fix 4K bytes
HEWLETT-PACKARD