
KA670

KA670 CPU System Maintenance

Order Number EK–347AB–MG–002

Digital Equipment Corporation
Maynard, Massachusetts

First Printing, March 1990
Revised, July 1991

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.

ML–S1635

This document was prepared using VAX DOCUMENT, Version 1.2.

Contents

Preface xi

Chapter 1 KA670 CPU and Memory Subsystems

1.1 Introduction 1-1
1.2 KA670 Module Components 1-4
1.2.1 DC520: P-Chip 1-4
1.2.2 DC523: F-Chip 1-5
1.2.3 DC592: C-Chip 1-5
1.2.4 DC250 (formerly DC561) G-Chip 1-6
1.2.5 DC256: CQBIC (formerly DC527) 1-6
1.2.6 DC541: SGEC 1-6
1.2.7 DC542: SHAC 1-6
1.2.8 DC511: SSC 1-7
1.2.9 Firmware ROMs 1-7
1.3 MS670 Memory Modules 1-8
1.4 Console Module 1-8

Chapter 2 Internal and External Configuration

2.1 Introduction 2-1
2.2 Internal Configuration 2-1
2.2.1 General Module Order 2-1
2.2.2 Installation of Supported Options 2-1
2.2.2.1 Recommended Order of Q-bus Options 2-3
2.2.3 Module Configuration and Naming 2-4
2.3 Mass Storage Configuration 2-5
2.3.1 Changing the DSSI Node Name of an ISE 2-6
2.3.2 Changing the DSSI Unit Number 2-8

2.3.3 Access to RF-Series Firmware in VMS Through DUP 2–10
2.3.3.1 Allocation Class 2–10
2.4 DSSI Cabling, Device Identity, and Bus Termination 2–11
2.5 External Configuration: DSSI VAXcluster (Dual-Host) Capability 2–11

Chapter 3 KA670 Firmware

3.1 Introduction 3–1
3.2 KA670 Firmware Features 3–1
3.2.1 General Description 3–2
3.3 Halt Entry/Exit/Dispatch Code 3–3
3.4 External Halts 3–4
3.5 Power-Up Sequence 3–5
3.5.1 Mode Switch Set to Test 3–6
3.5.2 Mode Switch Set to Language Inquiry 3–6
3.5.3 Mode Switch Set to Normal 3–8
3.5.4 LED Codes 3–9
3.6 Bootstrap 3–10
3.6.1 Bootstrap Initialization Sequence 3–10
3.6.2 VMB Boot Flags 3–11
3.6.3 Supported Boot Devices 3–12
3.6.4 Autoboot 3–13
3.7 Operating System Restart 3–14
3.7.1 Restart Sequence 3–14
3.7.2 Locating the RPB 3–15
3.8 Console I/O Mode Control Characters 3–16
3.8.1 Command Syntax 3–17
3.8.2 Address Specifiers 3–17
3.8.3 Symbolic Addresses 3–18
3.8.4 Console Numeric Expression Radix Specifiers 3–21
3.8.5 Console Command Qualifiers 3–21
3.8.6 Console Command Keywords 3–22
3.9 Console Commands 3–24
3.9.1 BOOT 3–24
3.9.2 CONFIGURE 3–26

3.9.3 CONTINUE 3–28
3.9.4 DEPOSIT 3–28
3.9.5 EXAMINE 3–29
3.9.6 FIND 3–30
3.9.7 HALT 3–31
3.9.8 HELP 3–31
3.9.9 INITIALIZE 3–33
3.9.10 MOVE 3–34
3.9.11 NEXT 3–35
3.9.12 REPEAT 3–37
3.9.13 SEARCH 3–37
3.9.14 SET 3–39
3.9.15 SHOW 3–43
3.9.16 START 3–47
3.9.17 TEST 3–47
3.9.18 UNJAM 3–48
3.9.19 X—Binary Load and Unload 3–48
3.9.20 ! (Comment) 3–50

Chapter 4 Troubleshooting and Diagnostics

4.1 Introduction 4–1
4.2 General Procedures 4–1
4.3 KA670 ROM-Based Diagnostics 4–3
4.3.1 Diagnostic Tests 4–4
4.3.2 Scripts 4–8
4.3.3 Script Calling Sequence 4–10
4.3.4 Test 9F: Creating Scripts 4–11
4.3.5 Modifying an Existing Script 4–17
4.3.6 Console Displays 4–18
4.4 Acceptance Testing 4–27
4.5 Troubleshooting 4–32
4.5.1 FE Utility 4–32
4.5.2 Overriding Halt Protection 4–33
4.5.3 Isolating Memory Failures 4–33
4.5.4 Additional Troubleshooting Suggestions 4–35

