KN220 CPU System Maintenance

Order Number EK–375AA–SM–001

Digital Equipment Corporation
Maynard, Massachusetts

First Printing, September 1990

Printed in U.S.A.

The Reader's Comments form at the end of this document requests your critical evaluation to assist in preparing future documentation.

FCC NOTICE: The equipment described in this manual generates, uses, and may emit radio frequency energy. The equipment has been type tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such radio frequency interference when operated in a commercial environment. Operation of this equipment in a residential area may cause interference, in which case the user at his own expense may be required to take measures to correct the interference.

S1304

This document was prepared using VAX DOCUMENT, Version 1.2.

Contents

Preface xiii

Chapter 1 KN220 Base System

1.1 Base System Overview 1–1
1.2 KN220 Features 1–6
1.2.1 R3000 RISC Processor 1–7
1.2.2 Cache Memory 1–8
1.2.3 Main Memory System 1–8
1.2.4 Console Serial Line 1–8
1.2.5 Time-of-Year Clock and Timers 1–9
1.2.6 Boot and Diagnostic Facility 1–9
1.2.7 Q22-Bus Interface 1–10
1.2.8 CVAX Diagnostic Processor 1–10
1.2.9 Network Interface 1–11
1.2.10 DSSI Interface 1–11
1.2.11 SCSI Interface 1–12
1.3 H3602–AC CPU I/O Panel 1–12
1.4 Using Console Security 1–16
1.4.1 Securing the System 1–16
1.4.2 Privileged Users 1–17
1.4.3 Unsecuring the System 1–17
1.4.4 The passwd Command 1–18
1.4.5 The unpriv Command 1–18
1.5 MS220 Memory 1–19

Chapter 2 KN220 Configuration

2.1 Introduction 2–1
2.2 General Module Order 2–1
2.3 Module and Bulkhead Order for KN220 Systems 2–2
2.4 Memory Module Configuration 2–2
2.5 Q-Bus Module Configuration 2–2
2.6 DSSI Configuration 2–4
2.6.1 Changing the Node Name 2–6
2.6.2 Changing the Unit Number 2–7
2.6.3 DSSI Cabling 2–9
2.6.4 DSSI Bus Termination and Length 2–10
2.7 SCSI Configuration and Cabling 2–10
2.7.1 Adding External Devices 2–10
2.7.2 Connecting Multiple Drives 2–11
2.7.3 Connecting Tabletop Drives 2–11
2.7.4 Connecting Internal Drive to Tabletop Drive 2–12
2.7.5 Assigning the Node ID 2–12
2.7.6 SCSI Interface ID Switches 2–12

Chapter 3 KN220 Firmware

3.1 Introduction 3–1
3.2 KN220 Firmware Features 3–2
3.3 CVAX Halt Entry and Dispatch Code 3–3
3.4 External Halts 3–4
3.5 Power-Up 3–5
3.5.1 Power-Up Sequence: Operation Switch Set to Normal 3–5
3.5.2 Power-Up Sequence: Operation Switch Set to Maintenance 3–6
3.5.3 Operation Switch Set to Action: Loopback Tests 3–6
3.5.4 Operation Switch Set to Action: Language Query 3–7
3.6 Bootstrap 3–8
3.6.1 ULTRIX–32 Bootstrap 3–9
3.6.1.1 ULTRIX–32 Bootstrap Procedure 3–10
3.6.1.2 On Installation 3–10
3.6.2 MDM Bootstrap 3–11
3.6.3 MDM Restart 3–15

3.7 Normal Mode Overview 3–16
3.7.1 Control Characters in Normal Mode 3–16
3.7.2 Environment Variables in Normal Mode 3–17
3.8 Normal Mode Commands 3–18
3.8.1 Conventions Used in This Section 3–20
3.8.2 Getting Help 3–20
3.8.3 boot 3–21
3.8.4 continue 3–23
3.8.5 d (deposit) 3–24
3.8.6 dump 3–25
3.8.7 e (examine) 3–27
3.8.8 fill 3–28
3.8.9 go 3–29
3.8.10 help 3–30
3.8.11 ? 3–31
3.8.12 init 3–32
3.8.13 printenv 3–33
3.8.14 setenv 3–34
3.8.15 test 3–35
3.8.16 unsetenv 3–36
3.8.17 x 3–37
3.9 Maintenance Mode Overview 3–38
3.9.1 Command Syntax in Maintenance Mode 3–39
3.9.2 Address Specifiers in Maintenance Mode 3–39
3.9.3 Symbolic Addresses in Maintenance Mode 3–40
3.9.4 Command Qualifiers in Maintenance Mode 3–42
3.9.5 Maintenance Mode Command Keywords 3–44
3.10 Maintenance Mode Commands 3–47
3.10.1 BOOT 3–47
3.10.2 CONFIGURE 3–48
3.10.3 CONTINUE 3–50
3.10.4 DC 3–51
3.10.5 DEPOSIT 3–53
3.10.6 EXAMINE 3–54
3.10.7 EXIT 3–56
3.10.8 FIND 3–57

3.10.9 HALT 3–58
3.10.10 HELP 3–59
3.10.11 INITIALIZE 3–61
3.10.12 MOVE 3–63
3.10.13 NEXT 3–65
3.10.14 REPEAT 3–67
3.10.15 SEARCH 3–68
3.10.16 SET 3–70
3.10.17 SHOW 3–73
3.10.18 START 3–77
3.10.19 TEST 3–78
3.10.20 UNJAM 3–79
3.10.21 X—Binary Load and Unload 3–80
3.10.22 ! (Comment) 3–82

Chapter 4 KN220 Troubleshooting and Diagnostics

4.1 Introduction 4–1
4.1.1 General Troubleshooting Procedures 4–1
4.2 KN220 ROM-Based Diagnostics 4–3
4.2.1 Diagnostic Tests 4–4
4.2.2 Scripts 4–7
4.2.3 Script Calling Sequence 4–10
4.2.4 Creating Scripts 4–12
4.2.5 Console Displays and LEDs 4–17
4.2.6 System Halt Messages 4–31
4.2.7 Console Error Messages 4–32
4.2.8 VMB Error Messages (CVAX) 4–33
4.3 Acceptance Testing 4–34
4.4 Troubleshooting 4–41
4.4.1 FE Utility 4–41
4.4.2 Isolating Memory Failures 4–44
4.4.3 Running a Memory Test 4–46
4.4.4 Additional Troubleshooting Suggestions 4–47
4.5 Loopback Tests 4–48
4.5.1 DSSI Problems 4–48

4.5.2 Ethernet Problems 4–48
4.5.3 Testing the Console Port 4–49
4.6 Module Self-Tests 4–49
4.7 RF-Series ISE Troubleshooting and Diagnostics 4–50
4.7.1 DRVTST 4–52
4.7.2 DRVEXR 4–53
4.7.3 HISTRY 4–54
4.7.4 ERASE 4–55
4.7.5 PARAMS 4–56
4.7.5.1 EXIT 4–57
4.7.5.2 HELP 4–57
4.7.5.3 SET 4–57
4.7.5.4 SHOW 4–58
4.7.5.5 STATUS 4–58
4.7.5.6 WRITE 4–58
4.7.6 Diagnostic Error Codes 4–59
4.8 Memory Diagnostics 4–59
4.8.1 Test 30 - Bitmap Placing Test 4–59
4.8.2 Test 4F - Memory Data Tests 4–62
4.8.3 Test 4E - Memory Byte Tests 4–62
4.8.4 Test 4D - Memory Address Uniqueness Test 4–62
4.8.5 Test 4C - Memory ECC Logic, Verify Error Detection and Reporting 4–62
4.8.6 Test 48 - Memory Address/Shorts Test 4–62
4.8.7 Test 47 - Memory Data Retention, Verify Refresh Logic . . . 4–63
4.8.8 Test 40 - Memory Count; Bad Pages Marked in Bitmap . . 4–63
4.8.9 Test 9A - Define Current Memory Configuration 4–63
4.9 SCSI Controller Chip Test 4–64
4.9.1 ASC Reset Test 4–64
4.9.2 ASC Register Test 4–64
4.9.3 ASC Interrupt Test 4–64
4.9.4 ASC FIFO Test 4–65
4.9.5 ASC DMA Counter Register Test 4–65

Appendix A ULTRIX–32 Exerciser and uerf Command Summary

A.1 On-line ULTRIX Exerciser A–1
A.1.1 Communications Exerciser (Asynchronous Serial Lines) . . A–2
A.1.2 Disk Exerciser A–3
A.1.3 File System Exerciser A–4
A.1.4 Line Printer Exerciser A–4
A.1.5 Memory Exerciser A–5
A.1.6 Magtape Exerciser A–5
A.1.7 TCP/IP Network Exerciser A–6
A.2 uerf Error Log Commands A–8

Appendix B KN220 Address Assignments

B.1 Accessing Physical Locations (R3000) B–1
B.2 R3000 Physical Address Space Map (M7637–AA) B–3
B.3 R3000 Physical I/O Address Space Map (M7638–AA) B–6
B.4 KN220 Diagnostic Processor Physical Address Space Map (M7638–AA) B–11
B.5 Diagnostic Processor Registers B–15
B.6 Global Q22-Bus Memory Space Map B–17

Appendix C Configuring the KFQSA

C.1 KFQSA Overview C–1
C.2 Configuring the KFQSA at Installation C–2
C.2.1 Entering Maintenance Mode C–4
C.2.2 Displaying Current Addresses C–5
C.2.3 Running the Configure Utility C–6
C.3 Programming the KFQSA C–8
C.4 Reprogramming the KFQSA C–13
C.5 Changing the ISE Unit Number C–15

Appendix D Prestoserve Software on the DECsystem 5500

D.1 Why Data Recovery Is Necessary D–1
D.2 Using the dc Commands D–1
D.2.1 Determining If a Cache Contains Data D–3
D.2.2 Saving the Cache Data to Tape with the dc/save Command D–3
D.2.3 Restore Data From Tape with the dc/restore Command . . . D–4
D.2.4 Clearing the Cache with the dc/zero Command D–4
D.3 Recover from Abnormal System Shutdowns D–4
D.3.1 Recovering Data From a System That Can Reboot D–5
D.3.2 Recovering Data From a System That Cannot Reboot D–5
D.3.2.1 Bad I/O Board D–6
D.3.2.2 Bad CPU Board D–6
D.3.2.3 Bad Boot Disk D–7
D.3.2.4 Other Hardware Problems D–8
D.3.3 Power-up Screen and Test 79 with NVRAM Battery Off . . D–8

Appendix E Field-Replaceable Units (FRUs)

Appendix F Related Documentation

Index

Examples

2–1 Changing a DSSI Node Name 2–6
2–2 Changing a DSSI Unit Number 2–8
3–1 Language Selection Menu 3–8
3–2 Command Procedure to Boot on Installation 3–11
4–1 Creating a Script with Utility 9F 4–13
4–2 Listing and Repeating Tests with Utility 9F, Help and Loop on A0 4–14
4–3 Console Display (No Errors) 4–17
4–4 Sample Output with Errors: R3000 4–19
4–5 Sample Output with Errors: CVAX 4–20

4–6 SHOW SCSI and SHOW SCSI/FULL 4–40
4–7 FE Utility Example 4–41
4–8 Isolating Bad Memory Using T 9C 4–45
C–1 Entering Console Mode Display C–5
C–2 SHOW QBUS Display C–5
C–3 Configure Display C–7
C–4 Display for Programming the First KFQSA C–8
C–5 SHOW QBUS Display C–11
C–6 SHOW DEVICE Display C–12
C–7 Reprogramming the KFQSA Display C–14
C–8 Display for Changing Unit Number C–16

Figures

1–1 KN220 CPU Module (M7637–AA) 1–2
1–2 KN220 I/O Module (M7638–AA) 1–3
1–3 KN220 Functional Block Diagram 1–5
1–4 H3602–AC CPU I/O Panel 1–14
4–1 KN220 I/O Module LEDs 4–24
B–1 KN220 Virtual Memory Map B–2
C–1 KFQSA Module Layout (M7769) C–3

Tables

1–1 H3602–AC Operation and Function Switch Settings 1–15
2–1 Conventional ISE and Tape Slots 2–4
2–2 ISE DIP Switch Settings 2–5
2–3 TLZ04 SCSI Address Node ID Number Settings 2–13
2–4 RRD40 SCSI Address ID Number Settings 2–14
2–5 SCSI Cables 2–14
3–1 KN220 Firmware Code 3–1
3–2 Actions Taken on a Halt 3–4
3–3 ULTRIX–32 Supported Boot Devices 3–10
3–4 VMB Boot Flags 3–13
3–5 Supported MDM Boot Devices 3–14
3–6 Normal-Mode Control Characters 3–16
3–7 Environmental Variables 3–17

3–8 KN220 Normal Mode Commands 3–18
3–9 Boot Device Names (Normal Mode) 3–21
3–10 KN220 Console Control Characters (Maintenance Mode) 3–38
3–11 Console Symbolic Addresses (Maintenance Mode) 3–40
3–12 Symbolic Addresses Used in Any Address Space 3–42
3–13 Console Command Qualifiers (Maintenance Mode) 3–43
3–14 Command Keywords by Type (Maintenance Mode) 3–44
3–15 Console Command Summary (Maintenance Mode) 3–44
4–1 Scripts Available to Customer Services 4–9
4–2 Commonly Used Customer Services Scripts 4–10
4–3 Tests Run During Power-up 4–18
4–4 Values Saved, Machine Check Exception During Executive (CVAX) 4–22
4–5 Values Saved, Exception During Executive (CVAX) 4–22
4–6 LED Codes 4–23
4–7 KN220 Console Displays and FRUs 4–25
4–8 System Halt Messages 4–31
4–9 Console Error Messages 4–32
4–10 VMB Error Messages 4–33
4–11 Hardware Error Summary Register 4–43
4–12 Running a Memory Test 4–46
4–13 Common Memory Parameters 4–47
4–14 Loopback Connectors for Q22-Bus Devices 4–50
4–15 DRVTST Messages 4–52
4–16 DRVEXR Messages 4–53
4–17 HISTRY Messages 4–54
4–18 ERASE Messages 4–56
4–19 RF-Series ISE Diagnostic Error Codes 4–59
B–1 R3000 Physical Address Space B–3
B–2 R3000 Physical I/O Addresses B–6
B–3 KN220 Diagnostic Processor Physical Addresses B–11
B–4 Diagnostic Processor Registers B–15
B–5 Q22-Bus Memory Space B–17
B–6 Q22-Bus I/O Space with BBS7 Asserted B–17
C–1 KFQSA (M7769) Service Mode Switch Settings C–4

Preface

This maintenance guide describes the base system, configuration, ROM-based diagnostics, and troubleshooting procedures for systems containing the KN220 CPU module and the KN220 I/O module.

Intended Audience

This guide is intended for use by Digital Customer Services personnel and qualified self-maintenance customers.

Organization

This guide has four chapters and six appendixes, as follows:

Chapter 1 describes the KN220 base system.

Chapter 2 contains system configuration guidelines and provides a table listing current, power, and bus loads for supported options. Chapter 2 also describes the Digital Storage System Interconnect (DSSI) and Small Computer Storage Interface (SCSI) bus interface cabling between the KN220 I/O module, the CPU I/O panel, the operator control panel (OCP), and the integrated storage elements (ISEs).

Chapter 3 describes the KN220 diagnostic firmware, including normal mode commands and maintenance mode commands.

Chapter 4 describes the KN220 diagnostics, including an error message and FRU cross-reference table. Chapter 4 also describes diagnostics that reside on the ISEs.

Appendix A describes the uerf commands.

Appendix B contains the address space maps for the KN220 CPU module and the KN220 I/O module.

Appendix C explains how to configure the KFQSA storage adapter.

Appendix D describes the Prestoserve™ commands for dealing with data in the NVRAM cache on the CPU board.

Appendix E lists the major field-replaceable units (FRUs) of the KN200 system.

Appendix F contains a list of related documentation.

Cautions, Notes, and Conventions

Cautions, differences, and notes appear throughout this guide. They have the following meanings:

CAUTION	Provides information to prevent damage to equipment or software.
NOTE	Provides general information about the current topic.
Boldface	User input is indicated by commands in **boldface type**. ULTRIX commands are lowercase, VMS commands are uppercase.

Chapter 1
KN220 Base System

This chapter describes the KN220 base system, which consists of the KN220 CPU module, the KN220 I/O module, the MS220–AA memory modules, and the H3602–AC I/O panel.

1.1 Base System Overview

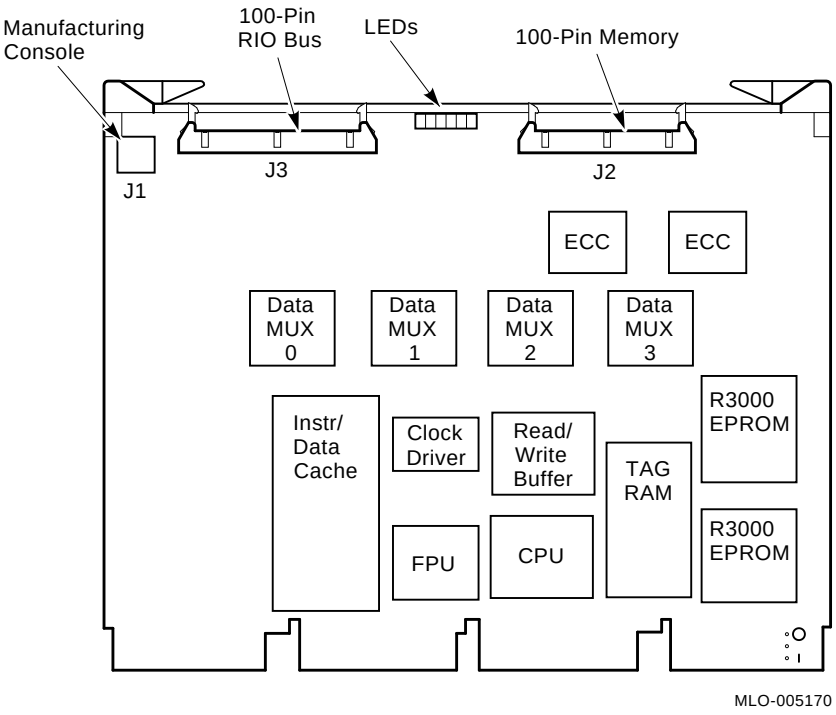
The KN220 system is designed for applications that require high-performance processing. The KN220 system supports only the ULTRIX–32 operating system (Version 4.0 and later). KN220 configurations include the capability for server and multiuser support.

The KN220 systems are enclosed in the DECsystem 5500 Pedestal (BA430 enclosure).

The KN220 CPU module (M7637–AA) is a quad-height processor module. The KN220 CPU operates at a 30 MHz clock rate and contains a reduced instruction set computer (RISC) processor based on the R3000 MIPS chipset. The RISC implementation is based on a CPU architecture that uses a pipelined design, a simple instruction set, and write buffering.

The major components on the KN220 CPU module are shown in Figure 1–1.

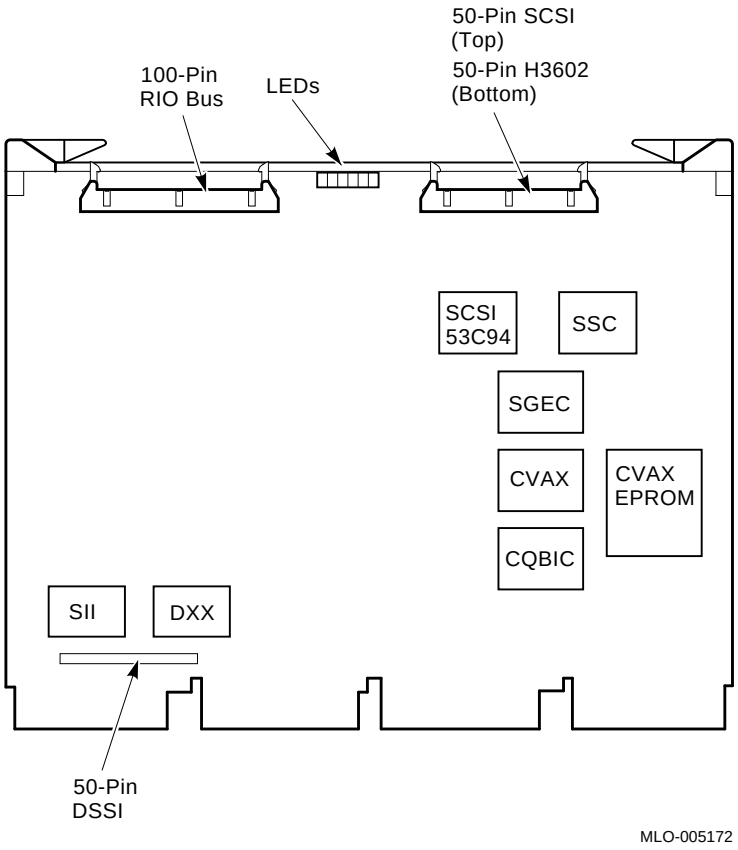
Figure 1–1: KN220 CPU Module (M7637–AA)



The KN220 I/O module (M7638–AA) is a quad-height module that contains mass storage and network interfaces. These interfaces provide higher performance than those available on the Q22-bus.

The major components on the KN220 I/O module are shown in Figure 1–2.

Figure 1–2: KN220 I/O Module (M7638–AA)



The KN220 I/O module appears as the following asynchronous devices on the KN220 CPU's buffered RIO bus, which is a private I/O bus:

- Master/slave devices:
  - —CVAX diagnostic processor
  - —CVAX Q22-bus interface chip (CQBIC)

—Second-generation Ethernet controller chip (SGEC)

- —Slave-only devices:

  Ethernet station address ROM
  —VAX-compatible console port
  —DSSI controller chip (SII)
  —DSSI buffer memory
  —SCSI controller chip (53C94)
  —SCSI buffer memory

The mass storage interface controls up to seven devices on a Digital Storage System Interconnect (DSSI) bus capable of a transfer rate of 4 Mbytes per second. The Small Computer Storage Interface (SCSI) port has 128 Kbytes of static RAM buffer space with a 32-bit data and address path to the CPU module.

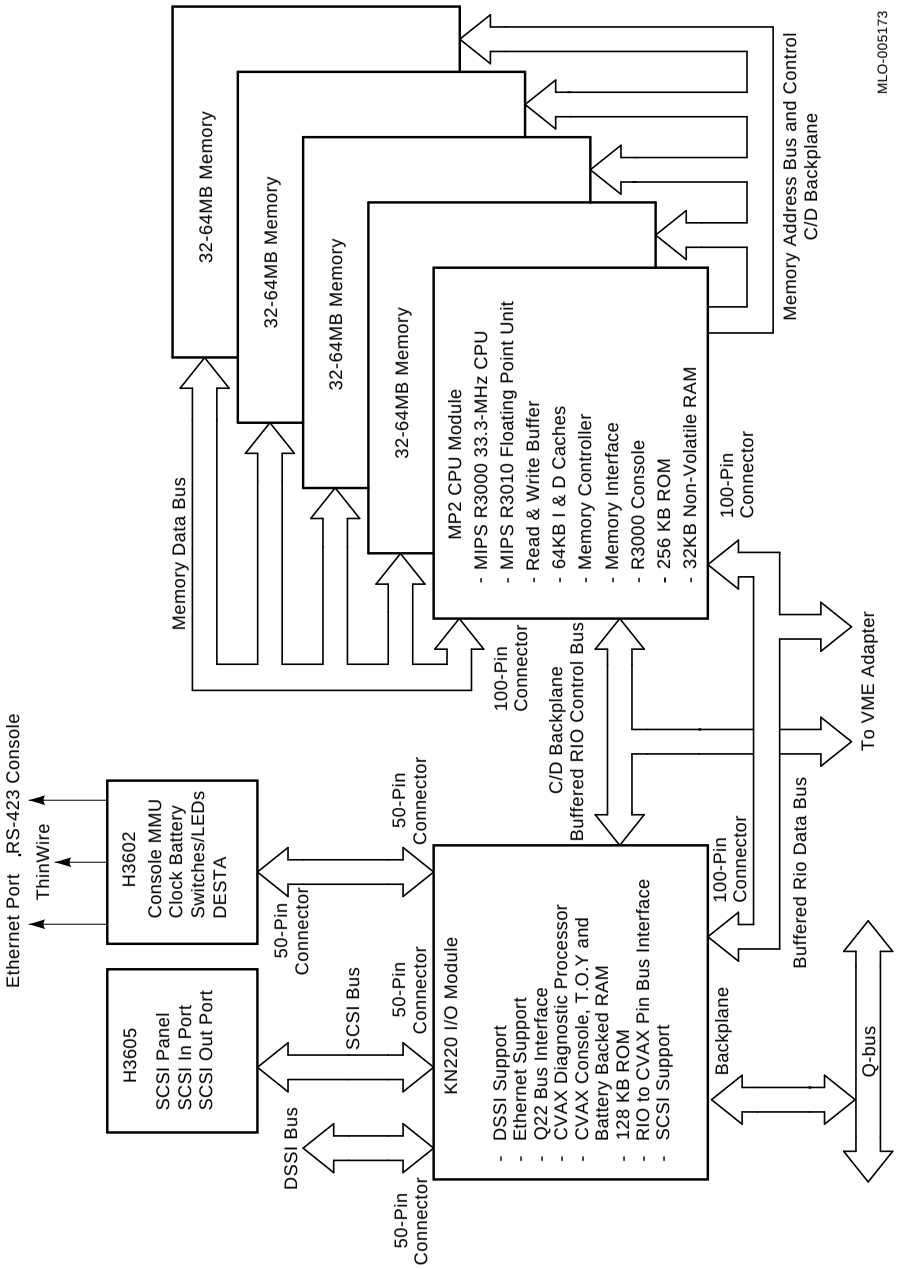
The mass storage interface also controls up to seven devices on a Small Computer Storage Interface (SCSI) bus. The bus has a transfer rate of 4 Mbytes per second. The DSSI port has 128 Kbytes of static RAM buffer space with a 32-bit data and address path to the CPU module. The network interface is an Ethernet controller.

The KN220 CPU module, the KN220 I/O module, and the MS220–AA memory module(s) combine to form a subsystem that contains an RIO bus and a Q22-bus for communicating with mass storage and I/O devices.

The KN220 CPU module set and the MS220–AA modules mount in standard Q22-bus backplane slots that implement the Q22-bus in the AB rows and the CD interconnect in the CD rows.

Figure 1–3 shows a functional block diagram of the KN220 CPU module, I/O module, and memory subsystem.

Figure 1–3: KN220 Functional Block Diagram



1.2 KN220 Features

The major features of the KN220 CPU-I/O module set are as follows:

- MIPS R3000-series RISC processor with a cycle time of 33 ns.
- MIPS R3010 floating-point unit.
- Enhanced read/write buffer for memory throughput.
- 64-Kbyte, 12-ns instruction cache.
- 64-Kbyte, 12-ns data cache.
- DSSI mass storage interface.
- SCSI (small computer system interface) mass storage interface.
- Ethernet interface that supports DMA.
- Main memory controller that supports up to 256 Mbytes of error correction code (ECC) memory. The ECC memory resides on one to four MS220–AA memory modules, depending on the system configuration.
- Console port featuring switch-selected baud rates.
- Console port for testing the CPU module as a standalone unit.
- RIO bus interface that supports DMA transfers from devices on the I/O module.
- System security and features.
- Q22-bus interface that supports up to 16-word, block mode transfers between a Q22-bus DMA device and main memory, and block mode transfers of up to 2 words between the CPU and Q22-bus devices. This Q22-bus interface contains:
  - 16-entry map cache for the 8192 entry, scatter-gather map that resides in main memory. This map translates 22-bit, Q22-bus addresses into 26-bit main memory addresses.
  - Interrupt arbitration logic that recognizes Q22-bus interrupt requests BR7 through BR4.
  - 240-ohm, Q22-bus termination.
- CVAX diagnostic processor.
- Two 128-Kbyte EPROMs (R3000).
- 128-Kbyte EPROM (CVAX).

- 512-Kbyte nonvolatile memory (NVRAM)

1.2.1 R3000 RISC Processor

The R3000 RISC processor plus its associated R3010 floating-point unit combine to form the KN220 central processor.

The R3000 chip resides on the KN220 CPU module (Figure 1–1) and implements two tightly coupled processors in a single VLSI chip. One processor is the 32-bit CPU and the other is the system control processor (CP0). The combined CPU/CP0 processors provide the following features:

- 32-bit operation. The R3000 contains thirty-two 32-bit registers that use 32-bit addressing.
- Pipelined design. The five-stage pipeline is capable of executing one instruction per 33-ns cycle.
- On-chip cache control. Separate instruction and data caches of 64 Kbytes each. Each cache can be accessed in a single CPU cycle.
- On-chip memory management. The 4-Gbyte virtual address space is mapped with a 64-entry, fully associative translation lookaside buffer (TLB).
- Coprocessor interface. A tightly coupled coprocessor interface for up to four coprocessors. CP0 is located on the CPU chip. CP1 is the floating-point accelerator. CP2 and CP3 are not used.
- Single read/write buffer. All CPU reads and writes pass through this write buffer.

DIFFERENCES: *For the KN220-based system, the terminology for various words is as follows:*

- *R3000: a halfword consists of 16 bits, and a word consists of 32 bits.*
- *CVAX: a word consists of 16 bits, and a longword consists of 32 bits.*

For similar systems, the terminology is the same as for the CVAX listed above: a word consists of 16 bits, and a longword consists of 32 bits.

The KN220 floating-point accelerator resides on the KN220 CPU module and is implemented by a single VLSI chip called the R3010.

1.2.2 Cache Memory

To maximize CPU performance, the KN220 CPU module contains a 64-Kbyte instruction cache and a 64-Kbyte data cache. Both caches have the same organization and are direct mapped, with a block size of one word (four bytes). The fill size is either one word (4 bytes) or four words (16 bytes).

1.2.3 Main Memory System

The KN220 CPU module contains a main memory controller with state machine and memory control signals.

The maximum amount of main memory supported by KN220 systems is 256 Mbytes. This memory resides on from one to four MS220–AA memory modules, depending on the system configuration. The MS220 modules communicate with the KN220 through the MS220 memory interconnect, which uses the CD interconnect for the address and control lines and a 100-pin ribbon cable for the data lines.

The main memory controller supports the following:

- Synchronous or asynchronous, 32-bit word read and write references
- Synchronous 4-word read references generated by R3000 cache references that miss the cache
- Longword, quadword, hexword, or octaword asynchronous reads generated by the CVAX or any DMA device on the RIO bus
- Masked or unmasked write references (synchronous or asynchronous) generated by the DMA devices or write buffer, as well as synchronous pagemode unmasked writes

1.2.4 Console Serial Line

The KN220 contains two console lines: one is located on the I/O module and one on the CPU module.

The console serial line on the KN220 I/O module is the standard VAX console implemented through the System Support Chip (SSC) and the H3602–AC CPU I/O panel.

The console serial line on the KN220 CPU module can be programmed to provide a full-duplex, RS–423 EIA serial line interface, which is also RS–232C compatible. This console serial line is implemented through the 2681 DUART chip. The port is available *only* during manufacturing; one channel of the DUART chip is available through an 8-pin MMJ connector mounted on the CPU module (not available in the field).

1.2.5 Time-of-Year Clock and Timers

The KN220 I/O module contains the time-of-year clock (TODR), two additional programmable timers, and a 100-Hz interval timer that serves as the R3000 interval clock.

1.2.6 Boot and Diagnostic Facility

The KN220 CPU boot and diagnostic facility is contained on both the I/O and CPU modules.

The KN220 CPU module contains firmware that consists of two EPROMs, each 64-Kbytes by 16 bits. Another 64-Kbyte EPROM resides on the KN220 I/O module. Both the CVAX and the R3000 CPU have access to all three EPROMs. The CPU module ROM address space extends from 1FC00000 to 1FC3FFFF in the R3000 memory map (2FC00000 to 2FC3FFFF in CVAX space).

The KN220 CPU module also contains 512 Kbytes of battery backed-up NVRAM, for use as a console scratchpad and NFS buffer. This array is not protected by parity; bus parity is neither checked nor generated on reads or writes.

NOTE: *The NVRAM battery jumper, in the upper rear corner of the CPU board as it is seated in the enclosure, must be set in the On position (1).*

The KN220 firmware gains control when the processor halts and contains programs that provide the following services:

- Module initialization
- Power-up self-testing of the KN220 and MS220 modules
- R3000 console program
- Emulation of a subset of the VAX standard console, which contains manual bootstrap and a simple command language for examining or altering the state of the processor
- Booting from supported Q22-bus devices, SCSI, Ethernet, and DSSI
- Multilingual capability

The KN220 firmware is described in detail in Chapter 3.

1.2.7 Q22-Bus Interface

The KN220 I/O module contains a Q22-bus interface, which is implemented by a single VLSI chip called the CQBIC (Figure 1–2). The CQBIC contains an interface between the CDAL bus and the Q22-bus and supports the following:

- A programmable mapping function (scatter-gather map) for translating 22-bit, Q22-bus addresses into 26-bit main memory addresses. This mapping function allows any page in the Q22-bus memory space to be mapped to any page in main memory.
- A direct mapping function for translating 26-bit main memory addresses into 22-bit, Q22-bus addresses. These main memory addresses are located in the local Q22-bus address space and the local Q22-bus I/O page.
- Masked and unmasked longword reads and writes from the CPU to the Q22-bus memory and I/O space and the Q22-bus interface registers.
  - Longword reads and writes of the local Q22-bus memory space are buffered and translated into two-word (16 bits), block mode transfers.
  - Longword reads and writes of the local Q22-bus I/O space are buffered and translated into two single-word transfers.
- Block-mode writes of up to 16 words from the Q22-bus to main memory.
- Transfers from the CPU to local Q22-bus memory space. The Q22-bus map translates the address back into main memory (local-miss, global-hit transactions).

1.2.8 CVAX Diagnostic Processor

The KN220 CPU diagnostic processor is located on the KN220 I/O module. The diagnostic processor is implemented by a single VLSI chip called the CVAX. The KN220 processor is used for the following:

- Power-up diagnostics
- Extended self-tests and scripts
- Booting and running MDM diagnostics

The CVAX supports the MicroVAX chip subset (plus six additional string instructions) of the VAX instruction set, data types, and full VAX memory management. The processor state is composed of 16 general purpose registers (GPRs), the processor status longword (PSL), and internal processor registers (IPRs).

The KN220 CPU diagnostic processor is capable of detecting the following types of error conditions during program execution:

- CDAL bus parity errors. MSER<6> (on a read) is set.
- Q22-bus NXM errors. DSER<7> is set.
- Q22-bus NO SACK errors. No indicator.
- Q22-bus NO GRANT errors. DSER<2> is set.
- Q22-bus device parity errors. DSER<5> is set.
- CDAL-bus timeout errors. DSER<4> (on DMA) is set.
- Main memory NXM errors. DSER<0> (on DMA) is set.
- Main memory correctable errors.
- Main memory uncorrectable errors. DSER<4> (on DMA) is set.

1.2.9 Network Interface

The KN220 I/O module contains a network interface implemented through the Ethernet controller and serial interface adapter chips. The H3602–AC CPU I/O panel interface allows you to connect the KN220 I/O module to either a ThinWire or standard Ethernet cable.

The second-generation Ethernet chip (SGEC) connects to the CP bus and the system through command and status registers (CSRs) and a system communication area in main memory. For data transfer, the SGEC contains a DMA controller that supports physical memory addresses.

The hardware address of the KN220 I/O module is determined during manufacture and is stored in the network interface station address (SAR) ROM.

1.2.10 DSSI Interface

The KN220 I/O module contains an SII chip and four 32K by 8-bit static RAMs that implement the Digital Storage System Interconnect (DSSI) bus interface. The DSSI interface allows the KN220 to transmit packets of data to, and receive packets of data from, up to seven RF-series integrated storage elements (ISEs). The DSSI bus improves system performance for two reasons:

- It is faster than the Q22-bus.
- It relieves the Q22-bus of disk traffic, thereby allowing more bandwidth for Q22-bus devices.

The physical characteristics of the DSSI bus are as follows:

- 4 Mbytes/second maximum bandwidth
- Distributed arbitration
- Synchronous operation
- Parity checking
- Six-meter total bus length (includes internal and external cabling)
- Maximum of eight nodes (KN220 I/O module counts as one)
- Eight data lines
- One parity line
- Eight control lines

See the following sections for more information about the DSSI bus and RF-series ISEs:

Section 2.6	Setting and changing DSSI node names, addresses, and unit numbers
Section 3.10.16	Console SET HOST command
Section 4.3	DSSI ISE acceptance testing
Section 4.7	RF-series resident diagnostics and local programs

1.2.11 SCSI Interface

The KN220 I/O module also contains a small computer storage interface (SCSI) bus that is implemented through the 53C94 chip and four 32K by 8-bit static RAMs.

The SCSI interface allows the KN220 I/O module to transmit packets of data to, and receive packets of data from, external SCSI devices. See Section 2.7 for more information.

1.3 H3602–AC CPU I/O Panel

The H3602–AC CPU I/O panel, shown in Figure 1–4, contains the following components:

- An operation switch
- A function switch
- Seven-segment LED for diagnostics
- A console serial line connector
- A console baud rate select switch

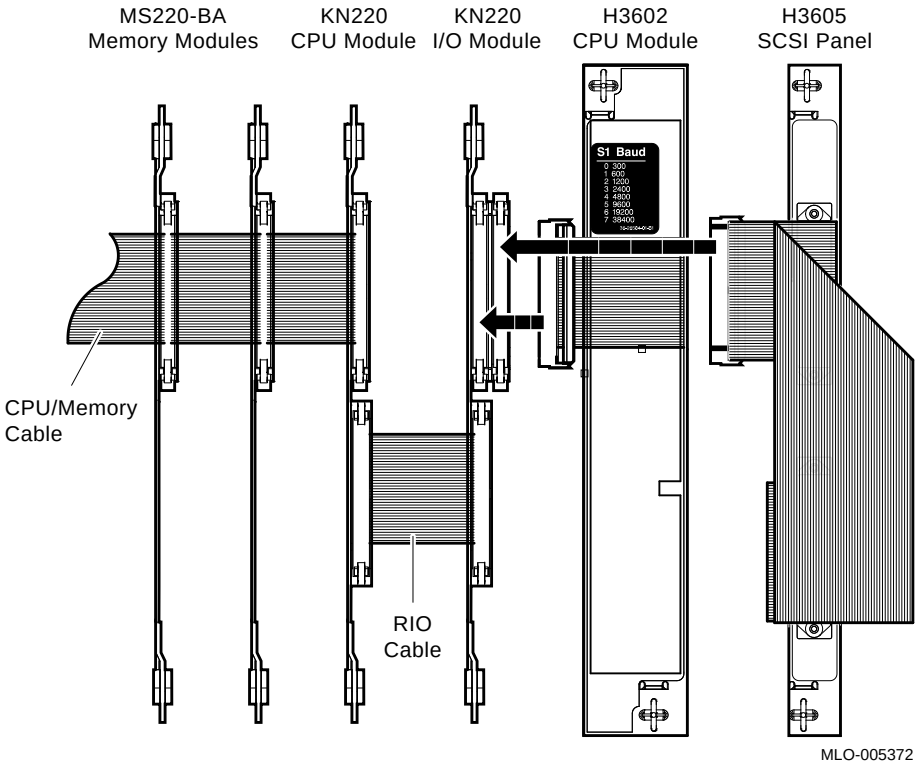
- A 15-conductor connector for standard Ethernet cable
- A BNC connector plug for a ThinWire Ethernet coaxial cable
- An Ethernet connector select switch
- Two LEDs that indicate the selected Ethernet connector (ThinWire or standard)
- One LED that indicates valid +12 Vdc for the selected Ethernet connector

The H3602–AC switches are read by the firmware when the processor starts. For this reason, if you change the baud rate on the H3602–AC, the new baud rate does not take effect until you power up or reset the system.

The hex LED display shows the individual test numbers during the power-up self-tests and bootstrap. Codes for the LED display are listed in Chapter 4, Table 4–6.

You connect the KN220 I/O module to the H3602–AC module cover through a ribbon cable and to the H3605 module cover through a ribbon cable. The H3602–AC connects to the bottom of the double-stack connector on the I/O module, and the H3605 connects to the top connector of the double stack on the I/O module.

Figure 1–4: H3602–AC CPU I/O Panel



DIFFERENCES FROM KN210 SYSTEMS: *The KN220 System has an interface module, M9715–AA, in slot 0, next to the power supply. KN210 systems had no M9715–AA module.*

For the KN220 system, use the switch settings that appear in Table 1–1.

The operation switch (three-position rotary) and the function switch (two-position slider) are described in Table 1–1. See Chapter 3 for a detailed description of power-up procedures and console commands.

Table 1–1: H3602–AC Operation and Function Switch Settings

Operation Switch Position	Function Switch Position	Action
{▫ Action mode	⊙	Test. The console serial line external loopback test is executed at the completion of the power-up self-tests. Use the H3103 MMJ loopback (12–25083–01). See Section 1.4 for a description of security features.
	◯	Query. The user is prompted for the language. Power-up self-tests are run.
→ Normal mode		Power-up self-tests run and, if successful, console enters normal mode (>>). If the *bootmode* environment variable is set to *a*, the R3000 processor attempts to locate a booting device specified through the bootpath environmental variable. If the *bootmode* environment variable is set to *d*, the R3000 processor enters normal mode without running any diagnostics, and prompts the user for commands.
	⊙	Breaks enabled.
	◯	Breaks disabled.
⊕ Maintenance mode		Power-up self-tests run and console enters maintenance mode (>>>).
	⊙	Breaks enabled.
	◯	Breaks disabled.

DIFFERENCES: *The KN220 system has no autoboot capability when the operation switch is set to the maintenance position. See Chapter 3, Section 3.6.1.1, for information on the ULTRIX–32 bootstrap procedure.*

Similar systems do have the autoboot capability when the operation switch is set to the maintenance position.

1.4 Using Console Security

DECsystem 5500 systems have a console security feature as part of the console firmware. The security feature allows you to secure the system. When the system is secure, unprivileged users (users who do not know the security password) are limited to just the **boot** command (with no arguments). Privileged users, knowing the security password, have access to all console commands.

1.4.1 Securing the System

To secure the system, use the **passwd** command as follows:

NOTE: *If unprivileged users are to be allowed to boot the system, the system manager should assign values to the bootpath and bootmode variables before securing the system. Once the system is secure, unprivileged users cannot issue the boot command with arguments or set the boot mode/boot path environment variables.*

1. At the console prompt (>>), enter the set password command,

   passwd -s.

2. At the "New password:" prompt, enter a password of 8–32 characters.

   You must retype the password for verification.

3. After the password has been accepted, enter the command

   passwd -u

   which causes the console module to display the unprivileged console prompt (s>). Unprivileged users are limited to the **boot** command with no arguments.

