digital

DIGITAL Network Appliance Reference Design

User's Guide

Order Number: EC-R8JLB-TE

Revision/Update Information: This is a revised document. It supersedes the *DIGITAL Network Appliance Reference Design User's Guide*, EC-R8JLA-TE.

Digital Equipment Corporation
Maynard, Massachusetts

http://www.digital.com/info/semiconductor

November 1997

Disclaimer: This board and platform is intended for the evaluation of the StrongARM CPU and is not intended to be included as part of any final product. No regulatory approvals as might apply to any finished product have been obtained or are implied by the availability of this board.

Contents

Preface

1 Introduction

2 FreeBSD Installation and Setup

3 Configuring an NC Bootserver

4 Hardware Functional Description

A Support, Products, and Documentation

Figures

Tables

Preface

Purpose

This manual is the user's guide for the DIGITAL Network Appliance Reference Design (DNARD).

Conventions

This section defines product-specific terminology, abbreviations, and other conventions used throughout this manual.

Abbreviations

- Binary Multiples

  The abbreviations K, M, and G (kilo, mega, and giga) represent binary multiples and have the following values.

  K = 2^{10} (1024)
  M = 2^{20} (1,048,576)
  G = 2^{30} (1,073,741,824)

  For example:

2KB	=	2 kilobytes	=	2×2^{10} bytes
4MB	=	4 megabytes	=	4×2^{20} bytes
8GB	=	8 gigabytes	=	8×2^{30} bytes
2K pixels	=	2 kilopixels	=	2×2^{10} pixels
4M pixels	=	4 megapixels	=	4×2^{20} pixels

- Register Access

  The abbreviations used to indicate the type of access to register fields and bits have the following definitions:

  MBZ — Must Be Zero

  Software must never place a nonzero value in bits and fields specified as MBZ. Reads return UNPREDICTABLE values. Such fields are reserved for future use.

  RES — Reserved

  Bits and fields specified as RES are reserved by DIGITAL Semiconductor and should not be used; however, zeros can be written to reserved fields that cannot be masked.

  RO — Read Only

  Bits and fields specified as RO can be read and are ignored (not written) on writes.

  RW — Read/Write

  Bits and fields specified as RW can be read and written.

  R/W1C — Read/Write One to Clear

  Bits and fields specified as R/W1C can be read. Writing a one clears these bits for the duration of the write; writing a zero has no effect.

  WO — Write Only

  Bits and fields specified as WO can be written but not read.

Addresses

Unless otherwise noted, all addresses and offsets are hexadecimal.

Aligned and Unaligned

The terms *aligned* and *naturally aligned* are interchangeable and refer to data objects that are powers of two in size. An aligned datum of size 2*n* is stored in memory at a byte address that is a multiple of 2*n*; that is, one that has *n* low-order zeros. For example, an aligned 64-byte stack frame has a memory address that is a multiple of 64.

A datum of size 2*n* is *unaligned* if it is stored in a byte address that is not a multiple of 2*n*.

Bit Notation

Multiple-bit fields can include contiguous and noncontiguous bits contained in brackets ([]). Multiple contiguous bits are indicated by a pair of numbers separated by a colon (:). For example, [9:7,5,2:0] specifies bits 9, 8, 7, 6, 5, 2, 1, and 0. Similarly, single bits are frequently indicated with angle brackets. For example, [27] specifies bit 27.

Caution

Cautions indicate potential damage to equipment or loss of data.

Data Units

The following data unit terminology is used throughout this manual.

Term	Words	Bytes	Bits	Other
Byte	—	1	8	—
Word	1	2	16	—
Dword	2	4	32	Longword
Quadword	4	8	64	2 Dwords

External

Unless otherwise stated, external means not contained in the 21xxx.

Note

Notes emphasize particularly important information.

Numbering

All numbers are decimal or hexadecimal unless otherwise indicated. The prefix 0x indicates a hexadecimal number. For example, 19 is decimal, but 0x19 and 0x19A are hexadecimal (also see Addresses). Otherwise, the base is indicated by a subscript; for example, 100_2 is a binary number.

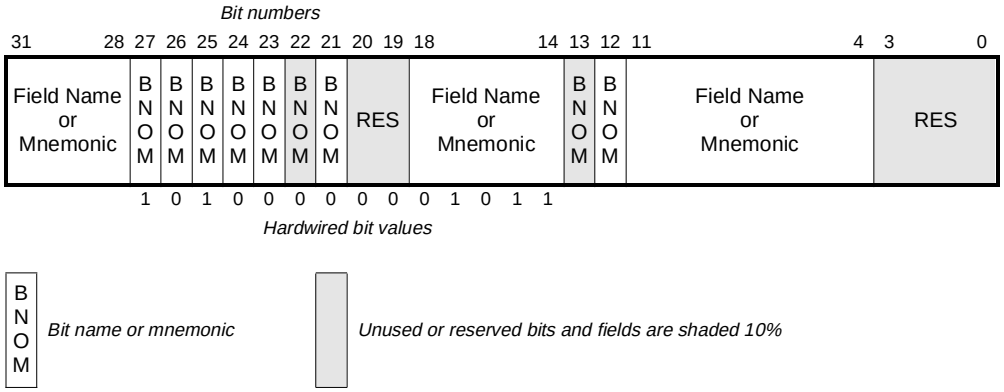
Ranges and Extents

Ranges are specified by a pair of numbers separated by two periods (..) and are inclusive. For example, a range of integers 0..4 includes the integers 0, 1, 2, 3, and 4.

Extents are specified by a pair of numbers in brackets ([]) separated by a colon (:) and are inclusive. Bit fields are often specified as extents. For example, bits [7:3] specifies bits 7, 6, 5, 4, and 3.

Register Figures

The following figure defines the conventions used in register format figures:



Signal Names

Signal names are printed in lowercase, boldface type. Low-asserted signals are indicated by the number sign (#) suffix. For example, **pll_clk_in** is a high-asserted signal, and **pll_clk_in#** is a low-asserted signal.

1
Introduction

The DIGITAL Network Appliance Reference Design (DNARD) from DIGITAL Semiconductor takes full advantage of the 233-MHz StrongARM CPU to achieve new heights in Network Computer (NC) performance with unmatched speed and quick user-interface response. This capability, combined with the economies of scale associated with the DNARD's use of industry-standard chips, offers the most cost-effective NC solution on the market today.

NC manufacturers are taking advantage of the DNARD and are leveraging this complete reference design to quickly create leadership products. By coupling the DIGITAL Semiconductor StrongARM processors with industry-standard components and buses, the DNARD provides a unique combination of power, low cost, and flexibility, and is adaptable to a wide range of NC and information appliance products.

There is no charge for the DNARD license. Design information is available on the World Wide Web at: http://www.research.digital.com/SRC/iag. The design information includes schematics, a circuit board layout, the complete mechanical design for the enclosure, and a complete port of the NetBSD operating system.

Important: The DNARD board is intended for the evaluation and development of NC solutions and is not intended to be included as part of any final product. No regulatory approvals as might apply to any finished product have been obtained or are implied for this board.

1.1 DNARD Overview

The DNARD package includes:

- Hardware:
  - DNARD printed circuit board. This single-sided board employs only two signal layers, optimizing economy of manufacture.
  - Packaging – a complete industrial design that can be used as-is in both vertical and horizontal orientations.
  - Cables
- Client software (executables):
  - OS: NetBSD, XFree86, and a simple window manager that provide diskless client operating system support during application development with easy subsequent porting to NC Desktop for final products
  - Java: Java Virtual Machine (JVM) and Just-In-Time (JIT) Java compiler
  - GNU tools and demonstration applications
- Server software (executables):
  - Boot services (DHCP, TFTP, NFS)
  - FreeBSD, which can be used on generic PCs to supply the client host services.
- Technical documentation

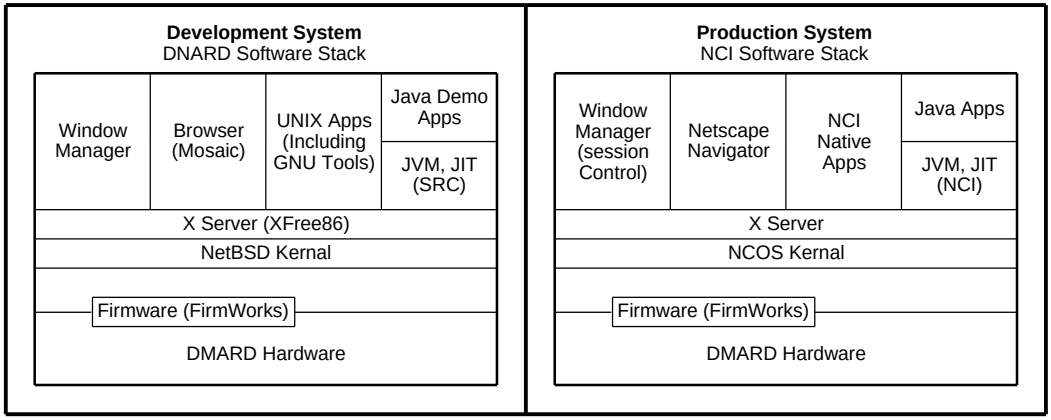
The DIGITAL Network Appliance Reference Design can be used directly to minimize time-to-market. The DNARD can also be scaled down gracefully, by deleting selected components, to satisfy a manufacturer's specific market requirements. In addition, the design's use of industry-standard buses makes it easy to enhance the design additing or substituting key components.