4.6 Loopback Tests 4–36
4.6.1 Testing the Console Port 4–36
4.6.2 SHAC Loopback Testing: 56 and 58 4–37
4.6.3 SGEC Loopback Testing: 5F and 59 4–38
4.7 Module Self-Tests 4–39
4.8 RF-Series ISE Troubleshooting and Diagnostics 4–40
4.8.1 DRVTST 4–41
4.8.2 DRVEXR 4–43
4.8.3 HISTRY 4–44
4.8.4 ERASE 4–45
4.8.5 PARAMS 4–46
4.8.5.1 EXIT 4–47
4.8.5.2 HELP 4–47
4.8.5.3 SET 4–47
4.8.5.4 SHOW 4–48
4.8.5.5 STATUS 4–48
4.8.5.6 WRITE 4–48
4.9 Diagnostic Error Codes 4–49

Appendix A Address Assignments

A.1 KA670 General Local Address Space Map A–1
A.2 KA670 Detailed Local Address Space Map A–2
A.3 External, Internal Processor Registers A–6
A.4 Global Q22-Bus Address Space Map A–7

Appendix B ROM Partitioning

B.1 Firmware EPROM Layout B–1
B.2 System ID Registers B–3
B.2.1 PR$_SID (IPR 3E) B–3
B.2.2 SIE (20040004) B–4
B.2.3 Call-Back Entry Points B–5
B.2.4 CP$GET_CHAR_R4 . . . . . B–5
B.2.5 CP$MSG_OUT_NOLF_R4 . . . . . B–6
B.2.6 CP$READ_WTH_PRMPT_R4 . . . . . B–6

B.3 Boot Information Pointers B–7

Appendix C System Support Chip RAM Partitioning

C.1 SSC RAM Layout C–1
C.1.1 Public Data Structures C–1
C.1.2 Console Program MailBoX (CPMBX) C–2
C.1.3 Firmware Stack C–3
C.1.4 Diagnostic State C–3
C.1.5 USER Area C–4

Appendix D Data Structures

D.1 Halt Dispatch State Machine D–1
D.2 RPB D–5
D.3 VMB Argument List D–8

Appendix E Error Messages

E.1 Halt Code Messages E–1
E.2 VMB Error Messages E–3
E.3 Console Error Messages E–5

Appendix F Machine State on Power-Up

F.1 Main Memory Layout and State F–2
F.1.1 Reserved Main Memory F–2
F.1.1.1 PFN Bitmap F–3
F.1.1.2 Scatter/Gather Map F–3
F.1.1.3 Firmware "Scratch Memory" F–4
F.1.2 Contents of Main Memory F–4
F.2 Memory Control Registers F–4
F.2.1 On-Chip Cache F–4
F.2.2 Translation Buffer F–5
F.2.3 Halt-Protected Space F–5

Appendix G MOP Support

G.1 Network "Listening" G–1
G.2 MOP Counters G–7

Appendix H Related Documentation

Glossary

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 3–7
3–2 Normal Diagnostic Countdown 3–8
3–3 Unsuccessful Diagnostic Countdown 3–8
3–4 Selecting a Boot Device 3–14
4–1 Test 9E 4–5
4–2 Creating a Script 4–13
4–3 Listing and Repeating Tests with Utility 9F 4–14
4–4 Modifying an Existing Script 4–17
4–5 Console Display (No Errors) 4–18
4–6 Sample Output with Errors 4–19
4–7 FE Utility Example 4–32

Figures

1–1 KA670 CPU Module Component Side 1–2
1–2 KA670 CPU Module Block Diagram 1–3
1–3 Console Module (Front) 1–9
1–4 Console Module (Back) 1–10
2–1 DSSI Cabling for a Generic Dual-Host Configuration 2–12
3–1 KA670 Firmware Structural Components 3–2