   The following example shows how to secure the system.

```
>> passwd -s
New password:
Retype new password:
New password accepted
>> passwd -u
Memory Size: 16777216 (0x1000000) bytes
Ethernet Address: 08-00-2b-12-81-22
S>
```

4. To maintain security, the Operation switch must remain set to Normal mode (indicated by the arrow) and the lower front door should be locked.

1.4.2 Privileged Users

By entering the security password, privileged users have access to all the console commands. In the example below, the **passwd** command is used to access the privileged console prompt (>>).

```
s> passwd
Password:
Password accepted.
Memory Size: 16777216 (0x1000000) bytes
Ethernet Address: 08-00-2b-12-81-22
>>
```

For a complete description of the **passwd** command, refer to Section 1.4.4.

1.4.3 Unsecuring the System

Privileged users can remove the security restrictions by using the clear password command, **passwd -c** at the console prompt (>>). For example,

```
>> passwd -c
>>
```

removes all security restrictions from the console firmware. The system is now unsecure.

If you forget the security password, you must use the following procedure to clear the password.

1. Set the Operation switch to the Maintenance mode setting (indicated by a T inside a circle).
2. Press the Restart button on the System Control Panel (SCP).
3. After the system completes self-tests, enter the maintenance command **unpriv** at the Maintenance mode prompt (>>>).
4. Reset the Operation switch to the Normal mode setting (indicated by and arrow).
5. Press the Restart button and wait for self-test to complete. You can now enter a new security password.

For a complete description of the **passwd** and **unpriv** commands, see Section 1.4.4 and Section 1.4.5.

1.4.4 The passwd Command

passwd [-(s | c | u])

The four variants of this command are used to control the console security feature. Using the console security feature, you can secure the system and limit unprivileged users (users who do not know the security password) to just the **boot** console command.

The use of the **passwd** command with flags is restricted to privileged mode (>>), while the use of **passwd** without flags is restricted to unprivileged mode (s>).

passwd —This command enables the console user to enter the security password to become privileged.
passwd s —This command is used to set a new security password. The security password can be from 8 to 32 characters long. This variant is available only in privileged mode (>>).
passwd c —This command removes security restrictions by clearing the security password.
passwd u —This command causes the console user to be unprivileged. The unprivileged console prompt (s>) is displayed.

1.4.5 The unpriv Command

unpriv

This Maintenance mode command clears the security password by setting it to zero. This command is used to unsecure or disable the console security feature if you forget the security password. To enter Maintenance mode, set the Operation switch to Maintenance mode (indicated by a T inside a circle). Press the Restart button on the SCP. After clearing the password, you must reset the Operation switch to Normal mode (indicated by an arrow) and press the Restart button again.

1.5 MS220 Memory

The MS220–AA (M7639–AA) is a 32-Mbyte memory module that provides memory for the KN220 system. The MS220–AA is a pseudo-intelligent memory array module.

The quad-height MS220–AA has a 100-ns, 78 bit-wide array (64-bit data and 14-bit ECC), implemented with 1-Mbit dynamic RAMs in dual in-line packages (SOIC).

The KN220 CPU module and up to four MS220–AA memory modules (128 Mbytes maximum) communicate through the MS220 memory interconnect. This interconnect uses the CD backplane interconnect for address and control signals and a 100-pin ribbon cable for data signals.

Ordering Information

MS220–AA	32-Mbyte module only (M7639–AA).

Diagnostic Support

MicroVAX Diagnostic Monitor	Release 133 (Version V4.4)
Self-test	KN220 self-test

Chapter 2

KN220 Configuration

2.1 Introduction

This chapter provides guidelines for changing the configuration of a KN220 system.

Before you change the system configuration, you must consider the following factors:

Module order in the backplane
Module configuration
Mass storage device configuration

If you are adding a device to a system, you must know the capacity of the system enclosure in the following areas:

Backplane
I/O panel
Power supply
Mass storage devices

2.2 General Module Order

The order of modules in the backplane depends on four factors:

- Relative use of devices in the system
- Expected performance of each device relative to other devices
- The ability of a device to tolerate delays between bus requests and bus grants (called delay tolerance or interrupt latency)
- The tendency of a device to prevent other devices farther from the CPU from accessing the bus

2.3 Module and Bulkhead Order for KN220 Systems

Observe the following rules about module order:

- Interface module (M9715) in slot 0.
- KN220 I/O module (M7638–AA) in slot 1.
- KN220 CPU module (M7637–AA) in slot 2.
- MS220–AA (M7639–AA) memory module in slot 3. Install any additional MS220–AA memory modules in slots 4, 5, and 6.
- Do not install dual-height modules in the CD rows.

Observe the following rules about bulkhead order:

- H3605 (70–27464–01), single-width bulkhead with two connector ports, is installed over slot 1 and connected to the I/O module in slot 1.
- H3602–AC (70–25775–03), double-width bulkhead, is installed over slots 2 and 3 and covers the CPU and the first memory module.

2.4 Memory Module Configuration

Memories have no registers in which the memory capacities can be configured. You need to run T 9A after installation of the memory if you want to view the board-specific memory configuration. (See Section 4.8.9.)

Test 9A allows you to enter the capacity of each individual memory module. The specifications you enter in T 9A stay in NVRAM as long as battery power is applied, or until you run T 9A and enter changes. See Section 4.3.

2.5 Q-Bus Module Configuration

The Q-bus passes through the backplane in a BA400-series enclosure.

Each Q-bus module in a system must use a unique device address and interrupt vector. The device address is also known as the control and status register (CSR) address. Most modules have switches or jumpers for setting the CSR address; most interrupt vector values are set by software. The value of a floating address depends on what other modules are housed in the system.

Set CSR addresses and interrupt vectors for a module as follows:

1. Determine the correct values for the module with the CONFIGURE command at the maintenance mode prompt (>>>). Type `CONFIGURE`, then `HELP` for the list of supported devices.

NOTE: *Some of the devices listed in the HELP display are not supported by the KN220 CPU module set.*

See the description of the CONFIGURE and HELP commands in Section 3.10.2 of how to obtain the correct CSR addresses and interrupt vectors, and Section 3.10.10 for a description of the help screen .

The LPV11–SA, which is the LPV11 version compatible with the BA400-series enclosures, has two sets of CSR address and interrupt vectors. To determine the correct values for an LPV11–SA, enter `LPV11,2` at the `Device, Number?` prompt for one LPV11–SA, or enter `LPV11,4` for two LPV11–SA modules.

2. See Appendix C for instructions on how to configure the KFQSA storage adapter. Appendix C explains how to do the following:
   - Set a four-position switchpack on the KFQSA
   - Program the CSR addresses for all the system's DSSI devices into the EEROM on the KFQSA
   - Reprogram the EEROM when you add DSSI devices
3. See *Microsystems Options* for switch and CSR and interrupt vector jumper settings for supported options.

2.6 DSSI Configuration

This section concerns the internal RF-series storage devices linked to the host CPU by means of the Digital Storage System Interconnect (DSSI). To link external RF-series storage devices to the host CPU by means of the KFQSA module, see Appendix C.

Whether internal or external to the host system, each storage device on a DSSI storage bus must have a unique DSSI node ID. The RF-series (ISE) receives its node ID from a plug on the operator control panel (OCP) on the front panel of each separate ISE. By convention, ISEs are mounted in the BA430 enclosure from right to left, as listed in Table 2–1.

Table 2–1: Conventional ISE and Tape Slots

Device	Position	Node ID[1]
TK70 is in rightmost slot	–	
First ISE	Right side	0
Second ISE	Center	1
Third ISE	Left side	2

[1]KN220 node ID = 7

If the cable between the ISE and the OCP is disconnected, the ISE reads the node ID from three DIP switches on its electronic controller module (ECM).

NOTE: *Pressing the system reset button on the front of the power supply has no effect on the ISEs. You must perform a power cycle.*

The node ID switches are located behind the 50-pin connector on the ECM. Switch 1 (the MSB) is nearest to the connector. Switch 3 (the LSB) is farthest from the connector. Refer to the RF71 section in *Microsystems Options* for an illustration and further information. Table 2–2 lists the switch settings for the eight possible node addresses.

NOTE: *The node ID plugs on the control panel of each ISE override the ISE DIP switches. It is good practice, however, to set the DIP switches to equal the node ID plugs.*

Table 2–2: ISE DIP Switch Settings

Node ID	S1	S2	S3
0	Down	Down	Down
1	Down	Down	Up
2	Down	Up	Down
3	Down	Up	Up
4	Up	Down	Down
5	Up	Down	Up
6	Up	Up	Down
7	Up	Up	Up

The maintenance mode SET HOST/DUP command creates ISE device names according to the following scheme:

`nodename $ DIA unit number.` For example, `KATH$DIA3`

You can use the device name for booting MDM, as follows:

```
>>> BOOT KATH$DIA3
```

You can access local programs in the ISE through the MicroVAX Diagnostic Monitor (MDM) or through the maintenance mode SET HOST/DUP command. This command creates a virtual connection to the storage device and the designated local program, using the Diagnostic and Utilities Protocol (DUP) standard dialog. Section 3.10.16 describes the SET HOST /DUP command.

2.6.1 Changing the Node Name

Each ISE has a node name that is maintained in the EEPROM on the controller module. This node name is determined in manufacturing from an algorithm based on the drive serial number. You can change the node name of the ISE to something more meaningful by following the procedure in Example 2–1. In the example, the node name for the RF71 ISE at DSSI node address 1 is changed from R3YBNE to DATADISK.

See Section 4.7.5 for further information about the PARAMS local program.

Example 2–1: Changing a DSSI Node Name

```
>>> SHOW DSSI
DSSI Node 0 (MDC)
-rf(0,0,*) (RF71)

DSSI Node 1 (R3YBNE)       !The node name for this drive will be
-rf(1,1,*) (RF71)    !changed from R3YBNE to DATADISK.

DSSI Node 7 (*)
>>>
>>> SET HOST/DUP/DSSI 1 PARAMS
Starting DUP server...
Copyright  1988  Digital Equipment Corporation

PARAMS> SHOW NODENAME

Parameter     Current           Default        Type     Radix
--------- ---------------  ---------------  --------  -----
NODENAME      R3YBNE             RF71         String    Ascii  B
```

Example 2–1 Cont'd on next page

Example 2–1 (Cont.): Changing a DSSI Node Name

```
PARAMS> SET NODENAME DATADISK

PARAMS> WRITE                   !This command writes the change
                                !to EEPROM.
Changes require controller initialization, ok? [Y/(N)] y

Stopping DUP server...
>>> SHOW DSSI
DSSI Node 0 (MDC)
-rf(0,0,*) (RF71)

DSSI Node 1 (DATADISK)          !The node name has changed from
-rf(1,1,*) (RF71)         !R3YBNE to DATADISK.

DSSI Node 7 (*)
```

2.6.2 Changing the Unit Number

By default, the ISE's unit number is the same value as the DSSI node address for that drive. This occurs whether the DSSI node address is determined from the bus ID plugs or from the three DIP switches on the ISE controller module.

ISEs conform to the Digital Storage Architecture (DSA). Each drive can be assigned a unit number from 0 to 16,383 (decimal). The unit number need not be the same as the DSSI node address.

Example 2–2 shows how to change the unit number of an ISE. This example changes the unit number for the RF71 at DSSI node address 1 from 1 to 14 (decimal). You must change two parameters: UNITNUM and FORCEUNI. Changing these parameters overrides the default, which assigns the unit number the same value as the node address.

See Section 4.7.5 for further information about the PARAMS local program.

Example 2–2: Changing a DSSI Unit Number

```
>>> SHOW DSSI
DSSI Node 0 (MDC)
-rf(0,0,*) (RF71)

DSSI Node 1 (R3QJNE)     !The unit number for this drive will be
-rf(1,1,*) (RF71)  !changed from 1 to 14

DSSI Node 7 (*)
>>>
>>> SET HOST/DUP/DSSI 1
Starting DUP server...
Copyright  1988  Digital Equipment Corporation
DRVEXR V1.0  D  2-JUN-1989 15:33:06
DRVTST V1.0  D  2-JUN-1989 15:33:06
HISTRY V1.0  D  2-JUN-1989 15:33:06
ERASE  V1.0  D  2-JUN-1989 15:33:06
PARAMS V1.0  D  2-JUN-1989 15:33:06
DIRECT V1.0  D  2-JUN-1989 15:33:06
End of directory

Task Name? PARAMS
Copyright 1988  Digital Equipment Corporation

PARAMS> SHOW UNITNUM

Parameter     Current           Default        Type     Radix
--------- ----------------  ----------------  --------  -----
UNITNUM         1                 1           Word     Dec   U

PARAMS> SHOW FORCEUNI

Parameter     Current           Default        Type     Radix
--------- ----------------  ----------------  --------  -----
FORCEUNI         1                 1          Boolean   0/1   U

PARAMS> SET UNITNUM 14

PARAMS> SET FORCEUNI 0

PARAMS> WRITE            !This command writes the changes
                         !to the EEPROM.

PARAMS> EX
Exiting...

Task Name?
```

Example 2–2 Cont'd on next page

Example 2–2 (Cont.): Changing a DSSI Unit Number

```
Stopping DUP server...
>>>
>>> SHOW DSSI
DSSI Node 0 (MDC)
-rf(0,0,*) (RF71)

DSSI Node 1 (R3QJNE)         !The unit number has changed
-rf(1,14,*) (RF71)    !and the node ID remains at 1.

DSSI Node 7 (*)
```

2.6.3 DSSI Cabling

Each ISE has a connector that connects to the backplane. The ISEs are connected to the internal DSSI bus by means of the backplane.

A 10-conductor cable connects each ISE to the OCP on its front panel.

A 50-conductor round cable is routed from the external DSSI connector, lower left of the front of the unit, to the backplane.

2.6.4 DSSI Bus Termination and Length

The DSSI bus has a maximum length of 6 m (19.8 ft), including internal and external cabling. The DSSI bus must be terminated at both ends. The KN220 I/O module terminates the DSSI bus at one end. A terminator on the left side of the media faceplate terminates the bus at the other end. This terminator can be removed if you need to expand the bus.

2.7 SCSI Configuration and Cabling

This subsection describes the Small Computer System Interface (SCSI) device configuration and SCSI bus cabling in a DECsystem 5500 system.

CAUTION: *If you have a TLZ04-GA, it is supplied with two cables. One cable is the BCO6P-06, which is a 1.83-m (6-foot) cable and should be used. The other cable is shorter in length and should not be used.*

2.7.1 Adding External Devices

CAUTION: *Before you proceed with step one below read the following list of general configuration rules.*

1. All external SCSI cables used with the DECsystem 5500 must be BC06P cables.
2. A maximum of two BC06P cables may be used in any one bus. BC06P cables are available in lengths of .76 m (2.5 ft), 1.83 m (6 ft), and 2.74 m (9 ft).
   - An exception: If a 2.74-m (9-foot) BC06P cable is used in the system, only one cable is allowed.
   - The .76 m (2.5 ft) cable that is used to connect the adapter to the storage shelf in the Q-bus-based BA430 enclosure is an external cable that must be counted in calculating the maximum length.
3. There are no restrictions on the number of devices or the type of devices under these rules as long as the devices are within the standard limits of the SCSI bus. For example, only seven devices maximum per bus are allowed.
4. Care must be taken not to exceed the industry standard maximum bus length of 6 meters. If the above steps are followed, there should be no problem. The BA430 enclosure's internal SCSI bus length is 1.2 m (47 in).
5. If there are tabletop drives connected to the DECsystem 5500, the internal cable length must be taken into account.

6. To connect the BA430 external drive to the SCSI bus, attach the 1.83 m (6 ft) SCSI cable (17–02659–02) to the bottom connector of the CPU module. Connect the terminator (12–30552–01) to the top connector. Pull the bail latches toward the cable and terminator to hold them in place.
7. Connect the other end of the 1.83-m (6-foot) SCSI cable to the top connector of the tabletop drive.
8. Connect the terminator to the lower connector of the drive.

   NOTE: *The two following steps are for internal drives only.*

9. To connect the BA430 internal drive to the SCSI bus, attach the .76 m (2.5 ft) SCSI cable (17–02659–03) to the top connector of the CPU module. Connect the terminator (12–30552–01) to the lower connector. Pull the bail latches toward the cable and terminator to hold them in place.
10. Connect the other end of the .76 m (2.5 ft) SCSI cable to the connector. (When connecting more than one drive refer to Chapter 6 for information on connecting multiple drives.)

2.7.2 Connecting Multiple Drives

Multiple drives are connected to the CPU module by a daisy-chain cabling configuration. TLZ04, RZ5x, or RRD40 drives may be daisy chained in any order. Each drive then must be assigned a unique node ID number. (See Section 6.3.)

Because the embedded drive has no power supply, it is physically smaller than the tabletop model, which does have a power supply.

2.7.3 Connecting Tabletop Drives

Connection of the tabletop drives proceeds as follows:

1. Remove the terminator from the lower connector of the first drive and replace it with 1.83 m (6 ft) SCSI cable.
2. Connect the other end of the cable to the top connector of the next drive.
3. Connect the SCSI terminator to the bottom connector. (The SCSI bus can accommodate up to three drives, with the last drive being terminated on the bottom connector.)

2.7.4 Connecting Internal Drive to Tabletop Drive

Because the internal drive is embedded in the system, the cable connections for additional drives begin at the CPU module. All additional drives must be tabletop models.

1. Remove the terminator from the bottom connector of the CPU module and replace it with a 1.83-m (6-foot) SCSI cable.
2. Connect the other end of the cable into the top connector of the next drive.
3. Connect the terminator to the bottom connector. (The CPU module can accommodate up to three drives, with the last terminated on the bottom connector.)

2.7.5 Assigning the Node ID

You must assign to each drive and to the CPU module a unique SCSI node ID.

NOTE: *The higher the node ID address number selected, the higher the bus priority.*

2.7.6 SCSI Interface ID Switches

Before proceeding, locate the SCSI switches on the back of your drive. The four numbered DIP switches set the SCSI address ID number that the drive will respond to in the system. The drive must be given a SCSI address ID number by setting the switch. See Tables 2–3 and 2–4.

1. Power down all drives before assigning a SCSI ID number.
2. Determine the SCSI address ID number that your drive will be assigned. The address ID number can be any number from 0 to 7. The default ID for the CPU module is 7.
3. Set the switches to the correct address ID number. (The TLZ04 and RRD40 drive switches are marked differently.)
4. After you complete the drive switch settings, use MDM to test your SCSI bus interconnects.

CAUTION: *Use a ballpoint pen or pointed object to set the switches. Never use a pencil to set the switches. The graphite used in pencils can damage the switches.*

Table 2–3: TLZ04 SCSI Address Node ID Number Settings

ID	Switch Settings		
No.	**SW4**	**SW2**	**SW1**
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Key to switch settings

1 = down
0 = up
Switch P is not used

Table 2–4: RRD40 SCSI Address ID Number Settings

ID	Switch Settings			
No.	**1**	**2**	**3**	**4**
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	0

Key to Switch Settings

1 = ON = UP
0 = OFF = DOWN
Switch 4 must be set to 0

Table 2–5 describes the SCSI cables.

Table 2–5: SCSI Cables

Cable Description	Order Number
2.74-m (9-foot) SCSI 50-pin cable	17–02659–01
1.83-m (6-foot) SCSI 50-pin cable	17–02659–02
.76-m (2.5-foot) SCSI 50-pin cable	17–02659–03
.31-m (12-inch) 50-pin ribbon cable	70–22834–03
.53-m (21-inch) 50-pin ribbon cable	70–22834–02
.92-m (36-inch) 50-pin ribbon cable	70–22834–01
SCSI terminator	12–30552–01

Chapter 3
KN220 Firmware

3.1 Introduction

This chapter describes the KN220 firmware.

The KN220 CPU-I/O module set contains a maximum of 384 Kbytes of EPROM for the firmware. This firmware is located as follows:

- R3000: two 128-Kbyte EPROMs on the KN220 CPU module (256 Kbytes)
- CVAX: one 128-Kbyte EPROM on the KN220 I/O module

The EPROMs are arranged as 32-bit words and are located at the following R3000 and CVAX restart locations:

- R3000: physical address 1FC00000
- CVAX: physical address 20040000

NOTE: *In the KN220 system, the firmware resides in three EPROMs: two for R3000 and one for CVAX.*

The CVAX and R3000 firmware contain the major functional blocks of code listed in Table 3–1.

Table 3–1: KN220 Firmware Code

CVAX Firmware Code	R3000 Firmware Code
Halt entry, dispatch, and exit	Primary and secondary bootstrap (ULTRIX–32)
Primary and secondary bootstrap (MDM)	Console program (normal mode commands)
Console program (maintenance mode commands)	ROM-based diagnostics
System restart	
ROM-based diagnostics	

The firmware uses the KN220 IO module LEDs and the console terminal to communicate diagnostic progress, display error conditions, and indicate the current mode of operation.

This chapter discusses the following:

- CVAX halt procedures
- Power-up procedures
- Bootstrap procedures for ULTRIX–32 and MDM
- R3000 console program and normal mode commands
- CVAX console program and maintenance mode commands

The CVAX/R3000 ROM-based diagnostics are described in Chapter 4.

3.2 KN220 Firmware Features

The KN220 CVAX and R3000 firmware provide the following features:

- Automatic or manual bootstrap of customer application images at power-up, reset, or conditionally after processor halts.
- A CVAX interactive command language (maintenance mode) that allows you to examine and alter the state of the processor.
- An R3000 interactive command language (normal mode) that allows you to make use of environment variables to pass information to the ULTRIX–32 operating system.
- Diagnostics and console utilities that test components on the KN220 and memory modules and test devices on the DSSI bus, SCSI bus, and Ethernet.
- LEDs and displays on the KN220 IO module and console terminal that display diagnostic progress and error reports.
- Multilingual support. In maintenance mode only, the firmware can issue system messages in several languages.

The processor must be functioning at a level capable of executing instructions from the maintenance program ROM for the maintenance program to operate.

3.3 CVAX Halt Entry and Dispatch Code

Unless a halt occurs when the R3000 is running, the CVAX processor enters the halt entry code at physical address 20040000 whenever the CVAX receives a halt signal. The halt entry code saves machine state, then transfers control to the firmware halt dispatcher.

After a halt, the halt entry code saves the current LED code, then writes an E to the LEDs. An E on the LEDs indicates that at least several instructions have been successfully executed, although if the CPU is functioning properly, the E occurs too quickly to be seen. The halt entry code saves the following registers. The console intercepts any direct reference to these registers and redirects it to the saved copies.

R0–R15	General purpose registers
PR$_SAVPSL	Saved processor status longword register
PR$_SCBB	System control block base register
DLEDR	Diagnostic LED register
SSCCR	SSC configuration register
ADxMCH	SSC address match registers
ADxMSK	SSC address mask registers

The halt entry code unconditionally sets the following registers to fixed values on any halt to ensure that the console itself can run:

SSCCR	SSC configuration register
ADxMCH	SSC address match registers
ADxMSK	SSC address mask registers
CBTCR	CDAL bus timeout control register
TIVRx	SSC timer interrupt vector registers

The console command interpreter does not modify actual processor registers. Instead, it saves the processor registers in console memory when it enters the halt entry code, then directs all references to the processor registers to the corresponding saved values, not to the registers themselves.

When the processor reenters normal mode, the saved registers are restored and any changes become operative only then. References to processor memory are handled normally. The binary load and unload command (X, Section 3.10.21) cannot reference the console memory pages.

After saving the registers, the halt entry code transfers control to the halt dispatch code. The halt dispatch code determines the cause of the halt by reading the halt field (PR$_SAVPSL <13:08>), the processor halt action field (PR$_CPMBX <01:00>), and the break enable switch on the H3602–AC panel. Table 3–2 lists the actions taken, by sequence. If an action fails, the next action is taken. There is no autoboot on the CVAX side.

Table 3–2: Actions Taken on a Halt

Breaks Enabled on H3602–AC	Power-Up Halt[1]	Halt Action[2]	Action
T[3]	T	X	Diagnostics, halt
T	F	0	Halt
F	T	X	Diagnostics, halt
F	F	0	Restart, halt
X	F	1	Restart, halt
X	F	2	Halt
X	F	3	Halt

[1]Power-up halt: PR$_SAVPSL<13:08>=3

[2]Halt action: PR$_CPMBX<01:00>

[3]T = condition is true, F = condition is false, X = does not matter

3.4 External Halts

Several conditions can trigger an external halt, and different actions are taken depending on the condition. The conditions are listed below.

- The function switch is set to enable, and you press Break on the system console terminal.
- Assertion of the BHALT line on the Q22-bus.
- Deassertion of DCOK. A halt is delivered if the processor is not running out of halt-protected space, and the BHALT ENB bit is set. The system restart switch deasserts DCOK. DCOK may also be deasserted by the DELQA sanity timer or any other Q22-bus module that chooses to implement the Q22-bus restart/reboot protocol.

When in maintenance mode, the processor halts on the deassertion of DCOK. If halts are enabled, the firmware enters maintenance mode. If halts are disabled, the firmware takes the action dictated by the halt action field.

The action taken by the halt dispatch code on a console Break or Q22-bus BHALT is the same: the firmware enters maintenance mode if halts are enabled.

The halt dispatch code distinguishes between DCOK deasserted and BHALT by assuming that BHALT must be asserted for at least 10 milliseconds, and that DCOK is deasserted for at most 9 microseconds. To determine if the BHALT line is asserted, the firmware steps out into halt-unprotected space after 9 milliseconds. If the processor halts again,

the firmware concludes that the halt was caused by the BHALT and not by the deassertion of DCOK.

3.5 Power-Up

On power-up, the firmware performs several unique actions. It runs the initial power-up test (IPT), locates and identifies the console device, performs a language inquiry, and runs the remaining diagnostics.

The IPT waits for power to stabilize by monitoring SCR<5>(POK). Once power is stable, the IPT verifies that the console private SSC NVRAM (System support chip nonvolatile RAM) is valid (backup battery is charged). If it is invalid or zero (battery is discharged), the IPT tests and initializes the SSC NVRAM.

NOTE: *The CPU board also contains NVRAM and a battery-backup system. For details of CPU verification, see Section D.3.3.*
After the battery check, the firmware tries to determine the type of terminal attached to the console serial line. If the terminal is a known type, it is treated as the system console.

Once a console device has been identified, the firmware displays the KN220 banner message:

```
KN220-A Vn.n
```

The banner message contains the processor name (KN220–A) and the version of the firmware (Vn.n), where n.n denotes the major and minor release numbers.

Power-up actions differ, depending on the state of the operation switch located on the H3602–AC CPU I/O panel, shown previously in Figure 1–4.

3.5.1 Power-Up Sequence: Operation Switch Set to Normal

If you set the operation switch on the H3602–AC CPU I/O panel to the normal position (→), the power sequence is as follows:

1. CVAX powers up (begins execution at a location pointed to by physical address 20040000).

   In addition, the console displays the language selection menu if the operation switch is set to the normal position (→) and the contents of SSC NVRAM are invalid. The console uses the saved console language if the operation switch is set to the normal position and the contents of SSC NVRAM are valid.

2. CVAX runs self-test diagnostics.

3. CVAX executes an EXIT command (40 000 000 is written into the SPR).
4. CVAX hangs on a DMA grant.
5. R3000 begins execution at address 1FC00000.
   - If the *bootmode* environmental variable is set to *a*, the R3000 attempts to autoboot. If the *bootpath* environment variable is valid, the autoboot succeeds. See Section 3.7.2 for a description of environmental variables.
   - If the *bootmode* environment variable is not initialized (*), the R3000 prompts you for a command at the normal mode prompt (>>).

     If you enter `maint` at the >> prompt, 80 000 000 is written in the SPR, the R3000 hangs on a RDBUSY stall, and the CVAX resumes execution (the >>> prompt is displayed).

3.5.2 Power-Up Sequence: Operation Switch Set to Maintenance

If you set the operation switch on the H3602–AC CPU I/O panel to the maintenance position (⊕), the power-up sequence is as follows:

1. CVAX powers up (begins execution at a location pointed to by physical address 20040000).

   In addition, the console displays the language selection menu if the operation switch is set to the normal position (→) and the contents of SSC NVRAM are invalid. The console uses the saved console language if the operation switch is set to the normal position and the contents of SSC NVRAM are valid.
2. CVAX runs self-test diagnostics.
3. CVAX enters maintenance mode and prompts you for commands at the maintenance mode prompt (>>>).
   - If you enter `EXIT` at the >>> prompt, the CVAX hangs on a DMA grant and the R3000 begins execution (the >> prompt is displayed).

3.5.3 Operation Switch Set to Action: Loopback Tests

You can verify the connection between the KN220 CPU module set and the console terminal, as follows:

1. Set the operation switch to the action position ({◦).
2. Set the function switch to enable (⊙).

3. To test the console terminal, connect the H3103 loopback connector to the H3602–AC console connector. (You must install the loopback connector to run the test.)
4. To test the console cable, connect the H8572 connector at the end of the console cable and connect the H3103 to the H8572.

During the test, the firmware toggles between the active and passive states. During the active state (3 seconds), the LED is set to 7. The firmware reads the baud rate and operation switch setting, then transmits and receives a character sequence.

During the passive state (7 seconds), the LED is set to 4.

If at any time the firmware detects an error (parity, framing, overflow, or no characters), the display hangs at 7. If you move the operation switch from the action position, the firmware continues as on a normal power-up.

3.5.4 Operation Switch Set to Action: Language Query

If you set the operation switch to the action position ({∘) and the function switch to query (◌̇), or if the firmware detects that the contents of NVRAM are invalid, the firmware prompts you for the language to be used for displaying the following system messages:

```
Failure.
Performing normal system tests.
Tests completed.
Normal operation not possible.
```

The selection menu for the language and keyboard type is shown in Example 3–1. If no response is received within 30 seconds, the firmware defaults to English.

Example 3–1: Language Selection Menu

```
KN220-A Vn.n
 1) Dansk
 2) Deutsch (Deutschland/Österreich)
 3) Deutsch (Schweiz)
 4) English (United Kingdom)
 5) English (United States/Canada)
 6) Español
 7) Français (Canada)
 8) Français (France/Belgique)
 9) Français (Suisse)
10) Italiano
11) Nederlands
12) Norsk
13) Português
14) Suomi
15) Svenska
 (1..15):
```

3.6 Bootstrap

Bootstrapping is the process of loading and transferring control to an operating system. The KN220 bootstrap support is as follows:

- CVAX (maintenance mode) supports the bootstrap of MDM diagnostics.
- R3000 (normal mode) supports the bootstrap of ULTRIX–32 as well as any user application image that conforms to the boot formats described in this section.

NOTE: *The KN220 system contains two console programs: maintenance mode (CVAX) and normal mode (R3000). See Section 3.8 for normal mode commands that you type at the >> prompt. Normal mode commands are case sensitive. See Section 3.10 for maintenance mode commands that you type at the >>> prompt.*

A KN220 bootstrap occurs under the following conditions:

- For MDM, enter `BOOT` at the maintenance mode prompt (`>>>`), MDM only.
- For ULTRIX, enter

  `>>`**`boot`**

  in lowercase letters at the normal mode prompt.

3.6.1 ULTRIX–32 Bootstrap

The ULTRIX–32 operating system is booted in the Normal operating mode under one of the following conditions:

1. Power is on; environmental variables are set as shown in Section 3.6.1.2. According to the variables set, the boot is either automatic or manual.
2. Power is on; no environmental variables are set. The commands are as follows, with the bootpath to the boot device is included in the command examples as shown.

   ```
   >> boot -f rf(0,0,0)vmunix
   ```

   ```
   >> boot -f rz(0,0,0)vmunix
   ```

3. The operating system initiates a reboot operation.

You can use one of the following ports as the ULTRIX–32 boot device:

- KN220 I/O module Ethernet controller
- KN220 I/O module DSSI controller
- KN220 I/O module SCSI controller
- KN220 Q22-bus MSCP or TMSCP controller

Table 3–3 lists the supported ULTRIX–32 boot devices. The table correlates the boot device names expected in a boot command with the corresponding supported devices.

Boot device names consist of a two- or three-letter device code (letters a through z).

Table 3–3: ULTRIX–32 Supported Boot Devices

Device Type	Protocol	Number of Units Per Storage Interface	Device Name
RA-series fixed disk	MSCP	4	ra
RF-series ISE	DSSI	7	rf
RZ-series fixed disk	SCSI	7	rz
TZ-series tape drive	SCSI	7	tz
TQK-series tape drive	TMSCP	1	tm
Ethernet adapter	MOP	1	mop
Ethernet adapter	TFTP	1	tftp

3.6.1.1 ULTRIX–32 Bootstrap Procedure

Boot the ULTRIX–32 operating system at the normal mode prompt (>>), using the commands explained in Section 3.7. Normal mode commands are case sensitive (see Section 3.8).

Boot the system as follows:

3.6.1.2 On Installation

1. On the H3602–AC CPU I/O panel, set the operation switch to the normal position (→).
2. Set the on/off power switch to 1 (on).
3. After the system has completed the power-up self-tests successfully, the normal mode prompt is displayed (>>).

   To define the bootpath, define the environmental variable for the desired boot device and boot mode, using lowercase letters, then boot the system to save the desired boot device. In Example 3–2, the boot device is an ISE at node 0.

Example 3–2: Command Procedure to Boot on Installation

```
>> setenv bootpath rf(0,0,0)vmunix   !For file name, type
                                     !vmunix or
                                     !application file name.

>> boot
```

To make the boot automatic, set the following environmental variable:

```
>>setenv bootmode a     !Boot will be automatic.
```

On Power-Up of Existing System

1. On the H3602–AC CPU I/O panel, set the operation switch to the normal position (→).
2. Set the on/off power switch to on (1).
3. After the system has completed the power-up self-tests successfully, the R3000 processor attempts to boot the operating system through the previously defined boot device.

3.6.2 MDM Bootstrap

When you enter maintenance mode and type `BOOT MUa0:` at the >>> prompt, the CVAX processor boots the MDM operating system from a TK tape cartridge.

The following example is for a TQKxx subsystem.

Before dispatching to the primary CVAX bootstrap (VMB), the KN220 CVAX processor firmware initializes the system to a known state, as follows:

1. Checks CPMBX<2>(RIP), bootstrap in progress. If it is set, bootstrap fails and the console displays the message `Failure.` in the selected console language.
2. Validates the boot device name. If none exists, supplies a list of available devices and issues a boot device prompt. If you do not specify a device within 30 seconds, uses EZA0.
3. Writes a form of this boot request, including active boot flags and boot device (BOOT/R5:0 EZA0, for example), to the console terminal.
4. Sets CPMBX<2>(BIP).

5. Initializes the Q22-bus scatter-gather map.
6. Validates the PFN bitmap. If invalid, rebuilds it.
7. Searches for a 128-Kbyte contiguous block of good memory as defined by the PFN bitmap. If 128 Kbytes cannot be found, the bootstrap fails.
8. Initializes the general purpose registers:

R0	Address of descriptor of the boot device name or 0 if none specified
R2	Length of PFN bitmap in bytes
R3	Address of PFN bitmap
R4	Time-of-day of bootstrap from PR$_TODR
R5	Boot flags
R10	Halt PC value
R11	Halt PSL value (without halt code and mapenable)
AP	Halt code
SP	Base of 128-Kbyte good memory block plus 512
PC	Base of 128-Kbyte good memory block plus 512
R1, R6, R7, R8, R9, FP	0

9. Copies the virtual memory bootstrap (VMB) image from EPROM to local memory, beginning at the base of the 128 Kbytes of good memory block plus 512.
10. Exits from the firmware to VMB residing in memory.

Virtual Memory Bootstrap (VMB) is the primary MDM bootstrap. The KN220 VMB resides in the CVAX firmware and is copied into main memory before control is transferred to it. VMB then loads the secondary bootstrap image and transfers control to it.

Table 3–4 lists the supported R5 boot flags.

Table 3–4: VMB Boot Flags

Bit	Name	Description
0	RPB$V_CONV	Conversational boot. At various points in the system boot procedure, the bootstrap code solicits parameters and other input from the console terminal.
1	RPB$V_DEBUG	Debug. If this flag is set, the code for the XDELTA debugger is mapped into the system page tables of the running system.
2	RPB$V_INIBPT	Initial breakpoint. If RPB$V_DEBUG is set, the VMS operating system executes a BPT instruction in module INIT immediately after enabling mapping.
3	RPB$V_BBLOCK	Secondary bootstrap from bootblock. When set, VMB reads logical block number 0 of the boot device and tests it for conformance with the bootblock format. If in conformance, the block is executed to continue the bootstrap. No attempt is made to perform a Files–11 bootstrap.
4	RPB$V_DIAG	Diagnostic bootstrap. When set, the load image requested over the network is [SYS0.SYSMAINT]DIAGBOOT.EXE.
5	RPB$V_BOOBPT	Bootstrap breakpoint. When set, a breakpoint instruction is executed in VMB and control is transferred to XDELTA before booting.
6	RPB$V_HEADER	Image header. When set, VMB transfers control to the address specified by the file's image header. When not set, VMB transfers control to the first location of the load image.
8	RPB$V_SOLICT	File name solicit. When set, VMB prompts the operator for the name of the application image file. The maximum file specification size is 17 characters.
9	RPB$V_HALT	Halt before transfer. When set, VMB halts before transferring control to the application image.
31:28	RPB$V_TOPSYS	This field can be any value from 0 through F. This flag changes the top-level directory name for system disks with multiple operating systems. For example, if TOPSYS is 1, the top-level directory name is [SYS1...].

Table 3–5 lists the supported MDM boot devices.

Table 3–5: Supported MDM Boot Devices

	Boot Name	Controller Type	Device Type(s)
Tape			
	MUcn	TQK70 MSCP	TK70
	DIAn	On-board SCSI	SCSI CDROM
Network			
	EZA0	On-board Ethernet	–
	XQcn	DELQA	–
		DESQA	–

3.6.3 MDM Restart

An MDM restart is the process of bringing up the MDM operating system from a known initialization state following a processor halt.

A restart occurs under the conditions listed in Table 3–2, earlier in this chapter.

To restart MDM, the firmware searches system memory for the Restart Parameter Block (RPB), a data structure constructed for this purpose by VMB. If the firmware finds a valid RPB, it passes control to the operating system at an address specified in the RPB.

The firmware keeps a RIP (restart-in-progress) flag in CPMBX, which it uses to avoid repeated attempts to restart a failing operating system. The operating system maintains an additional RIP flag in the RPB. The RPB is a page-aligned control block that can be identified by its signature in the first three longwords:

+00 (first longword) = physical address of the RPB
+04 (second longword) = physical address of the restart routine
+08 (third longword) = checksum of first 31 longwords of restart routine

The firmware finds a valid RPB as follows:

1. Searches for a page of memory that contains its address in the first longword. If none is found, the search for a valid RPB has failed.
2. Reads the second longword in the page (the physical address of the restart routine). If it is not a valid physical address, or if it is zero, returns to step 1. The check for zero is necessary to ensure that a page of zeros does not pass the test for a valid RPB.
3. Calculates the 32-bit two's-complement sum (ignoring overflows) of the first 31 longwords of the restart routine. If the sum does not match the third longword of the RPB, returns to step 1.
4. If the sum matches, a valid RPB has been found.

3.7 Normal Mode Overview

When the KN220 is in normal mode, the console reads and interprets commands received on the console terminal at the normal mode prompt (>>).

This R3000 interactive command language allows you to make use of environment variables to pass information to the ULTRIX–32 operating system.

You can use normal mode commands to boot the ULTRIX–32 operating system, set up automatic booting, and deposit or examine I/O address space and memory.

3.7.1 Control Characters in Normal Mode

Table 3–6 lists the characters that have special meaning in normal mode.

Table 3–6: Normal-Mode Control Characters

Character	Action
<CR>	Ends a command line. Command characters are buffered until you press Return.
X (delete or backspace)	Deletes the previously typed character. If you define the console terminal as hard copy (environmental variable *term* set to hardcopy), the deleted text is displayed surrounded by backslashes. If the console terminal is a CRT (environmental variable *term* set to crt), each delete is displayed with the sequence <BS><SP><BS>. Deletes received are ignored when there are no characters to be deleted.
Ctrl/C	Causes the console to abort the processing of a command.
Ctrl/O	Causes console output to be discarded until you enter the next Ctrl/O or until the next console prompt or error message is issued. Ctrl/O is also canceled when you enter Ctrl/C.
Ctrl/Q	Resumes console output that was suspended when you entered Ctrl/S.
Ctrl/R	Causes the current command line to be displayed without any deleted characters.
Ctrl/S	Suspends output on the console terminal until you enterCtrl/Q.
Ctrl/U	Discards all characters accumulated for the current line.
Ctrl/V	Suppresses any special meaning associated with the next character.

3.7.2 Environment Variables in Normal Mode

The KN220 console makes use of environmental variables to pass information to the operating system.

There are three types of variables:

- Volatile (lost when power resumes)
- Nonvolatile (maintained after power resumes)
- Fixed (rebuilt when power is turned on)

You can define additional environmental variables, but those you define will be lost when power is removed.

Table 3–7 lists the default environmental variables.

Table 3–7: Environmental Variables

Variable	Type	Description
baud	Fixed	The baud rate of the console terminal line is determined by the baud rate select switch inside the H3602–AC CPU I/O panel. The factory setting is 9600. Allowed values are 300, 600, 2400, 4800, 9600, 19,200, and 38,400.
bitmap	Fixed	Indicates the address of the memory bitmap. The bitmap keeps track of good and bad memory pages. Each bit corresponds to one page in memory; 1 indicates the page is good and 0 indicates the page is bad.
bitmaplen	Fixed	Indicates the length of the memory bitmap in words.
bootmode	Nonvolatile	Determines what programs run when the system is turned on or reset. Use one of the following codes: a Autoboots the operating system, using the *bootpath* variable d Halts the system with no diagnostics run, and goes to Normal mode prompt
bootpath	Nonvolatile	Indicates the default bootpath. The system uses this variable when you type the auto command. An example of a bootpath definition is: rf(0,0,0)vmunix.
memdescriptor	Volatile	Set by test 9A in Maintenance mode.

Table 3–7 (Cont.): Environmental Variables

Variable	Type	Description
console	Fixed	The system always selects TTY(0) as the console device.
osconsole	Fixed	The system always selects TTY(0) as the console device.
scsiid[0-7]	Fixed	The SCSI controller number. Defaults 7. The number in the variable is the SCSI controller number.
systype	Fixed	Contains information used to identify the processor. Bits 24:31 contain the CPU type. Bits 16:23 contain the system type (11 for KN220). Bits 8:15 contain the firmware revision level. Bits 0:7 contain the hardware version level.

3.8 Normal Mode Commands

The R3000 console program displays the normal mode prompt (>>) when it is ready to accept commands.

Table 3–8 lists the supported normal mode console commands.