1.2 DNARD Software Strategy

The DNARD uses a two-pronged software approach: a DNARD software stack is provided without charge to enable early evaluation, testing, and development work; then finished products, based on DNARD, can use the software stack available from Network Computer, Inc.(NCI).

Figure 1–1 compares these two software stacks.

Figure 1–1 Comparing Software Stacks: Development vs. Shipping Products



FM-06233.AI4

This two-pronged approach has several advantages:

- It provides the lowest barrier to entry because the DNARD software stack is free. The DNARD software stack also provides a native development environment. Developers are not required to wait for NCI to port their stack to a particular system configuration before development can begin.
- The elements used in the DNARD software stack are extremely similar to those provided in the NCI stack – therefore, a subsequent port of the development stack by NCI is made relatively easy.
- The NCI software stack (unlike the DNARD stack, which utilizes several free software components such as NetBSD) is fully supported and documented by NCI and thus is a solid basis for shipping products.

NC Desktop from the Oracle subsidiary Network Computer, Inc. is the recommended software for DNARD-based products. To help manufacturers create customized products DIGITAL provides, at no cost, a complete source-code port of the NetBSD operating system to the DNARD platform. Using this code, manufacturers can create and demonstrate customized DIGITAL Network Appliance Reference Design-based prototypes in record time. This prototype software then expedites porting of the NC Desktop to the final product.

Note: The approach outlined above does not preclude porting of other software stacks by strongly motivated developers.

1.3 DNARD Hardware Strategy

The DNARD approach takes advantage of the high performance, low power, and low cost provided by the StrongARM CPU and combines it with industry-standard components and interfaces so that developers can quickly and cost effectively bring to market configurations of DNARD that are customized for their application.

Table 1–1 summarizes the technical specifications of the DIGITAL Network Appliance Reference Design.

Table 1–1 DNARD Technical Specification Summary *(Sheet 1 of 2)*

Processor	DIGITAL Semiconductor StrongARM SA-110, 233 MHz
Memory	4-64MB of SDRAM in two industry-standard DIMMs (66 MHz memory bus)
	Up to 64MB of replaceable ROMcard (masked, EEPROM, or flash)
	128-512KB of on-board flash boot ROM
Firmware	OpenFirmware (IEEE Standard 1275) from FirmWorks
OS software Prototype Product:	
	NetBSD 1.2, in free source distribution
	NCOS 2.0, licensed from Network Computer, Inc.
Buses	VLB, ISA and (optional) PCI
Storage	100MB Iomega Zip drive (optional)
Network	10BASE-T Ethernet
	33.6-Kb/s soft modem (optional)
I/O options	By way of assembly-time “UMI” daughtercards
Video	VGA (24-bit), SVGA (16-bit), and XGA (16-bit) resolution
	75 Hz refresh
	2MB video RAM
	VGA, S-video, or composite output
	LCD panel connector (optional)

Table 1–1 DNARD Technical Specification Summary *(Sheet 2 of 2)*

Audio	8 or 16 bit, 44 KHz input (monaural) and output (stereo)
	FM synthesis
	SoundBlaster and SoundBlaster-Pro emulation
	Microphone input, line-level, and headphone outputs
Keyboard	PS-2 keyboard plus mouse ports
Smartcard	ISO standard, built into front panel
I/O ports	Industry-standard serial port, parallel port, and game port
Remote	Industry-standard (“consumer”) IR remote control
Power	External 5-V power supply
Form factor	9.25 x 11.35 x 2.0 in (231 x 284 x 50 mm)

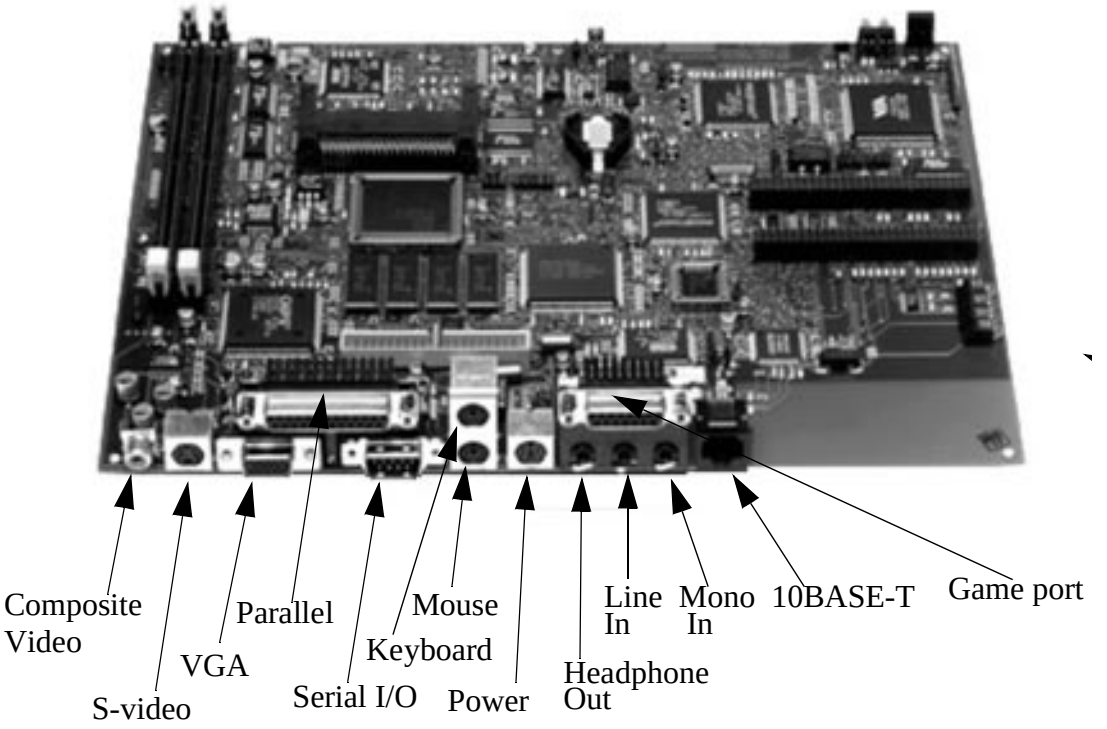
1.4 Physical Packaging and Connectors

The DIGITAL Network Appliance Reference Design (DNARD) package is designed to allow partners the option of easily customizing the appearance of their product by retooling the snap-on plastic bezel and/or changing the color. A complete redesign of the package is also an option for partners prepared to handle the necessary mechanical engineering work, FCC certification, and other aspects of this task.

The physical package included in the DIGITAL Network Appliance Reference Design is optimized for both flexibility and economy. This package supports the overall flexible, low-cost approach. The industrial design is tailored for both vertical and horizontal orientation. Two variations of the front bezel provide the flexibility to support units with or without the optional Iomega Zip drive. Also, NC manufacturers can emphasize individual brand identity by redesigning this snap-on bezel.

The DNARD provides a full complement of electrical interfaces to allow connections to all anticipated network interfaces and devices. Figure 1–2 shows the location of each of the DNARD connectors.