B–1 KA670 EPROM Layout B–2
B–2 SID: System Identification Register B–3
B–3 SIE: System Identification Extension (20040004) B–4
B–4 Boot Information Pointers B–8
C–1 KA670 SSC RAM Layout C–1
C–2 SSCR0 (20140400): Console Program MailBoX (CPMBX) . . . C–2
C–3 SSCR1 (20140401) C–3
C–4 SSCR2 (20140402) C–3
F–1 Memory Layout After Power-Up Diagnostics F–2

Tables

3–1 Actions Taken on a Halt 3–4
3–2 Language Inquiry on Power-Up or Reset 3–7
3–3 LED Codes 3–9
3–4 Virtual Memory Bootstrap (VMB) Boot Flags 3–11
3–5 Boot Devices Supported by the KA670 3–12
3–6 Console Symbolic Addresses 3–18
3–7 Symbolic Addresses Used in Any Address Space 3–20
3–8 Console Radix Specifiers 3–21
3–9 Console Command Qualifiers 3–22
3–10 Command Keywords by Type 3–23
3–11 Console Command Summary 3–23
4–1 First Things to Check 4–2
4–2 Power Supply Status Indicators 4–2
4–3 Scripts Available to Customer Services 4–9
4–4 Machine Check Exception During Executive 4–20
4–5 Exception During Executive with No Parameters 4–21
4–6 Other Exceptions with Parameters, No Machine Check 4–21
4–7 KA670 Console Displays As Pointers to FRUs 4–23
4–8 Signature Field Values 4–29
4–9 Memory Module as FRU 4–34
4–10 Loopback Connectors for Common Devices 4–39
4–11 DRVTST Messages 4–41
4–12 DRVEXR Messages 4–43
4–13 HISTRY Messages 4–44
4–14 ERASE Messages 4–46

4–15 RF-Series ISE 4–49
D–1 Firmware State Transition Table D–2
D–2 Restart Parameter Block Fields D–5
D–3 VMB Argument List D–8
E–1 Halt Messages E–2
E–2 VMB Error Messages E–3
E–3 Console Error Messages E–5
G–1 KA670 Network Maintenance Operations Summary G–2
G–2 Supported MOP Messages G–4
G–3 Ethernet and IEEE 802.3 Packet Headers G–6
G–4 MOP Multicast Addresses and Protocol Specifiers G–6
G–5 MOP Counter Block G–7

Preface

This guide describes the base system, configuration, ROM-based diagnostics, and troubleshooting procedures for VAX 4000 Model 300 systems, containing the KA670 CPU.

Intended Audience

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

Organization

This guide has four chapters, eight appendixes, and a glossary.

Chapter 1 describes the KA670/MS670 CPU and memory subsystem.

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) bus interface cabling.

Chapter 3 describes the firmware that resides in ROM on the KA670 and provides a list of console error messages and their meaning.

Chapter 4 describes the KA670 diagnostics, including an error message and an FRU cross-reference table. Chapter 4 also describes diagnostics that reside on the RF-series integrated storage element.

Appendix A describes address assignments.

Appendix B describes ROM partitioning

Appendix C describes RAM partitioning.

Appendix D describes data structures

Appendix E describes error messages.

Appendix F describes the KA670 machine state.

Appendix G describes MOP support

Appendix H lists related documentation.

The Glossary defines terms and acronyms used in this manual.