Table 3–8: KN220 Normal Mode Commands

Command	Description
boot	Boots the ULTRIX–32 operating system
continue	Returns control to the processes interrupted by a halt signal
d	Deposits data at a given address
dump	Dumps memory to the screen
e	Examines memory
fill	Deposits data in an address range
go	Resumes execution of the program in memory
help	Displays the syntax of console commands
?	Displays the syntax of console commands
init	Reinitializes memory
maint	Causes the console to enter maintenance mode
passwd	Prompts for the entry of a password, clears the old password, and sets security mode
printenv	Displays console environment variables
setenv	Sets console environment variables
test	Executes the CPU module ROM diagnostic referenced by the test number specified

Table 3–8 (Cont.): KN220 Normal Mode Commands

Command	Description
unsetenv	Unsets console environment variables

Observe the following rules when you type normal mode commands:

- All commands typed at normal mode level are case sensitive with respect to parsing commands; case is preserved when you assign values to environment variables.
- Type all normal mode console commands using ASCII characters only. Values that you enter for environment variables may contain any 8-bit character code.
- Command execution begins when you press Return.
- Enter numeric values as follows:
  - Enter *decimal values* as a string of decimal digits with no leading zeros (for example, 123).
  - Enter *octal values* as a string of octal digits with a leading zero (for example, 0177).
  - Enter *hexadecimal values* as a string of hexadecimal digits preceded by 0x (for example, 0x3ff).
  - Enter *binary values* as a string of binary digits preceded by 0b (for example, 0b1001).
- When reading or writing to memory, you have a choice of data sizes: byte, halfword, or word. Leading zeros are dropped.

  Because a word is 4 bytes, successive addresses, when referenced by a word, are successive multiples of 4. For example, the address following 0x80000004 is 0x80000008. An error occurs if you try to specify an address that is not on a boundary for the data size you are using.

3.8.1 Conventions Used in This Section

The following conventions apply in Section 3.8.

- Letters are to be typed exactly as they appear.
- Letters in italics represent arguments for which you supply values. (Note that the Help and menu screens display these arguments in all capital letters.)
- Arguments enclosed in brackets ([ ]) are optional.
- Ellipses (...) follow an argument that can be repeated.
- A vertical bar (|) separates choices.
- Parentheses are used as in algebraic expressions. For example, the following sequence means enter -b or -h or -w:

  ```
  -(b|h|w)
  ```

3.8.2 Getting Help

You can get help with console command syntax in several ways:

- You can type the word `help` or a question mark (`?`) to display a menu of all console commands.
- You can enter the name of the command for which you want help as an argument to help or as a question mark (?).

  For example, entering `? e` at the console prompt (>>) displays the syntax for the examine (e) command:

  ```
  e [-(b|h|w)] [ADDR]
  >>
  ```

- If you type an incorrect command line, you get a usage message.

  For example, the e command requires an *addr* argument. If you type `e -b` at the console prompt (>>) without entering an address, the screen will display the correct syntax for the command:

  ```
  e [-(b|h|w)] [ADDR]
  >>
  ```

3.8.3 boot

The boot command loads the file that contains the operating system.

Format:

boot [-f *file*] [-s] [-n] [*arg*...]

The optional -f flag followed by the *file* parameter specifies the file you want to use during a boot procedure. If you do not specify the -f flag and a file, the file specified by the environment variable *bootpath* is loaded.

The *file* parameter has the format:

dev([*controller*] [,*unit-number.logical-unit-number*] [,*logical-block-number*]) [*filename*]

- *dev* indicates the device from which you are booting the operating system. Typical devices are rf for RF-series ISEs, ra for RA-series hard disk drives, tm for a tape, and mop for a network. Typing `mop` nullifies the other arguments in the list. Table 3–9 lists the device names for each device.

Table 3–9: Boot Device Names (Normal Mode)

Device Type	Protocol	Number of Units Per Storage Interface	Device Name
RA-series fixed disk	MSCP	4[1]	ra
RF-series ISE	DSSI	7	rf
RZ-series fixed disk	SCSI	7	rz
TZ-series tape drive	SCSI	7	tz
Tape drive	TMSCP	1[1]	tm
Ethernet adapter	MOP	1	mop
Ethernet adapter	TFTP	1	tftp

[1]Up to 16 controllers and 32 units supported.

- *controller* indicates the ID number of the controller for the device from which you are booting the operating system.
- *unit-number* indicates the unit number of the device from which you are booting the operating system.
- *logical-unit-number* (LUN) has meaning for SCSI devices only; it defaults to zero.
- *logical-block-number* indicates the number (or other designator) of the block from which you are booting the operating system. When you are

booting from a tape, this number is not used because the boot file must be the first file on the tape. When you are booting from a disk, this number depends on how you partitioned the disk when you installed your operating system software. Refer to your software installation manual if you need a reminder about logical block indicators.

- *filename* indicates the name of the operating system file.

The optional -s flag causes the operating system to boot in single-user mode. Unless -s is specified, the system will boot in multiuser mode.

The optional -n flag causes the specified file to be loaded but not executed.

The optional *arg* parameter contains any information to be passed to the booted image.

Examples:

```
>> boot -f ra(0,0,0)vmunix
```

This command boots the file vmunix, located in the logical block number of the first hard disk (unit number 0), using controller 0.

```
>> boot -f rf(2,2,0)vmunix
```

This command boots the file vmunix, located in the logical block of the second RF-series ISE (unit number 2), using controller 2.

```
>> boot -f tm(0,5)
```

This command boots from the tape, which is unit 5 and controller 0.

```
>> boot -f rz(0, 0, 0)vmunix
```

This command boots from the tape, which is unit 0 and controller 0.

To display a list of devices, their unit numbers, and controller numbers, type the following sequence at the Normal mode prompt:

```
>> maint
>>> show dev (or dssi, scsi, ethernet, or uqssp)

>>> exit
>>
```

The show command displays all of the system device names, which are followed by the controller number and unit number in parentheses. The asterisk indicates the *logical-block-number* variable, which is determined during installation of the operating system software.

3.8.4 continue

CAUTION: *If the operating system state has not been properly saved (halted), entering the continue command may cause the processor to hang.*

Use the continue command if you inadvertently halt the system by pressing Break or the Halt button.

When a process is interrupted by a halt signal, the state of the process (R3000 and R3010 internal registers) are stored in the halt state memory block. When the continue command is entered, the saved state of the process is reloaded, and the process resumes execution.

The continue command returns control to the processes interrupted by a halt signal.

Format:

continue

3.8.5 d (deposit)

The d (deposit) command deposits a single byte, halfword, or word value at the specified address. If you repeat the command without specifying an address, the data will be deposited in the next word location.

Format:

d [[[-(b | h | w)] *[addr]]* | [-H *reg-name*]] val

The first parameter, which is optional, indicates the data size. If not given, data size defaults to word. If you do not specify a data size, a word is used.

- -b deposits 1 byte of data.
- -h deposits a halfword (2 bytes) of data.
- -w deposits a word (4 bytes) of data.

The *addr* parameter indicates the address to which you want data written. System address space is in the range 0x80000000 to 0xbf000000. If *addr* is not specified, it will default to one location beyond the last address accessed by either the examine or deposit commands.

The -H parameter specifies that the data is to be deposited to a register in the halt state memory block. This memory location is where all the R3000 internal registers are saved when the system is halted.

The *reg-name* parameter specifies the name of the particular R3000 internal register for which you want data written.

The *val* parameter contains the data you want deposited at the given address.

Example:

```
>> d -w 0x80000000 0xffffffff
```

This version of the command deposits the value 0xffffffff, with a data size of one word, at address 0x80000000.

3.8.6 dump

dump [-H] | [[[-(b | h | w)] [-(o | d | u | x | c | B)]] | [-I]] *rng*

This command shows a formatted display of the contents of memory.

The -H parameter displays the contents of the halt state memory block. All R3000 internal registers are stored in the halt state memory block when the system is halted. The -H parameter option cannot be used with any other command parameter.

The second parameter, which is optional, indicates the data size. If you do not specify a data size, the system uses a word.

- -b displays memory in bytes.
- -h displays memory in halfwords.
- -w displays memory in words.

The next parameter, also optional, determines how data is displayed.

- -o displays memory in octal format.
- -d displays memory in decimal format.
- -u displays memory in unsigned decimal format.
- -x displays memory in hexadecimal format.
- -c displays memory in ASCII format.
- -B displays memory in binary format.

If no format argument is given, hexadecimal format is used.

The -I parameter displays memory in assembly language format.

The *rng* parameter indicates the range of memory you want to see. You can specify the range in one of two ways:

- *addr#cnt* displays the number of addresses specified by *cnt*, beginning at *addr*.
- *addr:addr* displays all values between the specified addresses.

Examples:

```
>> dump 0x80000000#0xf
```

This command uses hexadecimal format to dump the first 15 words of memory to the screen.

```
>> dump -b 0x80000000#0xf
```

This command uses hexadecimal format to dump the first 15 bytes of memory to the screen. The dump display shows rows of address contents. The left-most column gives the address of the first field in each row.

```
>> dump -I 0x80030200:0x80030220
0x80030200:     c048228     jal         0x801208a0
0x80030204:        2021     addu        a0,zero,zero
0x80030208:    8fbf0014     lw          ra,0x14(sp)
0x8003020c:    27bd0018     addiu       sp,0x18
0x80030210:     3e00008     jr          ra
0x80030214:           0     nop
0x80030218:    27bdffe8     addiu       sp,0xffe8
0x8003021c:    afbf0014     sw          ra,0x14(sp)
>>
```

This command displays in assembly language format, all values between the specified addresses. The first column lists the memory location in hexadecimal, the second column lists the contents of the memory location, the third column lists the R3000 assembly language instruction, and the fourth column lists the corresponding operand.

3.8.7 e (examine)

The e (examine) command examines the byte, halfword, or word at the specified address.

Format:

e [-(b | h | w)] [addr]

The first parameter, which is optional, indicates the data size. If not given, data size defaults to word. If you do not specify the data size, a word is used.

- -b indicates a single byte.
- -h indicates a halfword.
- -w indicates a word.

The *addr* parameter indicates an address in the range 0x80000000 to 0xBF000000. The *addr* parameter is also optional. If it is not specified, it will default to one location beyond the last address accessed by either the Examine or Deposit commands.

When you enter the examine command, a display similar to the following appears:

```
0x80000005:  65 0x41    'A'
```

The left-hand field echoes the address you entered.

The next three fields display the contents of the address in decimal, hexadecimal, and ASCII formats, respectively. If the ASCII character is unprintable, it is displayed as an octal value preceded by a backslash: for example, '\032'.

Example:

```
>> e 0x80000000
```

This command examines the word at address 0x80000000. The resulting display might look like this:

```
0x80000000:       1008385985      0x3c1abfc1      '\301'
```

3.8.8 fill

The fill command writes a specified value to a range of memory. If you do not specify a value, the system puts zeros in the memory range.

Format:

fill [-(b | h | w)] [-v *val*] *rng*

The first parameter, which is optional, indicates the data size. If not given, data size defaults to word.

- -b indicates bytes.
- -h indicates halfwords.
- -w indicates words.

The optional parameter -v *val* specifies the numeric value to write to memory. If you do not specify a value, all zeros are written. If the size of *val* does not match the data size parameter, *val* is truncated or expanded as necessary.

The *rng* parameter indicates the memory range. You can specify the range in one of two ways:

- *addr#cnt* fills addresses beginning at *addr* and continuing for *cnt* locations.
- *addr:addr* fills all locations between the two given addresses.

Example:

```
>> fill -v 0xffffffff 0x80000010:0x800000ff
```

This command sets all bits to 1 at addresses 16 to 255.

3.8.9 go

The go command transfers control to the indicated entry-point address.

Format:

go [*pc*]

The optional *pc* parameter indicates the entry-point address you want to use.

If you do not specify an entry address, the system uses the entry point of the program module that was most recently loaded. If no program module was previously loaded, the system uses 0 as the entry-point address.

3.8.10 help

The help command displays the correct syntax for the console commands.

Format:

help [*cmd*]

The optional *cmd* parameter indicates the command for which you want information. If you do not specify *cmd*, the complete console menu appears.

Example:

```
>> help
CMD:
  x -(l|u) addr count
  boot [-f FILE] [-s] [-n] [ARG...]
  continue ADDRESS
  d [-(b|h|w)] [ADDR] VAL
  dump [-(b|h|w)] [-(o|d|u|x|c|B)] RNG
  e [-(b|h|w)] [ADDR]
  fill [-(b|h|w)] [-v VAL] RNG
  go [PC]
  help [CMD]
  init
  maint
  passwd -(c|s|u)
  printenv [EVAR...]
  setenv EVAR STR
  test [-r] TEST_NUMBER
  unsetenv EVAR
  ? [CMD]
RNG:
  ADDR#CNT
  ADDR:ADDR
>>
```

3.8.11 ?

The ? command functions exactly like the help command.

Format:

? [*cmd*]

3.8.12 init

The init command fully initializes the system.

Format:

init

The system performs the following:

- Clears memory.
- Resets I/O.
- Initializes all supported devices.

Example:

```
>> init
Memory Size: 33554432 (0x2000000) bytes
Ethernet Address: 08-00-2b-12-81-22
>>
```

3.8.13 printenv

The printenv command displays the current value for the specified environment variable.

Format:

printenv [*evar*...]

The optional *evar* parameter indicates the variable whose value you want to see. If you do not specify a variable, the complete environment variable table appears.

Example:

```
>> printenv
bootpath=rf()vmunix
bootmode=*
console=0
scsiid0=7
scsiid1=7
scsiid2=7
scsiid3=7
scsiid4=7
scsiid5=7
scsiid6=7
scsiid7=7
baud=9600
systype=0x820b0800
bitmap=0xa1ff2000
bitmaplen=0x800
memdescriptor=0x0
boot=rf()vmunix
osconsole=0
>>
```

3.8.14 setenv

The setenv command assigns new values to the specified environment variable. Refer to the discussion of the printenv command (Section 3.8.13) for a description of each variable.

Format:

setenv *evar str*

- The *evar* parameter indicates the variable you want to set.
- The *str* parameter indicates the value you want to specify.

Example:

```
>> setenv bootmode a
```

This command assigns a value of "a" to the *bootmode* variable. This will cause the system to autoboot at power-up.

You can also add your own environment variables, as explained in Section 3.8. These variables are stored in volatile memory. The environment variables table can contain up to 32 variables, for a total of 512 characters.

3.8.15 test

Test executes the CPU module ROM diagnostic referenced by test_number.

Format:

test [-r] test_number

The test number must be preceded by a '0x' (that is, specified in hex). If the -r flag is specified, the test repeats (even if failures occur).

3.8.16 unsetenv

The unsetenv command removes the specified variable from the environment variables table.

Format:

unsetenv *evar*

The *evar* parameter indicates the variable you are removing. Refer to Table 3–7 earlier in this section for a description of each variable.

The unsetenv command does not affect the environment variables stored in nonvolatile memory. These variables are reset at the next reset or power cycle.

3.8.17 x

The x command is a binary load/unload command for automatic testing systems that load programs through the console when the IO board is not present.

Format:

x [-l | -u] address count <CR> <LF> command_checksum

- -l specifies that data is to be loaded into memory through the console.
- -u specifies that data is to be unloaded from memory and output through the console.

The x command loads/unloads count number of bytes of data starting at address. After the command is entered x expects a byte of input which is the checksum of command line. If the command checksum is correct then x will begin accepting the input of count bytes of data or will begin outputing count bytes of data. Following the data, x will expect, or will output, an additional data checksum byte. If a load (-l option) and the checksum is incorrect, x will issue an error message.

When x is receiving the checksum or receiving data during a load, the data will not be echoed on the screen. When receiving data, control flow characters (CTRL-O, etc.) will be interpreted only as data and will not effect the flow of control.

The checksums will be validated by adding each byte of data to an 8-bit register. When the checksum is added the result should be zero. For the command checksum, each non-white_space character will be added to the register. Note that the whitespace includeing spaces, tabs, and the final return <CR> and linefeed <LF>, are excluded when calculating the checksum.

The address will be expected to be a virtual address so that count bytes of data will fit in the range 0000 0000 : 7fff ffff.

3.9 Maintenance Mode Overview

Whenever the CVAX console program is running, the maintenance mode prompt (>>>) is displayed on the console terminal and the KN220 is halted as described in Section 3.3 and Section 3.4.

In maintenance mode, you can examine and alter the state of the processor by typing certain console commands and characters. Table 3–10 lists the keypad control characters that have special meaning in maintenance mode.

Table 3–10: KN220 Console Control Characters (Maintenance Mode)

Character	Action
Return	Also <CR>. The carriage return ends a command line. No action is taken on a command until after it is terminated by a carriage return. A null line terminated by a carriage return is treated as a valid, null command. No action is taken, and the console prompts for input. Carriage return is echoed as carriage return, line feed.
X	When you press X (rubout), the console deletes the previously typed character. The resulting display differs, depending on whether the console is a video or a hardcopy terminal. For hardcopy terminals, the console echoes a backslash (\) followed by the character being deleted. If you press additional rubouts, the additional deleted characters are echoed. If you type a nonrubout character, the console echoes another backslash, followed by the character typed. The result is to echo the characters deleted, surrounding them with backslashes. For example: EXAMI;E[X][X]NE<CR> The console echoes: EXAMI;E\E;\NE<CR> The console sees the command line: EXAMINE<CR> For video terminals, the previous character is erased and the cursor is restored to its previous position. The console does not delete characters past the beginning of a command line. If you press more rubouts than there are characters on the line, the extra rubouts are ignored. A rubout entered on a blank line is ignored.
Ctrl/C	Echoes ^C<CR> and aborts processing of a command. Has no effect as part of a binary load data stream. Clears Ctrl/S and reenables output stopped by [Ctrl/O].
Ctrl/Q	Resumes output to the console terminal. Not echoed.
Ctrl/S	Stops output to the console terminal until you enter Ctrl/Q. Not echoed.
Ctrl/R	Echoes <CR><LF>, followed by the current command line. Can be used to improve the readability of a command line that has been heavily edited.

Table 3–10 (Cont.): KN220 Console Control Characters (Maintenance Mode)

Character	Action
Ctrl/U	Echoes ^U<CR> and deletes the entire line. Entered, but otherwise ignored if typed on an empty line.

3.9.1 Command Syntax in Maintenance Mode

The console accepts commands up to 80 characters long. Longer commands produce error messages. The character count does not include rubouts, rubbed-out characters, or the Return at the end of the command.

You can abbreviate a command by entering only as many characters as are required to make the command unique. Most commands can be recognized from their first character. See Table 3–14.

The console treats two or more consecutive spaces and tabs as a single space. Leading and trailing spaces and tabs are ignored. You can place command qualifiers after the command keyword or after any symbol or number in the command.

All numbers (addresses, data, counts) are hexadecimal (hex) except for GPR symbolic names, which are in decimal. The hex digits are 0 through 9 and A through F. You can use uppercase and lowercase letters in hex numbers (A through F) and commands.

The following symbols are qualifier and argument conventions:

[ ] an optional qualifier or argument

{} a required qualifier or argument

3.9.2 Address Specifiers in Maintenance Mode

Several commands take an address or addresses as arguments. An address defines the address space, and the offset into that space. The console supports six address spaces:

Physical memory
Virtual memory
Protected memory
General purpose registers (GPRs)
Internal processor registers (IPRs)
The PSL

The address space that the console references is inherited from the previous console reference, unless you explicitly specify another address space. The initial address space is physical memory.

3.9.3 Symbolic Addresses in Maintenance Mode

The console supports symbolic references to addresses. A symbolic reference defines the address space and the offset into that space. Table 3–11 lists symbolic references supported by the console, grouped according to address space. You do not have to use an address space qualifier when using a symbolic address.

Table 3–11: Console Symbolic Addresses (Maintenance Mode)

Symbol	Address	Symbol	Address
GPR Address Space (/G)			
R0	0	R1	1
R2	2	R3	3
R4	4	R5	5
R6	6	R7	7
R8	8	R9	9
R10	0A	R11	0B
R12	0C	R13	0D
R14	0E	R15	0F
AP	0C	FP	0D
SP	0D	PC	0E
PSL	–	–	–
IPR Address Space (/I)			
pr$_ksp	00	pr$_esp	01
pr$_ssp	02	pr$_usp	03
pr$_isp	04	pr$_p0br	08
pr$_p0lr	09	pr$_p1br	0A
pr$_p1lr	0B	pr$_sbr	0C
pr$_slr	0D	pr$_pcbb	10
pr$_scbb	11	pr$_ipl	12
pr$_astlv	13	pr$_sirr	14
pr$_sisr	15	pr$_iccr	18
pr$_nicr	19	pr$_icr	1A
pr$_todr	1B	pr$_rxcs	20
pr$_rxdb	21	pr$_txcs	22

Table 3–11 (Cont.): Console Symbolic Addresses (Maintenance Mode)

Symbol	Address	Symbol	Address
IPR Address Space (/I)			
pr$_txdb	23	pr$_tbdr	24
pr$_cadr	25	pr$_mcesr	26
pr$_mser	27	pr$_savpc	2A
pr$_savpsl	2B	pr$_ioreset	37
pr$_mapen	38	pr$_tbia	39
pr$_tbis	3A	pr$_sid	3E
pr$_tbchk	3F	–	–

Table 3–12 lists symbolic addresses that you can use in any address space.

Table 3–12: Symbolic Addresses Used in Any Address Space

Symbol	Description
*	The location last referenced in an EXAMINE or DEPOSIT command.
+	The location immediately following the last location referenced in an EXAMINE or DEPOSIT command. For references to physical or virtual memory spaces, the location referenced is the last address, plus the size of the last reference (1 for byte, 2 for word, 4 for longword, 8 for quadword). For other address spaces, the address is the last address referenced plus one.
–	The location immediately preceding the last location referenced in an EXAMINE or DEPOSIT command. For references to physical or virtual memory spaces, the location referenced is the last address minus the size of this reference (1 for byte, 2 for word, 4 for longword, 8 for quadword). For other address spaces, the address is the last address referenced minus one.
@	The location addressed by the last location referenced in an EXAMINE or DEPOSIT command.

3.9.4 Command Qualifiers in Maintenance Mode

You can enter console command qualifiers in any order on the command line after the command keyword. There are three types of qualifiers: data control, address space control, and command specific. Table 3–13 lists and describes the data control and address space control qualifiers. Command specific qualifiers are listed in the descriptions of individual commands.

Table 3–13: Console Command Qualifiers (Maintenance Mode)

Qualifier	Description
Data Control	
/B	The data size is byte.
/W	The data size is word.
/L	The data size is longword.
/Q	The data size is quadword.
/N:{count}	An unsigned hexadecimal integer that is evaluated as a longword. This qualifier determines the number of additional operations that are to take place on EXAMINE, DEPOSIT, MOVE, and SEARCH commands. An error message appears if the number overflows 32 bits.
/STEP:{size}	Step. Overrides the default increment of the console current reference. Commands that manipulate memory, such as EXAMINE, DEPOSIT, MOVE, and SEARCH normally increment the console current reference by the size of the data being used.
Address Space Control	
/G	General purpose register (GPR) address space, R0–R15. The data size is always longword.
/I	Internal processor register (IPR) address space. Accessible only by the MTPR and MFPR instructions. The data size is always longword.
/V	Virtual memory address space. All access and protection checking occur. If access to a program running with the current PSL is not allowed, the console issues an error message. Deposits to virtual space cause the PTE<M> bit to be set. If memory mapping is not enabled, virtual addresses are equal to physical addresses. Note that when you examine virtual memory, the address space and address in the response is the physical address of the virtual address.
/P	Physical memory address space.
/M	Processor status longword (PSL) address space. The data size is always longword.
/U	Access to console private memory is allowed. This qualifier also disables virtual address protection checks. On virtual address writes, the PTE<M> bit is not set if the /U qualifier is present. This qualifier is not inherited; it must be respecified on each command.

3.9.5 Maintenance Mode Command Keywords

Table 3–14 lists maintenance mode command keywords by type. Table 3–15 lists the parameters, qualifiers, and arguments for each console command. Parameters, used with the SET and SHOW commands only, are listed in the first column along with the command.

Although it is possible to abbreviate by using the minimum number of characters required to uniquely identify a command or parameter, these abbreviations may become ambiguous at a later time if a new command or parameter is added in an updated version of the firmware. For this reason you should not use abbreviations in programs.

Table 3–14: Command Keywords by Type (Maintenance Mode)

Processor Control	Data Transfer	Console Control
BOOT	EXAMINE	CONFIGURE
CONTINUE	DEPOSIT	FIND
HALT	MOVE	REPEAT
INITIALIZE	SEARCH	SET
NEXT	X	SHOW
START	DC [qualifier]	TEST
UNJAM		!
		DC
		UNPRIV

Table 3–15: Console Command Summary (Maintenance Mode)

Command	Qualifiers[1]	Argument	Other(s)
BOOT	/R5:{boot_flags} or /{boot_flags}	[{boot_device}]	–
CONFIGURE	–	–	–
CONTINUE	–	–	–
DC	–	–	–
DC/RESTORE	–	{tape_device}	–
DC/SAVE	–	{tape_device}	–
DC/ZERO	–	–	–
DEPOSIT	/B /W /L /Q /G /I /V /P /M /U /N:{count} /STEP:{size}	{address}	{data} [{data}]

[1]{} denotes a mandatory item that must be syntactically correct.
[ ] denotes an optional item.
! denotes an “or” condition.

Table 3–15 (Cont.): Console Command Summary (Maintenance Mode)

Command	Qualifiers[1]	Argument	Other(s)
EXAMINE	/B /W /L /Q /G /I /V /P /M /U /N:{count} /STEP:{size} /INSTRUCTION	[address]	–
EXIT	–	–	–
FIND	/MEM /RPB	–	–
HALT	–	–	–
HELP	–	–	–
INITIALIZE	–	–	–
MOVE	/B /W /L /Q /V /P /U /N:{count} /STEP:{size}	{src_address}	{dest_address}
NEXT	–	[count]	–
REPEAT	–	{command}	–
SEARCH	/B /W /L /Q /V /P /U /N:{count} /STEP:{size} /NOT	{start_address}	{pattern} [{mask}]
SET BFLAG	–	{boot_flags}	–
SET BOOT	–	{device_string}	–
SET HOST	/DUP {/DSSI n /UQSSP} {/DISK n /TAPE n csr_address} /{MAINTENANCE /UQSSP} {/SERVICE n csr_address}	{node} n {controller_number}	[{task}]
SET LANGUAGE	–	{language_type}	–
SHOW BFLAG	–	–	–
SHOW BOOT	–	–	–
SHOW DEVICE	–	–	–
SHOW DSSI	–	–	–
SHOW ETHERNET	–	–	–
SHOW LANGUAGE	–	–	–
SHOW MEMORY	/FULL	–	–
SHOW QBUS	–	–	–
SHOW SCSI	/FULL	–	–
SHOW UQSSP	–	–	–
SHOW VERSION	–	–	–

[1]{} denotes a mandatory item that must be syntactically correct.
[ ] denotes an optional item.
! denotes an "or" condition.

Table 3–15 (Cont.): Console Command Summary (Maintenance Mode)

Command	Qualifiers[1]	Argument	Other(s)
START	–	[{address}]	–
TEST	–	{test_number}	[{parameters}]
UNJAM	–	–	–
UNPRIV	–	–	–
X	–	{address}	{count}

[1]{} denotes a mandatory item that must be syntactically correct.
[ ] denotes an optional item.
! denotes an "or" condition.

3.10 Maintenance Mode Commands

This section describes the maintenance mode commands. Enter the commands at the maintenance mode prompt (>>>). These commands may be typed in uppercase or lowercase letters. However, they are shown in all uppercase letters in this document to differentiate them from the normal mode commands, which must be typed in all lowercase letters.

3.10.1 BOOT

The BOOT command initializes the processor and transfers execution to VMB. VMB attempts to boot MDM from the specified device or from the default boot device if none is specified. The console qualifies the bootstrap operation by passing a boot flags bitmap to VMB in R5.

Format:

BOOT [qualifier-list] [device_name]

If you do not enter either the qualifier or the device name, the default value is used. Explicitly stating the boot flags or the boot device overrides, but does not permanently change, the corresponding default value.

Set the default boot device and boot flags with the SET BOOT and SET BFLAG commands. If you do not set a default boot device, the processor times out after 30 seconds and attempts to boot from the on-board Ethernet port, EZA0.

Qualifiers:

Command specific:

/R5:{bitmap}	A 32-bit hex value passed to VMB in R5. The console does not interpret this value. Use the SET BFLAG command to specify a default boot flags longword. Use the SHOW BFLAG command to display the longword. Table 3–4 lists the supported R5 boot flags.
/{bitmap}	Same as /R5:{bitmap}
[device_name]	A character string of up to 39 characters. Longer strings cause a VAL TOO BIG error message. Apart from length, the console makes no attempt to interpret or validate the device name. The console converts the string to uppercase, then passes to VMB a string descriptor to this device name in R0.

Example:

```
>>> BOOT XQA0! Boot using default boot flags and
(BOOT/R5:10 XQA0)             ! specified device.
  2..
-XQA0
```

3.10.2 CONFIGURE

The CONFIGURE command invokes an interactive mode that permits you to enter Q22-bus device names, then generates a table of Q22-bus I/O page device CSR addresses and interrupt vectors. CONFIGURE is similar to the ULTRIX shoadrs utility. This command simplifies field configuration by providing information that is typically available only with a running operating system. Refer to the example below and use the CONFIGURE command as follows:

1. Enter `CONFIGURE` at the maintenance mode prompt (>>>).
2. Enter at the `Device, Number?` prompt to see a list of devices whose CSR addresses and interrupt vectors can be determined.
3. Enter the device names and number of devices.
4. Enter `EXIT` to obtain the CSR address and interrupt vector assignments.

The devices listed in the HELP display are not necessarily supported by the KN220–AA CPU.

Format:

CONFIGURE

Example:

```
>>> CONFIGURE

Enter device configuration, HELP, or EXIT
Device,Number? help
Devices:
 LPV11     KXJ11        DLV11J       DZQ11     DZV11      DFA01
 RLV12     TSV05        RXV21        DRV11W    DRV11B     DPV11
 DMV11     DELQA        DEQNA        DESQA     RQDX3      KDA50
 RRD50     RQC25        KFQSA-DISK   TQK50     TQK70      TU81E
 RV20      KFQSA-TAPE   KMV11        IEQ11     DHQ11      DHV11
 CXA16     CXB16        CXY08        VCB01     QVSS       LNV11
 LNV21     QPPSS        DSV11        ADV11C    AAV11C     AXV11C
 KWV11C    ADV11D       AAV11D       VCB02     QDSS       DRV11J
 DRQ3B     VSV21        IBQ01        IDV11A    IDV11B     IDV11C
 IDV11D    IAV11A       IAV11B       MIRA      ADQ32      DTC04
 DESNA     IGQ11        DIV32        KIV32     DTCN5      DTC05
 KWV32     KZQSA
Numbers:
 1 to 255, default is 1
Device,Number? cxa16,2
Device,Number? cxy08
Device,Number? tqk70
Device,Number? kda50
Device,Number? kzqsa
Device,Number? exit

Address/Vector Assignments
-772150/154 KDA50
-774500/260 TQK70
-760440/300 CXA16
-760460/310 CXA16
-760500/320 CXY08
-761400/330 KZQSA
>>>
```

3.10.3 CONTINUE

CAUTION: *If the operating system state has not been properly saved (halted), do not type* `CONTINUE`. *Doing so may cause the processor to hang.*

The CONTINUE command causes the processor to begin instruction execution at the address currently contained in the PC. It does not perform a processor initialization.

Format:

CONTINUE

Example:

```
>>> CONTINUE
```

3.10.4 DC

The DC command acts on the contents of the nonvolatile disk cache, by diagnosing (no modifier), restoring, saving, or zeroing.

Whenever the system notifies that "NVRAM dirty," back up the NVRAM and replace the CPU. Then restore the NVRAM to the new CPU before you boot the system. See Appendix D.

Format:

DC [/RESTORE] [/SAVE] [/ZERO]

Examples:

```
>>> DC/SAVE
```

```
>>> dc
```

At the dc command, if the CPU board's cache contains data the following message is displayed on the console terminal:

```
Disk Cache - Dirty
```

At the dc command, if the CPU board's cache does not contain any data the following message is displayed on the console terminal:

```
Disk Cache - Clean
```

If there is no data in the cache (cache is clean), you can follow normal procedures for rebooting or troubleshooting.

You should always run the dc command before replacing or using any DECsystem 5500 CPU board.

```
>>> dc/save mua0
Disk Cache - Dirty
Do you want to continue (y/n)? y

-MUA0
Zero Disk Cache (y/n)? y
>>>

>>> dc/restore mua0
Disk Cache - Dirty
Do you want to continue (y/n)? y

-MUA0
>>>
```

If the cache is clean, as in the following example, the contents of the tape (in this example, mua0) are loaded into the cache.

```
>>> dc/restore mua0
Disk Cache - Clean

-MUA0
>>>
```

When the system reboots, the contents of the cache are moved to the appropriate disks.

```
>>> dc/zero
Do you want to continue (y/n)? y
```

The command prompts you to confirm that you want to clear the contents of the cache and then fills the cache with zeroes. You can use this command as a security measure to ensure that the cache is cleared.

3.10.5 DEPOSIT

The DEPOSIT command deposits data into the address specified. If you do not specify an address space or data size qualifier, the console uses the last address space and data size used in a DEPOSIT, EXAMINE, MOVE, or SEARCH command. After processor initialization, the default address space is physical memory, the default data size is longword, and the default address is zero. If you specify conflicting address space or data sizes, the console ignores the command and issues an error message.

Format:

DEPOSIT [qualifier_list] {address} {data} [data...]

Qualifiers:

Data control: /B, /W, /L, /Q, /N:{count}, /STEP:{size}

Address space control: /G, /I, /P, /V, /U, /M

Arguments:

{address} A longword address that specifies the first location into which data is deposited. The address can be an actual address or a symbolic address.

{data} The data to be deposited. If the specified data is larger than the deposit data size, the firmware ignores the command and issues an error response. If the specified data is smaller than the deposit data size, it is extended on the left with zeros.

[data] Additional data to be deposited (as many as can fit on the command line).

Examples:

```
>>> D/P/B/N:1FF 0 0   ! Clear first 512 bytes of physical memory.

>>> D/V/L/N:3 1234 5  ! Deposit 5 into four longwords starting
                         ! at virtual memory address 1234.
>>> D/N:8 R0 FFFFFFFF ! Loads GPRs R0 through R8 with -1.

>>> D/L/P/N:10/S:200 0 8   ! Deposit 8 in the first longword of
         ! the first 17 pages in physical
         ! memory starting at location 0.
>>> D/N:200/8 - 0     ! Starting at previous address,
         ! clear 513 bytes.
```

3.10.6 EXAMINE

The EXAMINE command examines the contents of the memory location or register specified by the address. If no address is specified, + is assumed. The display line consists of a single character address specifier, the physical address to be examined, and the examined data.

EXAMINE uses the same qualifiers as DEPOSIT. The EXAMINE command also supports an /INSTRUCTION qualifier, which will disassemble the instructions at the current address.

Format:

EXAMINE [qualifier_list] [address]

Qualifiers:

Data control: /B, /W, /L, /Q, /N:{count}, /STEP:{size}

Address space control: /G, /I, /P, /V, /U, /M

Command specific:

/INSTRUCTION	Disassembles and displays the VAX MACRO–32 instruction at the specified address.

Arguments:

[address]	A longword address that specifies the first location to be examined. The address can be an actual or a symbolic address. If no address is specified, + is assumed.

Examples:

```
>>> EXAMINE PC     ! Examine the PC.
  G 0000000F FFFFFFFC
>>> EXAMINE SP     ! Examine the SP.
  G 0000000E 00000200
>>> EXAMINE PSL    ! Examine the PSL.
  M 00000000 041F0000
>>> EXAMINE/M     ! Examine PSL another way.
  M 00000000 041F0000
>>> EXAMINE R4/N:5 ! Examine R4 through R9.
  G 00000004 00000000
  G 00000005 00000000
  G 00000006 00000000
  G 00000007 00000000
  G 00000008 00000000
  G 00000009 801D9000

>>> EXAMINE PR$_SCBB ! Examine the SCBB, IPR 17
  I 00000011 2004A000   ! (decimal).

>>> EXAMINE/P 0 ! Examine local memory 0.
  P 00000000 00000000

>>> EXAMINE/INS 20040000 ! Examine 1st instruction
                                    ! in ROM.
  P 20040000   11 BRB     20040019

>>> EXAMINE/INS/N:5 20040019 ! Disassemble from branch.
  P 20040019   D0 MOVL    I^#20140000,@#20140000
  P 20040024   D2 MCOML   @#20140030,@#20140502
  P 2004002F   D2 MCOML   S^#0E,@#20140030
  P 20040036   7D MOVQ    R0,@#201404B2
  P 2004003D   D0 MOVL    I^#201404B2,R1
  P 20040044   DB MFPR    S^#2A,B^44(R1)

>>> EXAMINE/INS ! Look at next instruction.
  P 20040048   DB MFPR    S^#2B,B^48(R1)
>>>
```

3.10.7 EXIT

The EXIT command exits from maintenance mode (>>>) to the normal mode prompt (>>). This command idles the CVAX chip. The maint command, typed in lowercase letters from the normal mode prompt, returns you to maintenance mode.

Format:

EXIT

Example:

```
>>> EXIT ! From maintenance mode, exit to
                           ! normal mode.
>> maint ! From normal mode, exit to maintenance mode.
>>>
```

3.10.8 FIND

The FIND command searches main memory starting at address zero for a page-aligned 128-Kbyte segment of good memory, or a restart parameter block (RPB). If the command finds the segment or RPB, its address plus 512 is left in SP (R14). If it does not find the segment or RPB, the console issues an error message and preserves the contents of SP. If you do not specify a qualifier, /RPB is assumed.

Format:

FIND [qualifier-list]

Qualifiers:

Command specific:

/MEM	Searches memory for a page-aligned block of good memory, 128 Kbytes in length. The search looks only at memory that is deemed usable by the bitmap. This command leaves the contents of memory unchanged.
/RPB	Searches all of physical memory for an RPB. The search does not use the bitmap to qualify which pages are looked at. The command leaves the contents of memory unchanged.

Examples:

```
>>> EXAMINE SP  ! Check the SP.
  G 0000000E 00000000
>>> FIND /MEM   ! Look for a valid 128 Kbyte.
>>> EXAMINE SP  ! Note where it was found.
  G 0000000E 00000200
>>> FIND /RPB   ! Check for valid RPB.
?2C FND ERR 00C00004            ! None to be found here.
>>>
```

3.10.9 HALT

The HALT command has no effect. It is included for compatibility with other VAX consoles.

Format:

HALT

Example:

```
>>> HALT   ! Pretend to halt.
>>>
```

3.10.10 HELP

The HELP command provides information about command syntax and usage.

Format:

HELP

Example:

```
>>> HELP

Following is a brief summary of all the commands supported by the
console:

        UPPERCASE  denotes a keyword that you must type in
        |          denotes an OR condition
        []         denotes optional parameters
        <>         denotes a field that must be filled in
                   with a syntactically correct value

Valid qualifiers:
    /B /W /L /Q /INSTRUCTION
    /G /I /V /P /M
    /STEP: /N: /NOT /U

Valid commands:
    DEPOSIT [<QUALIFIERS>] <ADDRESS> [<DATUM> [<DATUM>]]
    EXAMINE [<QUALIFIERS>] [<ADDRESS>]
    MOVE [<QUALIFIERS>] <ADDRESS> <ADDRESS>
    SEARCH [<QUALIFIERS>] <ADDRESS> <PATTERN> [<MASK>]
    SET BFLG <BOOT_FLAGS>
    SET BOOT <BOOT_DEVICE>[:]
    SET HOST/DUP/DSSI <NODE_NUMBER> [<TASK>]
    SET HOST/DUP/UQSSP </DISK | /TAPE> <CONTROLLER_NUMBER> [<TASK>]
    SET HOST/DUP/UQSSP <PHYSICAL_CSR_ADDRESS> [<TASK>]
    SET HOST/MAINTENANCE/UQSSP/SERVICE <CONTROLLER_NUMBER>
    SET HOST/MAINTENANCE/UQSSP <PHYSICAL_CSR_ADDRESS>
    SET LANGUAGE <LANGUAGE_NUMBER>
    SHOW BFLG
    SHOW BOOT
    SHOW DEVICE
    SHOW DSSI
    SHOW SCSI [/FULL]
    SHOW ETHERNET
    SHOW LANGUAGE
    SHOW MEMORY [/FULL]
    SHOW QBUS
    SHOW UQSSP
    SHOW VME
    SHOW VERSION
    HALT
```

```
    INITIALIZE
    UNJAM
    CONTINUE
    START <ADDRESS>
    REPEAT
    X <ADDRESS> <COUNT>
    FIND [/MEM | /RPB]
    TEST [<TEST_CODE> [<PARAMETERS>]]
    BOOT [/R5:<BOOT_FLAGS> | /<BOOT_FLAGS>] [<BOOT_DEVICE>[:]]
    NEXT [count]
    CONFIGURE
    EXIT
    UNPRIV
    DC [[</SAVE | /RESTORE <TAPE_DEVICE>>] | /ZERO]
    HELP
>>>
```

3.10.11 INITIALIZE

The INITIALIZE command performs a processor initialization.

Format:

INITIALIZE

The following registers are initialized:

Register	State at Initialization
PSL	041F0000
IPL	1F
ASTLVL	4
SISR	0
ICCS	Bits <6> and <0> clear; the rest are unpredictable
RXCS	0
TXCS	80
MAPEN	0
CVAX cache	Disabled, all entries invalid
Instruction buffer	Unaffected
Console previous reference	Longword, physical, address 0
TODR	Unaffected
Main memory	Unaffected
General registers	Unaffected
Halt code	Unaffected
Bootstrap-in-progress flag	Unaffected
Internal restart-in-progress flag	Unaffected

The firmware clears all error status bits and initializes as follows:

1. Clears any interrupts.
2. Initializes pr$_scbb to console SCB.
3. Initializes the IPR:

ipri$1_ipr	= 0
ipri$1_val	= 4
ipri$s_ipri	= 8
pr$_ipl	= ^x0000001F
pr$_astlvl	= ^x00000004
pr$_sisr	= ^x00000000
pr$_iccs	= ^x00000000
pr$_rxcs	= ^x00000000
pr$_txcs	= ^x00000080
pr$_mapen	= ^x00000000
ctx_base plus ctx$1_psl	= ^x041F0000

4. Flushes and disables the chip cache.
5. Initializes the SSC.
6. Initializes the console state:

   Sets the current and previous reference to PHYSICAL and LONG at address 0; current datum is then 0; clears the exit flag:

cs_base plus cs$1_address	= 0
cs_base plus cs$1_prev_address	= 0
cs_base plus cs$1_datus_size	= 4
cs_base plus cs$q_datum	= 0
cs_base plus cs$1_qv plus 4*qual$v_n	= 0
ctx_base plus ctx$b_exit	= 0

Example:

```
>>> init
>>>
```

3.10.12 MOVE

The MOVE command copies the block of memory starting at the source address to a block beginning at the destination address. Typically, this command has an /N qualifier so that more than one datum is transferred. The destination correctly reflects the original contents of the source, regardless of the overlap between the source and the data.