Figure 1–2 DNARD Connectors



- Graphic connectors – three industry-standard interfaces
  - Composite video
  - S-video
  - VGA
- Parallel port – standard ECP/EPP 1284 parallel port
- Serial I/O – standard RS-232 serial port
- Keyboard – standard PS-2 interface and mini-DIN connector
- Mouse – mini-DIN connector

- Power – no power on/off switch. When power adapter is connected, DNARD is powered on
- Headphone out – with standard headphone connector
- Line out
- Mono In – audio monaural input with standard microphone connector
- Game port – standard 15-pin PC game port connector for joysticks, and so on
- 10BASE-T – standard Ethernet LAN interface and connector

2
FreeBSD Installation and Setup

This chapter describes how to install and set up FreeBSD on a PC so that it can be used as an NC server. The procedure described here does not attempt to to cover all configurations or possibilities. For a comprehensive description of FreeBSD installation, refer to the FreeBSD website (http://www.freebsd.org/).

Note: The procedure described here makes no attempt to preserve any other operating system that may be on your PC. If you require that kind of installation, read and carefully follow all documentation provided at the FreeBSD website. The following installation will DESTROY any data currently on the disk.

The steps to install and run FreeBSD on your PC follows:

1. Prepare a special FreeBSD installation program bootable floppy.
2. Boot your PC from the newly created floppy.
3. Configure the FreeBSD kernel to correspond to the hardware in your PC.
4. Install the FreeBSD system on your PC.
5. Boot your system into the FreeBSD OS.
6. Modify the FreeBSD kernel so that the system can operate as a DHCP server.

2.1 Preparing for FreeBSD Installation

This section describes the steps that must be completed before installing FreeBSD.

1. Obtain a FreeBSD (version 2.2.2) distribution. The software can be installed from a variety of media including CD-ROM, floppy disk, magnetic tape, an MS-DOS partition, or if you have a network connection (even a dial-up PPP to an Internet provider), you can install it directly over anonymous FTP or NFS. All you need is a single 1.44MB boot floppy disk.

2. Familiarize yourself with the supported hardware configurations (see http://www.freebsd.org/handbook/handbook8.html).

   FreeBSD runs on a wide variety of ISA, VLB, EISA, and PCI bus-based PCs, ranging from 386sx to Pentium class machines (although the 386sx is not recommended). Support for generic IDE or ESDI drive configurations, various SCSI controller, network, and serial cards is also provided.

   A minimum of four megabytes of RAM is required to run FreeBSD. To run the X Window System, eight megabytes of RAM is the recommended minimum memory.

   The following is a list of Digital Equipment Corporation hardware on which FreeBSD has been successfully installed:

   – Digital Equipment Corporation HiNote Ultra II with a 3Com 3C589C Ethernet LinkIII PCMCIA card installed using the boot.flp from the PAO web page at www.jp.FreeBSD.org/PAO/#faq with FreeBSD Version 2.2.1.

   – Digital Equipment Corporation Prioris XL Server 5200 with a PCI DE540 network card with FreeBSD Version 2.2.1 (the Toshiba CD-ROM drive is not supported [Apr. 23, 1997])

   – Digital Equipment Corporation HiNote CT450 and HiNote CT475.

   – Digital Equipment Corporation Celebris.

3. Ensure that your PC has a network interface and enough disk space to accomodate the client file system and swap area. You will need a minimum of one gigabyte for a small demonstration system, although it is recommended that two to four gigabytes be available. Access/connectivity to a network is a requirement – you are setting up a server for a “Network Appliance”.

4. If the PC has a supported CD-ROM drive, your installation can be performed using the CD-ROM distribution from FreeBSD directly on your PC. Your other option is to perform the installation by way of the network card using FTP or NFS from another host to gain access to the distribution.

Note: FreeBSD requires that the CD-ROM drive in the system be configured in a special way–it must be set up so that it is *neither* the bus master nor bus slave or it will not recognize that the drive is connected. On some systems, this may require the removal of configuration jumpers on the CD-ROM drive.

2.2 Creating the FreeBSD Boot Floppy Disk

Before you can install FreeBSD, you must create a special floppy disk from which you can boot the PC. This is *not* a DOS floppy, but you can use a PC to create it as follows:

1. Get the files Tools/fdimage.exe and Floppies/boot.flp (or Floppies/boot-pao.flp for laptops) from the FreeBSD distribution and copy it into a temporary folder on the PC.
2. Insert a blank 1.44MB floppy into the PC.
3. Open an MS-DOS prompt window on the PC and issue the following command:

   ```
   C > fdimage boot.flp a:
   ```

   The FreeBSD bootable installer is created on the floppy disk.

2.3 Booting from the FreeBSD Installation Floppy

Shut down the PC you want to install this software on, insert the floppy you just made into drive A (the floppy drive), put the FreeBSD CD-ROM into the CD-ROM drive, and boot the PC. You will see a screen display similar to the following:

```
>> FreeBSD BOOT ...
Usage: [[[0:][wd](0,a)]/kernel][-abcCdhrsv]
Use 1:sd(0,a)kernel to boot sd0 if it is BIOS drive 1
Use ? for file list or press Enter for defaults
Boot:
```

If you do not type anything, after a delay of about five seconds, FreeBSD will automatically boot with its default configuration. As FreeBSD boots, it probes your computer and displays on the screen the hardware that is installed. When the booting process is finished, the main FreeBSD installation menu is displayed.

2.4 Configuring the FreeBSD Kernel for Your PC

When the PC has booted from the FreeBSD Installation floppy, you will be presented with the beginning screen of the FreeBSD installation titled *Kernel Configuration Menu*. Due to limitations of the PC architecture, it is impossible for probing to be completely accurate. In the event that your hardware is incorrectly identified, or that the probing causes your computer to lock up, first check the supported configurations section of the installation guide to be sure that your hardware is supported by FreeBSD.

The FreeBSD kernel configuration mode lets you supply hints about your hardware. The FreeBSD kernel on the installation disk is configured with the assumption that most hardware devices are in their factory default configuration for IRQs, I/O addresses, and DMA channels. It is also possible that a probe for a device not present will cause a later probe for another device that is present to fail. In that case, the probes for the conflicting driver(s) should be disabled.

In the configuration mode, you can:

- List the device drivers installed in the kernel.
- Disable device drivers for hardware not present in your system.
- Change the IRQ, DRQ, and I/O port addresses used by a device driver

You can use the F1 key liberally to obtain help during the the installation. The description that follows presents broad, catch-all choices for novice help but is not intended to be comprehensive.

Note: The kernel configuration is designed to tune the kernel for optimal performance with your particular PC. Technically speaking, however, as long as FreeBSD can successfully boot on the PC, your newly created server can run on a network, provide NFS and bootp service, and has sufficient disk space for client file systems. This is the minimum functionality that is needed to support the DNARD.

1. Use the arrow keys to select the option *'Start kernel configuration in Visual Mode'*.

2. The next screen will display is similar to the following:

```
Active Drivers_____________________9 conflicts__Dev__IRQ__PORT__
```

   Now you can select/deselect device drivers based on what hardware is present in the PC. In particular, you must clear any device conflicts (CONF) that may be indicated.

   Each category of devices starts out with its display "collapsed" and can be expanded to view the list by moving to it (using the up/down arrow keys) and then pressing Enter. Within each category, devices can be removed using the Delete key. Use this to remove drivers that are *not* present but conflict (as indicated by the CONF tag on the device line) with the devices that are present in the system. When finished with a category, collapse it again by moving to the category title and pressing Enter. Note that conflicts can cross categories. Do not remove devices that are present on your hardware; instead, clear conflicts by removing conflicting non-existent devices.

3. Enter into each category in the Active Drivers and delete all devices that have conflicts (indicated by "CONF").

4. Ensure that the following three devices are present:

   – In the Network category, the Ethernet Adapter card (for example, le0 DEC EtherWORKS 2 or 3 Ethernet card)

   – In the Storage category, your hard disk controller.

   – In the Storage category, the CD-ROM controller.

5. Quit (press Q).

6. Save (press S).

   There will then be a pause for SCSI devices to settle followed by a long sequence of logging messages and interspersed delays. Be patient. Finally, you will see the "Welcome to FreeBSD" menu.

2.5 Installing the FreeBSD System

After the kernel configuration has been completed the *Welcome to FreeBSD* menu is presented. Here you can specify various installation options.