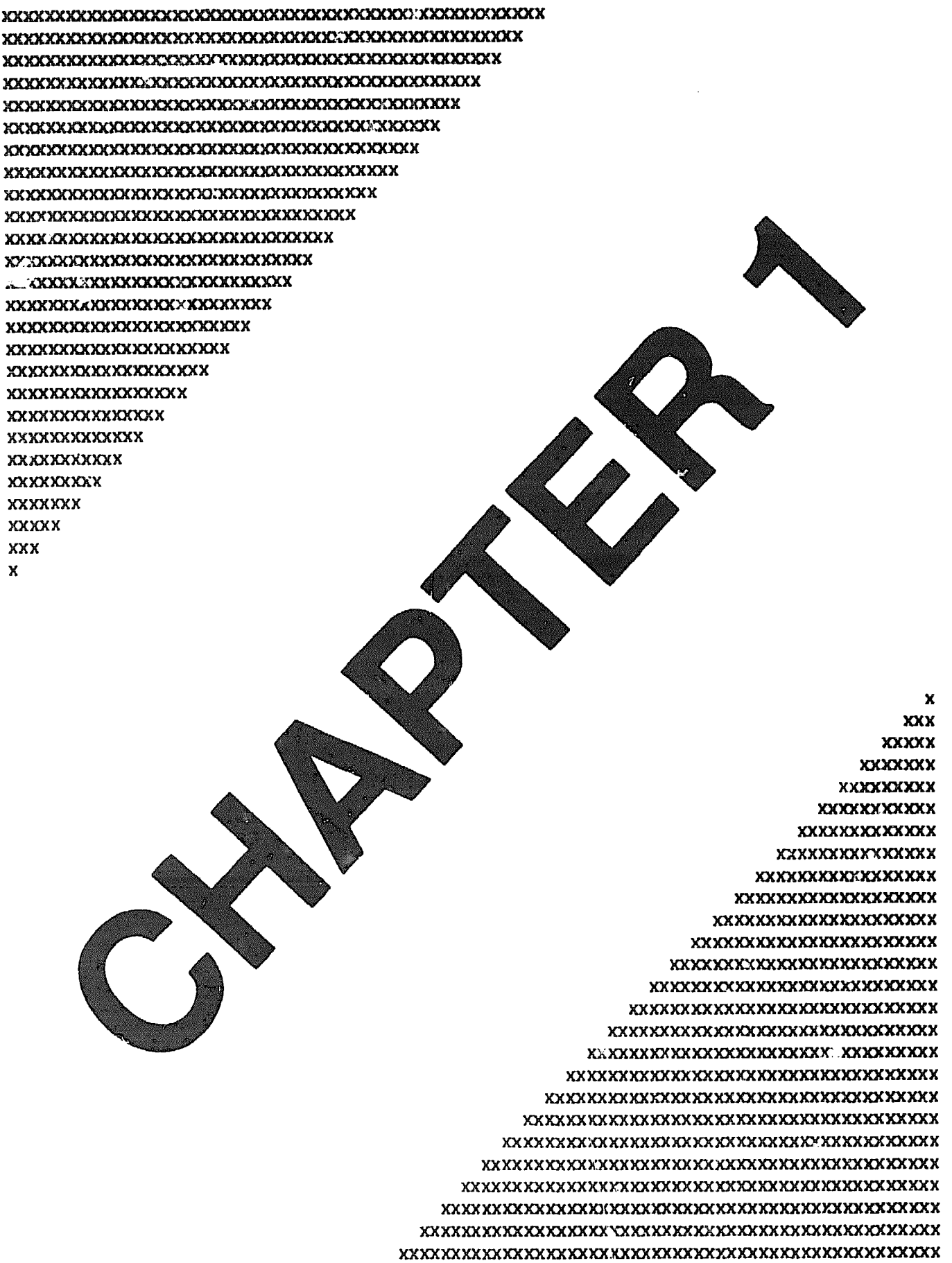
Warnings, Cautions, Notes

Warnings, cautions, and notes appear throughout this guide. They have the following meanings:

WARNING	Provides information to prevent personal injury.
CAUTION	Provides information to prevent damage to equipment or software.
NOTE	Provides general information about the current topic.

Conventions

A system prompt and a command in boldface, uppercase type, for example, **>>>SHOW DSSI**, show that the user enters the command at the system prompt.


CHAPTER 1

Chapter 1

KA670 CPU and Memory Subsystems

1.1 Introduction

This chapter describes the KA670 CPU. The KA670 is a quad-height VAX processor module for the Q22-bus. It is designed for use in high-speed, real-time applications and for multiuser, multitasking environments. The KA670 employs a cache memory to maximize performance. See Figure 1–1 for a view of the major chips on the board and Figure 1–2 for a block diagram of the major functions.

The two variants are the KA670–AA, which runs multiuser software, and the KA670–BA, which runs single-user software.

The KA670 is used in the VAX 4000 Model 300, which is housed in the BA440 enclosure. Configuration of the KA670 is provided by switches on the H3604 console module.

The KA670 CPU module and MS670 memory modules combine to form a VAX CPU and memory subsystem that uses the DSSI bus to communicate with mass storage devices, the Q22-bus to communicate with I/O devices, and the Ethernet to communicate across the network. The KA670 and MS670 modules mount in dedicated backplane slots covered by the H3604 console module. These modules communicate through the G-Chip Memory Interconnect (GMI) bus.

The KA670 can be configured only as an arbiter CPU on the Q22-bus, where it arbitrates bus mastership and fields bus interrupt requests and any on-board interrupt requests.