The MOVE command actually performs byte, word, longword, and quadword reads and writes as needed in the process of moving the data. Moves are supported only for the physical and virtual address spaces.

Format:

MOVE [qualifier-list] {src_address} {dest_address}

Qualifiers:

Data control: /B, /W, /L, /Q, /N:{count}, /STEP:{size},

Address space control: /V, /U, /P

Arguments:

{src_address} A longword address that specifies the first location of the source data to be copied.

{dest_address} A longword address that specifies the destination of the first byte of data. These addresses may be an actual address or a symbolic address. If no address is specified, + is assumed.

Examples:

```
>>> EXAMINE/N:4 0  ! Observe destination.
P 00000000 00000000
P 00000004 00000000
P 00000008 00000000
P 0000000C 00000000
P 00000010 00000000

>>> EXAMINE/N:4 200 ! Observe source data.
P 00000200 58DD0520
P 00000204 585E04C1
P 00000208 00FF8FBB
P 0000020C 5208A8D0
P 00000210 540CA8DE

>>> MOVE/N:4 200 0  ! Move the data.

>>> EXAMINE/N:4 0   ! Observe moved data.
P 00000000 58DD0520
P 00000004 585E04C1
P 00000008 00FF8FBB
P 0000000C 5208A8D0
P 00000010 540CA8DE
>>>
```

3.10.13 NEXT

The NEXT command executes the specified number of macro instructions. If no count is specified, 1 (one) is assumed.

After the last macro instruction is executed, the console reenters maintenance mode.

Format:

NEXT [count]

The console implements the NEXT command, using the trace trap enable and trace pending bits in the PSL and the trace pending vector in the SCB. The following restrictions apply:

- If memory management is enabled, the NEXT command works only if the first page in SSC RAM is mapped in S0 (system) space.
- Overhead associated with the NEXT command affects execution time of an instruction.
- The NEXT command elevates the IPL to 31 for long periods of time (milliseconds) while single stepping over several commands.
- Unpredictable results occur if the macro instruction being stepped over modifies either the SCBB or the trace trap entry. This means that you cannot use the NEXT command in conjunction with other debuggers.

Arguments:

[count] A value representing the number of macro instructions to execute.

Examples:

```
>>> EXAMINE PC
  G 0000000F 00000200
>>> NEXT
    PC = 00000202
>>> NEXT 4
    PC = 00000213
>>>
```

```
>>> EXAMINE/INS/N:10 0
  P 00000000   01 NOP
  P 00000001   01 NOP
  P 00000002   01 NOP
  P 00000003   01 NOP
  P 00000004   01 NOP
  P 00000005   01 NOP
  P 00000006   01 NOP
  P 00000007   01 NOP
  P 00000008   11 BRB     00000002
  P 0000000A   01 NOP
  P 0000000B   01 NOP
  P 0000000C   00 HALT
  P 0000000D   00 HALT
  P 0000000E   00 HALT
  P 0000000F   00 HALT
  P 00000010   00 HALT
  P 00000011   00 HALT

>>> DEP PC 0

>>> N
  P 00000001   01 NOP
>>> N
  P 00000002   01 NOP
>>> N
  P 00000003   01 NOP
>>> N
  P 00000004   01 NOP
>>> N
  P 00000005   01 NOP
>>> N 5
  P 00000006   01 NOP
  P 00000007   01 NOP
  P 00000008   11 BRB     00000002
  P 00000002   01 NOP
  P 00000003   01 NOP
```

3.10.14 REPEAT

The REPEAT command repeatedly displays and executes the specified command. Press Ctrl/C to stop the command. You can specify any valid console command except the REPEAT command.

Format:

REPEAT {command}

Arguments:

{command} A valid console command other than REPEAT.

Examples:

```
>>> REPEAT EXAMINE PR$_TODR ! Watch the clock.
  I 0000001B 5AFE78CE
  I 0000001B 5AFE78D1
  I 0000001B 5AFE78FD
  I 0000001B 5AFE7900
  I 0000001B 5AFE7903
  I 0000001B 5AFE7907
  I 0000001B 5AFE790A
  I 0000001B 5AFE790D
  I 0000001B 5AFE7910
  I 0000001B 5AFE793C
  I 0000001B 5AFE793F
  I 0000001B 5AFE7942
  I 0000001B 5AFE7946
  I 0000001B 5AFE7949
  I 0000001B 5AFE794C
  I 0000001B 5AFE794F
  I 0000001B 5^C
>>>
```

3.10.15 SEARCH

The SEARCH command finds all occurrences of a pattern and reports the addresses where the pattern was found. If the /NOT qualifier is present, the command reports all addresses in which the pattern did not match.

Format:

SEARCH [qualifier_list] {address} {pattern} [{mask}]

SEARCH accepts an optional mask that indicates bits to be ignored (*don't care* bits). For example, to ignore bit 0 in the comparison, specify a mask of 1. The mask, if not present, defaults to 0.

A match occurs if (pattern AND mask complement) = (data AND mask complement), where:

pattern is the target data
mask is the optional don't care bitmask (which defaults to 0)
data is the data at the current address

SEARCH reports the address under the following conditions:

/NOT Qualifier	Match Condition	Action
Absent	True	Report address
Absent	False	No report
Present	True	No report
Present	False	Report address

The address is advanced by the size of the pattern (byte, word, longword, or quadword), unless overridden by the /STEP qualifier.

Qualifiers:

Data control: /B, /W, /L, /Q, /N:{count}, /STEP:{size}

Address space control: /P, /V, /U

Command-specific:

/NOT Inverts the sense of the match.

Arguments:

{start_address}	A longword address that specifies the first location subject to the search. This address can be an actual address or a symbolic address. If no address is specified, + is assumed.
{pattern}	The target data.
[{mask}]	A longword containing the bits desired in the comparison.

Examples:

```
>>> DEPOSIT /P/L/N:1000 0 0 !Clear some memory.
>>>
>>> DEPOSIT 300 12345678 !Deposit some search data.
>>> DEPOSIT 401 12345678
>>> DEPOSIT 502 87654321
>>>

>>> SEARCH /N:1000 /ST:1 0 12345678 !Search for all occurrences
  P 00000300 12345678    !of 12345678 on any byte
  P 00000401 12345678    !boundary.

>>> SEARCH /N:1000 0 12345678 !Try on longword boundaries.
  P 00000300 12345678

>>> SEARCH /N:1000 /NOT 0 0 !Search for all nonzero
  P 00000300 12345678    !longwords.
  P 00000400 34567800
  P 00000404 00000012
  P 00000500 43210000
  P 00000504 00008765

>>> SEARCH /N:1000 /ST:1 0 1 FFFFFFFE  !Search for odd longwords
  P 00000502 87654321     !on any boundary.
  P 00000503 00875543
  P 00000504 00008765
  P 00000505 00000087

>>> SEARCH /N:1000 /B 0 12 !Search for all occurrences
  P 00000303 12        !of the byte 12.
  P 00000404 12

>>> SEARCH /N:1000 /ST:1 /W 0 FE11  !Search for all words that
>>>                                 !could be interpreted as a
>>>                                 !spin (10$:brb 10$).
>>>                                 !None were found.
```

3.10.16 SET

The SET command sets the parameter to the value you specify.

Format:

SET {parameter} {value}

Parameters:

BFLAG Set the default R5 boot flags. The value must be a hex number of up to 8 digits. See Table 3–4, VMB Boot Flags, for a list of the boot flags.

BOOT Set the default boot device. The value must be a valid device name as specified in the BOOT command description, Section 3.10.1.

HOST Connect to the DUP or MAINTENANCE driver on the selected node or device. Note the hierarchy of the SET HOST qualifiers below.

/DUP—Use the DUP driver to execute local programs of a device on either the DSSI bus or the Q22-bus.

/DSSI node—Attach to the DSSI node. A node is a name up to 8 characters in length or a number from 0 to 7.
/UQSSP—Attach to the UQSSP device specified using one of the following methods:

/DISK n—Specifies the disk controller number, where n is a number from 0 to 255. The resulting fixed address for n=0 is 20001468 for the first TMSCP controller, and the floating rank for n>0 is 26.
/TAPE n—Specifies the tape controller number, where n is a number from 0 to 255. The resulting fixed address for n=0 is 20001940 for the first MSCP controller, and the floating rank for n>0 is 30.
csr_address—Specifies the Q22-bus I/O page CSR address for the device.

/MAINTENANCE—Examines and modifies DSSI controller module configuration values. Does not accept a task value.

/UQSSP—

/SERVICE n—Specifies service for DSSI controller module n, where n is a value from 0 to 3. (The resulting fixed address of a DSSI controller module in maintenance mode is 20001910+4*n.)
/csr_address—Specifies the Q22-bus I/O page CSR address for the DSSI controller module.

LANGUAGE Sets console language and keyboard type. If the current console terminal does not support the Digital Multinational Character Set (MCS), then this command has no effect and the console message appears in English. Values are 1 through 15. Refer to Example 3–1 for the languages you can select.

Qualifiers: Listed in the parameter descriptions above.

Examples:

```
>>> SET BFLAG 220    ! Sets boot flags 5 and 9 (See boot flag
          ! table in the BOOT command description.)
>>> SET BOOT DUA0

>>> SET HOST/DUP/DSSI 0
Starting DUP server...

DSSI Node 0 (SUSAN)
DRVEXR V1.0  D 2-JUN-1989 10:01:35
DRVTST V1.0  D 2-JUN-1989 10:01:35
HISTRY V1.0  D 2-JUN-1989 10:01:35
ERASE  V1.0  D 2-JUN-1989 10:01:35
PARAMS V1.0  D 2-JUN-1989 10:01:35
DIRECT V1.0  D 2-JUN-1989 10:01:35
Copyright © 1988  Digital Equipment Corporation

Task Name? PARAMS
Copyright © 1988  Digital Equipment Corporation

PARAMS> STAT PATH

ID  Path Block     Remote Node    DGS_S  DGS_R   MSGS_S  MSGS_R
-- ------------  ---------------  ------ ------  ------  ------
0  PB  FF811ECC   Internal Path     0      0       0       0
1  PB  FF8120D4  KAREN  RFX V101    0      0       0       0
4  PB  FF8121D8  WILMA  RFX V101    0      0       0       0
5  PB  FF8120DC  BETTY  RFX V101    0      0       0       0
2  PB  FF8122E0  DSSI1  VMS V5.0    0      0      816    3045
3  PB  FF8124E4  3      VMB BOOT    0      0       50      52

PARAMS> EXIT
Exiting...

Task Name?

Stopping DUP server...
```

```
>>> SET HOST/DUP/DSSI 0 PARAMS
Starting DUP server...

DSSI Node 0 (SUSAN)
Copyright © 1988  Digital Equipment Corporation

PARAMS> SHOW NODE

Parameter     Current        Default     Type     Radix
--------- --------------  ------------  --------  -----
NODENAME      SUSAN          RF30        String   Ascii  B

PARAMS> SHOW ALLCLASS

Parameter     Current        Default     Type     Radix
--------- --------------  ------------  --------  -----
ALLCLASS         1              0        Byte     Dec  B

PARAMS> EXIT
Exiting...

Stopping DUP server...
>>>
```

3.10.17 SHOW

The SHOW command displays the console parameter you specify.

Format:

SHOW {parameter}

Parameters:

BFLAG Displays the default R5 boot flags.

BOOT Displays the default boot device.

DEVICE Displays all devices displayed by the SHOW DSSI, SHOW ETHERNET, SHOW SCSI, SHOW UQSSP, and SHOW VME commands.

DSSI Displays the status of all nodes that can be found on the DSSI bus. For each node on the DSSI bus, the firmware displays the node number, the node name, and the boot name and type of the device, if available. The command does not indicate whether the device contains a bootable image.

The node that issues the command is listed with a node name of * (asterisk).

The device information is obtained from the media type field of the MSCP command GET UNIT STATUS. If a node is not running or is not capable of running an MSCP server, then no device information is displayed.

ETHERNET Displays hardware Ethernet address for all Ethernet adapters that can be found, both on-board and on the Q22-bus. Displays as blank if no Ethernet adapter is present.

LANGUAGE Displays console language and keyboard type. Refer to the corresponding SET LANGUAGE command for the meaning.

MEMORY Displays main memory configuration board-by-board, if available. (See Section 4.8.9.)

/FULL—Additionally, displays the normally inaccessible areas of memory, such as the PFN bitmap pages, the console scratch memory pages, the Q22-bus scatter-gather map pages. Also reports the addresses of bad pages, as defined by the bitmap.

QBUS Displays all Q22-bus I/O addresses that respond to an aligned word read, and vector and device name information. For each address, the console displays the address in the VAX I/O space in hex, the address as it would appear in the Q22-bus I/O space in octal, and the word data that was read in hex. Also displays the vector that you should set up and device name(s) that could be associated with the CSR.

This command may take several minutes to complete. Press Ctrl/C to terminate the command. During execution, the command disables the scatter-gather map so that it can search for memory on the Q-bus.

SCSI — Displays the status of all nodes that can be found on the SCSI bus. For each node, the firmware displays the node number and the boot name and type of the device, if available. The command does not indicate whether the device contains a bootable image.

/*FULL*—Displays the boot path, device type, device capacity, product ID, revision, and fixed or removable.

UQSSP — Displays the status of all disks and tapes that can be found on the Q22-bus that support the UQSSP protocol. For each such disk or tape on the Q22-bus, the firmware displays the controller number, the controller CSR address, and the boot name and type of each device connected to the controller. The command does not indicate whether the device contains a bootable image.

This information is obtained from the media type field of the MSCP command GET UNIT STATUS. The console does not display device information if a node is not running (or cannot run) an MSCP server.

VERSION — Displays the current firmware version.

VME — Displays presence of VME interface board.

Qualifiers: Listed in the parameter descriptions above.

Examples:

```
>>> SHOW BFLAG
00000220

>>> SHOW BOOT
DUA0

>>> SHOW DEVICE
DSSI Node 0 (R7YRMS)
 -rf(0,0,*)  (RF71)

DSSI Node 7 (*)

SCSI Node 0
 -rz(0,0,*) (RZ56 )

SCSI Node 7 (*)

UQSSP Tape Controller 0 (774500)
 -tm(0,0)  (TK70) -MUA0

Ethernet Adapter
 -mop() -EZA0  (08-00-2B-12-81-22)

Ethernet Adapter 0 (774440)
        -XQA0 (08-00-2B-08-CB-5C)

VME Interface Board - Not Installed
>>>
```

```
>>> SHOW DSSI
DSSI Node 0 (SUSAN)
-rf(0,0,*) (RF71)

DSSI Node 1 (KAREN)
-rf(1,1,*) (RF71)

DSSI Node 7 (*)

>>> SHOW ETHERNET
Ethernet Adapter
 -mop() -EZA0  (08-00-2B-12-81-22)

Ethernet Adapter 0 (774440)
        -XQA0 (08-00-2B-08-CB-5C)
>>>

>>> SHOW LANGUAGE
English (United States/Canada)

>>> SHOW MEMORY
Memory 1: 00000000 to 01FFFFFF, 32MB, 0 bad pages
Memory 2: 02000000 to 03FFFFFF, 32MB, 0 bad pages
Memory 3: 04000000 to 05FFFFFF, 32MB, 0 bad pages
Memory 4: 06000000 to 07FFFFFF, 32MB, 0 bad pages

Total of 128MB, 0 bad pages, 160 reserved pages
>>>

>>> SHOW MEMORY/FULL
Memory 1: 00000000 to 01FFFFFF, 32MB, 0 bad pages
Memory 2: 02000000 to 03FFFFFF, 32MB, 0 bad pages
Memory 3: 04000000 to 05FFFFFF, 32MB, 0 bad pages
Memory 4: 06000000 to 07FFFFFF, 32MB, 0 bad pages

Total of 128MB, 0 bad pages, 160 reserved pages

Memory Bitmap
-07FEC000 to 07FF3FFF, 64 pages

Console Scratch Area
-07FF4000 to 07FF7FFF, 32 pages

Qbus Map
-07FF8000 to 07FFFFFF, 64 pages

Scan of Bad Pages
>>>
```

```
>>> SHOW QBUS
Scan of Qbus I/O Space
-20001468 (772150) = 4000 (154) RQDX3/KDA50/RRD50/RQC25/X-DISK
-2000146A (772152) = 0B40
-20001940 (774500) = 0000 (260) TQK50/TQK70/TU81E/RV20/X-TAPE
-20001942 (774502) = 0BC0
-20001F40 (777500) = 0020 (004) IPCR

Scan of Qbus Memory Space

>>> SHOW SCSI

SCSI Node 0
 -tz(0,0,*) (TLZ04) -DIA0

SCSI Node 1
 -rz(0,1,*) (RZ56 )

SCSI Node 2
 -rz(0,2,*) (RRD40) -DIA2

SCSI Node 4
 -tz(0,4,*) (.....) -DIA4

SCSI Node 7 (*)

>>>

>>> SHOW SCSI/FULL

Boot  Path    Dev    Cap (in Hex)       Product Id      Revs    r/f
-------------------------------------------------------------------
-tl(0,0,*)   TAPE       4B0 MBs      TLZ04 1989(C)DEC   0304     r
-rz(0,1,*)   DISK       27A MBs      RZ56     (C) DEC   0200     f
-rz(0,2,*)   CDROM      23B MBs      RRD40     TM DEC   250E     r
-tz(0,4,*)   TAPE        5A MBs      ................   ....     r

SCSI Node 7

>>>

>>> SHOW UQSSP
UQSSP Tape Controller 0 (772150)
-MUA0 -tm(0,0) (TK70)

>>> SHOW VERSION
KN220-A Vn.n
>>>
```

3.10.18 START

The START command starts instruction execution at the address you specify. If no address is given, the current PC is used. If memory mapping is enabled, macro instructions are executed from virtual memory, and the address is treated as a virtual address. The START command is equivalent to a DEPOSIT to PC, followed by a CONTINUE. It does not perform a processor initialization.

Format:

START [address]

Arguments:

[address]	The address at which to begin execution. This address is loaded into the user's PC.

Example:

```
>>> START  1000
```

3.10.19 TEST

The TEST command invokes a diagnostic test program specified by the test number. If you enter a test number of 0 (zero), all tests allowed to be executed from the console terminal are executed. The console accepts an optional list of up to five additional hexadecimal arguments.

Refer to Chapter 4 for a detailed explanation of the diagnostics.

Format:

TEST {test_number} [test_arguments]

Arguments:

{test_number}	A two-digit hex number specifying the test to be executed.
[test_arguments]	Up to five additional test arguments. These arguments are accepted but they have no meaning to the console.

Example:

```
>>> TEST 0 !Execute the power-up diagnostic script.

83..82..81..80..79..78..77..76..75..74..73..72..71..70..69..68..67..
66..65..64..63..62..61..60..59..58..57..56..55..54..53..52..51..50..
49..48..47..46..45..44..43..42..41..40..39..38..37..36..35..34..33..
32..31..30..29..28..27..26..25..24..23..22..21..20..19..18..17..16..
15..14..13..12..11..10..09..08..07..06..05..04..03..
Tests completed.
>>>
```

3.10.20 UNJAM

The UNJAM command performs an I/O bus reset, by writing a 1 (one) to IPR 55 (decimal).

Format:

UNJAM

Example:

```
>>> UNJAM
>>>
```

3.10.21 X—Binary Load and Unload

The X command is for use by automatic systems communicating with the console.

The X command loads or unloads (that is, writes to memory or reads from memory) the specified number of data bytes through the console serial line (regardless of console type) starting at the specified address.

Format:

X {address} {count} CR {line_checksum} {data} {data_checksum}

If bit 31 of the count is clear, data is received by the console and deposited into memory. If bit 31 is set, data is read from memory and sent by the console. The remaining bits in the count are a positive number indicating the number of bytes to load or unload.

The console accepts the command upon receiving the carriage return. The next byte the console receives is the command checksum, which is not echoed. The command checksum is verified by adding all command characters, including the checksum and separating space (but not including the terminating carriage return, rubouts, or characters deleted by rubout), into an 8-bit register initially set to zero. If no errors occur, the result is zero. If the command checksum is correct, the console responds with the input prompt and either sends data to the requester or prepares to receive data. If the command checksum is in error, the console responds with an error message. The intent is to prevent inadvertent operator entry into a mode where the console is accepting characters from the keyboard as data, with no escape mechanism possible.

If the command is a load (bit 31 of the count is clear), the console responds with the input prompt (>>>), then accepts the specified number of bytes of data for depositing to memory, and an additional byte of received data checksum. The data is verified by adding all data characters and the checksum character into an 8-bit register initially set to zero. If the final content of the register is nonzero, the data or checksum is in error, and the console responds with an error message.

If the command is a binary unload (bit 31 of the count is set), the console responds with the input prompt (>>>), followed by the specified number of bytes of binary data. As each byte is sent, it is added to a checksum register initially set to zero. At the end of the transmission, the two's complement of the low byte of the register is sent.

If the data checksum is incorrect on a load or if memory or line errors occur during the transmission of data, the entire transmission is completed, then the console issues an error message. If an error occurs during loading, the contents of the memory being loaded are unpredictable.

The console represses echo while it is receiving the data string and checksums.

The console terminates all flow control when it receives the carriage return at the end of the command line in order to avoid treating flow control characters from the terminal as valid command line checksums.

Using control characters (Ctrl/C, Ctrl/S, Ctrl/O, and so on), you can control the console serial line during a binary unload. You cannot control the console serial line during a binary load, since all received characters are valid binary data.

The console has the following timing requirements:

- It must receive data being loaded with a binary load command at a rate of at least one byte every 60 seconds.
- It must receive the command checksum that precedes the data within 60 seconds of the carriage return that terminates the command line.
- It must receive the data checksum within 60 seconds of the last data byte.

If any of these timing requirements is not met, the console aborts the transmission by issuing an error message and returning to the console prompt.

The entire command, including the checksum, can be sent to the console as a single burst of characters at the specified character rate of the console serial line. The console is able to receive at least 4 Kbytes of data in a single X command.

3.10.22 ! (Comment)

The comment character (an exclamation point) is used to document command sequences. It can appear anywhere on the command line. All characters following the comment character are ignored.

Format: **!**

Example:

```
>>> ! The console ignores this line.
>>>
```

Chapter 4

KN220 Troubleshooting and Diagnostics

4.1 Introduction

This chapter contains a description of KN220 ROM-based diagnostics, acceptance test and troubleshooting procedures, diagnostics, and power-up self-tests for common options.

NOTE: *Check the status of the nonvolatile RAM(NVRAM) disk cache and save the cache if necessary. Store the NVRAM and check the service history before removing any modules. See Appendix D.*

4.1.1 General Troubleshooting Procedures

Before troubleshooting any system problem, check the site maintenance guide for the system's service history. Ask the system manager two questions:

- Has the system been used before and did it work correctly?
- Have changes been made to the system recently?

Three problems commonly occur when you make a change to the system:

- Incorrect cabling
- Module configuration errors (incorrect CSR addresses and interrupt vectors)
- Incorrect grant continuity

Most communications modules use floating CSR addresses and interrupt vectors. If you remove a module from the system, you may have to change the addresses and vectors of other modules. *Microsystems Options* lists address and vector values for most options.

If you change the system configuration, run the Configure utility at the maintenance mode prompt (>>>) to determine the CSR addresses and interrupt vectors recommended by Digital.

See *MicroVAX Diagnostic Monitor User's Guide* for information about the Connect and Ignore commands, which are used to set up MDM for testing nonstandard configurations.

See Appendix A for a summary of ULTRIX–32 Exerciser and uerf commands to help you troubleshoot and diagnose errors. When troubleshooting, note the location of cables and connectors before you perform each step. Label cables before you disconnect them to save time and prevent you from introducing new problems.

If the system fails (or appears to fail) to boot the operating system, check the console terminal screen for an error message. If the terminal displays an error message, see Section 4.2. Check the LED display on the device you suspect is bad. If no errors are indicated by the device LED display, run the ROM-based diagnostics described in this chapter. In addition, check the following:

- If self-test fails, check the module interconnect cabling.
- If no message appears, make sure that the console terminal and the system are on. Check the on/off power switch on both the console terminal and the system. If the terminal has a DC OK LED, be sure it is on.
- Check the cabling to the console terminal.
- If you cannot get a display of any kind on the console terminal, try another terminal.
- If the system DC OK LED remains off, check the power supply and power supply cabling.
- Check the hex display on the H3602–AC. If the display is off, check the I/O module LED display and the CPU cabling. If a hex error message appears on the H3602–AC or the module, see Section 4.2.

If the system boots successfully, but a device seems to fail or an intermittent failure occurs, check the error log first for a device problem. The failing device is usually in one of the following areas:

- CPU
- Memory
- Mass storage
- Communications devices

4.2 KN220 ROM-Based Diagnostics

The KN220 ROM-based diagnostic facility, rather than the MicroVAX Diagnostic Monitor (MDM), is the primary diagnostic tool for troubleshooting and testing of the CPU, memory, Ethernet, SCSI, and DSSI subsystems. ROM-based diagnostics have significant advantages:

- Load time is virtually nonexistent.
- The boot path is more reliable.
- Diagnosis is done in a more primitive state. (MDM requires successful loading of the operating system.)

The ROM-based diagnostics can indicate several different FRUs, not just the CPU or I/O modules. (Table 4–7 lists the FRUs indicated by ROM-based diagnostic error messages.)

The diagnostics run automatically on power-up. While the diagnostics are running, the LEDs on the H3602–AC display a hexadecimal countdown of the tests from F to 4 (though not in precise reverse order) before booting the operating system, and 2 to 0 while booting the operating system. A different countdown appears on the console terminal.

The ROM-based diagnostics are a collection of individual tests with parameters that you can specify. A data structure called a *script* points to the tests. (See Section 4.2.2.) There are several field and manufacturing scripts. Qualified Customer Services personnel can also create their own scripts interactively.

A program called the *diagnostic executive* determines which of the available scripts to invoke. The script sequence varies if the KN220 is in a manufacturing environment. The diagnostic executive interprets the script to determine what tests to run, the correct order in which to run the tests, and the correct parameters to use for each test.

The diagnostic executive also controls tests so that errors can be detected and reported. It also ensures that when the tests are run, the machine is left in a consistent and well-defined state.

4.2.1 Diagnostic Tests

To get the list of ROM-based utilities, enter `T 9E` at the maintenance mode prompt (T is the abbreviation of TEST). The column headings have the following meanings:

- Test is the test code or utility code.
- Address is the test or utility's base address in ROM. This address varies. The addresses shown are examples only. If a test fails, entering T FE displays diagnostic state to the console. You can subtract the base address of the failing test from the last_exception_pc to find the index into the failing test's diagnostic listing.
- Name is a brief description of the test or utility.
- Parameters shows the parameters for each diagnostic test or utility. Tests accept up to 20 parameters. The asterisks (*) represent parameters that are used by the tests but that you cannot specify individually. These parameters are encoded in ROM and are provided by the diagnostic executive.

```
>>>T 9E

Test
#   Address   Name               Parameters
-------------------------------------------------------------------------------
    2004CA00  CP_SCB
    2004DCA8  De_executive
20  200583BC  R_D_Cache_Seg      ***
21  200583D4  R_D_Cache_Tag      ***
22  200583EC  R_D_Cache_Tag_Par  ***
23  20058408  R_D_Cache_Data_Par ***
24  20058425  R_D_Cache_Val_Bit  ***
25  20058441  R_D_Cache_RAM      ***
26  20058472  R_I_Cache_Seg      ***
27  2005848A  R_I_Cache_Tag      ***
28  200584A2  R_I_Cache_Tag_Par  ***
29  200584BE  R_I_Cache_Data_Par ***
2A  200584DB  R_I_Cache_Val_Bit  ***
2B  200584F7  R_I_Cache_RAM      ***
2C  2005850F  R_I_Cache_Inst     ***
2D  20058459  R_D_Cache_Inst     ***
2E  20058528  R_Cache_IStream    ***
2F  20058542  R_Cache_Exerciser  ***
30  20058058  MEM_Bitmap         *
34  2004F25C  ROM logic test     *
40  2005774C  Memory Count Errs  Soft_errs_allowed *******
41  20055FFC  CDE Cleanup        ******
42  2005617A  Check_for_intrs    ******
43  20057BD0  CVAX Mem Interface start_add end_add inc pat sel_tst
                                 sav_map ****
44  200561F8  CVAX Cache mem     addr_incr *********
45  20050490  C_cache_mem_cqbic  start_addr end_addr addr_incr ****
46  20050790  C_Cache_diag_mode  addr_incr *********
47  2005822B  MEM_Refresh        start end incr cont_on_err
                                 time_seconds
48  200581B4  MEM_Addr_shrts     start_add end_add * cont_on_err
                                 pat_2 pat_3
4C  2005819C  MEM_ECC_Error      start_add end_add add_incr
                                 cont_on_err
4D  2005811E  MEM_Address        start_add end_add add_incr
                                 cont_on_err
4E  2005810B  MEM_Byte           start_add end_add add_incr
                                 cont_on_err
4F  200580F8  MEM_Data           start_add end_add add_incr
                                 cont_on_err
52  2004FEEF  Prog timer         which_timer wait_time_us ***
53  200501D0  TOY clock          repeat_test_250ms_ea Tolerance ***
54  20055D19  Virtual mode       *********
55  20050387  Interval timer     which_timer **
56  20051624  SII_ext_loopbck    ***
57  20054124  SII_memory         incr test_pattern *******
```

```
58  20053D40  DSSI reset         port_no time_secs *
59  200557B8  SGEC_LPBCK_ASSIST  time_secs **
5B  2005450D  SII_registers      ****
5C  2005192F  SII_initiatior     ******
5D  200525DD  SII target         *******
5F  20054B08  SGEC               loopback_type no_ram_tests ******
60  2004FB6E  SSC Console Serial start_baud end_baud ******
65  20054139  SCSI_memory        incr test_pattern *******
66  20058834  R_SCSI_Test        **********
67  200546A8  SCSI Quick test    **********
71  2005855E  R3000_FPU          *
72  20058572  R3000_TLB          *
73  20058586  R_IO_Reg_Interface loop ***
74  200585A3  R_SII_Buf_Intrf    start_add end_add incr patrn **
75  200585BD  R_SCSI_Buf_Intrf   start_add end_add incr patrn **
76  200585D8  R3000_Interrupt    *
77  200585F2  R_Mem_Moving_Inver start_add end_add incr ***
78  2005860F  R3000_Write_Buffer *
79  2005862C  R3000_NVRAM        start_add end_add
7A  20058642  R3000_NVRAM_all    start_add end_add
7C  2005865C  R3000_DUART        *
7D  20058672  R3000_Interaction  loop_cnt sel_dev
80  2004F424  CQBIC_memory       **********
81  20050E2F  MSCP-QBUS test     IP_csr ******
82  20051003  DELQA              device_num_addr ****
83  200588D4  VME Test           *
90  2004F39E  CQBIC registers    *
91  2004F32C  CQBIC Powerup      **
92  20057988  CDAL_RIO Intrf     *****
9A  2005806F  Memory Descrip     Board1 Bd2 Bd3 Bd4 verify_only
9B  20057890  Init_memory_32MB   *
9C  20056662  List Registers     *
9D  2005655E  Utilities          Expnd_err_msg get_mode init_LEDs
                                 clr_ps_cnt
9E  20050E07  List diags         *
9F  20056FDF  Create Script      *******
C1  200511E4  SSC RAM            *
C2  200513B0  SSC RAM ALL        *
C5  20051529  SSC regs           *
C6  2004F0E0  SSC_powerup        *********
C7  2004F1A1  CBTCR timeout      ***
```

```
Scripts
#   Description

A0  Soft Script, created by 9F
A1  Powerup field
A3  Manuf FV
A4  Manuf Loop on A3
A7  Memory tests, called by A8
A8  Field memory acceptance, mark Hard SBEs
A9  Run memory tests, stop on any error
B3  Manuf APT, Do NOT use in field

>>>
```

Parameters that you can specify are written out, as shown in the following examples:

```
54  2004E557   Virtual mode    ******
52  2004FF38   Prog_timer      which_timer,wait_time_us***
```

The virtual mode test on the first line contains several parameters, but you cannot specify any of them. To run this test individually, enter:

```
>>> T 54
```

The Prog_timer test on the second line accepts 5 parameters, but you can specify only the first two to test the programmable timer.

```
>>> T 52 0 10
```

4.2.2 Scripts

Most of the tests shown by utility 9E are arranged into scripts. A *script* is a data structure that points to various tests and defines the order in which they are run. Different scripts can run the same set of tests, but in a different order and/or with different parameters and flags. A script also contains the following information:

- The parameters and flags that need to be passed to the test.
- Where the tests can be run from. For example, certain tests can be run only from EPROM. Other tests are program independent code and can be run from EPROM, cache diagnostic space, or main memory to enhance execution speed.
- What is to be shown, if anything, on the console.
- What is to be shown, if anything, in the LED display.
- What action to take on errors (halt, repeat, continue).

The power-up script runs every time the system is powered on. You can also invoke the power-up script at any time by entering T 0. Some of the common scripts are listed in test 9E.

```
Scripts
#   Description

A0  Soft script, created by 9F
A1  Powerup field
A3  Manuf FV
A4  Manuf Loop on A3
A7  Memory tests, called by A8
A8  Field memory acceptance, mark Hard SBEs
A9  Run memory tests, stop on any error
B3  Manuf APT, Do NOT use in field
```

Additional scripts are included in the ROMs for use in manufacturing and engineering environments.

Customer Services personnel can run scripts and tests individually, using the T command. When doing so, note that certain tests may be dependent upon a state set up from a previous test. For this reason, you should use the UNJAM and INITIALIZE commands, described in Chapter 3, before running an individual test after the operating system has crashed or has been halted. You do not need to use these commands on system power-up, however, because system power-up leaves the machine in a defined state.

Customer Services personnel with a detailed knowledge of the KN220 hardware and firmware can also create their own scripts by using the 9F utility. (See Section 4.2.4.)

Table 4–1 lists the scripts that are available to Customer Services.

Table 4–1: Scripts Available to Customer Services

Script[1]	Enter with TEST Command	Description
A0	A0	Soft script created by de_test9f. Enter T 9F to create.
A1	A1, B8, 0, 3	Common section of power-up script. Script B8 invokes this script at power-up.
A7	A7, A8	Memory test portion invoked by script A8. Reruns the memory tests without rebuilding and reinitializing the bitmap. Run script A8 once before running script A7 separately to allow mapping out of both single-bit and double-bit main memory ECC errors.
A8	A8	Memory acceptance. Running script A8 with script A7 tests main memory more extensively. It enables hard single-bit and multibit main memory ECC errors to be marked bad in the bitmap. Invokes script A7 when it has completed its tests.
A9	A9	Memory tests. Halts and reports the first error. Does not reset the bitmap or busmap.
AD	AD	Console program. Runs memory tests, marks bitmap, resets busmap, and resets caches. Calls script AE.
AE	AE, AD	Console program. Resets board registers and caches.
AF	AF	Console program. Resets busmap and resets caches.
B1	2, B8	Initial power-up script for console SLU before first console announcement. Invoked at power-up.

[1]Scripts A2–A6, B0, and B2–B5 are for manufacturing use. They should not be used by Customer Services. Scripts A2, A5, A6, AA, AB, AC, B2, B4, B7, BA–BF are unused. Scripts A3, A4, B0, B1, B3, B5, B6, and B9 are for manufacturing.

In most cases, Customer Services needs only the scripts shown in Table 4–2 for effective troubleshooting and acceptance testing.

Table 4–2: Commonly Used Customer Services Scripts

Command	Description
0	Automatically invokes the proper scripts; runs the same tests as during power-up.
A9	Primarily runs the memory tests; halts upon first hard or soft error.
A8	Memory acceptance script; marks hard multibit and single-bit ECC errors in the bitmap. Script A8 calls script A7 when this command is entered. Script A7 contains the memory tests that will continue on error.
A7	Can be run by itself; useful when you want to bypass the bitmap test.
A1	Power-up script that can be run by itself.

4.2.3 Script Calling Sequence

Actions at Power-Up

In a nonmanufacturing environment where the intended console device is the serial line unit (SLU), the console program (referred to as CP below) performs the following actions at power-up:

1. Runs the IPT.
2. Assumes console device is SLU.
3. Calls the diagnostic executive (DE) with Test Code = 2.
   a. DE determines that the environment is nonmanufacturing from H3602–AC. (Manufacturing sets a jumper on the H3602–AC for testing.)
   b. DE selects script sequence for console SLU.
   c. DE executes script B1. Script B1 directs DE to execute test. (Console announcements are off.)
   d. DE passes control back to the CP.
4. Establishes SLU as console device (whether or not SLU test passed).
5. Prints banner message.
6. Displays language inquiry menu on console if console supports MCS *and* any of the following is true:

a. Battery is dead.

b. H3602–AC switch set to action (language inquiry).

c. Contents of SSC NVRAM are invalid.

7. Calls DE with Test Code = 3.

   a. DE executes script A1. (Announcements are on.)

   b. DE passes control back to CP.

8. CP issues end message and >>> prompt. CP may exit to >> prompt.

Actions After You Enter T 0

In a nonmanufacturing environment where the intended console device is the SLU, the console program (CP) performs the following actions after you enter T 0 at the console prompt (>>> T 0):

1. Calls the diagnostic executive (DE) with Test Code = 0.

   a. DE determines environment is nonmanufacturing from H3602–AC switch setting.

   b. DE executes script B8.

      Script B8 directs DE to execute scripts B1 and A1.

      i Script B1 directs DE to execute tests. (Console announcements are off.)

      ii Script A1 directs DE to execute tests. (Console announcements are on.)

   c. DE passes control back to the CP.

2. CP prints end message and >>> prompt. Console may exit to >> prompt.

Note that although the sequence of actions is different in the two cases above, the same tests (those in scripts B1 and A1) are run both times.

4.2.4 Creating Scripts

You can create your own script, using utility 9F, to control the order in which tests are run and to select specific parameters and flags for individual tests. In this way you do not have to use the defaults provided by the hard-wired scripts.

Utility 9F also provides an easy way to see what flags and parameters are used by the diagnostic executive for each test.

Run test 9F first to build the user script (see Example 4–1). Press Return to use the default parameters or flags, which are shown in parentheses. Test 9F prompts you for the following information:

- Script location. The script can be located in the 1-Kbyte NVRAM in the SSC, in the 128-Kbyte mass storage interface (MSI) RAM, or in main memory. A script is limited by the size of the data structure that contains it. A larger script can be developed in main memory than in MSI RAM, and a larger script can be built in MSI RAM than in SSC RAM.

  A script cannot, however, always be located in main memory. For example, a script that runs memory tests will overwrite the user script, since the diagnostic executive cannot relocate the user script to another area. The diagnostic executive notifies you if you have violated this type of restriction by issuing a script incompatibility message.

- Test number.
- Run environment. This defines the environment from which the actual diagnostic test can be run. The choices are 0 = ROM, 1 = MSI RAM, 2 = Main Memory, and 3 = Fastest Possible. Choose number 3 to select the fastest possible environment that will not overwrite the test.
- Repeat code.
- Error severity level.
- Console error report.
- Script error treatment.
- LED display.
- Console display.
- Parameters.

Example 4–1 shows how to build and run a user script.

The utility displays the test name after you enter the test number and the number of bytes remaining after you enter the information for each test. When you have finished entering tests, press Return at the next `Next test number:` prompt to end the script-building session. Then type T A0 Return to run the new script.

You cannot review or edit a script you have created.

Example 4–1: Creating a Script with Utility 9F

```
>>> T 9F
SP=20140670
Create script in ?[0=SSC,1=SII_RAM,2=RAM] :
Script starts at 201407D4
 40 bytes left
Next test number :80
CQBIC_memory >>Run from ?[0=ROM,1=Diag_RAM,3=fastest possible] (0):
CQBIC_memory >>Addressing mode? [0=physical,1=virtual] (0):
CQBIC_memory >>Repeat? [0=no,1=forever,>1=count<FF] (0):
CQBIC_memory >>Error severity ? [0,1,2,3] (2):
CQBIC_memory >>Console error report? [0=none,1=full] (1):
CQBIC_memory >>Stop script on error? [0=NO,1=YES] (1):
CQBIC_memory >>LED on entry (08):
CQBIC_memory >>Console on entry (80):
 33 bytes left
Next test number :7d
R3000_Interaction>>Run from ?[0=ROM,1=Diag_RAM,2=RAM,3=fastest
 possible] (0):
R3000_Interaction>>Addressing mode? [0=physical,1=virtual] (0):
R3000_Interaction>>Repeat? [0=no,1=forever,>1=count<FF] (0):
R3000_Interaction>>Error severity ? [0,1,2,3] (2):
R3000_Interaction>>Console error report? [0=none,1=full] (1):
R3000_Interaction>>Stop script on error? [0=NO,1=YES] (1):
R3000_Interaction>>LED on entry (07):
R3000_Interaction>>Console on entry (7D):
R3000_Interaction>> loop_cnt : 00000001 - 0000FFFF ?(00000001) 3
R3000_Interaction>> sel_dev : 00000000 - 0000000F ?(0000000F)
 18 bytes left
Next test number :
>>> T A0
80..7D..
>>>
```

Example 4–2 shows the script-building procedure to follow if (a) you are unsure of the test number to specify, and (b) you want to run one test repeatedly. If you are not sure of the test number, enter ? at the `Next test number:` prompt to invoke test 9E and display test numbers, test names, and so on. To run one test repeatedly, enter the following sequence:

1. Enter the test number (55 in Example 4–2) at the `Next test number:` prompt.
2. Enter `A0` at the next `Next test number:` prompt.
3. Press Return at the next `Next test number:` prompt.
4. Enter `T A0` to begin running the script repeatedly.
5. Press Ctrl/C to stop the test.

The sequence above is a useful alternative to using the REPEAT console command to run a test, because REPEAT (test) displays line feeds only; it does not display the console test announcement.