1. Use the up/down arrows to select [5 Novice] and the press Enter to begin the installation setup process. Novice mode shows you complete information throughout the subsequent installation steps. Ignore the message about partitions that is displayed next and continue to the *FDISK Partition Editor* screen which displays information similar to the following:

```
Disk name: sd0
Disk Geometry: 261 cylinders/255 heads/63 sectors = 4192965
Offset  Size     End    Name   PType  Desc   Subtype Flags
     0  4194058  4194057   -       6  unused       0
```

   This display will be followed by several choices.

2. Select "A = Use entire disk." to make your hard disk (sd0) a single partition. The following message is then displayed:

```
Do you want to do this with a true partition entry so as to remain
cooperative with any future possible operation systems on the
drive(s)?
```

3. Select "YES" to make this a true partition. You will now see a display similar to the following (note that these are DOS-level partitions):

```
Offset  Size     End      Name   PType  Desc   SubtypeFlags
     0  63       62       -      6      unused    0
    63  4192902  4192964  sd0s1  3      freebsd 165   C
     0  4194058  4194057  -      6      unused    0
```

4. Select "Q = Quit" to accept the automatic configuration.

   You will now be presented with the following question:

```
Install Boot Manager for drive sd0?
```

   Use the up/down arrows and then the space bar to select "Standard Install a standard MBR (no boot manager.)" This will subsequently cause your PC to directly boot FreeBSD. Ignore the message about BSD partitions.

5. You will now get the *FreeBSD Disklabel Editor* and be presented with several choices.

   Select “A = Auto Defaults for All” to install the default directory configuration. You will see that the following directories are to be created on the hard disk: /, /swap, /var, /usr.

6. Select “Q = Quit” to exit this part of the installation preparation.

   You will now get the *Choose Distributions* screen where you will specify what files you want loaded onto your system as FreeBSD is installed, what encryption support to install, and lastly, the media from which the installation is to occur. Your first decision is to specify which distribution to install. The choices include such things as Developer, X-Developer, User, X-User, and All. For the purpose of this description, it is most convenient to select All – the only penalty is in the amount of disk space used and the amount of time needed to load all the files. An advantage is that you will have all of the source files on your system which may be useful to rebuild the kernel and to clean it up for your specific hardware.

7. Press the space bar to select “[8 All] = All sources and binaries (incl X-Window System)” to obtain a complete FreeBSD installation. You will then be presented with the following sequence of confirmation screens - reply as described below:

- Select “No” to “*Install DES*” screen
- Select “No” to “F*reeBSD Ports*” screen

8. You are now back at the Choose Distribution screen. This time just press Enter.
9. Select [1 CDROM] in the “*Installation Media*” menu to specify that FreeBSD is to be loaded from the CD-ROM drive (takes 5 to 10 mins).
10. Install FreeBSD CD-ROM in the CD-ROM drive.
11. Select “Yes” in the “*User Confirmation Requested*” screen.

    Now the installation of FreeBSD begins. It will take several minutes to complete and a long series of progress indicators will be displayed during the process.

12. Select OK at the “*Congratulation”* screen.

13. When the installation is complete, the *"Install Ethernet"* screen is displayed and you will be presented with the following sequence of screens – reply as described below:

- Select "Yes" to the "*Configure any Ethernet or SLIP/PPP network device?*" screen.
- Use the up/down arrows to select the appropriate network interface card. For example: "le0 DEC EtherWORKS 2 or 3 Ethernet card".
- Set the following Network parameters (note that the values shown below are only examples – check with your local network administrator for the approprite values for your installation.)
  - Host = bsdsrv2.pa.dec.com (or absolute IP address like 16.5.80.87). Then the "Domain = pa.dec.com or 5.80.87" will appear automatically.
  - Gateway = 16.5.80.253
  - Name Server = 16.4.0.250
  - IP = 16.5.80.87 (the same as the Host value)
  - Netmask = 255.255.255.0
- Then select "OK" and bring up the le0 interface for more configurations.

14. You will now be asked to define some additional operating and configuration information by way of the following sequence of screens. Reply as described below:

- Respond with "No" to "*Samba for NETBUI*" screen.
- Respond with "No" to "*IP Gateway*" screen.
- Respond with "No" to "*Anonymous FTP*" screen.
- Respond with "Yes" to "*NFS Server*" screen.
  - Respond with "OK" in the following message screen. The "ee" editor is invoked automatically to review the settings.
  - Press "ESC-Enter" to quit "ee" without modification.
- Respond with "No" to the "*NFS Client*" screen.
- Respond with "No" to the "*Customize Console*" screen.
- Respond with "Yes" to the "*Set Time*" screen, and then follow the on-screen instruction to set the time zone.

- Respond with “Yes” to the “*Mouse*” screen and use “space bar” to select the mouse type (typically, PS/2).
- Respond with “No” to the “*X-server at this time*” screen.
- Respond with “Yes” to the “*Browse package collection*” screen and then browse the “lang” group of packages. Select the kaffe-0.8.4 package within this group to ensure that the Java-related package needed by the NCBootServerConfig utility is installed.
- Respond with “No” to the “*Add user accounts*” screen.

15. You can now set up the root password. Respond with “Yes” to the “*Set Root Password*” screen and then set the root password to some easy-to-remember value (“root123” is suggested).
16. Respond with “No” to the “*Register Now*” screen and “OK” the registration message.
17. Respond with “No” to the “*General Configuration*” menu.
18. You are now returned to the “*FreeBSD Configuration*” menu. This time, use the left/right arrow keys to select [Exit Install] and then press Enter. The installation now proceeds to completion.
19. Once the installation has completed, you will be asked to remove the original boot floppy and the system will boot FreeBSD – a process that takes several minutes.

Now the FreeBSD installation in complete! You can now proceed to modify and rebuild the kernel to allow the system to run a DHCP server as described in the section that follows.

2.6 Enabling the System to Run a DHCP Server

The steps below modify the FreeBSD kernel to enable your system to run a DHCP server. You will need to rebuild the kernel to include the packet filter needed for the DHCP server.

1. Log in as root to FreeBSD and you will get the system prompt (#).
2. Make a working copy of the original kernel and name it MYKERNEL (must be all caps) by issuing the following commands:

```
# cd /usr/src/sys/i386/conf
# cp GENERIC MYKERNEL
```

3. Use an editor (for example, vi) to edit MYKERNEL:

```
# vi MYKERNEL
```

   – In the pseudo-device section of MYKERNEL, add a line:

```
pseudo-device bpfilter  4
```

   – In the device section of MYKERNEL, remove the word “disable” from the line:

```
device psm0 at isa? disable port"IO_KBD" conflicts tty irq 12
```

   – Comment out all device lines (by placing the “#” character at the beginning of the line) that uses the same address (for example, 0x300) as your Ethernet card, leaving only your Ethernet card (for example, le0) at that address.

Note: Search for the string “0x300” in MYKERNEL to find all those lines because they might not be in the same section.

   – Save MYKERNEL.

4. Rebuild the kernel by issuing the following sequence of commands:

```
# /usr/sbin/config MYKERNEL (about 30 sec)
# cd ../../compile/MYKERNEL
# make depend (about 2 min)
# make (about 5 min)
# make install (about 10 sec)
# reboot (about 2 min)
```

Note that the make commands can take many minutes to complete. When the kernel has been rebuilt and the system rebooted, you can configure the server to boot DNARD clients as described in Chapter 3.

3
Configuring an NC Bootserver

This chapter describes the steps used to configure FreeBSD so that it can boot a set of DNARD Network Computers.

This description assumes that you have a properly configured FreeBSD, which includes the packetfilter option in the kernel, with a working network configuration. You will need to know information about your network configuration (such as domain name, server name, IP addresses, and so on) to complete the task.

3.1 Configuring the X Window System

You will be using a utility (NCBootServerConfig) written in Java to perform the configuration. Since Java depends on the X Window System, you will first need to configure the X Window System for the mouse, keyboard, video card, and display that you are using. Therefore, you must know which mouse, keyboard, video card, and display you have in your system in order to respond to complete the configuration. The XF86Setup program handles the X Window System configuration, and it will prompt you for the information. Issue the following command:

```
XF86Setup
```

Note: It is important that you not touch the mouse until it has been configured. The mouse can be difficult to configure and some information displayed by the XF86Setup program (such as the function of some keys) is inaccurate. You should rely on the description provided in this document rather than the on-screen information displayed by XF86Setup. For example, the program indicates that pressing the A key will apply changes: this is not always true – sometimes pressing the A key causes an abort and subsequent hanging of the system.