The KA670 communicates with the console device through the H3604 CPU Console Module. The console module is described in Section 1.4.

CAUTION: *Static electricity can damage integrated circuits. Always use a grounded wrist strap (PN 29–11762–00) and grounded work surface when working with the internal parts of a computer system.*

Figure 1–1: KA670 CPU Module Component Side

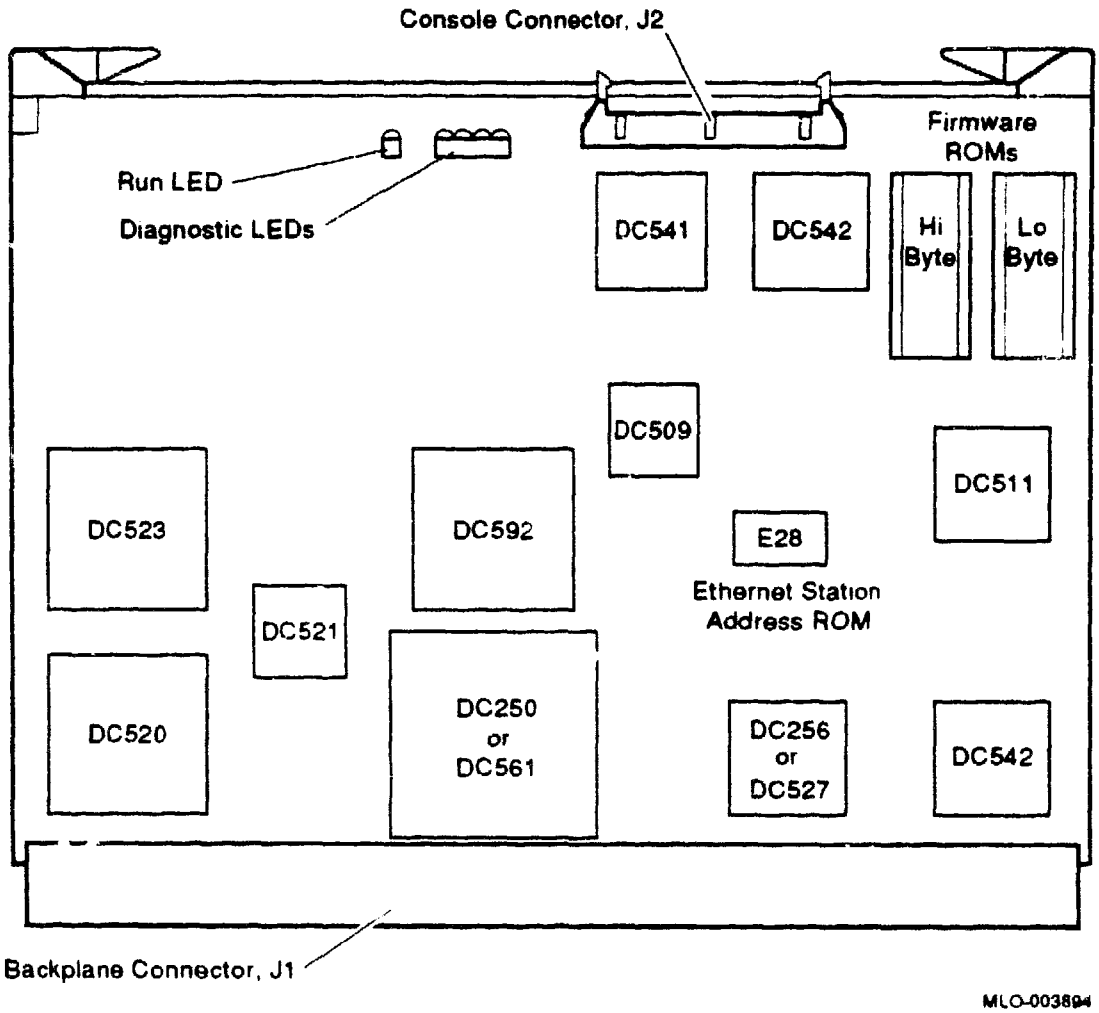
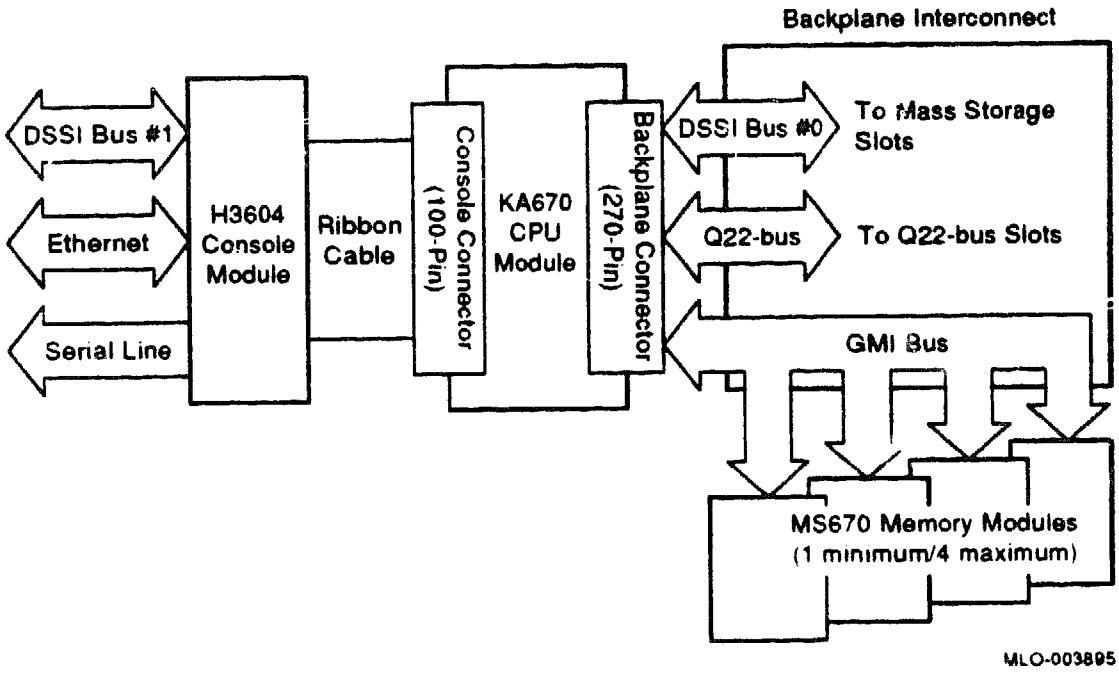