Example 4–2: Listing and Repeating Tests with Utility 9F, Help and Loop on A0

```
>>>T 9F
SP=20140670
Create script in ?[0=SSC,1=SII_RAM,2=RAM] :0
Script starts at 201407D4
 40 bytes left
Next test number :?
Test
#   Address   Name               Parameters
_____________________________________________________________________
    2004CA00  CP_SCB
    2004DCA8  De_executive
20  200583BC  R_D_Cache_Seg      ***
21  200583D4  R_D_Cache_Tag      ***
22  200583EC  R_D_Cache_Tag_Par  ***
23  20058408  R_D_Cache_Data_Par ***
24  20058425  R_D_Cache_Val_Bit  ***
25  20058441  R_D_Cache_RAM      ***
26  20058472  R_I_Cache_Seg      ***
27  2005848A  R_I_Cache_Tag      ***
28  200584A2  R_I_Cache_Tag_Par  ***
29  200584BE  R_I_Cache_Data_Par ***
2A  200584DB  R_I_Cache_Val_Bit  ***
2B  200584F7  R_I_Cache_RAM      ***
2C  2005850F  R_I_Cache_Inst     ***
2D  20058459  R_D_Cache_Inst     ***
2E  20058528  R_Cache_IStream    ***
2F  20058542  R_Cache_Exerciser  ***
30  20058058  MEM_Bitmap         *
```

Example 4–2 Cont'd on next page

Example 4–2 (Cont.): Listing and Repeating Tests with Utility 9F, Help and Loop on A0

```
34  2004F25C  ROM logic test     *
40  2005774C  Memory Count Errs  Soft_errs_allowed *******
41  20055FFC  CDE Cleanup        ******
42  2005617A  Check_for_intrs    ******
43  20057BD0  CVAX Mem Interface start_add end_add inc pat sel_tst
                                 sav_map ****
44  200561F8  CVAX Cache mem     addr_incr *********
45  20050490  C_cache_mem_cqbic  start_addr end_addr addr_incr ****
46  20050790  C_Cache_diag_mode  addr_incr *********
47  2005822B  MEM_Refresh        start end incr cont_on_err
                                 time_seconds
48  200581B4  MEM_Addr_shrts     start_add end_add * cont_on_err
                                 pat_2 pat_3
4C  2005819C  MEM_ECC_Error      start_add end_add add_incr
                                 cont_on_err
4D  2005811E  MEM_Address        start_add end_add add_incr
                                 cont_on_err
4E  2005810B  MEM_Byte           start_add end_add add_incr
                                 cont_on_err
4F  200580F8  MEM_Data           start_add end_add add_incr
                                 cont_on_err
52  2004FEEF  Prog timer         which_timer wait_time_us ***
53  200501D0  TOY clock          repeat_test_250ms_ea Tolerance ***
54  20055D19  Virtual mode       *********
55  20050387  Interval timer     which_timer **
56  20051624  SII_ext_loopbck    ***
57  20054124  SII_memory         incr test_pattern *******
58  20053D40  DSSI reset         port_no time_secs *
59  200557B8  SGEC_LPBCK_ASSIST  time_secs **
5B  2005450D  SII_registers      ****
5C  2005192F  SII_initiatior     ******
5D  200525DD  SII target         *******
5F  20054B08  SGEC               loopback_type no_ram_tests ******
60  2004FB6E  SSC Console Serial start_baud end_baud ******
65  20054139  SCSI_memory        incr test_pattern *******
66  20058834  R_SCSI_Test        **********
67  200546A8  SCSI Quick test    **********
71  2005855E  R3000_FPU          *
72  20058572  R3000_TLB          *
73  20058586  R_IO_Reg_Interface loop ***
74  200585A3  R_SII_Buf_Intrf    start_add end_add incr patrn **
75  200585BD  R_SCSI_Buf_Intrf   start_add end_add incr patrn **
76  200585D8  R3000_Interrupt    *
77  200585F2  R_Mem_Moving_Inver start_add end_add incr ***
```

Example 4–2 Cont'd on next page

Example 4–2 (Cont.): Listing and Repeating Tests with Utility 9F, Help and Loop on A0

```
78  2005860F  R3000_Write_Buffer *
79  2005862C  R3000_NVRAM        start_add end_add
7A  20058642  R3000_NVRAM_all    start_add end_add
7C  2005865C  R3000_DUART        *
7D  20058672  R3000_Interaction  loop_cnt sel_dev
80  2004F424  CQBIC_memory       **********
81  20050E2F  MSCP-QBUS test     IP_csr ******
82  20051003  DELQA              device_num_addr ****
83  200588D4  VME Test           *
90  2004F39E  CQBIC registers    *
91  2004F32C  CQBIC Powerup      **
92  20057988  CDAL_RIO Intrf     *****
9A  2005806F  Memory Descrip     Board1 Bd2 Bd3 Bd4 verify_only
9B  20057890  Init_memory        *
9C  20056662  List Registers     *
9D  2005655E  Utilities          Expnd_err_msg get_mode init_LEDs
                                 clr_ps_cnt
9E  20050E07  List diags         *
9F  20056FDF  Create Script      *******
C1  200511E4  SSC RAM            *
C2  200513B0  SSC RAM ALL        *
C5  20051529  SSC regs           *
C6  2004F0E0  SSC_powerup        *********
C7  2004F1A1  CBTCR timeout      ***

Scripts
#   Description

A0  Soft Script, created by 9F
A1  Powerup field
A3  Manuf FV
A4  Manuf Loop on A3
A7  Memory tests, called by A8
A8  Field memory acceptance, mark Hard SBEs
A9  Run memory tests, stop on any error
B3  Manuf APT, Do NOT use in field
```

Example 4–2 Cont'd on next page

Example 4–2 (Cont.): Listing and Repeating Tests with Utility 9F, Help and Loop on A0

```
 40 bytes left
Next test number :55
Interval timer >>Run from ?[0=ROM,1=Diag_RAM,2=RAM,3=fastest
 possible] (0):
Interval timer >>Addressing mode? [0=physical,1=virtual] (0):
Interval timer >>Repeat? [0=no,1=forever,>1=count<FF] (0):
Interval timer >>Error severity ? [0,1,2,3] (2):
Interval timer >>Console error report? [0=none,1=full] (1):
Interval timer >>Stop script on error? [0=NO,1=YES] (1):
Interval timer >>LED on entry (05):
Interval timer >>Console on entry (55):
Interval timer >> which_timer : 00000001 - 00000003 ?(00000003)
 29 bytes left
Next test number :a0 - script
 28 bytes left
Next test number :
>>>R T A0
55..55..55..55..55..55..55..55..55..55..55..55..55..55..55..55..55..
55..55..55..55..55..55..55..55..55..55..55..55..55..55..55..55..55..
55..55..55..
>>>
```

4.2.5 Console Displays and LEDs

Example 4–3 shows a typical console display during execution of the ROM-based diagnostics (power-up).

Example 4–3: Console Display (No Errors)

```
KN220-A Vn.n
Performing normal system tests.
83..82..81..80..79..78..77..76..75..74..73..72..71..70..69..68..67..
66..65..64..63..62..61..60..59..58..57..56..55..54..53..52..51..50..
49..48..47..46..45..44..43..42..41..40..39..38..37..36..35..34..33..
32..31..30..29..28..27..26..25..24..23..22..21..20..19..18..17..16..
15..14..13..12..11..10..09..08..07..06..05..04..03..
Tests completed.
>>>
```

The first line contains the module (KN220–A) and the firmware version (Vn.n).

The numbers on the console display do not refer to actual test numbers. Table 4–3 shows the actual ROM-based diagnostic tests that run during power-up.

Table 4–3: Tests Run During Power-up

Number Displayed/Test Run	Number Displayed/Test Run	Number Displayed/Test Run
83/91	56/25	30/48
82/90	55/2D	29/48
81/52	54/27	28/48
80/52	53/28	27/47
79/53	52/29	26/4C
78/C1	51/2A	25/40
77/34	50/2B	24/46
76/C5	49/2C	23/43
75/55	48/30	22/5F
74/5B	47/9A	21/7A
73/57	46/4F	20/20
72/C7	45/4E	19/26
71/C2	44/4D	18/2E
70/5C	43/48	17/2F
69/5D	42/48	16/73
68/65	41/48	15/74
67/67	40/48	14/75
66/92	39/48	13/76
65/79	38/48	12/66
64/78	37/48	11/44
63/7C	36/48	10/80
62/71	35/48	09/54
61/72	34/48	08/34
60/21	33/48	07/C5
59/22	32/48	06/45
58/23	31/48	05/83
57/24	30/48	04/7D
		03/41

During execution of the IPT, normal error messages are displayed if the console terminal is working. Console announcements, such as test numbers and countdown, however, are suppressed. Tests continue to run after the IPT, up to and including the appropriate console test.

Diagnostic test failures, if specified in the firmware script, produce an error display in the format shown in Example 4–4 and Example 4–5.

Example 4–4: Sample Output with Errors: R3000

```
>>>T 0
83..82..81..80..79..78..77..76..75..74..73..72..71..70..69..68..67..
66..65..

?79 2 06 FF 0000 0007

P1 =28000000  P2 =2807FFFC  P3 =00000000  P4 =00000000  P5 =00000000
P6 =00000000  P7 =00000000  P8 =00000000  P9 =00000000  P10=00000000
P11=00000000  P12=00000000  P13=00000000  P14=00000000  P15=00000000
P16=00000000  P17=00000000  P18=00000000  P19=00000000  P20=00000000
gp =1C270008  sp =B8001B1C  fp =00000000  sr =B048FF04
epc=BFC2903C  badvaddr =00000000  cause =00000000
64..63..62..61..60..59..58..57..56..55..54..53..52..51..50..
49..48..47..46..45..44..43..42..41..40..39..38..37..36..35..34..33..
32..31..30..29..28..27..26..25..24..23..22..21..20..19..18..17..16..
15..14..13..12..11..10..09..08..07..06..05..04..03..
Normal operation not possible.

>>>
```

Example 4–5: Sample Output with Errors: CVAX

```
>>>T 0
83..82..81..80..79..78..77..76..75..74..

?5B 2 01 FF 0000 0009

P1 =20160044  P2 =000080FF  P3 =00000000  P4 =FFFF0000  P5 =00000000
P6 =00000000  P7 =00000000  P8 =00000000  P9 =00000000  P10=00000000
P11=00000000  P12=00000000  P13=00000000  P14=00000000  P15=00000000
P16=00000000  P17=00000000  P18=00000000  P19=00000000  P20=00000000
r0 =00000001  r1 =2005462B  r2 =0000005B  r3 =201407AC  r4 =2005450D
r5 =2005452B  r6 =200589B3  r7 =00000000  r8 =0000000C  ERF=80000000
73..72..71..70..69..68..67..
66..65..64..63..62..61..60..59..58..57..56..55..54..53..52..51..50..
49..48..47..46..45..44..43..42..41..40..39..38..37..36..35..34..33..
32..31..30..29..28..27..26..25..24..23..22..21..20..19..18..17..16..
15..14..13..12..11..10..09..08..07..06..05..04..03..
Normal operation not possible.

>>>
```

The errors are printed in a seven-line display. The first line has six fields:

```
Test  Severity  Error  De_error  Vector  Count
```

- Test identifies the diagnostic test. In Example 4–5, the test is 5B.
- Severity is the severity level of a test failure, as dictated by the script. In Example 4–4 and Example 4–5, 2 is the severity level. Failure of a severity level 2 test causes the display of this five-line error printout and halts an autoboot. An error of severity level 1 causes a display of the first line of the error printout, but does not interrupt an autoboot. Most tests have a severity level of 2.
- Error is two hex digits identifying, usually within 10 instructions, where in the diagnostic the error occurred. This field is also called the subtestlog. In Example 4–5, 01 is the area where the error occurred.
- De_error (diagnostic executive error) signals the diagnostic's state and any illegal behavior. This field indicates a condition that the diagnostic expects on detecting a failure. FE, EE, or EF in this field means that an unexpected exception or interrupt was detected. FF indicates an error as a result of normal testing, such as a miscompare. The possible codes are:

  FF—Normal error exit from diagnostic
  FE—Unexpected interrupt
  FD—Interrupt in cleanup routine
  FC—Interrupt in interrupt handler
  FB—Script requirements not met
  FA—No such diagnostic
  EF—Unexpected exception in executive
  EE—Unexpected exception in console

- Vector identifies the SCB vector (0000 in the example above) through which the unexpected exception or interrupt trapped, when the de_ error field detects an unexpected exception or interrupt (FE or EF; for EE look at the CAUSE REGISTER).
- Count is four hex digits. It shows the number of previous errors that have occurred (seven in Example 4–4 and nine in Example 4–5).

Lines 2 through 5 of the error printout are parameters 1 through 20. When the diagnostics are running normally, these parameters are the same parameters that are listed in Table 4–5.

When an unexpected machine check exception or other type of exception occurs during the executive (de_error is EF), the stack is saved in the parameters on lines 2 and 3, as listed in Tables 4–4 and 4–5.

Table 4–4: Values Saved, Machine Check Exception During Executive (CVAX)

Parameter	Value
P1	Contents of SP, points to vector value in P2
P2	Vector = 04, vector of exception 04–FC, 00 = Q-bus
P3	Address of PC pointing to failed instruction, P9
P4	Byte count = 10
P5	Machine check code
P6	Most recent virtual address
P7	Internal state information 1
P8	Internal state information 2
P9	PC, points to failing instruction
P10	PSL

Table 4–5: Values Saved, Exception During Executive (CVAX)

Parameter	Value
P1	Contents of SP, points to vector value in P2
P2	Vector = nn, vector of exception 04–FC, 00 = Q-bus
P3	Address of PC pointing to failed instruction, P4
P4	PC, points to instruction following failed instruction
P5	PSL
P6	Contents of stack
P7	Contents of stack
P8	Contents of stack
P9	Contents of stack
P10	Contents of stack
P11-P20	Unused

In the example of CVAX errors (Example 4–5), lines 6 and 7 of the error printout are general registers R0 through R8 and the hardware error summary register. In the example of R3000 errors (Example 4–4) lines 6 and 7 are the general registers gp, sp, fp, sr, epc, badvaddr, and cause.

When returning a module for repair, record the first line of the error printout and the version of the ROMs on the module repair tag.

In addition to the console diagnostic countdown, a hexadecimal value is displayed on the diagnostic LEDs on the KN220 I/O module and on the H3602–AC CPU I/O panel.

Table 4–6 lists all LED codes and the associated actions that are performed at power-up.

Table 4–6: LED Codes

LED Value	Action
F	Initial state on power-up. No code has executed.
E	Entered ROM. Some instructions have executed.
D	Waiting for power to stabilize (POK).
C	SSC and ROM tests.
B	CVAX tests.
A	R3000 tests.
9	Memory controller and memory tests.
8	CQBIC (Q22-bus) tests.
7	Console loopback tests.
6	DSSI and SCSI subsystem tests.
5	Ethernet subsystem tests.
4	CVAX maintenance mode.
3	R3000 normal mode.
2	CVAX primary/secondary bootstrap.
1	R3000 primary/secondary bootstrap.
0	Operating system running.

Figure 4–1 shows the LEDs on the KN220 I/O module. They correspond to the hex display on the H3602–AC.

Figure 4–1: KN220 I/O Module LEDs

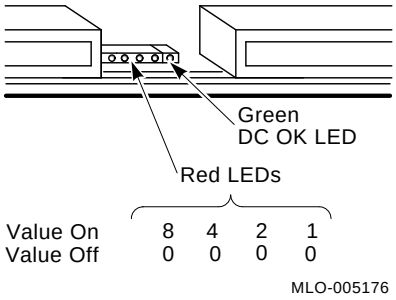


Table 4–7 lists the hex LED display, the default action on errors, and the most likely FRUs. It is divided into IPTs and scripts.

The Default Action on Error column refers to the action taken by the diagnostic executive under the following circumstances:

- The diagnostic executive detects an unexpected exception or interrupt.
- A test fails and that failure is reported to the diagnostic executive.

The Default on Error column does not refer to the action taken by the memory tests. The diagnostic executive either halts the script or continues execution at the next test in the script.

Most memory tests have a continue on error parameter (labeled cont_on_error, as shown in test 47 in Example 4–2). If you explicitly set cont_on_error, using parameter 4 in a memory test, the test marks bad pages in the bitmap and continues without notifying the diagnostic executive of the error. In this case, a halt on error does not occur even if you specify halt on error in the diagnostic executive (by answering `Yes` to `Stop script on error?` in Utility 9F), since the memory test does not notify the diagnostic executive that an error has occurred.

Table 4–7: KN220 Console Displays and FRUs

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU[1]
Initial Power-Up Tests					
F	None	None	Loop on test	Power-up	7, 2, 5, 6, 10
D	None	None	Loop on test	WAIT_POK	2
4	None	None	Loop on self	Entering IPT	2
7	None	None	Loop on test	SLU_EXT_LOOPBACK[2]	8, 9, 2
Script B1					
C	None	?9D	Continue	Utilities	2, 10
B	None	?42	Continue	Check_for_intrs	2, 1, 10
C	None	?C6	Continue	SSC_power-up	2, 10
7	None	?60	Continue	CONSOLE_SERIAL	2, 10
End of script.					
Script A1					
8	83	?91	Continue	CQBIC Powerup	2
8	82	?90	Continue	CQBIC registers	2
C	81	?52	Continue	Prog timer	2
C	80	?52	Continue	Prog timer	2
C	79	?53	Continue	TOY clock	2
C	78	?C1	Continue	SSC RAM	2

[1]In the case of multiple FRUs, refer to Section 4.4.2 for further information. If a problem recurs with the same FRU, check that the tolerances for system power supply +5 Vdc, +12 Vdc, and ac ripple are within specification.

[2]This test runs only if the power-up mode switch on the H3602–AC is set to select the test. See Section 4.5.3.

FRU key:

1 = KN220 CPU module
2 = KN220 I/O module
3 = MS220 memory module
4 = Memory interconnect cable
5 = Q22-bus device
6 = Q22/CD backplane
7 = System power supply
8 = H3602–AC CPU I/O panel
9 = H3602–AC cable
10 = CPU and I/O module interconnect cable
11 = CPU and VME interconnect cable

Table 4–7 (Cont.): KN220 Console Displays and FRUs

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU[1]
Script A1					
C	77	?34	Continue	ROM logic test	2, 10
C	76	?C5	Continue	SSC regs	2, 10
B	75	?55	Continue	Interval timer	2, 1, 10
6	74	?5B	Continue	SII_registers	2, 10
6	73	?57	Continue	SII_memory	2, 10
C	72	?C7	Continue	CBTCR timeout	2, 10
C	71	?C2	Continue	SSC RAM ALL	2, 10
6	70	?5C	Continue	SII_initiatior	2, 10
6	69	?5D	Continue	SII target	2, 10
6	68	?65	Continue	SCSI_memory	2, 10
6	67	?67	Continue	SCSI Quick test	2, 10
C	66	?92	Halt	CDAL_RIO Intrf	2, 1, 10
A	65	?79	Continue	R3000_NVRAM	1
A	64	?78	Continue	R3000_Write_Buffer	1
A	63	?7C	Continue	R3000_DUART	1
A	62	?71	Continue	R3000_FPU	1
A	61	?72	Continue	R3000_TLB	1
A	60	?21	Continue	R_D_Cache_Tag	1
A	59	?22	Continue	R_D_Cache_Tag_Par	1
A	58	?23	Continue	R_D_Cache_Data_Par	1
A	57	?24	Continue	R_D_Cache_Val_Bit	1
A	56	?25	Continue	R_D_Cache_RAM	1
A	55	?2D	Continue	R_D_Cache_Inst	1

[1]In the case of multiple FRUs, refer to Section 4.4.2 for further information.
If a problem recurs with the same FRU, check that the tolerances for system power supply +5 Vdc, +12 Vdc, and ac ripple are within specification.

FRU key:

1 = KN220 CPU module
2 = KN220 I/O module
3 = MS220 memory module
4 = Memory interconnect cable
5 = Q22-bus device
6 = Q22/CD backplane
7 = System power supply
8 = H3602–AC CPU I/O panel
9 = H3602–AC cable
10 = CPU and I/O module interconnect cable
11 = CPU and VME interconnect cable

Table 4–7 (Cont.): KN220 Console Displays and FRUs

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU[1]
Script A1					
A	54	?27	Continue	R_I_Cache_Tag	1
A	53	?28	Continue	R_I_Cache_Tag_Par	1
A	52	?29	Continue	R_I_Cache_Data_Par	1
A	51	?2A	Continue	R_I_Cache_Val_Bit	1
A	50	?2B	Continue	R_I_Cache_RAM	1
A	49	?2C	Continue	R_I_Cache_Inst	1
9	48	?30	Halt	MEM_Bitmap	3,1,2,4,6
9	47	?9A	Continue	Memory Descrip	3,1,2,4,6
9	46	?4F	Continue	MEM_Data	3,1,2,4,6
9	45	?4E	Continue	MEM_Byte	3,1,2,4,6
9	44	?4D	Continue	MEM_Address	3,1,2,4,6
9	43	?48	Continue	MEM_Addr_shrts	3,1,2,4,6
9	42	?48	Continue	MEM_Addr_shrts	3,1,2,4,6
9	41	?48	Continue	MEM_Addr_shrts	3,1,2,4,6
9	40	?48	Continue	MEM_Addr_shrts	3,1,2,4,6
9	39	?48	Continue	MEM_Addr_shrts	3,1,2,4,6
9	38	?48	Continue	MEM_Addr_shrts	3,1,2,4,6
9	37	?48	Continue	MEM_Addr_shrts	3,1,2,4,6
9	36	?48	Continue	MEM_Addr_shrts	3,1,2,4,6
9	35	?48	Continue	MEM_Addr_shrts	3,1,2,4,6
9	34	?48	Continue	MEM_Addr_shrts	3,1,2,4,6
9	33	?48	Continue	MEM_Addr_shrts	3,1,2,4,6
9	32	?48	Continue	MEM_Addr_shrts	3,1,2,4,6

[1]In the case of multiple FRUs, refer to Section 4.4.2 for further information.
If a problem recurs with the same FRU, check that the tolerances for system power supply +5 Vdc, +12 Vdc, and ac ripple are within specification.

FRU key:

1 = KN220 CPU module
2 = KN220 I/O module
3 = MS220 memory module
4 = Memory interconnect cable
5 = Q22-bus device
6 = Q22/CD backplane
7 = System power supply
8 = H3602–AC CPU I/O panel
9 = H3602–AC cable
10 = CPU and I/O module interconnect cable
11 = CPU and VME interconnect cable

Table 4–7 (Cont.): KN220 Console Displays and FRUs

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU[1]
Script A1					
9	31	?48	Continue	MEM_Addr_shrts	3,1,2,4,6
9	30	?48	Continue	MEM_Addr_shrts	3,1,2,4,6
9	29	?48	Continue	MEM_Addr_shrts	3,1,2,4,6
9	28	?48	Continue	MEM_Addr_shrts	3,1,2,4,6
9	27	?47	Continue	MEM_Refresh	3,1,2,4,6
9	26	?4C	Continue	MEM_ECC_Error	3,1,2,4,6
9	25	?40	Continue	Memory Count Errs	3,1,2,4,6
B	24	?46	Continue	C_Cache_diag_mode	2
9	23	?43	Continue	CVAX Mem Interface	3,2,1,4,6
5	22	?5F	Continue	SGEC	2
A	21	?7a	Continue	R3000_NVRAM_all	1
A	20	?20	Continue	R_D_Cache_Seg	1
A	19	?26	Continue	R_I_Cache_Seg	1
A	18	?2e	Continue	R_Cache_IStream	1
A	17	?2f	Continue	R_Cache_Exerciser	1
A	16	?73	Continue	R_IO_Reg_Interface	1
A	15	?74	Continue	R_SII_Buf_Intrf	2,1
A	14	?75	Continue	R_SCSI_Buf_Intrf	2,1
A	13	?76	Continue	R3000_Interrupt	1,2
6	12	?66	Continue	R_SCSI_Test	2,1
B	11	?44	Continue	CVAX Cache mem	1,3,4,6
8	10	?80	Continue	CQBIC_memory	1,3,5,4,6
B	09	?54	Continue	Virtual mode	2,1,3,4,6

[1]In the case of multiple FRUs, refer to Section 4.4.2 for further information.
If a problem recurs with the same FRU, check that the tolerances for system power supply +5 Vdc, +12 Vdc, and ac ripple are within specification.

FRU key:

1 = KN220 CPU module
2 = KN220 I/O module
3 = MS220 memory module
4 = Memory interconnect cable
5 = Q22-bus device
6 = Q22/CD backplane
7 = System power supply
8 = H3602–AC CPU I/O panel
9 = H3602–AC cable
10 = CPU and I/O module interconnect cable
11 = CPU and VME interconnect cable

Table 4–7 (Cont.): KN220 Console Displays and FRUs

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU[1]
Script A1					
C	08	?34	Continue	ROM logic test	2
C	07	?C5	Continue	SSC regs	2
8	06	?45	Continue	C_cache_mem_cqbic	2,1,3,5,4,6
A	05	?83	Continue	VME Test	11
A	04	?7d	Continue	R3000_Interaction	2,1,3,4,10
C	03	?41	Continue	CDE Cleanup	2,1,5
Script A9					
9	4F	?4F	Halt	MEM_data	3, 1, 2, 4, 6
9	4E	?4E	Halt	MEM_byte	3, 1, 2, 4, 6
9	4D	?4D	Halt	MEM_addr	3, 1, 2, 4, 6
9	4C	?4C	Halt	MEM_ECC_error	3, 1, 4, 6
9	48	?48	Continue	MEM_Addr_shrts	3, 1, 4, 6
9	47	?47	Continue	MEM_refresh	3, 1, 4, 6
9	40	?40	Continue	MEM_count_errs	3, 1, 4, 6
C	41	?41	Continue	Board reset	2, 1, 5
End of script.					
Script A8					
9	30	?30	Halt	MEM_bitmap	1, 3, 2, 4, 6
Invoke script A7.					

[1]In the case of multiple FRUs, refer to Section 4.4.2 for further information.
If a problem recurs with the same FRU, check that the tolerances for system power supply +5 Vdc, +12 Vdc, and ac ripple are within specification.

FRU key:

1 = KN220 CPU module
2 = KN220 I/O module
3 = MS220 memory module
4 = Memory interconnect cable
5 = Q22-bus device
6 = Q22/CD backplane
7 = System power supply
8 = H3602–AC CPU I/O panel
9 = H3602–AC cable
10 = CPU and I/O module interconnect cable
11 = CPU and VME interconnect cable

Table 4–7 (Cont.): KN220 Console Displays and FRUs

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU[1]
Script A8					
End of script.					
Script A7					
9	4F	?4F	Halt	MEM_data	3, 1, 2, 4, 6
9	4E	?4E	Halt	MEM_byte	3, 1, 2, 4, 6
9	4D	?4D	Halt	MEM_addr	3, 1, 2, 4, 6
9	4C	?4C	Halt	MEM_ECC_error	3, 1, 2, 4, 6
9	48	?48	Halt	MEM_address_shorts	3, 1, 2, 4, 6
9	47	?47	Halt	MEM_refresh	3, 1, 2, 4, 6
9	40	?40	Cont	MEM_count_bad pages	3, 1, 2, 4, 6
8	80	?80	Cont	CQBIC_memory	1, 3, 5, 4, 6
C	41	?41	Halt	Board reset	2, 1, 5
End of script.					

[1]In the case of multiple FRUs, refer to Section 4.4.2 for further information.
If a problem recurs with the same FRU, check that the tolerances for system power supply +5 Vdc, +12 Vdc, and ac ripple are within specification.

FRU key:

1 = KN220 CPU module
2 = KN220 I/O module
3 = MS220 memory module
4 = Memory interconnect cable
5 = Q22-bus device
6 = Q22/CD backplane
7 = System power supply
8 = H3602–AC CPU I/O panel
9 = H3602–AC cable
10 = CPU and I/O module interconnect cable
11 = CPU and VME interconnect cable

4.2.6 System Halt Messages

Table 4–8 lists messages that may appear on the console terminal when a system error occurs.

Table 4–8: System Halt Messages

Code	Message	Explanation
?02	EXT HLT	External halt, caused either by console BREAK condition or because Q22-bus BHALT_L or DBR<AUX_HLT> bit was set while enabled.
?03	–	Power-up; no halt message displayed. The presence of the firmware banner and diagnostic countdown indicates this halt.
?04	ISP ERR	Caused by attempt to push interrupt or exception state onto the interrupt stack when the interrupt stack was mapped NO ACCESS or NOT VALID.
?05	DBL ERR	A second machine check occurred while the processor was attempting to service a normal exception.
?06	HLT INST	The processor executed a HALT instruction in kernel mode.
?07	SCB ERR3	The vector had bits <1:0> = 3.
?08	SCB ERR2	The vector had bits <1:0> = 2.
?0A	CHM FR ISTK	A change mode instruction was executed when PSL<IS> was set.
?OB	CHM TO ISTK	The SCB vector for a change mode had bit <0> set.
?OC	SCB RD ERR	A hard memory error occurred during a processor read of an exception or interrupt vector.
?10	MCHK AV	An access violation or an invalid translation occurred during machine check exception processing.
?11	KSP AV	An access violation or an invalid translation occurred during invalid kernel stack pointer exception processing.
?12	DBL ERR2	Double machine check error. A machine check occurred during an attempt to service a machine check.
?13	DBL ERR3	Double machine check error. A machine check occurred during an attempt to service a kernel stack not valid exception.
?19	PSL EXC5	PSL <26:24> = 5 on interrupt or exception.
?1A	PSL EXC6	PSL <26:24> = 6 on interrupt or exception.
?1B	PSL EXC7	PSL <26:24> = 7 on interrupt or exception.
?1D	PSL REI5	PSL <26:24> = 5 on an rei instruction.
?1E	PSL REI6	PSL <26:24> = 6 on an rei instruction.
?1F	PSL REI7	PSL <26:24> = 7 on an rei instruction.
?38	SECENB	Security is enabled.

4.2.7 Console Error Messages

Table 4–9 lists messages issued in response to an error or to a console command that was entered incorrectly.

Table 4–9: Console Error Messages

Code	Message	Explanation
?20	CORRPTN	The console data base was corrupted. The console simulates a power-up sequence and rebuilds its data base.
?21	ILL REF	The requested reference would violate virtual memory protection, address is not mapped or is invalid in the specified address space, or value is invalid in the specified destination.
?22	ILL CMD	The command string cannot be parsed.
?23	INV DGT	A number has an invalid digit.
?24	LTL	The command was too large for the console to buffer. The message is sent only after the console receives the Return at the end of the command.
?25	ILL ADR	The specified address is not in the address space.
?26	VAL TOO LRG	The specified value does not fit in the destination.
?27	SW CONF	Switch conflict. For example, an EXAMINE command specifies two different data sizes.
?28	UNK SW	The switch is not recognized.
?29	UNK SYM	The EXAMINE or DEPOSIT symbolic address is not recognized.
?2A	CHKSM	An X command has an incorrect command or data checksum. If the data checksum is incorrect, this message is issued and is not abbreviated to “Illegal command.”
?2B	HLTED	The operator entered a HALT command.
?2C	FND ERR	A FIND command failed either to find the RPB or 64 Kbytes of good memory.
?2D	TMOUT	Data failed to arrive in the expected time during an X command.
?2E	MEM ERR	Memory error or machine check occurred.
?2F	UNXINT	An unexpected interrupt or exception occurred.
?30	UNIMPLEMENTED	Unimplemented function.
?31	QUAL NOVAL	Qualifier does not take a value.
?32	QUAL AMBG	Ambiguous qualifier.
?33	QUAL REQ VAL	Qualifier requires a value.
?34	QUAL OVERF	Too many qualifiers.
?35	ARG OVERF	Too many arguments.
?36	AMBG CMD	Ambiguous command.
?37	INSUF ARG	Too few arguments.
?64	INVSCSIID	Invalid SCSI host ID configuration.

4.2.8 VMB Error Messages (CVAX)

If VMB is unable to boot MDM, it returns an error message to the console. Table 4–10 lists the error messages and their descriptions.

Table 4–10: VMB Error Messages

Message Number	Mnemonic	Interpretation
?40	NOSUCHDEV	No bootable devices found
?41	DEVASSIGN	Device is not present
?42	NOSUCHFILE	Program image not found
?43	FILESTRUCT	Invalid boot device file structure
?44	BADCHKSUM	Bad checksum on header file
?45	BADFILEHDR	Bad file header
?46	BADDIRECTORY	Bad directory file
?47	FILNOTCNTG	Invalid program image format
?48	ENDOFFILE	Premature end-of-file encountered
?49	BADFILENAME	Bad file name given
?4A	BUFFEROVF	Program image does not fit in available memory
?4B	CTRLERR	Boot device I/O error
?4C	DEVINACT	Failed to initialize boot device
?4D	DEVOFFLINE	Device is off line
?4E	MEMERR	Memory initialization error
?4F	SCBINT	Unexpected SCB exception or machine check
?50	SCB2NDINT	Unexpected exception after starting program image
?51	NOROM	No valid ROM image found
?52	NOSUCHNODE	No response from load server
?53	INSFMAPREG	Insufficient Q-bus mapping registers due to invalid memory configuration, bad memory, or because Q-bus map was not relocated to main memory
?54	RETRY	No devices bootable, retrying
?55	IVDEVNAM	Invalid device boot name

4.3 Acceptance Testing

Perform the acceptance testing procedure listed below, after installing a system or whenever replacing the following:

KN220 CPU module
KN220 I/O module
MS220–AA memory module
Memory data interconnect cable
Backplane
DSSI device
SCSI device
H3602–AC

1. Run five error-free passes of the power-up scripts by entering the following command:

   ```
   >>> R T 0
   ```

2. Press Ctrl/C to terminate the scripts.

3. Perform the next two steps for a more thorough test of memory.

   ```
   >>> T A8
   ```

   Script A8 runs for one pass. This command enables mapping out of solid single-bit ECC as well as multibit ECC errors. It will also run script A7 for one pass.

If any of the memory tests fail, they mark the bitmap and continue with no error printout to the console. An exception is test 40 (count bad pages). If any single-bit or multibit ECC errors are detected, they are reported in test 40. Such a failure indicates that pages in memory have been marked bad in the bitmap because of solid single-bit and /or multibit ECC errors. The error printout does not display the 20 longwords, since it is a severity level 1 error.

4. Run T 9A to check the memory configuration again, since T 9A prompts you for the correct memory configuration.

   NOTE: *The specifications you entered in T 9A stay in NVRAM as long as battery power is applied, or until you run T 9A and enter changes.*

```
>>> T 9A

                        0 MB = 0;
   32 MB, M7639-A, MS220-AA = 1;
   64 MB, M7639-B, MS220-BA = 2;
MEM FRU 0:3

MEM FRU 0 = 1
MEM FRU 1 = 1
MEM FRU 2 = 0

MEM FRU = 0  32 MB, M7639-A, MS220-AA  00000000 01FFFFFF
MEM FRU = 1  32 MB, M7639-A, MS220-AA  02000000 03FFFFFF
>>>
```

5. To determine the exact configuration, run the SHOW MEMORY/FULL command.

```
>>>SHOW MEM/FULL

Memory 0: 00000000 to 01FFFFFF, 32MB, 0 bad pages
Memory 1: 02000000 to 03FFFFFF, 32MB, 0 bad pages

Total of 64MB, 0 bad pages, 128 reserved pages

Memory Bitmap
-03FF0000 to 03FF3FFF, 32 pages

Console Scratch Area
-03FF4000 to 03FF7FFF, 32 pages

Qbus Map
-03FF8000 to 03FFFFFF, 64 pages

Scan of Bad Pages
>>>
```

Memories 0 through 3 are the MS220 memory modules. The Q22-bus map always spans the top 32 Kbytes of good memory. The memory bitmap always spans two pages (1 Kbyte) per 4 Mbytes of memory configured.

6. Check the Q22-bus and the Q22-bus logic in the KN220 CQBIC chip and the configuration of the Q22-bus, as follows:

```
>>> SHOW QBUS
Scan of Qbus I/O Space
-20000120 (760440) = 0080 (300) DHQ11/DHV11/CXA16/CXB16/CXY08
-20000122 (760442) = F081
-20000124 (760444) = DD18
-20000126 (760446) = 0200
-20000128 (760450) = 0000
-2000012A (760452) = 0000
-2000012C (760454) = 8000
-2000012E (760456) = 0000
-20001920 (774440) = FF08 (120) DELQA/DEQNA
-20001922 (774442) = FF00
-20001924 (774444) = FF2B
-20001926 (774446) = FF06
-20001928 (774450) = FF16
-2000192A (774452) = FFF2
-2000192C (774454) = 00F8
-2000192E (774456) = 1030
-20001940 (774500) = 0000 (260) TQK50/TQK70/TU81E/RV20/K-TAPE
-20001942 (774502) = 0BC0
-20001F40 (777500) = 0020 (004) IPCR

Scan of Qbus Memory Space
>>>
```

The columns are described below. The examples listed are from the last line of the example above.

First column = the VAX I/O address of the CSR, in hex (20001F40).
Second column = the Q22-bus address of the CSR, in octal (777500).
Third column = the data, contained at the CSR address, in hex (0020).
Fourth column = the device vector in octal, according to the fixed or floating Q22-bus and UNIBUS algorithm (004).
Fifth column = the device name (IPCR, the KN220 interprocessor communications register).

Additional lines for the device are displayed if more than one CSR exists.

The last line, Scan of Qbus Memory Space, displays memory residing on the Q22-bus, if present. VAX memory mapped by the Q22-bus map is not displayed.

If the system contains an MSCP or TMSCP controller, run test 81. This test performs the following functions:

Performs step one of the UQ port initialization sequence
Performs the SA wraparound test
Checks the Q22-bus interrupt logic

If you do not specify the CSR address, the test searches for and runs on the first MSCP device by default. To test the first TMSCP device, you must specify the first parameter:

```
>>> T 81 20001940
```

You can specify other addresses if you have multiple MSCP or TMSCP devices in the first parameter. This action may be useful to isolate a problem with a controller, the KN220, or the backplane. Use the VAX address provided by the SHOW QBUS command to determine the CSR value. If you do not specify a value, the MSCP device at address 20001468 is tested by default.

5. Check that all UQSSP, MSCP, TMSCP, and Ethernet controllers and devices are visible by typing the following command line:

```
>>> SHOW DEVICE

DSSI Node 0 (R7YRMS)
 -rf(0,0,*)  (RF71)

DSSI Node 7 (*)

SCSI Node 0
 -rz(0,0,*) (RZ56)

SCSI Node 7 (*)

UQSSP Tape Controller 0 (774500)
 -tm(0,0)  (TK70) -MUA0

Ethernet Adapter
 -mop() -EZA0  (08-00-2B-12-81-22)

Ethernet Adapter 0 (774440)
        -XQA0 (08-00-2B-08-CB-5C)

VME Interface Board - Not Installed
>>>
```

   In the example above, the console displays the DSSI node names and numbers, then the R3000 boot names, which are followed by the controller number and unit number in parentheses, then the device type.

   DSSI Node 7 (*) is the DSSI adapter located on the KN220 I/O module. In most cases, the KN220 I/O module is the local DSSI node shown by the asterisk and has a node number of 7. DSSI node names, node numbers, and unit numbers should be unique.

   The console also displays the SCSI node number, the R3000 boot name, the controller number, the unit number, and the device type. SCSI node number (7 (*)), is the SCSI controller number contained in the environment variables described in Section 3.7.2.

   The UQSSP (TQK70) tape controller and its CSR address are also shown. The line below this display shows a TK70 connected.

   The next two lines show the logical name and station address for the Ethernet adapter located on the KN220 I/O module. A Q-bus Ethernet adapter is listed with Q-bus address and station address.

6. Test the DSSI subsystem, using the KN220 ROM-based Diagnostics and Utilities Protocol (DUP) facility. This facility allows you to connect to the DUP server in the RF drive controller. Here are some examples:

```
>>> SET HOST/DUP/DSSI 7
Starting DUP server...

Stopping DUP server...
```

In this example, a DUP connection was made with DSSI node 7, the KN220 I/O module. The DUP server times out, since no local programs exist and no response packet was received.

```
>>> SET HOST/DUP/DSSI 1
Starting DUP server...

DSSI Node 1 (R3VBNC)
DRVEXR V1.0  D 21-FEB-1988 21:27:54
DRVTST V1.0  D 21-FEB-1988 21:27:54
HISTRY V1.0  D 21-FEB-1988 21:27:54
ERASE  V1.0  D 21-FEB-1988 21:27:54
PARAMS V1.0  D 21-FEB-1988 21:27:54
DIRECT V1.0  D 21-FEB-1988 21:27:54
End of directory

Task Name? DRVTST
Write/read anywhere on medium? [1=Yes/(0=No)]: <CR>
5 minutes for test to complete.
Compare failed on head  1 track 1091.
Compare failed on head  0 track  529.

Task Name? DRVEXR
Write/read anywhere on medium? [1=Yes/(0=No)]: <CR>
Test time in minutes? [(10)-100]:
10 minutes for test to complete.
R3VBNC::MSCP$DUP 21-FEB-1988 21:37:35 DRVEXR CPU=00:00:01.88 PI=43
R3VBNC::MSCP$DUP 21-FEB-1988 21:37:38 DRVEXR CPU=00:00:03.38 PI=79
Compare failed on head  1 track 1091.
R3VBNC::MSCP$DUP 21-FEB-1988 21:37:40 DRVEXR CPU=00:00:04.97 PI=116
^C
>>>
```

In the example above, the local programs DRVTST and DRVEXR are run on drive 1. Do not enter 1 in response to the question `Write/read anywhere on medium?`. Doing so destroys data on the disk. Press Return, which uses the default, allowing reads and writes to the DBNs only. Ctrl/T or Ctrl/G displays a message as shown in the DRVEXR example above (the lines beginning with R3VBNC::). In the example, Ctrl/T has been pressed twice to show the difference in the time and in the value of the progress indicator (PI).

Press Ctrl/C to terminate the program.

Use the local programs HISTRY (Section 4.7.3) and PARAMS (Section 4.7.5) to determine the cause of errors displayed during DRVTST or DRVEXR. DRVTST should run successfully for one pass on each drive. Customer Services can refer to the *RF71 Disk Drive Service Manual* for details about the DUP local programs and corrective action.

7. Enter the SHOW SCSI/FULL command.

Example 4–6: SHOW SCSI and SHOW SCSI/FULL

```
>>> SHOW SCSI

SCSI Node 0
 -tz(0,0,*) (TLZ04) -DIA0

SCSI Node 1
 -rz(0,1,*) (RZ56 )

SCSI Node 2
 -rz(0,2,*) (RRD40) -DIA2

SCSI Node 4
 -tz(0,4,*) (.....) -DIA4

SCSI Node 7 (*)

>>> SHOW SCSI/FULL
Boot  Path    Dev    Cap (in Hex)       Product Id      Revs    r/f
-------------------------------------------------------------------
-tl(0,0,*)   TAPE       4B0 MBs      TLZ04 1989(C)DEC   0304     r
-rz(0,1,*)   DISK       27A MBs      RZ56     (C) DEC   0200     f
-rz(0,2,*)   CDROM      23B MBs      RRD40     TM DEC   250E     r
-tz(0,4,*)   TAPE        5A MBs      ..............   ....     r

SCSI Node 7
```

This will list the R3000 boot path, the device type, the device capacity (in hexadecimal), the product identification (if available), the revision number of the drive, and whether it is removable or fixed. If the capacity field returns zero, then a problem exists with the specified drive.

8. If there are one or more DELQA modules in the system, use test 82 to invoke the Ethernet option's self-test and receive status from the firmware. Test 82 is useful for acceptance testing if you cannot access the system enclosure to see the DELQA LEDs.

9. After the steps above have completed successfully, load MDM and run the system tests from the Main Menu. If they run successfully, the system has gone through its basic checkout and you can load the software.

4.4 Troubleshooting

This section contains suggestions for determining the cause of ROM-based diagnostic test failures.