1. After you start the XF86Setup program, press the Enter key to start configuring the mouse.
2. A help screen appears: press Enter to dismiss the help screen and proceed to the mouse configuration screen.
3. Now press Alt/P to move the cursor to the correct protocol (for example, PS/2).
4. When the correct protocol is highlighted, press Enter. Now you need to move the cursor to the Apply box by repeatedly pressing the Tab key.
5. When the cursor is on the Apply box, press Enter. You should be able to move the mouse and see the cursor move appropriately
6. When the mouse has been configured, you can then use it for navigation and selection. Proceed to set the rest of the configuration for the keyboard, video card, and display that you are using.
7. Approve the setup.
8. Select Yes (to create x link to the server.)

After you have set the configuration information required by X86, quit the XF86Setup program.

3.2 Setting Up the Root File System

Install the DNARD CD-ROM.

1. The NCBootServerConfig utility sets up a single root file system (shared by all NCs) and a set of swap files (one per NC). You need to create a directory under /usr to hold these files, for example: /usr/dnard. Create the directory now by issuing the following command:

   ```
   mkdir /usr/dnard
   ```

2. The NC root file system is provided as a set of compressed tar files. You can download these files from the web, copy them from another server, or get them on a CD-ROM. The configuration utility will decompress the files for you. It assumes that the tar files are in the /tar directory. Create the directory now, with mkdir, and copy or download the compressed tar files to the directory. The example below assumes that the files are copied from a CD-ROM.

3. Mount the CD-ROM drive. Use the vi editor to look at the file /etc/fstab to determine the location of the CD-ROM. Use these commands:

   ```
   cd /etc
   vi fstab
   ```

4. The fstab file will look like this:

   ```
   //dev/wd0a              /ufs          rw 1 1  root file system

   //dev/wd0ac             /cdrom  cd9660  ro 0 0  CD rom
   ```

5. Write down the CD-ROM drive information (from the above example, /dev/wd0ac cd9660)

6. Quit the vi editor.

7. Mount the CD-ROM drive by issuing the following commands:

   ```
   mkdir  /cd1
   mount -t cd9660 -o ro /dev/wd0ac /cd1
   ```

   The CD-ROM is mounted. No errors should appear.

8. Issue the following command to create the directory for the tar files:

   ```
   mkdir /usr/tar
   ```

9. Issue the following command to link the directories.

   ```
   ln -s /usr/tar /tar
   ```

10. List the files on the CD-ROM.

```
ls /cd1
```

11. You will see five files that you need to copy from the CD-ROM to the hard drive. The files must be copied and renamed exactly as shown here. Issue the following commands to copy the files from the CD-ROM:

```
cd /tar
cp /cd1/rootta~1.gz root.tar.gz
cp /cd1/usrta~1.gz  usr.tar.gz
cp /cd1/x11tar~1.gz X11.tar.gz
cp /cd1/config~1.gz config.tar.gz
cp /cd1/dhcpta~1.gz dhcp-freebsd.tar.gz
```

12. The configuration utility cannot decompress itself, so you will need to decompress it using tar. Issue the following commands:

```
cd /usr/dnard
tar xpzf /tar/config-util.tar.gz
```

13. Java is not included on the FreeBSD CD-ROM. A Java implementation is included in the config–util tar file, along with the installJava shell script used to install it. Install Java now by issuing the following command:

```
./installJava
```

    A licensing agreement from Sun Microsystems will be displayed when the installation completes. Respond as appropriate.

14. Next, run the X Window System by issuing the following command:

```
startx
```

3.3 Running the NCBootServerConfig Utility

The NCBootServerConfig utility is written in Java and provides a relatively easy way to specify various parameters for your installation. The utility also automates a variety of file editing tasks to eliminate the need for using an editor to manually modify certain system files.

1. Run the NCBootServerConfig utility by issuing the doConfig shell script. The doConfig script is included in the config-util.tar.gz tarfile, so that when you unpacked it, this shell script was placed in your /usr/dnard (or whatever you named it) directory.

   ```
   cd /usr/dnard
   ./doConfig
   ```

2. The utility brings up an X Window where you can enter the rest of the information for your site and installation. (Note: when this initial X Window for the utility is first displayed, you must position it to your desired position on the screen and then click the mouse to activate it.)

3. You can now enter the following information for your site and installation in the appropriate fields displayed on the screen:

   <domainName>
   <serverName>
   <server>
   <subnet> <subnetMask>
   <gateway>
   <firstNC>
   <numberNCs>

   a. Note that the <subnet> and the <subnetMask> must be separated by a space in the field labelled "Subnet and Mask". For example, here is a valid specification of <subnet> and <subnetMask>:

      ```
      16.4.208.0 255.255.255.0
      ```

      Thus, the example subnet value is 16.4.208.0 and the subnetMask value is 255.255.255.0 and a space character is entered to separate the two values.

   b. The <firstNC> and <numberNCs> fields are used together to assign a range of IP addresses for the DNARD units. Thus you specify the beginning IP address (<firstNC>) and how many DNARD units are to be connected (<numberNCs>) and the utility calculates and supplies the IP addresses for the range of supported units.

4. The check box labelled “Configure and install dhcp?" allows you to bypass installation of dhcp. The default is to install it. If you already have DHCP running on this host, you can uncheck the box and a template configuration file will be created that you can use to update your existing DHCP configuration.

Note: The check boxes provided by this utility program are displayed as colored or gray-out when they are “true” or checked.

5. The check box labelled “Update /etc/resolv.conf?” has no effect unless you change the value of the <domain> or <nameServer> fields. You need to do this if the NCs must use a different domain name or name server than the FreeBSD server, or if you entered the wrong information when you configured the FreeBSD server. If you check this box, the information you supply here is used to update both the NCs and the FreeBSD server. If you do not check this box, information entered in the <domain> and <nameServer> fields are used only for the NCs and not for the FreeBSD server.
6. Use the Configure button to apply the configuration information you have just specified and initiate unpacking of all required files. The utility will catch many inadvertent errors (such as invalid address values, and so on). If it detects a problem, a message about the problem will appear in the bottom window. After an error, you can correct the appropriate field, and press the Configure button to try again.

Note: Unpacking the tar files will take quite a long time. Be patient.

7. After all the files have been unpacked, you may want to close the X Window. If so, proceed as follows (if otherwise, issue the reboot command now).
    a. Click the mouse on the desktop outside the window.
    b. Hold down the mouse button and select “Kill” from the pop-up menu displayed.
    c. A skull and crossbones cursor is displayed. Move it to the window that should be killed and click the mouse.

Your server is now configured to boot NCs. You should now issue the reboot command. Then, power up a DNARD unit. It should come up at the FreeBSD login prompt.

4
Hardware Functional Description

This chapter provides an overview of the DNARD hardware and a description of how each of the major functional units on the board is configured and used. Up-to-date descriptions of circuit level details are always available at:

www.research.digital.com:80/SRC/iag/

4.1 DNARD Hardware Overview

The core subsystem of DNARD is comprised of a very high-performance high-speed CPU/memory system that also attempts to take advantage of optimal price/performance component trade-offs to deliver maximum value to the final end-user system. The StrongARM processor and its high-speed memory subsystem use 3.3-V power and run at the maximum SA-110 bus clock frequency of 66-MHz. A PAL is used to create a 33-MHz VL-Bus (or as close as is needed for the devices used). The rest of the DNARD system is similar to a 486 PC design, thus taking advantage of the many low-cost devices available for this architecture. The video and PCI subsystems are connected off the VL-Bus, and the National Semiconductor Sequoia core-logic chip set provides the main system controller. The Sequoia chip also produces an ISA bus to which the low-speed peripherals are connected.

Both the hardware design and the software/development environment are intended to simplify the introduction of developers so that they can quickly design and implement products based on this system.

4.2 The CPU

To avoid unnecessary current sourcing, CPU input pins that need to be tied to a particular logic state should be directly connected to power/ground (rather than through a resistor). The only exceptions on this design are the straps used to set the core CPU speed and the Abort signal. The core CPU speed is set using resistors i3 through i10. Only four of these should ever be fitted: the default setting is 12 (i3, i4, i9, and i10 are not fitted), thus producing a 233-MHz core clock. Consult the StrongARM CPU technical reference manual for other possible values.