Figure 1–2: KA670 CPU Module Block Diagram



1.2 KA670 Module Components

The KA670 CPU is a single, quad-height module (L4000–xA). The major hardware components of the KA670 CPU module are listed below. The chip identification numbers (DCxxx) are reflected in Figure 1–1.

1. DC520 (P-chip): VAX central processor
2. DC523 (F-chip): Floating-point accelerator
3. DC592 (C-chip): Backup cache controller
4. DC250 or DC561 (G-chip): Main memory controller
5. DC521: Clock generator
6. DC256 (CQBIC): Q22-bus interface (replaced DC527)
7. DC541 (SGEC): Ethernet interface
8. DC542 (SHAC): DSSI interface chips (2)
9. DC511 (SSC): System support chip
10. DC509: Clock generator
11. Firmware ROMs (2): 256 Kbytes; each 128-Kbyte EPROMs
12. 100-pin connector to the console module
13. 270-pin connector to the backplane carrying signals for the Q-bus, the DSSI bus, and the memory interconnect

1.2.1 DC520: P-Chip

The P-chip, or CPU, executes the 181 instructions defined by the MicroVAX subset of the VAX instruction set. The VAX central processor has a 2-Kbyte primary cache and a 143-MHz clock, and supports full VAX memory management with demand paging and a 4-Gbyte virtual address space. The P-chip supports the MicroVAX instruction set, with the following string instructions included:

- CMPC3
- CMPC5
- LOCC
- SCANC
- SKPC

- SPANC

The P-chip provides the following subsets of the VAX data types:

- Byte
- Word
- Longword
- Quadword
- Character string
- Variable-length bit field
- Absolute queues
- Self-relative queues
- F-floating
- G-floating
- D-floating

Support for the remaining VAX data types and instructions is provided through macrocode emulation.

1.2.2 DC523: F-Chip

The floating-point accelerator is implemented by the F-chip that executes the VAX f-, d-, and g-floating-point instructions. The F-chip receives opcode information from the P-chip and receives operands directly from memory, backup cache, or the P-chip primary cache. The result of the floating point is always returned to the P-chip.