4.4.1 FE Utility

If any of the tests that run after the IPTs and up to the primary console test fail, the major test code is displayed on the LEDs. Run the FE utility if the message `Normal operation not possible` is displayed after the tests are completed and there is no other error indication, or if you need more information than what is provided in the error display.

The FE utility dumps the diagnostic state to the console (Example 4–7). This state indicates the major and minor test code of the test that failed, the 20 parameters associated with the test, and the hardware error summary register.

Example 4–7: FE Utility Example

```
>>> T FE

 bitmap=07FEC000, length=00008000, checksum=0000
 busmap=07FF8000
 return_stack=20140670
 subtest_pc=2005668D
 timeout=000003E8, error=00, de_error=00
 de_error_vector=0000, severity_code=02, total_error_count=0000
 previous_error=00000000, 00000000, 00000000, 00000000
 last_exception_pc=00000000
 flags=000FFFFF, test_flags=0020
 highest_severity=00
 led_display=09
 console_display=9C
 save_mchk_code=80, save_err_flags=000000
 Interrupted test number = 48, Subtestlog=02, Error_type=FF
 gp =C10AE000  sp =B8001B14  fp =00000000  sr =BFC29A0C
 epc=B0482004  badvaddr =00000000  cause =10002000
 parameter_1 =00000000  2=00000000  3=00000000  4=00000000
```

Example 4–7 Cont'd on next page

Example 4–7 (Cont.): FE Utility Example

```
 5=00000000
 parameter_6 =00000000  7=00000000  8=00000000  9=00000000
  10=00000000
 parameter_11=00000000 12=00000000 13=00000000 14=00000000
  15=00000000
 parameter_16=00000000 17=00000000 18=00000000 19=00000000
  20=00000000
>>>
```

The most useful fields displayed above are as follows:

- De_error_vector, which is the SCB vector through which the unexpected interrupt or exception is trapped if de_error equals FE or EF.
- Total_error_count. Four hex digits showing the number of previous errors that have occurred.
- Previous_error. Contains the history of the last four errors. Each longword contains four bytes of information. From left to right these are the de_error, subtest_log, and test number (copied in both bytes).
- Save machine check code (save_mchk_code). Valid only if the test halts on error. This field has the same format as the hardware error summary register.
- Save error flags (save_err_flags). Valid only if the test halts on error. This field has the same format as the hardware error summary register.
- Parameters 1 through 20. Valid only if the test halts on error. The parameters have the same format as the hardware error summary register.

FE in the previous_error field indicates that an unexpected exception has occurred. If any of the tests that announce to the console fails, and the error code is FE, examine the last longword of the error printout. The last longword is the hardware error summary register and contains the machine check code (<31:24>) and KN220 error status bits (<23:0>). Table 4–11 lists the status bits.

Table 4–11: Hardware Error Summary Register

Bit	Register	Description
31	Machine check code	
30	Machine check code	
29	Machine check code	
28	Machine check code	
27	Machine check code	
26	Machine check code	
25	Machine check code	
24	Machine check code	
23	MSER <6>	; CDAL data parity error
22	MSER <5>	; Mchn chck CDAL parity error
21	MSER <4>	; Machine check cache parity
20	MSER <1>	; Cache data parity error
19	MSER <0>	; Cache tag parity error
18	Unused	
8	CBCTR <31>	; CDAL bus timeout.
7	CBCTR <30>	; CPU read/write bus timeout.
6	DSER <7>	; Q22-bus NXM.
4	DSER <5>	; Q22-bus parity error.
3	DSER <4>	; Read main memory error.
2	DSER <3>	; Lost error.
1	DSER <2>	; No grant timeout.
0	IPCRn <15>	; DMA Q22-bus memory error.

4.4.2 Isolating Memory Failures

This section describes procedures for isolating memory subsystem failures, particularly when the system contains more than one MS220 memory module.

1. SHOW MEMORY/FULL.

   Use the SHOW MEMORY/FULL command to examine failures detected by the memory tests. If test 40 fails, indicating that pages have been marked bad in the bitmap, use this test.

   Memory failures normally show up as errors during test 40, which counts memory pages marked bad in the bitmap. If this test fails, you should use the SHOW MEMORY/FULL command to describe the number of bad pages on each individual memory module. If the system contains more than one memory module and SHOW MEMORY/FULL does not identify each module, then you should run test 9A.

2. T 9A to define capacity of memory modules.

   Test 9A allows you to enter the capacity of each individual memory module. When 9A is completed, enter the SHOW MEMORY/FULL command again to display the individual memory modules that contain errors.

   The utility prompts for a description of each board. The descriptions are as follows:

   0 = no memory
   1 = 32-Mbyte board
   2 = 64-Mbyte board

   You can avoid the prompt by entering the specified command:

   ```
   >>> T 9A mem_board_0 mem_board_1 mem_board_2 mem_board_3
   ```

   To specify a system with two 32-Mbyte boards, the command is:

   ```
   >>> T 9A 1 1
   >>>
   ```

   See Section 4.3 for an example of running T 9A interactively.

3. T 40.

   Specify the first parameter in test 40 to be the threshold for soft errors. To allow 0 (zero) errors, enter the following: `>>> T 40 0`. The parameter is a threshold of allowable software errors on all boards. The default is to ignore soft errors.

4. Test 40 always fails if any hard errors are marked in the bitmap.

This command tests the memory all memory installed modules. Use it after running memory tests individually or within a script. If test 40 fails with subtestlog = 6, enter the SHOW MEMORY/FULL command to see the error summary.

Run test A9 after a failure of test 40.

5. T A9.

```
>>> T [memory test] starting_address ending_address
```

Script A9 stops on any error, hard or soft. This script is useful in determining the first memory test that failed. After finding a failure in test 40, T A9 runs the power-up memory test and prints the first error. When an error is detected, run the failed test on each memory module, one at a time. Enter parameters 1 and 2 of tests 40, 47, 48, and 4A–4F as the starting and ending addresses for testing.

All memory tests with the exception of 40 save the MEAR, MESR, and CAUSE registers.

6. T 9C.

Example 4–8: Isolating Bad Memory Using T 9C

```
>>> T 9C
TOY  =000402C8  ICCS =00000000
TCR0 =00000000  TIR0 =00000000  TNIR0=00000000  TIVR0=00000078
TCR1 =00000081  TIR1 =01EE03BC  TNIR1=00000000  TIVR1=0000007C
RXCS =00000000  RXDB =0000000D  TXCS =00000000  TXDB =00000030
MSER =00000000  CADR =0000000C
BDR  =FFFFFFD2  DLEDR=0000000B  SSCCR=00D45077  CBTCR=00000004
SCR  =0000D000  DSER =00000000  QBEAR=0000000F  DEAR =00000000
QBMBR=07FF8000  IPCR =0000
MESR =FFFC0614  MEAR =3E030003  ITR  =FFFFFF8F
IOPRE=EFFFFFFF  ISR  =7FFFFFF1

NICSR0 =3FFF0003   3=00008380   4=00008360   5=803BFF00   6=09E0F108
NICSR7 =00000000   9=04E204E2  10=00030000  11=00000000  12=00000000
NICSR13=00000000  15=0000FFFF

Ethernet  SA = 08-00-2B-12-81-22

MSIDR0 =0000  MSIDR1 =0000  MSIDR2 =0000  MSICSR =0000
MSIIDR =0007  MSITR  =80CE  MSITLP =0032  MSIILP =79D6
MSIDSCR=80FF  MSIDSSR=AC00  MSIDCR =0004
```

Example 4–8 Cont'd on next page

Example 4–8 (Cont.): Isolating Bad Memory Using T 9C

```
XCTRL =00  XCTRH =00  FIFO =00  COMD =12  STS  =00  SEQST =C4
INTR  =00  FFLAG =80  CON1 =07  CON2 =00  CON3 =00
>>>
```

MEAR contains the address of the failure. MESR indicates the type of failure. These registers are valid only after a memory test has failed. Bit 0 of MEAR is set if a non-existent memory error was detected. Bits 2 through 31 contain the failing address.

7. T A8 for testing a new memory module.

   Test A8 is used in the field when a new memory module is installed. It runs the same tests as the power-up script, but with different parameters. In addition, near the end when it runs test 40 to check the bitmap, it also counts soft errors and allows only one.

4.4.3 Running a Memory Test

To run a memory test, check your entries against Table 4–12.

The command is:

```
>>> T [test] [start_address] [ending_address] [address_increment]
```

Table 4–12: Running a Memory Test

Entry	Description
4F	Memory test, floating 1s and 0s pattern
4E	Memory byte test; use masked write cycles
4D	Memory address uniqueness
4C	ECC logic
48	Memory address shorts
47	Memory refresh logic

Table 4–13 describes the common memory parameters.

Table 4–13: Common Memory Parameters

Entry	Description
Start_address	First address to test; default is 0
Ending_address	Last address to test plus one; default is all memory
Address_increment	How often to run the test; minimum value is always 0x10 or greater. Test 48 tests every location in memory; for tests 4F, 4E, and 4C set this parameter low since the test will take a long time; tests 47 and 4D run fairly quickly so you can test all locations. To see the parameters used, run T 9E after the test finishes.

4.4.4 Additional Troubleshooting Suggestions

Note the following additional suggestions when diagnosing a possible memory failure.

If more than one memory module is failing, you should suspect the KN220 CPU module, KN220 I/O module, CPU/memory cable, or the backplane.

Always check the seating of the memory cable first before replacing a KN220 or MS220 module. If the seating appears to be improper, remove the cable and check the pins on the connector.

If you are rotating MS220 modules to verify that a particular memory module is causing the failure, be aware that a module may fail in a different way when in a different slot.

Be sure to put the modules back in their original positions when you are finished.

If memory errors are found in the error log, use the KN220 ROM-based diagnostics to see if it is an MS220 problem or if it is related to the KN220, CPU/memory interconnect cable, or backplane. Follow steps 1–3 of Section 4.3 and Section 4.4.2 to aid in isolating the failure.

Use the SHOW QBUS, SHOW DEVICE, and SET HOST/DUP commands when troubleshooting I/O subsystem problems.

Use the CONFIG command to help with configuration problems or when installing new options onto the Q-bus. See the command descriptions in Chapter 3.

You can run a DSSI device power-up diagnostic without performing a cold restart or spinning the disk drives down and back up.

Type the following at the console program:

```
>>> T 58 <node_number>
```

A CI RESET command is issued to the DSSI device, causing the device to perform its power-up diagnostics.

Parameter 1 is the DSSI node or port number. It must be in the range of 0–7 (0 is the default). Use the default for parameter 2.

You can perform this test repeatedly with the REPEAT command (R T 58 <node_number>). In that case, the drive's self-tests run repeatedly until you press Ctrl/C to terminate the test.

Once the test has completed successfully, you can examine the DSSI device's internal error logs by running the DUP local programs HISTRY and PARAMS. Refer to Section 4.7.3 and Section 4.7.5 for further information.

4.5 Loopback Tests

You can use external loopback tests to localize problems with the Ethernet, console, and DSSI subsystems.

4.5.1 DSSI Problems

For DSSI problems, run the SII external loopback test (test 56). To check the DSSI out to the KN220 I/O module connector, plug one end of the cable (17–02216–01) into the H3281 loopback connector and the other end into the DSSI connector on the KN220 I/O module. To test out to the end of the DSSI bus, power down the system, remove all DSSI devices with the exception of the KN220 from the bus, and replace the DSSI bus terminator plug with the external DSSI loopback connector 12–30702–01.

4.5.2 Ethernet Problems

For Ethernet problems, run the SGEC external loopback test by entering the following:

```
>>> T 5F 1 <CR>
```

Set the ThinWire/standard Ethernet switch on the H3602–AC to the appropriate position.

Use two 50-ohm H8225 terminators connected to an H8223 T-connector. Before running the test, attach this assembly to the H3602–AC ThinWire port.

To test the standard Ethernet connector, attach loopback connector 12–22196–02 to the H3602–AC standard Ethernet port.

To test further, connect the Ethernet port to a live network and run the SGEC_LPBCK_ASSIST test, number 59. To display responses from other network nodes, enter the following line:

```
>>> T 59 <CR>
```

4.5.3 Testing the Console Port

To test the console port at power-up, set the operation switch on the H3602–AC, using the procedure in Section 3.5.3. The H3103 connects the console port transmit and receive lines. At power-up, the SLU_EXT_LOOPBACK IPT then runs a continuous loopback test.

To test the end of the console terminal cable:

1. Plug the MMJ end of the console terminal cable into the H3602–AC.
2. Disconnect the other end of the cable from the terminal.
3. Plug an H8572 adapter into the disconnected end of the cable.
4. Connect the H3103 to the H8572.

While the test is running, the LED display on the CPU I/O insert should alternate between 7 and 4. A value of 7 latched in the display indicates a test failure. If the test fails, one of the following parts is faulty: the KN220, the H3602–AC, the cabling, or the I/O module.

4.6 Module Self-Tests

Module self-tests run when you power up the system. A module self-test can detect hard or repeatable errors, but usually not intermittent errors.

Module LEDs display pass/fail test results.

A pass by a module self-test does not guarantee that the module is good, because the test usually checks only the controller logic. The test usually does not check the module Q22-bus interface, the line drivers and receivers, or the connector pins—all of which have relatively high failure rates.

A fail by a module self-test is accurate, because the test does not require any other part of the system to be working.

The following modules do not have LED self-test indicators:

KLESI
LPV11
TSV05

The following modules have one green LED, which indicates that the module is receiving +5 and +12 Vdc:

CXA16
CXB16
CXY08

Table 4–14 lists loopback connectors for common KN220 system modules. See *Microsystems Options* for a description of specific module self-tests.

Table 4–14: Loopback Connectors for Q22-Bus Devices

Device	Module Loopback	Cable Loopback
CXA16/CXB16	H3103 + H8572[1]	
CXY08	H3046 (50-pin)	H3197 (25-pin)
DELQA	12–22196–02	
KN220/H3602–AC	H3103	H3103 + H8572

[1]For DSSI to KN220 or RF-series connector, use 17–02216–01 plus H3281 loopback. For connection to end of bus, use the DSSI loopback connector 12–30702–01.

4.7 RF-Series ISE Troubleshooting and Diagnostics

An RF-series integrated storage element (ISE) may fail either during initial power-up or during normal operation. In both cases, the failure is indicated by the lighting of the red fault LED on the operator control panel (OCP) on the enclosure front panel. The ISE also has a red fault LED, but it is not visible from the outside of the system enclosure.

If the drive is unable to execute the Power-On Self-Test (POST) successfully, the red fault LED remains lit and the ready LED does not come on, or both LEDs remain on.

POST is also used to handle the following two types of error conditions in the drive:

1. *Controller errors* are caused by the hardware associated with the controller function of the drive module. A controller error is fatal to the operation of the drive, since the controller cannot establish a logical connection to the host. If the red fault LED remains lit, replace the drive module.

2. *Drive errors* are caused by the hardware associated with the drive control function of the drive module. These errors are not fatal to the drive, since the drive can establish a logical connection and report the error to the host. Both LEDs go out for about 1 second, then the red fault LED lights. In this case, run either DRVTST, DRVEXR, or PARAMS (described in the next sections) to determine the error code.

Three configuration errors also commonly occur:

- More than one node with the same node number
- Identical node names
- Identical unit numbers

The first error cannot be detected by software. Use the SHOW DSSI command to display the second and third errors. This command lists each device connected to the DSSI bus by node name and unit number.

You must install a bus node ID plug in the bus node ID socket on the OCP. If the ISE is not connected to the OCP, the ISE reads its bus node ID from the three-switch DIP switchpack on the side of the drive.

The RF-series ISE contains the following local programs (described in the following sections):

DIRECT	A directory, in DUP specified format, of available local programs
DRVTST	A comprehensive drive functionality verification test
DRVEXR	A utility that exercises the ISE
HISTRY	A utility that saves information retained by the drive
ERASE	A utility that erases all user data from the disk
PARAMS	A utility that allows you to look at or change drive status, history, and parameters

A description of each local program follows, including a table showing the dialog of each program. The table also indicates the type of messages contained in the dialog, although the screen display will not indicate the message type. Message types are abbreviated as follows:

Q—Question
I—Information
T—Termination
FE—Fatal error

To access these local programs, use the Maintenance mode SET HOST /DUP command, which creates a virtual terminal connection to the storage device and the designated local program, using the Diagnostic and Utilities Protocol (DUP) standard dialog.

Once the connection is established, the local program is in control. When the program terminates, control is returned to the KN220 console. To abort or prematurely terminate a program and return control to the KN220 console, press Ctrl/C or Ctrl/Y.

4.7.1 DRVTST

DRVTST is a comprehensive functionality test. Errors detected by this test are isolated to the FRU level. The messages are listed in Table 4–15.

Table 4–15: DRVTST Messages

Message Type	Message
I	Copyright © 1988 Digital Equipment Corporation
Q	Write/read anywhere on the medium? [1=yes/(0=no)]
Q	User data will be corrupted. Proceed? [1=yes/(0=no)]
I	Five minutes to complete.
T	Test passed.
or	
FE	Unit is currently in use.[1]
FE	Operation aborted by user.
FE	xxxx—Unit diagnostics failed.[2]
FE	xxxx—Unit read/write test failed.[2]

[1]Either the drive is inoperative, in use by a host, or is currently running another local program.
[2]Refer to the diagnostic error code list at the end of this chapter.

Answering No to the first question (“Write/read...?”) or pressing Return results in a read-only test. DRVTST, however, writes to a diagnostic area on the disk. Answering Yes to the first question or pressing Return causes the second question to be displayed.

Answering No to the second question (“Proceed?”) or pressing Return is the same as answering No to the first question. Answering Yes to the second question permits write and read operations anywhere on the medium.

DRVTST resets the ECC error counters, then calls the timed I/O routine. After the timed I/O routine ends (5 minutes), DRVTST saves the counters again. It computes the uncorrectable error rate and byte (symbol) error rate. If either rate is too high, the test fails and the appropriate error code is displayed.

4.7.2 DRVEXR

The DRVEXR local program exercises the ISE. The test is data transfer intensive and indicates the overall integrity of the device. Table 4–16 lists the DRVEXR messages.

Table 4–16: DRVEXR Messages

Message Type	Message
I	Copyright © 1988 Digital Equipment Corporation
Q	Write/read anywhere on the medium? [1=yes/(0=no)]
Q	User data will be corrupted. Proceed? [1=yes/(0=no)]
Q	Test time in minutes? [(10)-100]
I	ddd minutes to complete.
I	ddddddddd blocks (512 bytes) transferred.
I	ddddddddd bytes in error (soft).
I	ddddddddd uncorrectable ECC errors (recoverable).
T	Complete.
or	
FE	Unit is currently in use.[1]
FE	Operation aborted by user.
FE	xxxx—Unit diagnostics failed.[2]
FE	xxxx—Unit read/write test failed.[2]

[1]Either the drive is inoperative, in use by a host, or is currently running another local program.
[2]Refer to the diagnostic error list at the end of this chapter.

Answering No to the first question (“Write/read...?”) results in a read-only test. DRVEXR, however, writes to a diagnostic area on the disk. Answering Yes to the first question causes the second question to be displayed.

Answering No to the second question (“Proceed?”) is the same as answering No to the first question. Answering Yes to the second question permits write and read operations anywhere on the medium.

NOTE: *If the write-protect switch on the OCP is pressed in (LED on) and you answer Yes to the second question, the drive does not allow the test to run. DRVEXR displays the error message 2006—Unit read/write test failed. In this case, the test has not failed but has been prevented from running.*

DRVEXR saves the error counters, then calls the timed I/O routine. After the timed I/O routine ends, DRVEXR saves the counters again. It then reports the total number of blocks transferred, bits in error, bytes in error, and uncorrectable errors.

DRVEXR uses the same timed I/O routine as DRVTST, with two exceptions:

- DRVTST always uses a fixed time of five minutes, whereas you specify the time of the DRVEXR routine.
- DRVTST determines whether the drive is good or bad. DRVEXR reports the data but does not determine the condition of the drive.

4.7.3 HISTRY

The HISTRY local program displays information about the history of the ISE. Table 4–17 lists the HISTRY messages.

Table 4–17: HISTRY Messages

Message Type	Field Length	Field Meaning
I	47 ASCII characters	Copyright notice
I	4 ASCII characters	Product name
I	12 ASCII characters	Drive serial number
I	6 ASCII characters	Node name
I	1 ASCII character	Allocation class
I	8 ASCII characters	Firmware revision level
I	17 ASCII characters	Hardware revision level
I	6 ASCII characters	Power-on hours
I	5 ASCII characters	Power cycles
I[1]	4 ASCII characters	Hexadecimal fault code
T		Complete

[1]Displays the last 11 fault codes as informational messages. Refer to the diagnostic error code list at the end of this chapter.

The following example shows a typical screen display when you run HISTRY:

```
Copyright © 1988 Digital Equipment Corporation
RF71
EN01082
SUSAN
0
RFX V101
RF71 PCB-5/ECO-00
617
21
A04F
A04F
A103
A04F
A404
A04F
A404
A04F
A404
A04F
A404
Complete.
```

If no errors have been logged, no hexadecimal fault codes are displayed.

4.7.4 ERASE

The ERASE local program overwrites application data on the drive while leaving the replacement control table (RCT) intact. This local program is used if an HDA must be replaced and the customer wants to protect any confidential or sensitive data.

Use ERASE only if the HDA must be replaced and only after you have backed up the customer's data.

Table 4–18 lists the ERASE messages.

Table 4–18: ERASE Messages

Message Type	Message
I	Copyright © 1988 Digital Equipment Corporation
Q	Write/read anywhere on the medium? [1=yes/(0=no)]
Q	User data will be corrupted. Proceed? [1=yes/(0=no)]
I	6 minutes to complete.
T	Complete.
or	
FE	Unit is currently in use.
FE	Operation aborted by user.
FE	xxxx—Unit diagnostics failed.[1]
FE	xxxx—Operation failed.[2]

[1]Refer to the diagnostic error code list at the end of this chapter.

[2]xxxx = one of the following error codes:
000D : Cannot write the RCT.
000E : Cannot read the RCT.
000F : Cannot find an RBN to which to revector.
0010 : The RAM copy of the bad block table is full.

If a failure is detected, the message indicating the failure will be followed by one or more messages containing error codes.

4.7.5 PARAMS

The PARAMS local program supports modifications to device parameters that you may need to change, such as device node name and allocation class. You invoke it in the same way as the other local programs. However, you use the following commands to make the modifications you need:

EXIT	Terminates PARAMS program
HELP	Prints a brief list of commands and their syntax
SET	Sets a parameter to a value
SHOW	Displays a parameter or a class of parameters
STATUS	Displays module configuration, history, or current counters, depending on the status type chosen
WRITE	Alters the device parameters

4.7.5.1 EXIT

Use the EXIT command to terminate the PARAMS local program.

4.7.5.2 HELP

Use the HELP command to display a brief list of available PARAMS commands, as shown in the example below.

```
PARAMS> HELP
EXIT
HELP
SET {parameter | .} value
SHOW {parameter | . | /class}
  /ALL     /CONST   /DRIVE
  /SERVO   /SCS     /MSCP
  /DUP
STATUS [type]
  CONFIG   LOGS     DATALINK
  PATHS
WRITE

PARAMS>
```

4.7.5.3 SET

Use the SET command to change the value of a given parameter. *Parameter* is the name or abbreviation of the parameter to be changed. *Value* is the value assigned to the parameter.

For example, SET NODE SUSAN sets the NODENAME parameter to SUSAN.

The following parameters are useful to Customer Services:

ALLCLASS	The controller allocation class. The allocation class should be set to match that of the host.
FIVEDIME	True (1) if MSCP should support five connections with ten credits each. False (0) if MSCP should support seven connections with seven credits each.
UNITNUM	The MSCP unit number.
FORCEUNI	True (1) if the unit number should be taken from the DSSI ID. False (0) if the UNITNUM value should be used instead.
NODENAME	The controller's SCS node name.
FORCENAM	True (1) if the SCS node name should be forced to the string RF71x (where x is a letter from A to H corresponding to the DSSI bus ID) instead of using the NODENAME value. False (0) if NODENAME is to be used.

4.7.5.4 SHOW

Use the SHOW command to display the settings of a parameter or a class of parameters. It displays the full name of the parameter (8 characters or less), the current value, the default value, radix and type, and any flags associated with each parameter.

4.7.5.5 STATUS

Use the STATUS command to display module configuration, history, or current counters, depending on the type specified. *Type* is the optional ASCII string that denotes the type of display desired. If you omit *Type*, all available status information is displayed. If present, it may be abbreviated. The following types are available:

CONFIG	Displays the module name, node name, power-up hours, power cycles, and other such configuration information. Unit failures are also displayed, if applicable.
LOGS	Displays the last 11 machine and bug checks on the module. The display includes the processor registers (D0–D7, A0–A7), the time and date of each failure (if available; otherwise the date 17 November 1858 is displayed), and some of the hardware registers.
DATALINK	Displays the data link counters.
PATHS	Displays available path information (open virtual circuits) from the point of view of the controller. The display includes the remote node names, DSSI IDs, software type and version, and counters for the messages and datagrams sent and/or received.

4.7.5.6 WRITE

Use the WRITE command to write the changes made while in PARAMS to the drive nonvolatile memory. The WRITE command is similar to the VMS SYSGEN WRITE command. Parameters are not available, but you must be aware of the system and/or drive requirements and use the WRITE command accordingly or it may not succeed in writing the changes.

The WRITE command may fail for one of the following reasons:

- You altered a parameter that required the unit, and the unit cannot be acquired (that is, the unit is not available to the host). Changing the unit number is an example of a parameter that requires the unit.
- You altered a parameter that required a controller initialization, and you replied negatively to the request for reboot. Changing the node name or the allocation class are examples of parameters that require controller initialization.
- Initial drive calibrations were in progress on the unit. The use of the WRITE command is inhibited while these calibrations are running.

4.7.6 Diagnostic Error Codes

Diagnostic error codes appear when you are running DRVTST, DRVEXR, or PARAMS. Most of the error codes indicate a failure of the drive module. The exceptions are listed below. The error codes are listed in Table 4–19. If you see any error code other than those listed below, replace the module.

Table 4–19: RF-Series ISE Diagnostic Error Codes

Code	Message	Meaning
2032/A032	Failed to see FLT go away	FLT bit of the spindle control status register was asserted for one of the following reasons: 1. Reference clock not present 2. Stuck rotor 3. Bad connection between HDA and module
203A/A03A	Cannot spin up, ACLOW is set in WrtFlt	Did not see ACOK signal, which is supplied by the host system power supply for staggered spin-up.
1314/9314	Front panel is broken	Could be either the module or the operator control panel or both.

4.8 Memory Diagnostics

The following subsections describe the memory diagnostics.

4.8.1 Test 30 - Bitmap Placing Test

The purpose of test 30 is to determine the size of memory and set up a bitmap in memory to be used by the memory tests. After all memory tests are complete, the bitmap defines memory that is good and available and memory that is bad or being used for the bitmap, busmap, etc.

An error is marked in the bitmap when a multibit error or a hard single bit error occurs. Soft single bit errors are not marked, but they are counted.

This test is run before other memory tests to make sure the memory bitmap is present. Though this test is usually run only once, other memory tests may be run more than once if desired.

The main purpose of this test is to find a good block of memory for the maps. It fails only when all memory is bad or the CPU memory logic is bad. If it fails, troubleshooting should be done by running the individual memory tests, do not try to troubleshoot using this test. The easiest way to run all individual tests is to run the A9 memory script (>>>T A9).

The bitmap placing test calls the data test (4F), the byte test (4E) and the address shorts test (48). During this time the parameters are determined by the test being run. First the memory is sized to determine the amount of memory available and describe any holes in memory if present.

The test looks for a 256 KB section of memory to be used for the bitmap, busmap and CVAX reserved console area. The bitmap ranges in size from 8 KB (for 32 MB of memory) to 128 KB (for 512 MB of memory), the busmap is always 32 KB and the CVAX reserved console area is 16 Kb. The other 80 KB may be used by diagnostics. The actual size of the bitmap is determined by the results of the memory sizing routine previously run. The test starts at the top of available memory and tests the highest 256 KB section of each 8 MB of memory until a good section is found for the maps or the bottom of memory is reached in which case a failure is reported. When a good section is found, the bitmap is initialized to mark all memory other than the bitmap, busmap and CVAX reserved console area as good, these are marked as taken (same as being marked bad). Load the bitmap address, bitmap length and busmap address in the diagnostic state. Save address of CVAX reserved console area. The busmap itself is not properly setup at this time, it is left cleared.

NOTE: *This test reports an error only if it fails to find a good section of memory for the maps or if the size of memory is 0.*

It is important to note that the bitmap is always initialized to all good memory when this test is run except for memory holes which are marked as bad. You must run all memory tests after this test to insure that bad memory is marked correctly.

The following list contains a detailed description of the procedure.

1. Clear bitmap address, bitmap checksum, bitmap length and busmap address in diagnostic state. Mark RAM available flag (DST$V_RAM) as not available.
2. Size memory from address 0 upwards. Memory is present if it responds to a write cycle without timing out (NXM). If memory is not present in 0 then there is error. Size every 32 MB from the beginning to end of memory. Store the results in a 16 bit mask with each bit defining one 32 MB memory bank, 0 = not present, 1 = present. Bit 0 is first 32 MB, etc.

   NOTE: *This routine is a destructive sizing routine, it will destroy any contents present in the memory locations sized. The sizing routine does not write below 192 KB to protect R3000 console code.*

3. From the top of memory -256KB, test the highest 256KB block in each 8MB section of memory until either a good block is found (success) or the bottom of memory is reached (failure).
4. Call the memory data test (4F) with the starting and ending address of the current 256KB section of memory to test. The address increment is set to 32 KB to test eight QWs in section.
5. Call the memory byte test (4E) with the starting and ending address of the current 256KB section of memory to test. The address increment is set to 64 KB to test four QWs in section.
6. Call the memory address shorts test (48) with the starting and ending address of the current 256KB section of memory to test. The address shorts test will always test every word selected. The misc parameter is set to use both I and D cache and to only run the first three passes of the address shorts test. This will leave the contents of the last pattern (0x55555555) left unchecked in the section of memory.
7. For all of memory from beginning to end by (4 KB + 4) clear the first QW in each 16 KB block. Ignore any errors if they occur. Do not clear any locations in the 256 KB area under test. Do not start below 192 KB to protect the R3000 console.
8. Call the memory address shorts test (48) with the starting and ending address of the current 256KB section of memory to test. The address shorts test will always test every word selected. The misc parameter is set to use both I and D cache and to only run the fourth pass of the address shorts test. This will verify that the pattern of 0x55555555 left by running passes 1,2 and 3 of the test was not disturbed by accessing other locations in memory.
9. If any error occurs during test then subtract 8 MB from the current base address and continue trying.
10. If a good 256 KB section of memory was not found then report error.
11. If a good 256 KB section of memory was found then continue.
12. Clear the 256 KB block of memory found.
13. Mark all locations in bitmap as good. Save length of bitmap and address of bitmap in diagnostic state. Determine correct checksum for bitmap and then save the complement of the good checksum. This makes sure the bitmap checksum is bad after running this test. The correct checksum is not placed in the diagnostic state until the test which counts bad pages in memory is run which is after all other memory tests. At this time the bitmap should be valid. Save the address of

the 16 KB CVAX reserved area for the console program in the console portion of SSC RAM. Mark RAM available flag good.

4.8.2 Test 4F - Memory Data Tests

The purpose of Test 4F is to verify that each bit in the data path can be written to a one and a zero individually. This test also checks for shorts between individual paths. The test always checks a QW at a time to make sure that both sides of the memory array are checked. The test need only be run once for each array of memory chips.

4.8.3 Test 4E - Memory Byte Tests

Test 4E verifies that masked write cycles work correctly. The test writes to each byte of a QW and verifies the data. The test need only be run once for each array of memory chips.

4.8.4 Test 4D - Memory Address Uniqueness Test

The main purpose of test 4D is to verify that each location in memory can be uniquely addressed. Write each LW from the starting address to ending address with its own R3000 physical address.

From starting address to ending address read back each LW and verify that it contains its address.

4.8.5 Test 4C - Memory ECC Logic, Verify Error Detection and Reporting

The main purpose of test 4C is to test ECC logic. It is not intended to test the memory boards explicitly. The test introduces single and multiple bit errors and then reads back or tries to do a masked write to the location and verifies the proper responses and error logging.

4.8.6 Test 48 - Memory Address/Shorts Test

Test 48 verifies that all locations in each bank can be uniquely written and that each of the 39 data bits in each LW can be written to a one and a zero. This test also writes all locations in memory with good ECC.

This test takes 12 seconds to verify each 32 MBs of memory.

T 48 runs address shorts test across every LW from beginning to end address.

4.8.7 Test 47 - Memory Data Retention, Verify Refresh Logic

Test 47 verifies that the refresh logic is working for all memory boards.

4.8.8 Test 40 - Memory Count; Bad Pages Marked in Bitmap

Test 40 counts the number of pages marked in the memory bitmap. It also places the correct bitmap checksum into the diagnostic state.

4.8.9 Test 9A - Define Current Memory Configuration

Test 9A is an optional test which allows a user to define exactly what size memory boards are present in the system and in what order they are. This information is useful to the memory tests to allow them to associate memory addresses with physical memory boards in the backplane (FRUs).

NOTE: *It is not possible for a sizing routine to associate accurately memory addresses with board numbers for all possible memory configurations.*

This is an optional test that the user can invoke to prompt the user with questions to describe the actual memory configuration of the system. It then verifies the information and if it is reasonable it saves it in SSC RAM. Otherwise an error is reported.

The memory configuration information is used by memory diagnostics to allow the test to identify which memory board is bad if a failure occurs.

This routine must be manually invoked and is not run automatically by any of the powerup scripts. The routine is not required to run any of the test scripts.

The following is a example of the interaction with the user. In this case the system has two 32-Mbyte memory boards. The memory board number (M7639-x) is shown because it is readable from the handle end of the module.

```
                        0 MB = 0;
   32 MB, M7639-A, MS220-AA = 1;
   64 MB, M7639-B, MS220-BA = 2;
MEM FRU 0:3

MEM FRU 0 = 1
MEM FRU 1 = 1
MEM FRU 2 = 0

MEM FRU = 0  32 MB, M7639-A, MS220-AA  00000000 01FFFFFF
MEM FRU = 1  32 MB, M7639-A, MS220-AA  02000000 03FFFFFF
>>>
```

4.9 SCSI Controller Chip Test

There are five SCSI controller chip tests, as shown below.

4.9.1 ASC Reset Test

The ASC reset test checks whether the Software can reset the ASC and bring it to a known state.

Reset the ASC through the COMMAND register and check Initial values of some of the registers.

Reset the ASC through the RESET command and the COMMAND register. The following registers will ZERO after reset.

1. Command Register
2. First element of FIFO
3. Status Register
4. Interrupt Register
5. FIFO Flag and Sequence register
6. Configuration Registers 1, 2 and 3

4.9.2 ASC Register Test

The purpose of this test is find whether the software can "Access" internal bits in a register. This is a read write test done on a register that can be read as well as written to.

The configuration registers are chosen for this test. Each register is run through an 8 bit up-counter and the values are tested at each increment. Configuration 2 register is run through an 8 bit down-counter.

4.9.3 ASC Interrupt Test

ASC interrupt tests the ability of the ASC to generate an Interrupt and tests the ability of the hardware to field an interrupt generated by the SCSI subsystem.

There are two ways that the ASC can be 'faked' into generating an interrupt without actually having any devices connected to the SCSI bus. These interrupts are generated and fielded to prove that the interrupts can be handled by the SCSI subsystem.

After Poweron/Reset, ASC will be in disconnected mode. Issuing a command from another mode (either Initiator or target mode) will cause an Illegal command interrupt.

The ASC is reset through the software. An ISR is setup to handle the interrupt and a flag is setup. The interrupt is initiated by writing an Initiator command to the command register. The flag is checked and reset in the ISR. If the flag is not reset within a reasonable time, it signifies error.

Without an actual device (target) on the bus, it is difficult to test the Disconnect Interrupt. The Target Disconnect interrupt is the same as the interrupt that indicates Selection or Reselection Timeout, which can be tested.

Set the SELECT/RESELECT TIMEOUT register to 1 (minimum value), then issue a select command with the SCSI ID the same as the ASC. The effect of this is to setup a short timeout and selecting the host itself. After the timeout, expect the Disconnected Interrupt. The interrupt is tested with a flag being set before causing the interrupt and checking/resetting the flag in the ISR.

4.9.4 ASC FIFO Test

This is to test the working of the FIFO and associated FIFO flags inside the ASC.

The FIFO inside the ASC helps speed up the transfer of Command, data and messages to and from the SCSI bus. The FIFO test checks out the operation of the 16 by 9 FIFO along with the FIFO flags inside the ASC.

The FIFO registers tells how many bytes are in the FIFO. Load a number of bytes into the FIFO and check the flags. The number in the register should agree with the number of bytes sent. This is checked for numbers 0 through 15 and the flags are checked. Also in this test, the bytes read from out of FIFO should follow the order in which it was written.

4.9.5 ASC DMA Counter Register Test

This is to test the working of the DMA counter and count register inside ASC.

The DMA register stores the DMA transfer count. A DMA transfer command instructs the ASC to use this counter to do DMA operation. This test will check whether a DMA command will transfer this value to the internal count-down counter.

Setup the COUNT register with a 'starting value'. Issue a DMA NOP instruction to the ASC. This command will transfer the COUNT value into the internal count-down counter. A read from the COUNTER register will return the number of bytes remaining to be transferred. In this case no bytes will have been transferred, so the values read from COUNT should be identical to the value written to the COUNT register. Repeat the test from the 'starting value' being the lowest possible number to the highest possible count value.

Appendix A
ULTRIX–32 Exerciser and uerf Command Summary

This appendix contains a summary of ULTRIX–32 exerciser and uerf commands to help you troubleshoot and diagnose errors in the DECsystem 5500.

See the following documents for detailed information on the commands:

- *ULTRIX–32 Guide to System Exercisers*
- *ULTRIX–32 Guide to the Error Logger System*

A.1 On-line ULTRIX Exerciser

The ULTRIX exercisers perform functional system and device testing. The exercisers are run in single- or multiuser mode from an account with *root* privileges.

The exercisers log status information in *LOG* files. Normal device errors are handled by the error log and *uerf*. You can run each of the exercisers in the background by ending each command line with an *&*. This allows many (the same or different) exercisers to be run concurrently, which enhances your ability to perform system testing.

To run the exercisers, your current directory must be the *field* account. To terminate the exercisers, enter Ctrl/C if the job is in the foreground, or **kill -15** *pid* if in the background. When you run an exerciser in the background, *pid* is displayed when the command is envoked.

A time stamp entry is made in the system error log each time you stop or start an exerciser. Use the *uerf* option **-r** *350* to include these in an error report. All the system exercisers, except *netx*, have the **-o** option. The **-o** option allows you to specify a file where diagnostic output is saved when the exerciser terminates.

Exercising More Than One Part of the System

You can run more than one exerciser at the same time. Keep in mind, however, that the more processes you have running, the slower the system

performs. Before exercising the system extensively, make sure there are no other users on the system.

To exercise more than one part of the system simultaneously, use the **syscript** maintenance command. The **syscript** command asks you which exercisers you want to run, how long you want to run each exerciser, and how many exercisers you want to run at one time. The **syscript** command allows you to exercise a device, a subsystem, or the entire system.

You can start each exerciser by using either of the following methods:

- Manually, by specifying the time parameter (**-t** option) and by placing each command in the background before executing the next command
- By typing the **syscript** command as follows:

```
# syscript
```

Once the **syscript** command is running, answer the questions displayed on the console. The **syscript** command then executes the individual exercisers and creates a file called testsuite, which contains all the answers you entered. You can reexecute the commands in the testsuite file by entering the following, which causes testsuite to execute using the original commands and parameters that you specified:

```
# sh testsuite
```

A.1.1 Communications Exerciser (Asynchronous Serial Lines)

The communications exerciser writes, reads, and validates random data and packet lengths on communication lines as specified.

Syntax

cmx [-h] [-o*file***] [-t** *m***] -l***line#*

Options

-h	Prints a help message.
-o*file*	Writes run-time statistics to *file*. Default file is *#LOG_CMX_##*.
-t*min*	Runs the exerciser for *x* minutes. Default is run continuously.
-l*line#*	Specifies the line number to exercise. For example, if the line to be exercised is */dev/tty03*, *line#=03*.

Usage

Any line to be exercised must have a loopback connector on the communication option's bulkhead panel or the end of the cable. Any line to be exercised must be disabled in the *letc/ttys* file by setting the *status* to off.

Exercise line tty01 and tty03 for 10 minutes in the background:

```
cmx -t10 -1 01 03
```

A.1.2 Disk Exerciser

CAUTION: *This exerciser can DESTRUCTIVELY WRITE on a disk. Do not use this exerciser on any portion of a disk that contains customer data.*

*The **-p** and **-c** options destroy data on a disk. The **-r**dev command does not overwrite data.*

Syntax

dskx [*options***] -r***dev*
dskx [*options***] -p***devpart*
dskx [*options***] -c***dev*

Arguments

r*dev*	Random read-only test on all but the **c** partition.
-p*devpart*	Writes, reads, and validates on device *dev* on partition *part*.
-c*dev*	Writes, reads, and validates on devide *dev* on all but the **c** partition.

Options

-h	Prints a help message.
-o*file*	Writes run-time statistics to *file*. The default file is #*LOG_DSKX_*##.
-t*m*	Runs the exerciser for *m* minutes. The default is run continuously.

Test (read only) the first RA disk in the system (ra0) for 20 minutes in the background. Diagnostics display every five minutes:

```
dskx -rra0 -t20 -d5 &
```

A.1.3 File System Exerciser

The file system exerciser initiates multiple processes and creates, writes, closes, opens, and reads a test file of random data.

Syntax

fsx [-h] [-o*file***] [-t***m* **] -f***path*) **[-p***n***]**

Options

-h	Prints a help message.
-o*file*	Writes run-time statistics to *file*. The default file is #*LOG_FSX_##*.
-t*min*	Runs the exerciser for *x* minutes. The default is run continuously.
-f*path*	Path name of the file system directory to test. Default is **/usr/field**.
-p*n*	Number of *fsx* processes to spawn. Maximum is 250. Default is 20.

Usage

This test writes and reads data on the disk; it is not destructive to the customer's data. The file system exerciser can also be used on an NFS-mounted file system.

Exercise the */usr/tmp* file system continuously using 10 processes in the background:

```
fsx -p10 -f/usr/tmp &
```

A.1.4 Line Printer Exerciser

Syntax

lpx [-h] [-o*file***] [-t***m***] -f***path* **[-p***n***]**

Arguments

-d*dev*	Printer device name to exercise.

Options

-h	Prints a help message.
-o*file*	Writes run-time statistics to *file*. Default file is #*LOG_LPX_##*.
-t*min*	Runs the exerciser for *x* minutes. Default is run continuously.