Table 4–1 provides comments and notes on how certain pins on the CPU should be connected.

Table 4–1 CPU Pin Connection Notes

Signal(s)	Comments/Usage Notes
JTAG interface	Not used, so the input pins are tied to a hard ground and the output is unconnected.
Test clock	Not used, tied low.
SNA	Tied low (asynchronous) since the CPU bus clock is derived from the VL-bus clock produced by the Sequoia-1 core logic chip.
Power down	Not supported, so PWRSLP is tied high.
MSE	Tied high since the PAL requires that MREQ always be driven to indicate CPU bus requests (even during DMA).
MCCFG pins	Not used since the bus clock is asynchronous. These pins are tied low.
CONFIG pin	Tied high because the bus clock is asynchronous to enable enhanced bus mode. This improves performance by enabling write buffer merging and required-word-first cache line fetching. It also enables the use of byte enables for each byte lane on the data bus (instead of having address bits 0 and 1 and size information).
APE signal	Tied low to cause the address and address control signals to be synchronized to falling edge of MCLK.
SPDF	Reserved for DIGITAL use only, and must be tied low.

5-V level signals must be buffered (in most cases by the LVC04, which has 5-V tolerant inputs) into the 3.3-V domain of the board because the CPU input buffers have a tight maximum input voltage specification. (Exceptions are NWAIT and DBE: see the PAL notes in the section that follows.)

The standard INTR interrupt line is used to generate the CPU ~IRQ signal. The service routine must access the Sequoia and clear the source (StongARM has no automatic interrupt acknowledge cycle).

The Sequoia System Management Interrupt (SMI) signal is used to generate the StrongARM ~FIQ input. This was chosen over the non-maskable interrupt because it has more potential sources. The Sequoia power control line 4 is connected back to Sequoia's SMI-active signal to allow the CPU to clear the interrupt.

4.3 PAL Notes

The PAL used in the DNARD is an AMD MACH231-6. Although this is a 5-V PAL it has NMOS pull-ups in its output stage allowing it to directly drive all the 3.3-V parts in the design. AMD's characterization data were used with SPICE to confirm that the PAL is capable of this kind of operation. The only exceptions are the inputs to the StrongARM processor, which should not rise above 3.3V in normal operation. Particular care must be taken to avoid the extra power dissipation that would be caused by forward biasing the ESD protection diodes. Using both AMD's model and a model of the StrongARM procedssor input stage, it was determined that using a 1k pull-down to sink current from the PAL is sufficient to prevent its output stage from overdriving the CPU during transitions. If these pull-downs are removed or increased in value, then the CPU inputs will be driven beyond the normal operating specification. Both of the signals (nWAIT and DBE) that are driven in this way need the direct connection to allow full bus speed operation.

The complete, commented PAL source is available in MACH-XL format (a subset that is basically PALASM is used). A free version of MACH-XL 2.1 obtained from Vantis.

4.4 Bus Buffers

The buffers on both the data and address lines are 3.3V-to-5V bidirectional transceivers and are normally set to drive the VL-bus. The data buffers are turned around for CPU reads of the VL-bus and DMA writes into the SDRAM. The address buffers are turned around (and the CPU address signals floated) for DMA activity. The buffers are disabled for a cycle during turnaround to avoid bus clashes.

4.5 SDRAM Subsystem

The design provides two SDRAM DIMM sockets and includes the buffering and address multiplexer required by the SDRAM devices. This memory subsystem uses only 3.3V and requires one set of bypass capacitors.

The design uses 100MHz (10ns) SDRAMs clocked at 66MHz in CAS latency 2 mode. Each 168-pin DIMM slot is designed to work with modules between 4MB and 32MB. To enable this, the address lines are connected in a careful manner as described in the PAL source code.

The data bus for the StrongARM processor is only 32 bits wide, whereas the JEDEC DIMM standard is for 64 bit-wide modules. According to the standard, the ~CS0 input enables devices that drive data lines [0..15,32..47] as does ~CS1 for larger modules. The ~CS2 line enables devices that drive data lines [16..31,48..63] as does ~CS3 on larger modules. For 32-bit-wide operation the design combine these blocks of data lines and ensures that only one ~CS is ever asserted.

The modules have a small serial EEPROM containing information about the module configuration, speed, and so on. This is accessed using a I2C bus controlled by software. The I2C bus address for the EEPROMs is 1010000 for one socket and 1010001 for the other: the low three bits of the address are set by the SA0-2 lines.

4.5.1 Supported DIMMs

The design uses JEDEC standard 168-pin unbuffered 3.3-V 100-MHz SDRAM DIMMs. This frequency grade has been used for all testing. The provisional specifications for the Samsung DIMMs indicate that their 83-MHz (12-ns) parts should also work. However, this has not been tested, and the specifications are subject to change.

The following module configurations are supported (although not all configurations have been tested):

- 8MB 1 bank, 1Mx64 bit
- 16MB 1 bank, 2Mx64 bit
- 16MB 2 bank, 1Mx64 bit
- 32MB 2 bank, 2Mx64 bit
- 32MB 1 bank, 4Mx64 bit
- 64MB 2 bank, 4Mx64 bit

One caveat on the 64MB module is that the firmware currently only sets up page tables to map a total of 64MB of memory. So unless or until the firmware sets up more page tables, there's no point in putting more than 64MB into a board.

Caution: Not all DIMMs conform to the JEDEC standard. These DIMMs are 64-bits wide, but each select line controls 32 bits. Nonconforming DIMMs may not have the select lines controlling the correct 32 bits. These DIMMs will work fine in systems that access 64 bits at a time (which is how these DIMMs are most often used), but they do not work in the DNARD.

4.5.2 Data Buffers

Two 18-bit-wide data buffers are used rather than 16-bit-wide devices. This is necessary because the 16-bit devices are byte oriented and each would place four loads on the clock, rather than the two loads the chosen buffers use.

The byte enable lines are buffered through a tri-state buffer to provide the fan-out drive for modules with many chips and to allow easy de-assertion of all DQMB lines during reset and machine initialization. The BOOTMD (boot mode) signal is used to disable the DIMMs' data path while the hardware state machines are being initialized.

4.5.3 Address Latch and Multiplexer

As with all DRAMs, a multiplexed address is required. Two simple multiplexer devices are used to do the higher address bits and the PAL is used to generate the three low order bits. For the range of modules supported A11-12 are only used at RAS time and so need not be multiplexed.

To allow both the CPU burst sequence and a 486 DMA burst, the PAL generates the address for all CAS cycles, so the CAS address bits must be valid at the start of all CAS cycles. The address latch is closed to hold the column address because the CPU may change the address early prior to the last word of an access.

The a10ap signal is used to provide SDRAM address A10 during RAS time, and the auto-precharge command during CAS time. Since auto-precharge is never used, a logic low is always sent during CAS.

4.6 Video Subsystem

The video subsystem consists of the display controller, the frame buffer memory and the VGA video output stage.

4.6.1 Video Controller

The CT65550 video controller is used in its VL-bus mode. The controller's standby or low-power mode will not work because while in this state it does not notice the RESET signal, and therefore does not get correctly configured, jamming the bus.

The manufacturer of this device recommends that the core not be run above 3.3 V to avoid high power dissipation. The latest revision of the chip works with faster memory clock frequencies at 3.3 V, so it should be satisfactory. Since the device's analog supply voltage (AVCC) and the PLL supply voltage (CVCC0 and CVCC1) must never be more than the core supply, these are also 3.3 V.

The power to the bus interface (BVCC), memory interface (MVCC) and the digital interface (DVCC) are all run at 5 V to match the rest of the system. Use of appropriate DRAMs would allow the memory interface to also be reduced to 3.3 V.

The LCD interface is brought out to pins for experimentation only. No testing has been done on this interface, the recommended filter components have been omitted, and there is no software support for its use.

The video output is a current source to drive into a doubly-terminated 75R line (as is standard for VGA video). To allow for ease of connecting the TV-out signal, the source termination is provided using a pair of 37R5 resistors. This allows the correct levels to be passed to the TV encoder when no monitor is attached (since only half of the termination is in place the output voltage on the R,G,B will be twice the required value.) If both a TV and monitor are connected then the TV signal will probably appear washed-out. If TV out is not required, each pair of resistors can be combined into a single 75R resistor.