-f*path*	Path name of the file system directory to test. Default is **/usr/field**.
-p*n*	To save paper, pauses printing for *n* minutes and only exercises the controller. Default is 15. A value of 0 indicates no pause.

Exercise **lp1**: lpx -dlp 1

A.1.5 Memory Exerciser

Syntax

memx [-h] [-s] [-o*file***] [-t***m***] [-m***j***] [-p***k***]**

Options

-h	Prints a help message.
-s	Disables shared memory testing. Shared memory is software functionality, not hardware.
-o*file*	Writes run-time statistics to *file*. Default file is *#LOG_MEMX_##*.
-t*min*	Runs exerciser for *x* minutes. Default is run continuously.
-m*j*	Memory size in *j* bytes to be tested by each spawned process. Default is (total memory)/20.
-p*k*	Number of *memx* processes to spawn. Maximum is 20. Default is 20.

Usage

The memory exerciser is restricted by available swap space. Errors like `out of memory` generally indicate swap space was used up. If you have more physical memory than swap space, you may see this problem. If so, reduce the number of spawned processes and/or the size of memory you are testing. Running the memory exerciser can also cause other users to have the same memory problem.

Exercise all of memory and the shared memory functionality for 10 minutes in the background:

```
memx -t10 &
```

A.1.6 Magtape Exerciser

The magtape exerciser reads, writes, and validates random data from the beginning of the tape (BOT) to the end of the tape (EOT).

Syntax

mtx [*options***] -a***dev*

mtx [*options***] -s***dev*
mtx [*options***] -l***dev*
mtx [*options***] -v***dev*

Arguments

-a*dev*	Use short-, long-, and variable-length record tests on raw device *dev*.
-s*dev*	Use short records on raw *dev*.
-l*dev*	Use long records on raw *dev*.
-v*dev*	Use variable records on raw *dev*.

Options

-h	Prints a help message.
-o*file*	Writes run-time statistics to *file*. Default file is #*LOG_MTX_##*.
-t*i*	Runs exerciser for *i* minutes. Default is run continuously.
-r*j*	Record length for long record test. Range is 1 to 20480. Default is 10240.
-t*k*	Size of file in *k* number of records. Default: -1, go to EOT.

Run all record lengths on tape drive rmt0h for five minutes in the background:

```
mtx -armt0h -t5 &
```

A.1.7 TCP/IP Network Exerciser

Syntax

netx [-h] [-t*m***] [-p***m***]***nodename*

Arguments

nodename	Node name of target system to test. May also be the host system name.

Options

-h	Prints a help message.
-t*min*	Runs exerciser for *x* minutes. Default is run continuously.
-p*m*	Port number.

Usage

The **TCP** echo service defined in the */etc/inetd.conf* file must not be commented out (# at start of line) on the host and target systems.

Exercise the network from the local host to the remote node **max** continuously in the background:

```
netx max &
```

A.2 uerf Error Log Commands

The uerf utility generates error log reports and does bit-to-text translation for hardware device registers and messages, similar to *ERF* on VMS. Syntax is case sensitive. If no options are specified, all errors are reported.

To disable error logging to an error log file, type:

 # /etc/eli -d

To enable error logging in multiuser mode, type:

 # /etc/eli -e

Syntax

/etc/uerf [options...]

Options

-A *adapter_type*	*Example:* /etc/uerf -A uba,nmi
aie	BVP controller
aio	BVP controller
bia	BI LESI adapter
bua	BI UNIBUS adapter
nmi	NMI errors
uba	VAX UNIBUS adapter
default	Report all error types

-c *classes*	*Example:* /etc/uerf -c oper
err	All hardware and software errors
maint	Maintenance events
oper	System status; startup/shutdown; configuration

-D	Reports errors for MSCP disks (*ra, rd*). Default: all MSCP disks are reported. *Example*: /etc/uerf -D ra60
-f	Specifies the error log file to be used to generate the report. *Example:* /etc/uerf -f old.errorlog
-h	Displays a brief help message.
-H	Selects errors only for the specified system name. *Example:* /etc/uerf -H guru

---

-M *mainframe_errors*	*Example:* /etc/uerf -M mem
cpu	Reports CPU errors and machine checks.
mem	Reports memory errors (SBE and DBE).
default	Reports all error types.

-n	Uerf runs. Waits for errors to be logged and immediately reports them.

-o *output*	*Example*: /etc/uerf -o full
brief	Reports errors in brief format (*default*).
full	Reports all information for each error.
terse	No bit-to-text translation for register values.

-O *operating_system_events*	*Example*: /etc/uerf -O seg,raf
aef	Arithmetic exception faults
ast	Asynchronous trap exception faults
bpt	Breakpoint instruction faults
cmp	Compatibility mode faults
pag	Page faults

pif	Privileged instruction faults
pro	Protection faults
ptf	Page table faults
raf	Reserved address faults
rof	Reserved operand faults
scf	System call exception faults
seg	Segmentation faults
tra	Trace exception faults
xfc	Reports xfc instruction faults

-R	Reports errors in reverse chronological order.

-r *record_type* *Example*: /etc/uerf -r 102,210,250

Hardware Detected Error Types:

100	Machine check
101	Memory CRD/RDS errors
102	Disk errors
103	Tape errors
104	Device controller errors
105	Adapter errors
106	Bus errors
107	Stray interrupts
108	Asynchronous write errors
109	Exceptions/faults
112	Stack dump

Software Detected Error Types:

200	Panics (bug checks)
201	CI pdd information

Informational ASCII Message Types:

250	Informational

Operational Message Types:

300	Startup
301	Shutdown
310	Time change
350	Diagnostic information
351	Repair information

-s *sequence numbers* — Reports errors for the specified sequence numbers.
EXAMPLE: /etc/uerf -s 1011,1320

-S — Summarizes error information.
Example: /etc/uerf -S -o full

-t *s:dd-mmm-yyyy,hh:mm:ss e:dd-mmmm-yyyy,hh:mm:ss*

Example: /etc/uerf -t s:08-aug-1989:13:20:00

s	Starting date and time
e	Ending date and time
dd	Day
mmm	Month
yyyy	Year
hh	Hour
mm	Minute
ss	Second

-T	Reports errors for TMSCP tapes (*tk, tu*). Default: all TMSCP tapes are reported. *Example*: /etc/uerf -T tu81
-x	Excludes specified error types from the report. *Example*: /etc/uerf -x -r 102,103
-Z	Displays the entire error record as hexadecimal data. Used only for debugging.

Appendix B
KN220 Address Assignments

This appendix explains how to access R3000 physical address locations and provides physical address space maps for the KN220 CPU module set.

B.1 Accessing Physical Locations (R3000)

From the R3000 processor, you must use virtual addresses to access physical locations.

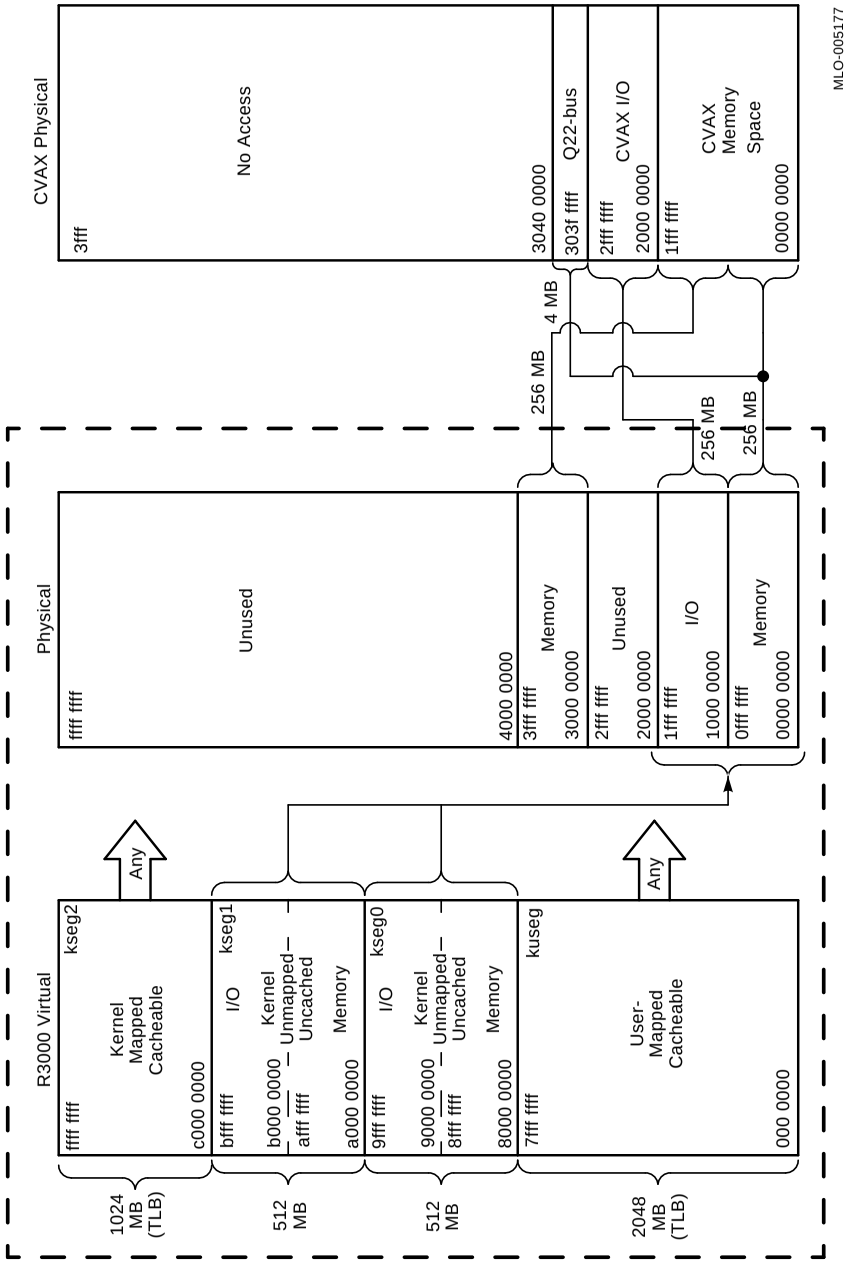
Figure B–1 shows the virtual memory map for the R3000 and CVAX processors. Note that the R3000 virtual addresses are separated into kernel segments (ksegs):

- To address physical locations, use the upper four bits of the kernel segment.
- Always reference kernel segment 1 for I/O addresses, which are unmapped and uncached.

For example, from the R3000 processor:

- If you are using kernel segment 0 (80000000; unmapped and cached), use 8001A340 to access physical location 0001A340.
- If you are using kernel segment 1 (a0000000; unmapped and uncached), use a001A340 to access physical location 0001A340.
- Use a008000C to access DMA error address register 1008000C (physical).

Figure B–1: KN220 Virtual Memory Map



Sections B.2 through B.6 list contents and address ranges for the KN220 CPU module (M7637–AA).

B.2 R3000 Physical Address Space Map (M7637–AA)

Table B–1: R3000 Physical Address Space

Contents	Address Range
Local memory space (up to 256 Mbytes)	0000 0000–0FFF FFFF
Local Q22-Bus I/O Space	
Reserved Q22-bus I/O space	1000 0000–1000 0007
Q22-bus floating address space	1000 0008–1000 07FF
User reserved Q22-bus I/O space	1000 0800–1000 0FFF
Reserved and fixed CSR Q22-bus I/O space	1000 1000–1000 1F3F
Interprocessor communication register	1000 1F40
Reserved Q22-bus I/O space	1000 1F48–1000 1FFF
Reserved I/O module address space	1000 2000–1000 7FFF
SGEC internal registers	1000 8000–1000 803C
Reserved (copies of SGEC regs)	1000 8040–1001 FFFF
Reserved I/O module address space	1002 0000–1003 FFFF
Two copies of CVAX ROM	1004 0000–1007 FFFF
Q22 system configuration register	1008 0000
Q22 system error register	1008 0004
Q22 master error address register	1008 0008
Q22 slave error address register	1008 000C
Q22-bus map base register	1008 0010
Reserved	1008 0014–1008 3FFF
Interrupt status register	1008 4000
Boot and diagnostic register	1008 4004
Select processor register	1008 4008
Interval timer register	1008 4010
Reserved I/O module address space	1008 4014–1008 7FFF
Q22-bus map registers	1008 8000–1008 FFFF
Reserved I/O module address space	1009 0000–1009 FFFF
DSSI buffer RAM	1010 0000–1011 FFFF
NI station address ROM	1012 0000–1012 007C
Reserved I/O module address space	1012 0080–1013 FFFF
SSC base address register	1014 0000
SSC configuration register	1014 0010

Table B–1 (Cont.): R3000 Physical Address Space

Contents	Address Range
Local Q22-Bus I/O Space	
CDAL bus timeout control register	1014 0020
Diagnostic LED register	1014 0030
Reserved I/O module address space	1014 0034–1014 0068
Time-of-year register	1014 006C
Reserved	1014 0070–1014 007C
CVAX console receiver control/status	1014 0080
CVAX console receiver data buffer	1014 0084
CVAX console transmitter control/status	1014 0088
CVAX console transmitter data buffer	1014 008C
Reserved	1014 0090–1014 00DB
I/O system reset register	1014 00DC
Reserved	1014 00E0
ROM data register	1014 00F0
Bus timeout counter	1014 00F4
Interval timer	1014 00F8
Reserved	1014 00FC–1014 00FF
Timer 0 control register	1014 0100
Timer 0 interval register	1014 0104
Timer 0 next interval register	1014 0108
Timer 0 interrupt vector	1014 010C
Timer 1 control register	1014 0110
Timer 1 interval register	1014 0114
Timer 1 next interval register	1014 0118
Timer 1 interrupt vector	1014 011C
MSIDB address decode match register	1014 0130
MSIDB address decode mask register	1014 0134
LIOD address decode match register	1014 0140
LIOD address decode match register	1014 0144
Reserved	1014 0148–1014 033F
CVAX battery backed-up RAM	1014 0400–1014 07FF
Reserved I/O module address space	1014 0800–1015 FFFF
SII internal registers	1016 0000–1016 007C
Reserved I/O module address space	1016 0080–1017 FFFF
Reserved	1018 0000–13FF FFFF
Local Q22-bus memory space	1400 0000–143F FFFF
Reserved (4 copies local Q22-bus memory)	1440 0000–14FF FFFF
Reserved	1500 0000–1600 004F
Vector read register 0[1]	1600 0050

[1]Accessible only from R3000 processor.

Table B–1 (Cont.): R3000 Physical Address Space

Contents	Address Range
Local Q22-Bus I/O Space	
Vector read register 1[1]	1600 0054
Vector read register 2[1]	1600 0058
Vector read register 3[1]	1600 005C
Reserved	1600 0060–1610 0058
Vector write register	1610 005C
Reserved	1610 0060–FFFF FFFF
I/O presence register	1700 0000–1703 FFFF
Memory error syndrome register	1704 0000–1707 FFFF
Memory error address register	1708 0000–170B FFFF
Memory ID register	170C 0000–170F FFFF
SCSI 53C94 registers	1710 0000–1710 0028
Reserved	1710 002C–1710 003C
Reserved (copies of SCSI registers)	1710 0040–1713 FFFF
SCSI DMA register	1714 0000
Reserved (copies of DMA register)	1714 0004–1717 FFFF
SCSI buffer RAM	1718 0000–1719 FFFF
Reserved (copy of SCSI RAM)	171A 0000–171B FFFF
Reserved SCSI address space	171C 0000–171F FFFF
R3000 LED register	1720 0000–1723 FFFF
Reserved I/O module address space	1724 0000–17FF FFFF
R3000 nonvolatile RAM	1800 0000–1807 FFFF
Reserved I/O module address space	1808 0000–180F FFFF
R3000 UART registers	1810 0000–1810 003C
Reserved (copies of UART registers)	1810 0040–1813 FFFF
Reserved I/O module address space	1814 0000–18FF FFFF
VME option memory and I/O space	1900 0000–1AFF FFFF
FDDI option memory and I/O space	1B00 0000–1CFF FFFF
Reserved I/O module address space	1D00 0000–1FBF FFFF
R3000 ROM	1FC0 0000–1FC3 FFFF
Reserved I/O module address space	1FC4 0000–1FFF FFFF
Reserved	2000 0000–2FFF FFFF
Memory space (up to 256 Mbytes)	3000 0000–3FFF FFFF
Reserved	4000 0000–FFFF FFFF

[1]Accessible only from R3000 processor.

B.3 R3000 Physical I/O Address Space Map (M7638–AA)

Table B–2: R3000 Physical I/O Addresses

Contents	Address Range
Local Q22-Bus I/O Space	
Reserved Q22-bus I/O space	1000 0000–1000 0007
Q22-bus floating address space	1000 0008–1000 07FF
User reserved I/O module address space	1000 0800–1000 0FFF
Reserved Q22-bus I/O space	1000 1000–1000 1F3F
Interprocessor communication register	1000 1F40
Reserved Q22-bus I/O space	1000 1F48–1000 1FFF
Reserved I/O module address space	1000 2000–1000 7FFF
SGEC CSR0: vector, IPL, mode	1000 8000
SGEC CSR1: polling demand	1000 8004
SGEC CSR2: reserved register	1000 8008
SGEC CSR3: receive descriptor list	1000 800C
SGEC CSR4: transmit descriptor list	1000 8010
SGEC CSR5: status register	1000 8014
SGEC CSR6: command and mode register	1000 8018
SGEC CSR7: system base register	1000 801C
SGEC CSR8: reserved register	1000 8020
SGEC CSR9: watchdog timers	1000 8024
SGEC CSR10: revision number and missed frame count	1000 8028
SGEC boot message registers	1000 802C–1000 8034
SGEC diagnostic registers	1000 8038–1000 803C
Reserved (copies of SGEC registers)	1000 8040–1011 FFFF
Reserved I/O module address space	1002 0000–1003 FFFF
Two copies of CVAX ROM	1004 0000–1007 FFFF
Q22 system configuration register	1008 0000
Q22 system error register	1008 0004
Q22 master error address register	1008 0008
Q22 slave error address register	1008 000C
Q22-bus map base register	1008 0010
Reserved	1008 0014–1008 3FFF
Interrupt status register	1008 4000
Boot and diagnostic register	1008 4004
Select processor register	1008 4008
Interval timer register	1008 4010

Table B–2 (Cont.): R3000 Physical I/O Addresses

Contents	Address Range
Local Q22-Bus I/O Space	
Reserved I/O module address space	1008 4014–1008 7FFF
Q22-bus map registers	1008 8000–1008 FFFF
Reserved I/O module address space	1009 0000–1009 FFFF
DSSI buffer RAM	1010 0000–1011 FFFF
NI station address ROM	1012 0000–1012 007C
Reserved I/O module address space	1012 0080–1013 FFFF
SSC base address register	1014 0000
SSC configuration register	1014 0010
CDAL bus timeout control register	1014 0020
Diagnostic LED register	1014 0030
Reserved I/O module address space	1014 0034–1014 0068
Time-of-year register	1014 006C
Reserved	1014 0070–1014 007C
CVAX console receiver control/status	1014 0080
CVAX console receiver data buffer	1014 0084
CVAX console transmitter control/status	1014 0088
CVAX console transmitter data buffer	1014 008C
Reserved	1014 0090–1014 00DB
I/O system reset register	1014 00DC
Reserved	1014 00E0
ROM data register	1014 00F0
Bus timeout counter	1014 00F4
Interval timer	1014 00F8
Reserved	1014 00FC–1014 00FF
Timer 0 control register	1014 0100
Timer 0 interval register	1014 0104
Timer 0 next interval register	1014 0108
Timer 0 interrupt vector	1014 010C
Timer 1 control register	1014 0110
Timer 1 interval register	1014 0114
Timer 1 next interval register	1014 0118
Timer 1 interrupt vector	1014 011C
DSSIDB address decode match register	1014 0130
DSSIDB address decode match register	1014 0134
LIOD address decode match register	1014 0140
LIOD address decode match register	1014 0144
Reserved	1014 0148–1014 033F
CVAX battery backed-up RAM	1014 0400–1014 07FF

Table B–2 (Cont.): R3000 Physical I/O Addresses

Contents	Address Range
Local Q22-Bus I/O Space	
Reserved I/O module address space	1014 0800–1015 FFFF
DSSI diagnostic register 0	1016 0000
DSSI diagnostic register 1	1016 0004
DSSI diagnostic register 2	1016 0008
DSSI control and status register	1016 000C
DSSI ID register	1016 0010
Reserved DSSI register	1016 0014
Reserved DSSI register	1016 0018
DSSI timeout register	1016 001C
Reserved DSSI register	1016 0020
Reserved DSSI register	1016 0024
Reserved DSSI register	1016 0028
Reserved DSSI register	1016 002C
Reserved DSSI register	1016 0030
Reserved DSSI register	1016 0034
DSSI short target list pointer	1016 0038
DSSI long target list pointer	1016 003C
DSSI initiator list pointer	1016 0040
DSSI DSSI control register	1016 0044
DSSI DSSI status register	1016 004C
Reserved DSSI register	1016 004C
Reserved DSSI register	1016 0050
DSSI diagnostic control register	1016 0054
DSSI clock control register	1016 0058
DSSI internal state register 0	1016 005C
DSSI internal state register 1	1016 0060
DSSI internal state register 2	1016 0064
DSSI internal state register 3	1016 0068
Reserved DSSI register	1016 006C
Reserved DSSI register	1016 0070
Reserved DSSI register	1016 0074
Reserved DSSI register	1016 0078
Reserved DSSI register	1016 007C
Reserved I/O module address space	1016 0080–1017 FFFF
Reserved	1018 0000–13FF FFFF
Local Q22-bus memory space	1400 0000–143F FFFF
Reserved (4 copies local Q22 memory)	1440 0000–14FF FFFF
Reserved	1500 0000–1600 004F

Table B–2 (Cont.): R3000 Physical I/O Addresses

Contents	Address Range
Local Q22-Bus I/O Space	
Vector read register 0[1]	1600 0050
Vector read register 1[1]	1600 0054
Vector read register 2[1]	1600 0058
Vector read register 3[1]	1600 005C
Reserved	1600 0060–1610 0058
Vector write register[1]	1610 005C
Reserved	1600 0060–16FF FFFF
I/O presence register	1700 0000–1703 FFFF
Memory error syndrome register	1704 0000–1707 FFFF
Memory error address register	1708 0000–170B FFFF
Memory ID register	170C 0000–170F FFFF
53C94 transfer counter low register	1710 0000 (read only)
53C94 transfer count low register	1710 0000 (write only)
53C94 transfer counter high register	1710 0004 (read only)
53C94 transfer count high register	1710 0004 (write only)
53C94 FIFO register	1710 0008
53C94 command register	1710 000C
53C94 status register	1710 0010 (read only)
53C94 select bus ID register	1710 0010 (write only)
53C94 interrupt status register	1710 0014 (read only)
53C94 select timeout register	1710 0014 (write only)
53C94 sequence step register	1710 0018 (read only)
53C94 synchronous transfer period register	1710 0018 (write only)
53C94 FIFO flags register	1710 001C (read only)
53C94 synchronous offset register	1710 001C (write only)
53C94 configuration register	1710 0020
53C94 clock conversion register	1710 0024 (write only)
53C94 test register	1710 0028 (write only)
Reserved	1710 002C–1710 003C
Reserved (copies of SCSI registers)	1710 0040–1713 FFFF
SCSI DMA register	1714 0000 (write only)
Reserved (copies of DMA register)	1714 0004–1717 FFFF
SCSI buffer RAM	1718 0000–1719 FFFF
Reserved (copy of SCSI RAM)	171A 0000–171B FFFF
Reserved SCSI address space	171C 0000–171F FFFF
R3000 LED register	1720 0000–1723 FFFF
Reserved I/O module address space	1724 0000–17FF FFFF
R3000 nonvolatile RAM	1800 0000–1807 FFFF

[1]Accessible only from R3000 processor.

Table B–2 (Cont.): R3000 Physical I/O Addresses

Contents	Address Range
Local Q22-Bus I/O Space	
Reserved I/O module address space	1808 0000–180F FFFF
R3000 UART registers	1810 0000–1810 003C
Reserved (copies of UART registers)	1810 0040–1813 FFFF
Reserved I/O module address space	1814 0000–18FF FFFF
VME option memory and I/O space	1900 0000–1AFF FFFF
FDDI option memory and I/O space	1B00 0000–1CFF FFFF
Reserved I/O module address space	1D00 0000–1FBF FFFF
R3000 ROM	1FC0 0000–1FC3 FFFF
Reserved I/O module address space	1FC4 0000–1FFF FFFF
Reserved	2000 0000–2FFF FFFF

B.4 KN220 Diagnostic Processor Physical Address Space Map (M7638–AA)

Table B–3: KN220 Diagnostic Processor Physical Addresses

Contents	Address Range
Local Memory Space (up to 256 Mbytes)	0000 0000–0FFF FFFF
Local Q22-Bus I/O Space	
Reserved Q22-bus I/O space	2000 0000–2000 0007
Q22-bus floating address space	2000 0008–2000 07FF
User reserved I/O module address space	2000 0800–2000 0FFF
Reserved Q22-bus I/O space	2000 1000–2000 1F3F
Interprocessor communication register	2000 1F40
Reserved Q22-bus I/O space	2000 1F48–2000 1FFF
Reserved I/O module address space	2000 2000–2000 7FFF
SGEC CSR0: vector, IPL, mode	2000 8000
SGEC CSR1: polling demand	2000 8004
SGEC CSR2: reserved register	2000 8008
SGEC CSR3: receive descriptor list	2000 800C
SGEC CSR4: transmit descriptor list	2000 8010
SGEC CSR5: status register	2000 8014
SGEC CSR6: command and mode register	2000 8018
SGEC CSR7: system base register	2000 801C
SGEC CSR8: reserved register	2000 8020
SGEC CSR9: watchdog timers	2000 8024
SGEC CSR10: revision number and missed frame count	2000 8028
SGEC boot message registers	2000 802C–2000 8034
SGEC diagnostic registers	2000 8038–2000 803C
Reserved (copies of SGEC registers)	2000 8040–2001 FFFF
Reserved I/O module address space	2002 0000–2003 FFFF
Two copies of CVAX ROM	2004 0000–2007 FFFF
Q22 system configuration register	2008 0000
Q22 system error register	2008 0004
Q22 master error address register	2008 0008
Q22 slave error address register	2008 000C
Q22-bus map base register	2008 0010
Reserved	2008 0014–2008 3FFF
Interrupt status register	2008 4000
Boot and diagnostic register	2008 4004

Table B–3 (Cont.): KN220 Diagnostic Processor Physical Addresses

Contents	Address Range
Local Q22-Bus I/O Space	
Select processor register	2008 4008
Interval timer register	2008 4010
Reserved I/O module address space	2008 4014–2008 7FFF
Q22-bus map registers	2008 8000–2008 FFFF
Reserved I/O module address space	2009 0000–2009 FFFF
DSSI buffer RAM	2010 0000–2011 FFFF
NI station address ROM	2012 0000–2012 007C
Reserved I/O module address space	2012 0080–2013 FFFF
SSC base address register	2014 0000
SSC configuration register	2014 0010
CDAL bus timeout control register	2014 0020
Diagnostic LED register	2014 0030
Reserved I/O module address space	2014 0034–2014 0068
Time-of-year register	2014 006C
Reserved	2014 0070–2014 007C
CVAX console receiver control/status	2014 0080
CVAX console receiver data buffer	2014 0084
CVAX console transmitter control/status	2014 0088
CVAX console transmitter data buffer	2014 008C
Reserved	2014 0090–2014 00DB
I/O system reset register	2014 00DC
Reserved	2014 00E0
ROM data register	2014 00F0
Bus timeout counter	2014 00F4
Interval timer	2014 00F8
Reserved	2014 00FC–2014 00FF
Timer 0 control register	2014 0100
Timer 0 interval register	2014 0104
Timer 0 next interval register	2014 0108
Timer 0 interrupt vector	2014 010C
Timer 1 control register	2014 0110
Timer 1 interval register	2014 0114
Timer 1 next interval register	2014 0118
Timer 1 interrupt vector	2014 011C
DSSIDB address decode match register	2014 0130
DSSIDB address decode mask register	2014 0134
LIOD address decode match register	2014 0140
LIOD address decode match register	2014 0144
Reserved	2014 0148–2014 033F

Table B–3 (Cont.): KN220 Diagnostic Processor Physical Addresses

Contents	Address Range
Local Q22-Bus I/O Space	
CVAX battery backed-up RAM	2014 0400–2014 07FF
Reserved I/O module address space	2014 0800–2015 FFFF
DSSI diagnostic register 0	2016 0000
DSSI diagnostic register 1	2016 0004
DSSI diagnostic register 2	2016 0008
DSSI control and status register	2016 000C
DSSI ID register	2016 0010
Reserved DSSI register	2016 0014
Reserved DSSI register	2016 0018
DSSI timeout register	2016 001C
Reserved DSSI register	2016 0020
Reserved DSSI register	2016 0024
Reserved DSSI register	2016 0028
Reserved DSSI register	2016 002C
Reserved DSSI register	2016 0030
Reserved DSSI register	2016 0034
DSSI short target list pointer	2016 0038
DSSI long target list pointer	2016 003C
DSSI initiator list pointer	2016 0040
DSSI DSSI control register	2016 0044
DSSI DSSI status register	2016 004C
Reserved DSSI register	2016 004C
Reserved DSSI register	2016 0050
DSSI diagnostic control register	2016 0054
DSSI clock control register	2016 0058
DSSI internal state register 0	2016 005C
DSSI internal state register 1	2016 0060
DSSI internal state register 2	2016 0064
DSSI internal state register 3	2016 0068
Reserved DSSI register	2016 006C
Reserved DSSI register	2016 0070
Reserved DSSI register	2016 0074
Reserved DSSI register	2016 0078
Reserved DSSI register	2016 007C
Reserved I/O module address space	2016 0080–2017 FFFF
Reserved	2018 0000–23FF FFFF
Reserved	2400 0000–26FF FFFF
I/O presence register	2700 0000–2703 FFFF
Memory error syndrome register	2704 0000–2707 FFFF

Table B–3 (Cont.): KN220 Diagnostic Processor Physical Addresses

Contents	Address Range
Local Q22-Bus I/O Space	
Memory error address register	2708 0000–270B FFFF
Memory ID register	270C 0000–270F FFFF
53C94 transfer counter low register	2710 0000 (read only)
53C94 transfer count low register	2710 0000 (write only)
53C94 transfer counter high register	2710 0004 (read only)
53C94 transfer count high register	2710 0004 (write only)
53C94 FIFO register	2710 0008
53C94 command register	2710 000C
53C94 status register	2710 0010 (read only)
53C94 select bus ID register	2710 0010 (write only)
53C94 interrupt status register	2710 0014 (read only)
53C94 select timeout register	2710 0014 (write only)
53C94 sequence step register	2710 0018 (read only)
53C94 synchronous transfer period register	2710 0018 (write only)
53C94 FIFO flags register	2710 001C (read only)
53C94 synchronous offset register	2710 001C (write only)
53C94 configuration register	2710 0020
53C94 clock conversion register	2710 0024 (write only)
53C94 test register	2710 0028 (write only)
Reserved	2710 002C–2710 003C
Reserved (copies of SCSI registers)	2710 0040–2713 FFFF
SCSI DMA register	2714 0000 (write only)
Reserved (copies of DMA register)	2714 0004–2717 FFFF
SCSI buffer RAM	2718 0000–2719 FFFF
Reserved (copy of SCSI RAM)	271A 0000–271B FFFF
Reserved SCSI address space	271C 0000–271F FFFF
R3000 LED register	2720 0000–2723 FFFF
Reserved I/O module address space	2724 0000–27FF FFFF
R3000 nonvolatile RAM	2800 0000–2807 FFFF
Reserved I/O module address space	2808 0000–280F FFFF
R3000 UART registers	2810 0000–2810 003C
Reserved (copies of UART registers)	2810 0040–2813 FFFF
Reserved I/O module address space	2814 0000–28FF FFFF
VME option memory and I/O space	2900 0000–2AFF FFFF
FDDI option memory and I/O space	2B00 0000–2CFF FFFF
Reserved I/O module address space	2D00 0000–2FBF FFFF

Table B–3 (Cont.): KN220 Diagnostic Processor Physical Addresses

Contents	Address Range
Local Q22-Bus I/O Space	
R3000 ROM	2FC0 0000–2FC3 FFFF
Reserved I/O module address space	2FC4 0000–2FFF FFFF
Local Q22-bus memory space	3000 0000–303F FFFF
Reserved	3040 0000–3FFF FFFF

B.5 Diagnostic Processor Registers

Several KN220 internal processor registers (IPRs) are implemented in the SSC chip rather than the CVAX chip. These registers are listed in Table B–4. The R3000 accesses these registers through R3000 memory space.

Table B–4: Diagnostic Processor Registers

Dec	Hex	Register Name	Mnemonic	Type	Location
0	0	Kernel Stack Pointer	KSP	r/w	CVAX
1	1	Executive Stack Pointer	ESP	r/w	CVAX
2	2	Supervisor Stack Pointer	SSP	r/w	CVAX
3	3	User Stack Pointer	USP	r/w	CVAX
4	4	Interrupt Stack Pointer	ISP	r/w	CVAX
5	5	Reserved			CVAX
6	6	Reserved			CVAX
7	7	Reserved			CVAX
8	8	P0 Base Register	P0B	r/w	CVAX
9	9	P0 Length Register	P0LR	r/w	CVAX
10	A	P1 Base Register	P1BR	r/w	CVAX
11	B	P1 Length Register	P1LR	r/w	CVAX
12	C	System Base Register	SBR	r/w	CVAX
13	D	System Length Register	SLR	r/w	CVAX
14	E	Reserved			CVAX
15	F	Reserved			CVAX
16	10	Process Control Block Base	PCBB	r/w	CVAX
17	11	System Control Block Base	SCBB	r/w	CVAX
18	12	Interrupt Priority Level	IPL	r/w	CVAX
19	13	AST Level	ASTLVL	r/w	CVAX
20	14	Software Interrupt Request	SIRR	w	CVAX
21	15	Software Interrupt Summary	SISR	r/w	CVAX

Table B–4 (Cont.): Diagnostic Processor Registers

Dec	Hex	Register Name	Mnemonic	Type	Location
22	16	Reserved			CVAX
23	17	Reserved			CVAX
24	18	Interval Clock Control Status	ICCS	r/w	CVAX
25	19	Next Interval Count	NICR	w	CVAX
26	1A	Interval Count	ICR	r	CVAX
27	1B	Time-of-year Register	TOY	r/w	SSC
28	1C	Console Storage Receiver Status	CSRS	r/w	SSC
29	1D	Console Storage Receiver Data	CSRD	r	SSC
30	1E	Console Storage Transmitter Status	CSTS	r/w	SSC
31	1F	Console Storage Transmitter Data	CSDB	w	SSC
32	20	Console Receiver Control Status	RXCS	r/w	SSC
33	21	Console Receiver Data Buffer	RXDB	r	SSC
34	22	Console Transmitter Control Status	TXCS	r/w	SSC
35	23	Console Transmitter Data Buffer	TXDB	w	SSC
36	24	Translation Buffer Disable	TBDR	r/w	CVAX
37	25	Cache Disable	CADR	r/w	CVAX
38	26	Machine Check Error Summary	MCESR	r/w	CVAX
39	27	Memory System Error	MSER	r/w	CVAX
40	28	Reserved			CVAX
41	29	Reserved			CVAX
42	2A	Console Saved PC	SAVPC	r	CVAX
43	2B	Console Saved PSL	SAVPSL	r	CVAX
44	2C	Reserved			CVAX
45	2D	Reserved			CVAX
46	2E	Reserved			CVAX
47	2F	Reserved			CVAX
55	47	I/O System Reset Register	IORESET	–	CVAX

B.6 Global Q22-Bus Memory Space Map

Table B–5: Q22-Bus Memory Space

Contents	Address Range (Octal)
Q22-bus memory space	0000 0000–1777 7777

Table B–6: Q22-Bus I/O Space with BBS7 Asserted

Contents	Address Range (Octal)
Q22-bus I/O space	1776 0000–1777 7777
Reserved Q22-bus I/O space	1776 0000–1776 0007
Q22-bus floating address space	1776 0010–1776 3777
User-reserved Q22-bus I/O space	1776 4000–1776 7777
Reserved and fixed CSR Q22-bus I/O space	1777 0000–1777 7477
Interprocessor communication register	1777 7500
Reserved Q22-bus I/O space	1777 7502–1777 7777

Appendix C
Configuring the KFQSA

This appendix describes the KFQSA storage adapter and explains how to:

- Configure the KFQSA storage adapter at installation
- Enter console I/O mode
- Run the Configure utility
- Program the EEROM on the KFQSA
- Reprogram the EEROM on the KFQSA
- Change the ISE's allocation class and unit number

C.1 KFQSA Overview

The KN220 CPU module contains the DSSI adapter that interfaces six DSSI devices, both internal and external to the DECsystem 5500. At the same time, the KFQSA adapter module can be added to interface six DSSI devices external to the DECsystem 5500.

The KFQSA module is a storage adapter that allows Q-bus systems that support the KFQSA to communicate with storage peripherals based on the Digital Storage Architecture (DSA), using the Digital Storage System Interconnect (DSSI).

The KFQSA contains the addressing logic required to make a connection between the host and a requested ISE on the DSSI bus. Each ISE has its own controller, which contains the intelligence and logic necessary to control data transfers over the DSSI bus. The KFQSA presents a mass storage control protocol (MSCP) U/Q port for each ISE.

The EEROM on the KFQSA contains a configuration table. After you install the KFQSA, you program the EEROM with the CSR address for each ISE in the system.

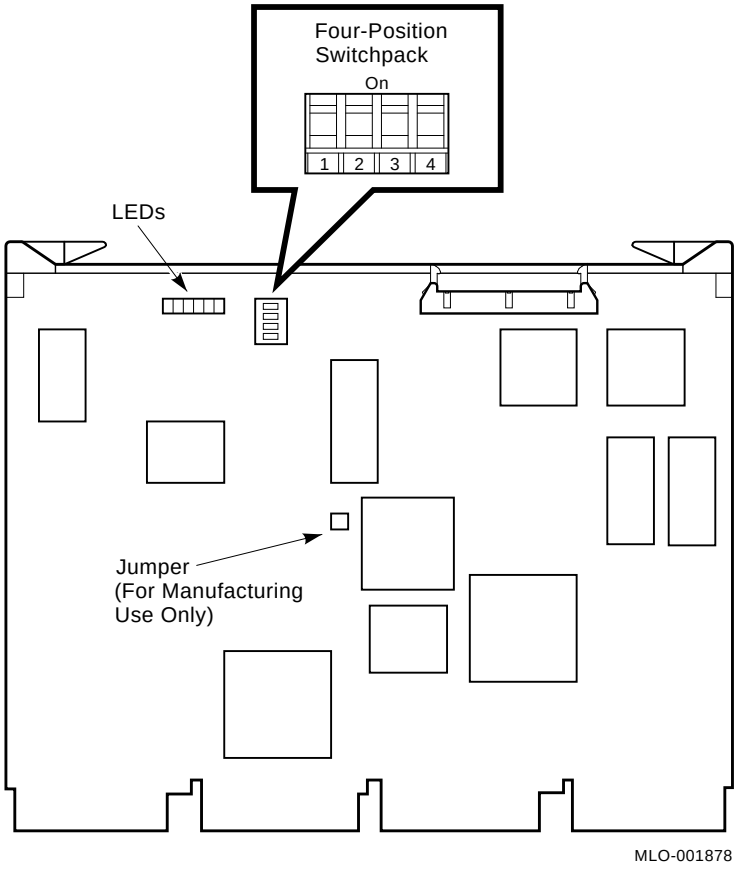
C.2 Configuring the KFQSA at Installation

At installation, configure the KFQSA as follows:

> **CAUTION:** *Static electricity can damage integrated circuits. Use the wrist strap and antistatic mat found in the Antistatic Kit (29–26246) when you work with the internal parts of a computer system.*

1. Check the KFQSA module for the presence of a jumper intended for manufacturing use only. The location of this jumper is shown in Figure C–1. Remove the jumper.
2. Use the four-position DIP switchpack shown in Figure C–1 as follows to set a temporary CSR address that enables you to access the EEROM:
   a. Set switches 1, 2, 3, and 4 to reflect a fixed CSR address to allow the KFQSA to be programmed. Table C–1 shows the switch settings for the KFQSA.
   b. Install the KFQSA adapter module into the backplane according to the procedures in the appropriate enclosure maintenance manual.

Figure C–1: KFQSA Module Layout (M7769)



The positions of the KFQSA four-position switchpack are shown in Table C–1.

Table C–1: KFQSA (M7769) Service Mode Switch Settings

Switches	S/N Mode 1	Fx/F1 2	MSB 3	LSB 4	Fixed Address
First KFQSA	on	off	on	on	0774420
Additional	on	off	on	off	0774424
"	on	off	off	on	0774430
"	on	off	off	off	0774434

S/N = Service mode/Normal operating mode

Fx/Fl = fixed/floating CSR address

C.2.1 Entering Maintenance Mode

After installing the KFQSA, you issue a series of commands to the KN220 system at the maintenance mode prompt (>>>) to program the EEROM on the KFQSA. You may type these commands in either uppercase or lowercase letters. Unless otherwise specified, type each command, then press Return.

Enter maintenance mode as follows:

1. Set the operation switch on the H3602–AC CPU cover panel to the maintenance position (⊕).
2. Set the function switch on the CPU cover panel to the enable position (⊙).
3. Set the on/off power switch to on (1).
4. When the power-up self-tests complete, the console prompt appears, as shown in Example C–1.

Example C–1: Entering Console Mode Display

```
KN220-A Vn.n

Performing normal system tests.
83..82..81..80..79..78..77..76..75..74..73..72..71..70..69..68..67..
66..65..64..63..62..61..60..59..58..57..56..55..54..53..52..51..50..
49..48..47..46..45..44..43..42..41..40..39..38..37..36..35..34..33..
32..31..30..29..28..27..26..25..24..23..22..21..20..19..18..17..16..
15..14..13..12..11..10..09..08..07..06..05..04..03..
Tests completed.
>>>
```

C.2.2 Displaying Current Addresses

Type SHOW QBUS to display the current Q22-bus addresses (Example C–2). The KFQSA adapter appears in service mode as KFQSA #0.

Example C–2: SHOW QBUS Display

```
>>> SHOW QBUS
Scan of Qbus I/O Space
-20001910 (774420) = 0000 (000) KFQSA #0
-20001912 (774422) = 0AA0
-20001920 (774440) = FF08 (120) DELQA/DEQNA/DESQA
-20001922 (774442) = FF00
-20001924 (774444) = FF2B
-20001926 (774446) = FF09
-20001928 (774450) = FFA3
-2000192A (774452) = FF96
-2000192C (774454) = 8000
-2000192E (774456) = 1030
-20001940 (774500) = 0000 (260) TQK50/TQK70/TU81E/RV20/KFQSA-TAPE
-20001942 (774502) = 0BC0
-20001F40 (777500) = 0020 (004) IPCR

Scan of Qbus Memory Space
>>>
```

C.2.3 Running the Configure Utility

Now that you have physically reconfigured the system by installing the KFQSA storage adapter, you must run the Configure utility to find the correct address for each device and module in the system. The Configure utility uses Q-bus/Unibus fixed and floating address space rules.