The filter on the PLL power (CVCC0 and CVCC1) requires special layout, as detailed in its datasheet. It is important that a wide trace be used to connect the pin to the small capacitor and then to the larger capacitor and into the 10R resistor. The supply side of the resistor should not be directly connected to the power plane; instead, it should connect to the pad of the bypass capacitor and from there go to a single via to the plane.

Since the R, G, and B output signals are potentially high frequency, it is advisable to keep their traces short to avoid emissions problems.

4.6.2 Video Memory

Up to 2MB of video memory may be installed. It is possible to remove one bank to obtain 1MB with the corresponding loss of video modes.

The analog device AD724 is used as an NTSC encoder (PAL is possible by supplying an appropriate clock frequency and linking the STND pin to ground). Flicker removal must be done in software; this chip provides only the encoder functions.

4.7 ROMcard Slot

The ROMcard slot is intended for storing system code and applications in a modem-based device. To allow execution from the ROM, the interface is 32 bits wide and sits directly on the VL-bus. Up to 64MB of ROM are supported.

The timing of an access to the ROM is controlled by the main PAL.

Flash memory may also be used (for development), and a write protect line is provided to support this. If writing is enabled then the PAL will pass writes to the ROM space through to the card, if writing is disabled the writes are discarded.

The connector used is based on the 88-pin DRAM card standard. This is like an enlarged PCMCIA connector and was chosen to balance the expense of the connector against something that would be easy for end users to change.

4.8 Clocks

The 32-kHz clock is made using a CMOS oscillator. It can run off the backup battery to allow the RTC to maintain the time while power is off. Even in a system with no backup battery, this clock is required for operation of the core logic, Super I/O, and video components.

The 33-MHz oscillator module sets the clock speed for the entire system. It is propagated through the core logic chips to set the VL-Bus speed, and the CPU/SDRAM bus runs at twice this speed.

The 14.318-MHz clock, which is the reference clock for the ISA peripherals and the video section, is divided by four to give a 3.5795-MHz clock for the CPU core clock generator and the smart card interface.

The PLL generates the high speed clocks for the 3.3-V section of the board. It uses the VL-bus clock from Sequoia as its reference and generates edge locked clocks from this. The 4-times clock is not used. There are five copies of the 2-times clock (66 MHz) and one inverted copy. There is also a 3.3-V copy of the reference clock which is used to close the feedback loop.

Eight copies of the 66-MHz clock are needed for the SDRAM DIMMs. These are generated using the split series terminated lines from four of the outputs. The DIMMs provide loading equivalent to a clock load on any clock inputs that are not used, so these signals will remain balanced across a range of module sizes.

One output at 66 MHz is split to feed the data bus buffers and the PAL. Care should be taken if different bus buffers are used since with the four loads (two clock inputs to each buffer) this signal is already well loaded.

The CPU's main clock edge is the falling edge of MCLK. It is given an inverted copy of the 66-MHz clock to align this with the rising edge used elsewhere.

The copy of the VLCLOCK is used for the PAL and is conveniently sychronized with the fast clock.

4.9 Sequoia Chip Set

The core PC support logic is provided by the National Semiconductor Sequoia two-chip set which communicate with each other using an undocumented burst bus and can be regarded as a single device. The Sequoia chips are configured for a 486-style interface. On the VL-bus, the ~LRDY (Local Ready) signal is not used. It allows re-timing of the end of cycle to be done by the core logic, but for all the devices used in this design, the timing of their ~RDY signal is appropriate for direct connection to the PAL.

The DRAM control interface is not used; a lower cost variant of the design omits the SDRAM and uses the Sequoia DRAM interface for standard FPM DRAM (at 33 MHz using EDO with Sequoia has no advantage). It is estimated that the performance of such a system will be slightly better than half that of the SDRAM solution but this variant has not been built nor tested.

None of the power management features of Sequoia are used.

The ~FERR signal, which on a PC can be used to signal a floating-point exception, is reclaimed and used as a regular interrupt line.

The Sequoia power control lines are used to control system functions. Table 4–2 provides a brief description of how the signals are used in this system.

Table 4–2 Sequoia Power Control Lines – System Function Control

Sequoia Signal	Use
pc[0]	Not used. Rev 1, 2, 3. Drives beep speaker. Rev 4 and later.
pc[1]	Brought to test point to use as scope trigger.
pc[2]	Used as ~gpcs1 to as chip select for the CS8900 Ethernet controller.
pc[3]	Set high to enable NTSC encoder for TV out. Defaults to low.
pct[4]	Connected to ~SMIACT. Used as ~gpcs3 to allow software to provide a pulse to clear SMI/FIQ.
pc[5]	Drives the green section of the bicolored front panel LED, low to light LED. Sequoia allows this signal to flash the LED.
pc[6]	Drives the yellow section of the bicolored front panel LED, low to light LED. Sequoia allows this signal to flash the LED.
pc[7]	Drives the yellow board mounted developer LED (normally not populated on boxed boards).
pc[8]	Drives the green board mounted developer LED (normally not populated on boxed boards).
pc[9]	The clock signal for the software generated I2C bus to the SDRAM DIMM information EEPROMs. The SCL signal on the bus is the inverse of this pin.

The Sequoia GPIO lines (signals Sequoia.smart[0..7]) are primarily used to allow software to drive the smart card interface. Table 4–3 provides a brief description of how the signals are used in this system.

Table 4–3 Sequoia GPIO Signals – Smart Card Interface Control

Signal	Use
smart[0]	Set high to apply power to the smart card. Defaults to input, resistor will pull low.
smart[1]	Set high to reset the smart card
smart[2]	Input data from smart card
smart[3]	Used to read the data line of the I2C bus to the SDRAM DIMM information EEPROM. The data line uses open collector drivers and has a pullup.
smart[4]	Output clock to smart card in CPU-driven configuration
smart[5]	Output data to smart card in CPU-driven configuration
smart[6..7]	Sets smart card interface options as shown in Table 4–4.

The Sequoia gpiob[0] signal is used as WAKE0 to raise an SMI/FIQ on both smart card insertion and removal.

Table 4–4 Smart Card Interface Options

[7]	[6]	Data	Clock
0	0	Not driven	Not driven
0	1	From smart[5]	3.579 MHz
1	0	From smart[5]	From smart[4]
1	1	From Super I/O UART 2	3.579 MHz

The Sequoia gpiob[1] signal is used to write data to the software-generated I2C bus for the SDRAM DIMM information EEPROMs. If this bit is set, the data line will be driven low; if this bit is clear, the data line will not be driven. This bit should therefore be clear when reading the bus, and should be set to the inverse of the data for writes.

Sequoia-1 provides two copies of the master clock, one of these is used as the reference for the clock distribution PLL (sheet 5) the other is split three ways (as recommended in the reference design) to drive the other Sequoia and VL-bus devices.

Sequoia-1 provides resets for the whole system. It takes the power good signal and the backup battery good signal and will generate a system reset if either are low. The backup battery (RCRST) signal causes a “harder” reset since all the Sequoia configuration registers are also reset (something that does not happen at power-up if the backup battery is good).

The Sequoia RTC is used by the system, so it uses IRQ8.

The Sequoia keyboard controller interface is not used (the Super I/O provides this), so the KBRST signal is used in its alternative function as ~GPCSC0 to provide a chip select line to the ISA UMI slot. The availability of this signal as a chip select line, reduces the need for an address decode PAL on the UMI card.

The VL-bus LOCAL signals are used by other devices to claim a cycle as described in Table 4–5.

Table 4–5 Sequoia VL-Bus LOCAL Signal Usage

~LOCAL	Device Using
0	SDRAM. Used by the PAL to claim a DMA (or PCI) cycle as an access to main memory
1	Video. Used by the 65550 to claim the bus as an access to its configuration registers or video memory.
2	PCI. Used by the PCI bridge to claim the cycle.

The ISA interface is standard, due to loading an address buffer is provided on all lines except adr[18].

The standard timer 2 output is taken to drive the “beep” speaker. A power control line is used and software must run the timing loop to generate a tone.

The output from timer 2 (normally used on a PC to drive the beep speaker) is passed to the SWTCH input on Sequoia-1. This can be used to give a frequent low-overhead SIM/FIQ interrupt for profiling and software timing.

4.9.1 Boot ROM

The boot ROM is 8 bits wide and is expanded by Sequoia (which will do four accesses for every read). Sequoia decodes the boot ROM based on BA31, BA27, and BA25 being set, so during boot time (BOOTMD is set), the PAL forces these bits to be set, thus allowing the processor (which on reset executes at location 0) to access the ROM.

The ROM may be mask ROM, EPROM, or flash memory up to 512k bytes.

Flash memory that can be programmed at 5 V may be programmed in place. The Firmware supports reprogramming an Am29F040 512kx8 Flash device. Increasing to 16-bit-wide boot ROM is possible since Sequoia has a configuration setting to enable this.

4.9.2 IDE Interface

A standard IDE interface is provided using Sequoia. The “Turbo” or extended modes are not supported (although they could be with additional buffering; see the Sequoia reference design).

The data is buffered to allow driving a cable to the drive. An integrated device on the motherboard could probably use a direct connection (although this has not been verified or tested). HCT drivers were chosen since they have relatively slow edges and should help reduce EMI.

4.10 PCI and UMIs

The design implements a PCI bridge and UMI slots. The UMI slots are inexpensive connectors intended for option boards that are inserted at the final manufacturing step. They allow manufactures to do just-in-time binding of the network interface (for example ISDN or POTS modem).

4.10.1 ISA UMI

The ISA UMI interface provides a manufacturing-time option slot for ISA devices. The upper address bits to the slot are connected directly to the VL-bus; these are not used by many ISA devices and are lightly loaded in the system. The UMICS line is provided by Sequoia as a decode of upper address bits. Using this may remove the need for a decode PAL on the option card. It is the responsibility of the firmware or operating system driver to program the Sequoia to correctly decode the address range.

4.10.2 PCI UMI

The PCI UMI provides a manufacturing time option slot for PCI devices. The Via bridge connects the option slot to the VL-bus. The bridge chip is connected for "compatible" operation (used when the core logic chip set is not one of the Via family). This mode is not documented in the device's datasheet, but details are given in an application note available from the manufacturer.

The bridge chip will claim many cycles that are not for PCI devices. When it does this, it issues a back-off on VL.~BOFF, which causes the cycle to be rerun. The bridge learns and will not claim the cycle a second time.

At the current time, there are no drivers for the PCI. Simple tests have successfully read the configuration registers of a card behind the bridge. Only 5 V is supplied to the option slot.

4.11 Super I/O Device

The National PC87307 Super I/O device provides all the low-speed I/O for DNARD and was chosen because it includes support for consumer IR (remote control). The 87307 was selected over the similar but more expensive 87308 because there is no need for the fast IRDA IR supported by the 87308 device.

- The floppy interface is not used.
- The RTC (Real Time Clock) is not used (the Sequoia devices provide this function).
- The backup battery power is provided to the Super I/O device to allow use of the protected RAM and to avoid problems with the power management subsystem.
- The Super I/O clock is generated by multiplying up the 32-kHz RTC clock.
- Two serial EEPROMs are connected to the GPIO lines. One provides the Ethernet UID, and it is programmed according to the DIGITAL format. The other provides nonvolatile storage: it is partitioned with 4kbits for the firmware (to hold environment and boot options), 4kbits for the NCI software stack (to hold ISP information), and the rest for allocation by DIGITAL.

4.11.1 Keyboard/Mouse

The Super I/O device includes a standard keyboard controller running the AMI keyboard code. This interfaces to a PS/2 keyboard and mouse. The output is filtered against EMI and protected against ESD. The power to the keyboard and mouse is protected by a PTC cutout thermistor.

4.11.2 Parallel Port

The Super I/O device provides a standard ECP/EPP parallel port, which is filtered in the usual way.

4.11.3 Serial I/O

Serial UART 1 on the Super I/O device is used to drive a standard RS-232 serial port. The MAXIM (or equivalent) transceiver allows for 5V-only operation.

A Super I/O GPIO bit allows the transceiver to be shutdown, which results in very low current drain. The transceiver with built-in ESD protection should be used with this interface.

4.11.4 LEDs

Four of the Sequoia power control lines drive status LEDs. Two LEDs are for developers only, and are only populated on developer boards; they are not visible through the standard DNARD case. The two Sequoia lines which have built-in LED flashing capability are used to drive a green/yellow bicolor LED to allow liveness and activity indications. (Red was not used because of its interpretation as a warning in some countries.) Two of the LED driver open-collector buffers are used to make the software-controlled I2C bus used to access the SDRAM information EEPROM.

4.12 Smart Card

The smart card interface allows various sources for the data and clock for the smart card.

Smart cards have a special way of signalling parity errors, which prevents use of the UART as the transmitter or receiver. The smart card detect contacts are connected to the Sequoia WAKE0 signal. This signal can be debounced and set to cause a SMI/FIQ on both edges (that is, on card insertion and card removal). The use of a high-priority interrupt will allow the system to respond in a timely manner to card removal (which must revoke access rights).

4.13 Consumer IR

The IR logic allows consumer remote control style IR transmission and reception. The first section of the IR plug_and_play recommendation is supported. Although support is shown for only consumer IR, frequency deduction is not possible. (It would require an additional gate; see the National Semiconductor IR recommendations.) The reception module produces logic level outputs, and it combines a photodetector, amplifier, and discriminator on a single device.

Two transmit LEDs are used; lower transmit power is possible by removing one of the LEDs and one of the 14R resistors. The transmitter frequency is set in the Super I/O, but the receiver is tuned. A 38-kHz receiver is used because it is common in remote controls and because the pass filter should be wide enough to allow acceptable behavior at 36 kHz and 40 kHz.

No driver code currently exists for the IR subsystem.

A

Support, Products, and Documentation

If you need technical support, a *DIGITAL Semiconductor Product Catalog*, or help deciding which documentation best meets your needs, visit the DIGITAL Semiconductor World Wide Web Internet site:

http://www.digital.com/semiconductor

You can also call the DIGITAL Semiconductor Information Line or the DIGITAL Semiconductor Customer Technology Center. Please use the following information lines for support.

For documentation and general information:	
DIGITAL Semiconductor Information Line	
United States and Canada:	1–800–332–2717
Outside North America:	1–510–490–4753
Electronic mail address:	semiconductor@digital.com

For technical support:	
DIGITAL Semiconductor Customer Technology Center	
Phone (U.S. and international):	1–978–568–7474
Fax:	1–978–568–6698
Electronic mail address:	ctc@hlo.mts.dec.com

DIGITAL Semiconductor Products

Note: The following products and order numbers might have been revised. For the latest versions, contact your local distributor.

To order the SA-110 microprocessors contact your local distributor.
The following tables list some of the semiconductor products available from DIGITAL Semiconductor.

Chips	Order Number
DIGITAL Semiconductor SA-110 160 MHZ Microprocessor	21281-AA
DIGITAL Semiconductor SA-110 100 MHZ Microprocessor	21281-BA
DIGITAL Semiconductor SA-110 200 MHZ Microprocessor	21281-CA
DIGITAL Semiconductor SA-110 166 MHZ Microprocessor	21281-DA
DIGITAL Semiconductor SA-110 233 MHZ Microprocessor	21281-EA

Evaluation board kits include an evaluation board, and can include a complete design kit, an installation kit, or an accessories kit.

Evaluation Board Kits	Order Number
DIGITAL Network Appliance Reference Design	21A81-05

DIGITAL Semiconductor Documentation

The following table lists some of the documentation available from DIGITAL Semiconductor.

Title	Order Number
DIGITAL Semiconductor SA-110 Microprocessor Technical Reference Manual	EC-QPWLC-TE
Memory Management on the StrongARM SA-110, An Application Note (MB)	EC-R4WCA-TE
SA-110 Microprocessor Instruction Timing: An Application Note	EC-R6M6A-TE
ARM Architecture Reference Manual	EC-QV44B-TE
StrongARM/ARM Technical Document Kit (includes the SA-110 Technical Reference Manual, EC-QPWLC-TE and the ARM Architecture Reference Manual, EC-QV44A-TE)	QR-ARMKT-TE

Third–Party Documentation

You can order the following third-party documentation directly from the vendor.

Title	Vendor
PCI Local Bus Specification, Revision 2.1 PCI Multimedia Design Guide, Revision 1.0 PCI System Design Guide PCI-to-PCI Bridge Architecture Specification, Revision 1.0 PCI BIOS Specification, Revision 2.1	PCI Special Interest Group U.S. 1–800–433–5177 International 1–503–797–4207 Fax 1–503–234–6762