Run the Configure utility as follows. Refer to Example C–3.

1. At the maintenance mode prompt, type `CONFIGURE`, then type `HELP` at the `Device,Number?` prompt for a list of devices that can be configured.

   NOTE: *Some of the devices listed in the HELP display are not supported by the KN220–A CPU.*

2. For each device in the system, type the device name at the `Device,Number?` prompt. If you have more than one of the same type, type a comma followed by the total number of that device. In Example C–3, the system contains one KFQSA with six ISEs.

   Be sure you list *all* the devices: those already installed and those you plan to install.

3. Type `EXIT`. The Configure utility displays an address and vector assignment for each device. Example C–3 shows the address and vector assignments and the device input. Write down the addresses for the KFQSA devices.

4. For all modules except the KFQSA, verify that the CSR addresses are set correctly by comparing the addresses listed in the SHOW QBUS command with those listed in the Configure utility display. If necessary, remove modules from the backplane and reset switches or jumpers to the addresses in your Configure display, using module removal and replacement procedures in *BA430/BA440 Enclosure Maintenance*.

Example C–3: Configure Display

```
>>> CONFIGURE

Enter device configuration, HELP, or EXIT
Device,Number? help
Devices:
 LPV11      KXJ11        DLV11J       DZQ11      DZV11       DFA01
 RLV12      TSV05        RXV21        DRV11W     DRV11B      DPV11
 DMV11      DELQA        DEQNA        DESQA      RQDX3       KDA50
 RRD50      RQC25        KFQSA-DISK   TQK50      TQK70       TU81E
 RV20       KFQSA-TAPE   KMV11        IEQ11      DHQ11       DHV11
 CXA16      CXB16        CXY08        VCB01      QVSS        LNV11
 LNV21      QPSS         DSV11        ADV11C     AAV11C      AXV11C
 KWV11C     ADV11D       AAV11D       VCB02      QDSS        DRV11J
 DRQ3B      VSV21        IBQ01        IDV11A     IDV11B      IDV11C
 IDV11D     IAV11A       IAV11B       MIRA       ADQ32       DTC04
 DESNA      IGQ11        DIV32        KIV32      DTCN5       DTC05
 KWV32      KZQSA
Numbers:
 1 to 255, default is 1
Device,Number? KFQSA-DISK, 6
Device,Number? DESQA
Device,Number? TQK70
Device,Number? EXIT

Address/Vector Assignments
-774440/120 DESQA
-772150/154 KFQSA-DISK  !NODE 0
-760334/300 KFQSA-DISK  !NODE 1
-760340/304 KFQSA-DISK  !NODE 2
-760344/310 KFQSA-DISK  !NODE 3
-760350/314 KFQSA-DISK  !NODE 4
-760354/320 KFQSA-DISK  !NODE 5
-774500/260 TQK70
>>>
```

NOTE: *KN220 does not support all devices in this list. See ULTRIX Installation Guide for complete information.*

C.3 Programming the KFQSA

Program the configuration table in the EEROM of the KFQSA to include all ISEs on the DSSI bus, as follows. Refer to Examples C–4 through C–6.

1. Determine the DSSI node plug address for each ISE you are configuring. Start with node 0, then continue up through node 5. In Example C–3, nodes 0, 1, 2, 3, 4, and 5 are used; node 7 is saved for the KFQSA adapter.
2. At the console prompt on each system, type `SET HOST/UQSSP/MAINT /SERV 0` to program the KFQSA.
3. Type `HELP` to display a list of supported commands.
4. Program the KFQSA to include each DSSI device in the system:
   a. For each ISE, type `SET`, followed by the node number, the CSR address (from the list of addresses you obtained from the Configure utility), and the model number (ISEs are model 21).
   b. Type `SHOW` to display the configuration table you just programmed.
   c. Check the display to make sure the addresses are correct.
   d. Type `EXIT` to save the configuration table or `QUIT` to leave the table unchanged.

Example C–4: Display for Programming the First KFQSA

```
>>> SET HOST/UQSSP/MAINT/SERV 0      !0 refers to first KFQSA
            !in the system.

UQSSP Controller (772150)

Enter SET, CLEAR, SHOW, HELP, EXIT, or QUIT
```

Example C–4 Cont'd on next page

Example C–4 (Cont.): Display for Programming the First KFQSA

```
Node    CSR Address     Model
 7      ------- KFQSA -------
? HELP
Commands:
     SET <node> /KFQSA                !Sets KFQSA DSSI node
                                      !number
     SET <node> <CSR_address> <model> !Enables a DSSI device
     CLEAR <node>                     !Disables a DSSI device
     SHOW                             !Displays current
                                      !configuration
     HELP                             !Displays this display
     EXIT                             !Saves the KFQSA program
     QUIT                             !Does not save the KFQSA
                                      !program
Parameters:
     <node>                           !0 through 7
     <CSR_address>                    !760010 to 777774
     <model>                          !21 (disk) or 22 (tape)
? SET 0 772150 21
? SET 1 760334 21
? SET 2 760340 21
? SET 3 760344 21
? SET 4 760350 21
? SET 5 760354 21
? SHOW
Node    CSR Address    Model
 0        772150 21
 1        760334        21
 2        760340        21
 3        760344        21
 4        760350        21
 5        760354        21
 7     ------ KFQSA -------
? exit
Programming the KFQSA...              !Note from the system that
                                      !the KFQSA is
                                      !being programmed.
```

5. Turn the system power off by setting the on/off switch to off (0).
6. Remove the KFQSA from the backplane.
7. Confirm that the unit ID plugs on the enclosure's operator control panel (OCP) match the node IDs you just programmed.
8. On the KFQSA, set switch 1 on the four-position switchpack to on (1). (Figure C–1 shows the location and position of the switchpack.) This action sets the KFQSA to the normal operating mode; switches 2, 3, and 4 are ignored since switch 1 is set to off, and the DSSI addresses are read from the EEROM.
9. Reinstall the KFQSA in the backplane.
10. Power on the system by setting the on/off switch to on (1). Wait for the self-tests to complete.
11. At the maintenance mode prompt, type `SHOW QBUS` to verify that all addresses are present and correct, as shown in Example C–5.
12. Type `SHOW DEVICE` to verify that all ISEs are displayed correctly, as shown in Example C–6.

Example C–5: SHOW QBUS Display

```
>>> show qbus
Scan of Qbus I/O Space
-200000DC (760334) = 0000 (300) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000DE (760336) = 0AA0
-200000E0 (760340) = 0000 (304) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000E2 (760342) = 0AA0
-200000E4 (760344) = 0000 (310) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000E6 (760346) = 0AA0
-200000E8 (760350) = 0000 (314) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000EA (760352) = 0AA0
-200000EC (760354) = 0000 (320) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000EE (760356) = 0AA0
-20001468 (772150) = 0000 (154) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-2000146A (772152) = 0AA0
-20001920 (774440) = FF08 (120) DESQA
-20001922 (774442) = FF00
-20001924 (774444) = FF2B
-20001926 (774446) = FF09
-20001928 (774450) = FFA3
-2000192A (774452) = FF96
-2000192C (774454) = 0050
-2000192E (774456) = 1030
-20001940 (774500) = 0000 (260) TQK70/TU81E/RV20/KFQSA-TAPE
-20001942 (774502) = 0BC0
-20001F40 (775500) = 0020 (004) IPCR

Scan of Qbus Memory Space
>>>
```

Example C–6: SHOW DEVICE Display

```
>>> SHOW DEVICE

DSSI Node 0 (772150)
-rf(0,0,*) (RF71)

DSSI Node 1 (760334)
-rf(1,1,*) (RF71)

DSSI Node 2 (760340)
-rf(2,2,*) (RF71)

DSSI Node 3 (760344)
-rf(3,3,*) (RF71)

DSSI Node 4 (760350)
-rf(4,4,*) (RF71)

DSSI Node 5 (760354)
-rf(5,5,*) (RF71)

DSSI Node 7 (*)

SCSI Node 0
-rz(0,0,*) (RZ56)

SCSI Node 7 (*)

UQSSP Tape Controller 0 (774500)
-tm(0,0) (TK70) -MUA0

Ethernet Adapter 0
-tftp() -mop() -EZA0 (08-00-2B-0C-C4-75)

VME Interface Board - Not Installed
>>>
```

C.4 Reprogramming the KFQSA

When you add a new DSSI device to a system with at least one RF-series ISE that has been programmed correctly, you must reprogram each KFQSA on the DSSI bus to include the new device(s) as follows:

1. Enter maintenance mode, using the procedure in Section C.2.1.
2. At the maintenance mode prompt, type `SHOW DEVICE` for a display of all devices currently in the system. The display includes tape drives and the Ethernet adapter, as shown in Section C.3, Example C–6.

   This display lists the Q-bus address and port name of the device.
3. Type `SHOW QBUS` for a display of the eight-digit Q-bus address (hex) for each device, as shown in Section C.3, Example C–5.
4. Find the eight-digit Q-bus address for an ISE attached to the KFQSA that you are reprogramming. Use the SET HOST command to enter the KFQSA through an existing port and edit the configuration table as follows. Refer to Example C–7.

   a. Type `SET HOST/UQSSP/MAINT` followed by the Q-bus address.

   b. Use the SET and CLEAR commands to reconfigure the KFQSA, as shown in Example C–7.

   c. Type `SHOW` to display the new KFQSA configuration table setting.

   d. Type `EXIT` to save the configuration table or `QUIT` to cancel the reprogramming.

Example C–7: Reprogramming the KFQSA Display

```
>>> SET HOST/UQSSP/MAINT 20001468
UQSSP Controller (772150)

Node    CSR Address    Model
 0        772150        21
 1        760334        21
 2        760340        21
 3        760344        21
 4        760350        21
 5        760354        21
 7      ----- KFQSA ------
? CLEAR 5
? SHOW
Node    CSR Address    Model
 0        772150        21
 1        760334        21
 2        760340        21
 3        760344        21
 4        760350        21
 7      ----- KFQSA ------
? SET 5 760354 21
? SHOW
Node    CSR Address    Model
 0        772150        21
 1        760334        21
 2        760340        21
 3        760344        21
 4        760350        21
 5        760354        21
 7      ----- KFQSA ------
? EXIT
Programming the KFQSA...      !Note from the system that the
                              !KFQSA is being programmed.
```

C.5 Changing the ISE Unit Number

This section describes how to change the ISE unit number. For most configurations, you will not need to change the default unit numbers.

Change the allocation class and unit number parameters, using the console-based DUP driver utility, as follows. Refer to Example C–8.

1. Enter maintenance mode, using the procedure in Section C.2.1.
2. At the maintenance mode prompt, type `SET HOST/DUP/UQSSP/DISK 0 PARAMS` (0 through 6 for the ISE to which you want to connect) to start the DUP server.
3. Type `SHOW UNITNUM` to check the unit number.
4. To change the ISE's unit number from the default value, type `SET UNITNUM n` (where `n` is the new unit number). For example, type `SET UNITNUM 20` to change the unit number from 0 to 20.
5. Type `SET FORCEUNI 0` to set the forceunit flag to zero in order to use a nondefault value. If you do not change the FORCEUNI parameter, the drive unit number defaults to the number of the corresponding DSSI plug on the operator control panel (OCP).
6. Type `SHOW UNITNUM` to show the new unit number.
7. Type `SHOW FORCEUNI` to show the new forceunit flag values.
8. Type `WRITE`, then type `Y` to save the new values into the EEROM or `N` to cancel the reprogramming.
9. Type `SHOW DEVICE` to make sure you have programmed the first ISE to have a unit number of 20. When you boot the operating system, the display shows the new unit number.

Example C–8: Display for Changing Unit Number

```
>>> SET HOST/DUP/UQSSP/DISK 0 PARAMS
Starting DUP server...

UQSSP Disk Controller 0 (772150)
Copyright (c) 1988 Digital Equipment Corporation

PARAMS> SHOW UNITNUM

Parameter    Current     Default    Type   Radix
---------  ----------  ----------  ------  ------
UNITNUM            0           0   Word    Dec   B

PARAMS> SET UNITNUM 20

PARAMS> SET FORCEUNI 0

PARAMS> SHOW UNITNUM

Parameter    Current     Default    Type   Radix
---------  ----------  ----------  ------  ------
UNITNUM           20           0   Word    Dec   U

PARAMS> SHOW FORCEUNI

Parameter    Current     Default    Type   Radix
---------  ----------  ----------  ------  ------
FORCEUNI           0           1  Boolean  0/1   U

PARAMS> WRITE

Stopping DUP server...

>>> SHOW DEVICE
DSSI Node 0 (772150)
-rf(0,20,*) (RF71)

DSSI Node 1 (760334)
-rf(1,1,*) (RF71)

DSSI Node 2 (760340)
-rf(2,2,*) (RF71)

DSSI Node 3 (760344)
-rf(3,3,*) (RF71)

DSSI Node 4 (760350)
-rf(4,4,*) (RF71)

DSSI Node 5 (760354)
-rf(5,5,*) (RF71)

DSSI Node 7 (*)
```

Example C–8 Cont'd on next page

Example C–8 (Cont.): Display for Changing Unit Number

```
SCSI Node 0
-rz(0,0,*)  (RZ56)

SCSI Node 7 (*)

UQSSP Tape Controller 0 (774500)
-tm(0,0) (TK70) -MUA0

Ethernet Adapter 0 (774440)
-mop() -EZA0 (08-00-2B-0C-C4-75)

VME Interface Board - Not Installed
>>>
```

Appendix D
Prestoserve Software on the DECsystem 5500

This appendix contains information about using Prestoserve on the DECsystem 5500, especially in regard to firmware support.

D.1 Why Data Recovery Is Necessary

Prestoserve is a software facility for disk acceleration. It uses non-volatile storage on the CPU board to cache data writes to disk. In the event of an abnormal system shutdown, data intended for disk may be left in the NVRAM cache. It is necessary to assure that the data intended for disk is ultimately written to the disk to maintain disk integrity. See the *ULTRIX Guide to Prestoserve* for a more detailed description of Prestoserve.

Data in the cache is automatically recovered and moved to the appropriate disks in the following cases:

- If you follow the normal ULTRIX shutdown procedures for the DECstation 5500
- If you unmount a removable device that uses Prestoserve software while running ULTRIX

If your system shuts down abnormally because of a power failure, hardware failure, or software failure, data intended for disk may remain in the cache after the shutdown has completed. A cache containing data is referred to as dirty. The data intended for disk will be written to the disk automatically during reboot (if the system is able to boot). This section describes procedures for recovering data both in the case when reboot is possible, and in the case when reboot is not possible.

D.2 Using the dc Commands

Commands executed from the Maintenance mode prompt (>>>) allow you to try to recover any data in the cache and avoid corrupting your disks. The commands allow you to:

- Determine if the cache contains data

- Save cache data to tape
- Restore data from tape
- Zero (clear) the data in the cache

The commands that access tape drives depend on the availability of a functioning tape device. To determine the tape devices that are available on your system, enter the show device command at the Maintenance mode prompt. For example:

```
>>> show device
>>>show dev
DSSI Node 0 (R7YRMS)
 -rf(0,0,*)  (RF71)

DSSI Node 7 (*)

SCSI Node 0
 -tz(0,0,*) (TLZ04) -DIA0

SCSI Node 1
 -rz(0,1,*) (RZ56 )

SCSI Node 2
 -rr(0,2,*) (RRD40) -DIA2

SCSI Node 4
 -tz(0,4,*) (.....) -DIA4

SCSI Node 7 (*)

UQSSP Tape Controller 0 (774500)
 -tm(0,0)  (TK70) -MUA0

Ethernet Adapter
 -mop() -EZA0 (08-00-2B-12-81-22)

Ethernet Adapter
 -XQA0 (08-00-2B-08-CB-5C)

VME Interface Board - Not Installed
>>>
```

The available devices are displayed on the console terminal. Note the unit number of the tape device that you want to use. Use the Maintenance mode boot path. For example, for UQSSP tape controller 0, the Maintenance mode boot path is shown at the end of the line containing the ULTRIX boot path, and it is MUA0.

D.2.1 Determining If a Cache Contains Data

To determine if the cache contains data, use the dc command with no qualifier, executed at the Maintenance mode >>> prompt. For example:

```
>>> dc
```

At the dc command, if the CPU board's cache contains data the following message is displayed on the console terminal:

```
Disk Cache - Dirty
```

At the dc command, if the CPU board's cache does not contain any data the following message is displayed on the console terminal:

```
Disk Cache - Clean
```

If there is no data in the cache (cache is clean), you can follow normal procedures for rebooting or troubleshooting.

NOTE: *You should always run the dc command before replacing or using any DECsystem 5500 CPU board.*

D.2.2 Saving the Cache Data to Tape with the dc/save Command

If a problem exists with the CPU board, and there is data in the Prestoserve cache, save the data to tape before you replace the board. You can save data by using the dc/save command, and specifying the tape device. The dc/save command checks if the cache is clean or dirty and displays the status. If the cache is clean, no additional action is taken. After the data has been saved on tape, the dc/save command will attempt to clear the cache. Prior to this action it will query. The command is executed at the Maintenance mode (>>>) prompt. If the cache is dirty, the system responds to the command as follows:

```
>>> dc/save mua0
Disk Cache - Dirty
Do you want to continue (y/n)? y

-MUA0
Zero Disk Cache (y/n)? y
>>>
```

D.2.3 Restore Data From Tape with the dc/restore Command

After you correct or replace the CPU board, you must move the data from the specified tape device to the new CPU board's NVRAM cache, using the dc/restore command. The command is executed at the Maintenance mode (>>>) prompt. The command checks that the new cache is clear. If the cache is dirty, you will be prompted to continue, to make sure that you want to overwrite the cache contents, as in the following example.

```
>>> dc/restore mua0
Disk Cache - Dirty
Do you want to continue (y/n)? y

-MUA0
>>>
```

If the cache is clean, as in the following example, the contents of the tape (in this example, mua0) are loaded into the cache.

```
>>> dc/restore mua0
Disk Cache - Clean

-MUA0
>>>
```

When the system reboots, the contents of the cache are moved to the appropriate disks.

D.2.4 Clearing the Cache with the dc/zero Command

If you want to clear the contents of the cache because the data is not wanted or because the data cannot be saved to tape with the dc/save command, use the dc/zero command. The command is executed at the Maintenance mode (>>>) prompt. For example:

```
>>> dc/zero
Do you want to continue (y/n)? y
```

The command prompts you to confirm that you want to clear the contents of the cache and then fills the cache with zeroes. You can use this command as a security measure to ensure that the cache is cleared.

D.3 Recover from Abnormal System Shutdowns

The following sections describe in detail how to use the dc commands in data recovery. Data recovery applies to systems that were abnormally shut down.

In some circumstances, you may not be able to recover the data.

D.3.1 Recovering Data From a System That Can Reboot

If the system was shut down uncleanly but can still reboot, the contents of the cache are automatically moved to the appropriate disks during the boot process. If a particular disk that is needed is unavailable, you will be given the opportunity to discard the cache data for that disk; then you must recreate that disk by using backups or by re-entering the data.

D.3.2 Recovering Data From a System That Cannot Reboot

If the system was shut down uncleanly and cannot reboot, the following message is displayed on the console terminal after the execution of the system self-tests:

```
Normal operation not possible.
>>>
```

The Maintenance mode prompt (>>>) is then displayed.

NOTE: *Make sure that the CPU cover panel Operation switch is set to Maintenance mode before you attempt to recover data.*

If you are unable to enter any commands, you will be unable to recover the data in the cache. Also, if the cache fails, you will be unable to recover the data.

To recover any data that may be in the cache, you must determine the contents of the CPU board's cache by using the dc command.

If the display on the console terminal indicates that the cache is clean, no file systems were being accelerated when the system was shut down; therefore you will not lose data or corrupt your disks when the system reboots. You should continue troubleshooting the system.

If the display on the console terminal indicates that the cache is dirty (that is, it contains data), you should attempt to recover the data by the following steps. Note that you should not remove any disks from the system, and you should not change the system configuration until the data has been recovered.

You can use the maint command at the console prompt (>>) to set the system to Maintenance mode. For example:

```
>> maint
>>>
```

You should run the full system self-tests to determine the hardware problem. The self-test command is t 0, or press the Reset button.

Note the results of the system self-tests and refer to the appropriate section to correct the hardware error.

D.3.2.1 Bad I/O Board

If the system self-tests indicate a bad I/O board (Table 4–7), follow these steps:

1. Replace the I/O board.
2. Run the system self-tests again and note the results.
3. If a problem still exists, perform the appropriate corrective action. If the problem is corrected, attempt to reboot.

   If you are able to reboot, a message indicating the CPU board and I/O board mismatch is displayed on the console terminal. At the prompt, you should confirm that you want to continue with the reboot. Then, the contents of the cache are moved to the appropriate disks if available.

D.3.2.2 Bad CPU Board

If the system self-tests indicate a bad CPU board (Table 4–7), follow the steps in this section.

Before you replace the CPU board, you must make sure that the bootmode environment variable is not set to autoboot. You must also make sure that the CPU cover panel is switched to Maintenance mode.

To install a new CPU board and recover data in the cache, follow these steps:

1. Determine what tape devices are available on your system by typing the show device command at the Maintenance mode prompt. For example:

   ```
   >>> show device
   ```

   The available devices are displayed on the console terminal, as shown in Section D.2. Note the unit number of the tape device that you want to use.

2. Move the contents of the bad CPU board's cache to tape by using the dc/save command. For example:

   ```
   >>> dc/save [tape device]
   ```

3. Use the dc/zero command to clear the contents of the cache. For example:

   ```
   >>> dc/zero
   Do you want to continue (y/n)? y
   ```

4. If you want to install a new CPU board, place the battery jumper on the new CPU board to the two top jumper pins (position 1, On). To prevent battery drain, new CPU boards are shipped with the battery jumper on the bottom two jumper pins (position 0, Off).

   Clear the old board's cache either by setting the battery jumper to position 0 for two seconds or by using the dc/zero command.

   NOTE: *Before installing a CPU board, reposition the jumper to the On position.*

5. Use the dc/restore command to move the contents of the tape to the newly installed CPU board's cache. For example:

```
>>> dc/restore [tape device]
Do you want to continue (y/n)? y
```

6. Run the system self-tests again and note the results.

7. If a problem still exists, perform the appropriate corrective action; then attempt to reboot.

   Once the reboot is successful, the data in the cache is moved to the appropriate disks.

D.3.2.3 Bad Boot Disk

If you have a bad boot disk, you should attempt to reboot the system and move the contents of the cache to the appropriate disks. However, any data destined for the boot disk will be lost.

To reboot the system, follow these steps:

1. Boot the memory-based operating system from the installation tape.

2. Make a system special file for any file system by using the MAKEDEV command. Refer to MAKEDEV(8) in the *ULTRIX Reference Pages* for more information.

3. Use the fsck command to check the file system. Refer to fsck(8) in the *ULTRIX Reference Pages* for more information.

4. During the file system checking, the system moves the data in the cache to the appropriate disks. If a disk is unavailable, the system prompts you to confirm that you want to discard the data for those disks. Once the data is discarded the cache is empty.

5. Reinstall the ULTRIX operating system on a viable boot disk. Refer to the appropriate installation guide for more information.

D.3.2.4 Other Hardware Problems

Once you correct a hardware problem, you should be able to reboot, and the data in the cache will be moved to the appropriate disks.

D.3.3 Power-up Screen and Test 79 with NVRAM Battery Off

If the battery jumper on the CPU module is installed in the ON position (1), there is no message. If the battery jumper is installed in the OFF position (0), there is an error message in the first line of the error display.

The following screen messages are displayed after the initial power-up, and then after running test 79.

This is the display when the cache is dirty.

```
>>>

KN220-A VX.x
Performing normal system tests.
83..82..81..80..79..78..77..76..75..74..73..72..71..70..69..68..67..
66..65..

?79 1 07 FF 0000 0000
64..63..62..61..60..59..58..57..56..55..54..53..52..51..50..
49..48..47..46..45..44..43..42..41..40..39..38..37..36..35..34..33..
32..31..30..29..28..27..26..25..24..23..22..21..20..19..18..17..16..
15..14..13..12..11..10..09..08..07..06..05..04..03..
Tests completed.
>>>
>>>T 79

?79 1 07 FF 0000 0001

>>>
```

This is the display when the battery jumper is in the OFF (0) position.

```
KN220-A VX.x
Performing normal system tests.
83..82..81..80..79..78..77..76..75..74..73..72..71..70..69..68..67..
66..65..

?79 1 0A FF 0000 0000
64..63..62..61..60..59..58..57..56..55..54..53..52..51..50..
49..48..47..46..45..44..43..42..41..40..39..38..37..36..35..34..33..
32..31..30..29..28..27..26..25..24..23..22..21..20..19..18..17..16..
15..14..13..12..11..10..09..08..07..06..05..04..03..
Tests completed.
>>>
>>>T 79
```

```
?79 1 0A FF 0000 0001
>>>
```

Appendix E
Field-Replaceable Units (FRUs)

This appendix lists the major field-replaceable units (FRUs) for the KN220-based system (DECsystem 5500).

FRU Description	Part Number
KN220 Base System	
H3602–AC bulkhead assembly	70–25775–03
H3605 bulkhead assembly	70–27464–01
Interface (ISE terminal power) module	M9715–AA
KN220 I/O module	M7638–AA
KN220 CPU module	M7637–AA
MS220–AA memory module, 32 Mbyte	M7639–AA
NVRAM battery	12–33140–01

FRU Description	Part Number
BA430 Enclosure	
Battery pack	12–19245–01
Blank labels	36–26883–01
Backplane	54–20181–01
Bulkhead assy	70–28083–01
Bulkhead cover, single	70–23981–01
Bulkhead cover, dual	70–23982–02
Control assembly	70–27044–01
Door assembly, top	70–27047–01
Door assembly, bot	70–27048–01
EOS clip	12–26922–01
Fan, 6"	12–31500–01
FCC clip/handles	12–26340–01
Indicator panel	70–27044–02
Interface module, terminal power	M9715–AA
Key, plastic	12–17119–01
One quarter-turn fastener/handle	12–26948–01
Operator Control Panel	70–25752–01
Power supply assy (120/240)	H7874–00
Side gap filler panel (2)	70–24505–01
Terminator, bus	12–29258–01
Wheel, base	74–34068–01
Wheel, shaft	74–34069–01
Wheel, pin	90–09385–00

FRU Description	Part Number
Cables, Base System	
Backplane cable to operator control panel (OCP)	17–01964–01
CPU and I/O module interconnect cable to H3605	17–02665–01
Data cable, with MMJ	BC16E–43
50-cond flat cable	17–01836–01
H3602–AC cable	17–02666–01
H3605 cable	17–02665–01
KN220 CPU module cable to KN220 I/O module	17–02700–01
Memory daisy chain cable (1 memory board)	17–02700–01
Memory daisy chain cable (2 memory boards)	17–02700–02
Memory daisy chain cable (3 memory boards)	17–02700–03
Memory daisy chain cable (4 memory boards)	17–02700–04
Power cord, 120 V (USA)	17–00083–43
Power control cable, BA400-BA400	17–02638–01
Power control cable, BA400-BA200	17–02637–01
Digital Storage System Interconnect (DSSI)	
Cable, external	17–02152–03
Cable, round	17–02059–01
Cable, conns, bulkhead to backplane	70–27458–01
Daisy-chain cable, flat	17–01836–01
Operator control panel (OCP)	70–25752–01
Port protector	12–33902–01
Terminator, bus	12–29258–01
Unit ID plugs	12–28766–19
KFQSA module	M7769–00
RF30/71 drive bracket	70–36498–01
RF31 integrated storage element	70–
RF71 drive module	54–18316–01
RF71 head disk assembly (HDA)	70–23557–01
RF71 integrated storage element (ISE)	RF71–EA
RF71 shock mount, bottom	70–25452–03
RF71 shock assembly, top	70–25452–04
RF-series lens, encoder set	12–28766–19

FRU Description	Part Number
Small Computer Storage Interface (SCSI) ISEs, Modules, and Cables	
SCSI cable, external	17–02659–01
Cable, conns, bulkhead to backplane	70–27459–01
Port cover	12–33377–01
Unit ID plugs	12–28766–28
Q22-Bus Devices	
CXA16–M module	M3118–YA
CXB16–M module	M3118–YB
CXY08–M module	M3119–YA
DESQA–SA module	M3127–PA
Load module	M9060–YA
LPV11–SA module	M8086–PA
Q-bus expansion modules	M9405–PA/M9405–PA
Q-bus expansion cable	17–02048–01
Loopback Connectors	
CXY08 loopback	12–26964–01
DEQNA Ethernet loopback	12–22196–02
H3103–00 (MMJ) loopback	12–25083–01
H3197–00 CXY08 loopback	12–15336–07

Appendix F
Related Documentation

The following documents contain information relating to the KN220 CPU system:

Document Title	Order Number
Modules	
CXA16 Technical Manual	EK–CAB16–TM
CXY08 Technical Manual	EK–CXY08–TM
DESQA Technical Manual	EK–DESQA–TM
DSV11–S Communications Option User Guide	EK–DSV11–UG
DSV11 Communications Option Technical Description	EK–DSV11–TD
KDA50–Q CPU Module User's Guide	EK–KDA5Q–UG
KFQSA Installation Guide	EK–KFQSA–IN
Disk and Tape Drives	
RF31 Installation Guide	EK–RF31–IN
RF71 Disk Drive User's Guide	EK–RF71D–UG
RZ56 Manual	TBD
RZ57 Manual	TBD
RRD40 Owner's Manual	EK-RRD40-OM
TLZ04 Cassette Tape Drive Owner's Manual	EK-TLZ04-OM-001
Enclosures	
BA430/BA440 Enclosure Maintenance	EK–348AB–MG
Microsystems Options	EK–192AA–MG
Microsystems Site Preparation Guide	EK–067AB–PG

Document Title	Order Number
Diagnostics	
MDM User's Guide	AA-FM7AB-DN
MicroVAX Diagnostic Monitor Ethernet Server User's Guide	AA–FNTAC–DN
MicroVAX Diagnostic Monitor Reference Card	AV–FMXAA–DN
Expanders	
B400X Expander Installation	EK–400AA–MG
R400X Expander Installation and Maintenance	EK–349AA–MG
Networks	
Ethernet Transceiver Tester User's Manual	EK–ETHTT–UG

Index

! (comment command), 3–82
9E utility, 4–12
  examples, 4–13

A

Acceptance testing, 4–34
Address assignments, B–3 to B–17

B

BA430 enclosure
  termination with internal drive, 2–11
  termination with tabletop drive, 2–11
Boot and diagnostic facility, on KN220 CPU, 1–9
BOOT command, in maintenance mode, 3–47
boot command, in normal mode, 3–21
Bootstrap
  of MDM, description of, 3–11
  of MDM, supported boot devices, 3–13
  of MDM, supported boot flags, 3–13
  of ULTRIX–32, description of, 3–9
  of ULTRIX–32, procedures for, 3–10
  of ULTRIX–32, supported devices, 3–10
  support, for KN220 systems, 3–8
Bus length (DSSI), 2–10

C

53C94 chip, on KN220 I/O module, 1–12
Cables available, SCSI, 2–14
Cabling
  DSSI, 2–9
  ISE, 2–9
Cache data, clearing, D–2
Cache memory, on KN220 CPU, 1–8
? command, in normal mode, 3–31
Comment command (!), 3–82
Configuration
  and module order, 2–1
  DSSI, 2–4
  rules, 2–2
CONFIGURE command, 2–3, 3–48
Console commands
  address space control qualifiers, in maintenance mode, 3–43
  address specifiers, in maintenance mode, 3–39
  binary load and unload (X), 3–80
  BOOT, in maintenance mode, 3–47
  boot, in normal mode, 3–21
  ! (comment), 3–82
  CONFIGURE, in maintenance mode, 3–48
  CONTINUE, in maintenance mode, 3–50
  continue, in normal mode, 3–23
  d (deposit), in normal mode, 3–24
  data control qualifiers, 3–42
  DC, in maintenance mode, 3–51
  DEPOSIT, in maintenance mode, 3–53
  dump, in normal mode, 3–25

Console commands (Cont.)
e (examine), in normal mode, 3–27
EXAMINE, 3–54
EXIT, 3–56
fill, in normal mode, 3–28
FIND, 3–57
go, in normal mode, 3–29
HALT, 3–58
HELP, 3–59
help, in normal mode, 3–30
init, in normal mode, 3–32
INITIALIZE, 3–61
?, in normal mode, 3–31
keywords, 3–44
list of, 3–44
MOVE, 3–63
NEXT, 3–65
passwd, 1–18
printenv, in normal mode, 3–33
qualifier and argument conventions, in maintenance mode, 3–39
qualifiers, 3–42
REPEAT, 3–67
SEARCH, 3–68
SET, 3–70
setenv, in normal mode, 3–34
SHOW, 3–73
START, 3–77
symbolic addresses, 3–40
syntax, in maintenance mode, 3–39
TEST, 3–78
test, in normal mode, 3–35
UNJAM, 3–79
unpriv, 1–18
unsetenv, in normal mode, 3–36
X (binary load and unload), 3–80
x, in normal mode, 3–37
Console displays
and FRUs, 4–24
Console displays, ROM-based diagnostics, 4–17
Console error messages, 4–31
Console error messages (Cont.)
list of, 4–32
sample of, 4–19
Console port, testing, 4–49
Console security, 1–16
privileged, 1–17
unprivileged, 1–16
Console serial lines on KN220 CPU, 1–8
CONTINUE command, 3–50
continue command, in normal mode, 3–23
CQBIC chip, on KN220 I/O, 1–10
CVAX
console displays, 4–17
halt entry and dispatch code, 3–3
on KN220 CPU, 1–10
CVAX ROM-based diagnostics
parameters for, 4–4

D

d (deposit) command, in normal mode, 3–24
DC command, 3–51
DEPOSIT command, 3–53
Diagnostic executive, 4–3
error field, 4–21
Diagnostics
CVAX, description of, 1–10
KN220 CVAX and boot and diagnostic facility, on KN220 CPU, 1–9
Diagnostic tests
list of, 4–4
parameters for, 4–4
Differences, between KN220 and other systems
in autoboot capability, 1–15
in H3602–AC switch meanings, 1–15
in modules and slot locations, 1–15
in terminology surrounding word size, 1–7
Displays

Displays (Cont.)
console banner, 3–5
Documentation, relating to KN220 systems, list of, F–1
Drive connect
BA430, 2–11
Drives, 2–11
descriptions, 2–11
ID switches, 2–12
internal to tabletop connect, 2–12
RRD40 table, 2–14
tabletop to tabletop connect, 2–11
TLZ04 table, 2–12
DRVEXR local program, 4–39, 4–53
DRVTST local program, 4–39, 4–52
DSSI
bus length, 2–10
bus termination, 2–10
cabling, 2–9
configuration, 2–4
interface, on KN220 I/O module, 1–11
ISE order, 2–4
node ID, 2–4
node name, changing, 2–6
testing with H3281 loopback, 4–48
unique addresses, 4–38
unit number, changing, 2–7
dump command, in normal mode, 3–25

E

e (examine) command, in normal mode, 3–27
Embedded drive, 2–11
description, 2–11
Environmental variables, 3–17
ERASE local program, 4–55
Error messages
console, list of, 4–32
console, sample of, 4–19
halt, 4–31
VMB, 4–33
Ethernet
See Network interface
Ethernet connectors
location of, on H3602–AC, 1–12
EXAMINE command, 3–54
EXIT command, 3–56

F

FE utility, 4–41
Field replaceable units (FRUs) for DECsystem 5500, E–1 to E–4
fill command, in normal mode, 3–28
FIND command, 3–57
Firmware
description of, on KN220 CPU, 3–1
features of, 3–2
power-up sequence, 3–5
Firmware, on KN220 CPU, 1–9
Floating-point accelerator, on KN220 CPU, 1–7
FRUs and console display, 4–24
Function switch
and loopback tests at power-up, 3–6
and query position at power-up, 3–7
Function switch, on H3602–AC, 1–15

G

General purpose registers (GPRs)
in error display, 4–23
initialization of, 3–12
symbolic addresses for, 3–40
go command, in normal mode, 3–29

H

H3103 loopback connector, 4–49
H3281 loopback connector for DSSI, 4–48
H3602–AC CPU I/O panel
description of, 1–12

H3602–AC I/O panel, 4–49
H8572 loopback connector, 4–49
HALT command, 3–58
Halts
and actions taken, 3–3
conditions for, external, 3–4
CVAX, halt entry and dispatch code, 3–3
messages, list of, 4–31
registers saved, 3–3
registers set to fixed values, 3–3
Hardware error summary register, 4–42
HELP command, 3–59
help command, in normal mode, 3–30
HISTRY local program, 4–39, 4–54

I

init command, in normal mode, 3–32
INITIALIZE command, 3–61
Initial power-up test
See IPT
Installation
of KN220 and MS220–AA modules, 1–15
Internal processor registers (IPRs)
symbolic addresses for, 3–40
IPT, 3–5, 4–25
ISE
cabling, 2–9
node ID switches, 2–4

K

KFQSA storage adapter
and running Configure utility, C–6
changing the ISE unit number, C–15
configuring, C–2
programming, C–8
reprogramming, C–13
KN220
features of, 1–6
KN220 (Cont.)
LEDs, 4–30
KN220 CPU module
description of, 1–1, 1–3
firmware, features of, 3–2

L

Loopback connectors
H3103, 4–49
H8572, 4–49
list of, 4–50
tests, 4–48
Loopback tests
at power-up, 3–6
for DSSI problems, 4–48
for Ethernet problems, 4–48

M

Maintenance mode, 3–21
address specifiers, 3–39
command keywords, 3–44
command qualifiers, 3–42
command syntax, 3–39
console commands, 3–47
description of, 3–38
special characters for, 3–38
symbolic addresses, 3–40
Maintenance mode commands
CONFIGURE, 2–3
Mass storage
See DSSI
MDM operating system
and restart procedures, 3–15
boot devices, supported, 3–13
bootstrap, 3–11
supported boot flags, 3–13
Memory
cache, on KN220 CPU, 1–8
isolating FRU, 4–44
main memory system, on KN220 CPU, 1–8
maximum supported, in KN220 systems, 1–8

Memory (Cont.)
MS220–AA, description of, 1–19
testing, 4–44
Messages
console error, 4–32
system halt, 4–31
VMB error, 4–33
Module
configuration, 2–2
order, in backplane, 2–1
self-tests, 4–49
MOVE command, 3–63
MS220–AF option kit, contents of, 1–19
MS220 memory module, description of, 1–19

N

Network interface
on KN220 I/O module, 1–11
NEXT command, 3–65
Node name
DSSI, changing, 2–6
Normal mode
command syntax in, 3–19
console commands for, 3–18
conventions used for description purposes, 3–20
description of, 3–16
environmental variables for, 3–17
special characters for, 3–16

O

OCP, 4–51
cabling, 2–9
Operating system bootstrap
and bootstrap support, 3–8
and ULTRIX–32 procedures, 3–10
conditions for, 3–8
of MDM, description of, 3–11
of ULTRIX–32, 3–9
Operating system restart, MDM, 3–15
Operating system support, 1–1
Operation switch
and action position at power-up, 3–6, 3–7
and maintenance position at power-up, 3–6
and normal position at power-up, 3–5
Operation switch, on H3602–AC, 1–15
Operator console panel
See OCP

P

Panel, CPU I/O
See H3602–AC CPU I/O panel
Parameters
for diagnostic tests, 4–7
in error display, 4–21
PARAMS local program, 4–39, 4–56
commands, 4–56
passwd command, 1–18
Physical address locations, and accessing through R3000 processor, B–1
Power-up sequence, 3–5
Prestoserve software, D–1
printenv command, in normal mode, 3–33

Q

Q22-bus
interface chip (CQBIC), 1–10

R

R3000 RISC chip, on KN220 CPU, 1–7
REPEAT command, 3–67
RF30 local programs
DRVEXR, 4–39
DRVTST, 4–39
HISTRY, 4–39
PARAMS, 4–39
RF-series ISE, local programs

RF-series ISE, local programs (Cont.)
DRVEXR, 4–53
DRVTST, 4–52
ERASE, 4–55
HISTRY, 4–54
list of, 4–51
PARAMS, 4–56
RF-series ISEs
configuration errors, 4–51
diagnostic error codes, 4–59
diagnostics, 4–50
RISC chip, R3000, on KN220 CPU, 1–7
ROM-based diagnostics
description of, 4–3
list of, 4–4
utilities, 4–4

S

Scripts
calling sequence for, 4–10
commonly used, 4–10
creating with 9E utility, 4–12
description of, 4–7
list of, 4–9
SCSI
interface, on KN220 I/O module, 1–12
SEARCH command, 3–68
Securing the system, 1–16
Security password
forgotten, 1–17
Self-test, for modules, 4–49
SET command, 3–70
setenv command, in normal mode, 3–34
SET HOST/DUP command, 3–70
SHOW command, 3–73
Shutdown, abnormal, recovery from, D–1
SII chip, on KN220 I/O module, 1–11
START command, 3–77
Switch
function, on H3602–AC, 1–15
function, set to test position at power-up, 3–6
operation, on H3602–AC, 1–15
operation, set to action position at power-up, 3–6, 3–7
operation, set to maintenance position at power-up, 3–6
operation, set to normal position at power-up, 3–5
Symbolic addresses, 3–40
for any address space, 3–42
for GPRs, 3–40
for IPRs, 3–40
for physical memory, 3–41
Systems, for KN220 CPU module set, 1–1

T

Tabletop drive, 2–11
description, 2–11
TEST command, 3–78
test command, in normal mode, 3–35
Tests, diagnostic
See also Troubleshooting; Loopback
list of, 4–4
parameters for, 4–7
Time-of-year clock, on KN220 I/O, 1–9
Timers, on KN220 I/O, 1–9
Troubleshooting
memory failures, 4–44
memory failures, additional suggestions for, 4–47
with FE utility, 4–41

U

ULTRIX–32 operating system
boot devices, supported, 3–10
bootstrap, 3–9
bootstrap procedure for, 3–10

ULTRIX–32 operating system (Cont.)
Exerciser and Uerf commands, list of, A–1
Unit number
DSSI, changing, 2–7
UNJAM command, 3–79
unpriv command, 1–18
Unsecuring the system, 1–17
unsetenv command, in normal mode, 3–36
Utilities, diagnostic, 4–4

V

Virtual memory bootstrap
See VMB
VMB, 3–12
boot flags, 3–13
error messages, 4–33

X

X command (binary load and unload), 3–80
x command, in normal mode, 3–37