1.2.3 DC592: C-Chip

The KA670 module incorporates a two-level, hierarchical cache memory to maximize CPU performance. The first level cache consists of a 2-Kbyte primary cache contained in the P-chip. The second level or backup cache, consists of the C-chip and the twenty-four (24) 16-Kbytes-by-4 static RAMs (total 128 Kbytes plus parity). The C-chip contains the tag store and the control logic for the backup cache RAMs, as well as a copy of the primary cache tag store to guarantee primary cache coherence between memory and processor.

1.2.4 DC250 (formerly DC561): G-Chip

The DC250 replaces the earlier DC561 and controls communication among the RDAL, CPDAL, and GMI busses:

- RDAL connects to the DC520, DC523, DC592, and the backup cache.
- CPDAL connects to the DC256, DC541, DC542, and the DC511.
- GMI connects to the main memory.

A memory module must be populated with RAM in chips of equal capacity. When a memory module contains 1-Mbit or 4-Mbit RAMs, the bank sizes are 4 Mbytes or 16 Mbytes, respectively.

1.2.5 DC256: CQBIC (formerly DC527)

The KA670 includes a Q22-bus interface that supports the following functions:

- Programmable and direct mapping functions
- Masked and unmasked longword reads and writes from CPU to the Q22-bus memory and I/O space and the interface registers
- Up to 16-word, block-mode writes from Q22-bus to main memory
- Transfers from CPU to local Q22-bus memory space

1.2.6 DC541: SGEC

The second-generation Ethernet chip implements an on-board network interface. Used in connection with the console module, the SGEC allows the KA670 to connect to either a ThinWire or standard Ethernet. The SGEC supports the Ethernet Data Link Layer as specified in DEC STD 134 and also supports CP bus parity protection.

1.2.7 DC542: SHAC

The two single-host adapter chips implement the DSSI (Digital Storage Systems Interconnect) bus interfaces. The DSSI interface allows each DSSI bus on the KA670 to transmit packets of data to, and receive packets from, up to seven other DSSI devices. These devices include the RF-series integrated storage elements (ISEs), a KFQSA module, a second KA670 module, a KA660 module, or a KA640 module.

The DSSI bus improves system performance because it has a higher transfer rate than the Q22-bus and it relieves the Q22-bus of disk traffic. The DSSI bus has eight data lines, one parity line, and eight control lines. Controllers are built into the ISEs, enabling many functions to be handled

without host or adapter intervention. The SHAC chips also support CP bus parity protection.

1.2.8 DC511: SSC

The system support chip provides console and boot code support functions, operating system support functions, timers, and the following features:

- Word-wide ROM unpacking
- 1-Kbyte battery backed-up RAM
- Halt-arbitration logic
- Console serial line
- Interval timer with 10 ms interrupts
- VAX-standard time-of-year clock with battery backup
- IORESET register
- Programmable CDAL bus timeout
- Two programmable timers
- Register controlling the diagnostic LEDs

1.2.9 Firmware ROMs

Resident firmware ROM is located on two chips, each 128-Kbyte EPROMs. The firmware gains control when the processor halts and contains programs that provide the following services:

- Board initialization
- Power-up self-testing of the KA670 and MS670 modules
- Emulation of a subset of the VAX standard console (auto or manual bootstrap, auto restart, and a simple command language for examining or altering the state of the processor)
- Booting from supported Q22-bus devices
- Multilingual capability

See Chapter 3 for details on KA670 firmware.

1.3 MS670 Memory Modules

The MS670 memory module is a single, quad-height, standard Fingerless Circuit Size (FCS) array that uses a 150-pin, high-density connector to communicate to the KA670. Memory access is through a 39-bit CPU /Memory data interconnect (through the GMI bus) consisting of 32 data, plus seven Error Checking/Correcting (ECC) bits.

The MS670 memories are available in two variations: 32 Mbytes and 64 Mbytes. The MS670–BA (module designation L4001–BA) is a fully populated, 32-Mbyte MOS memory using 100 ns, 1 megabit x 1, Small Outline "J" (SOJ) Dynamic RAMs (DRAM) in surface-mount packages. The MS670–CA module (module designation L4001–CA) is a half-populated, 64-Mbyte MOS memory using 100 ns, 4 megabits x 1 SOJ DRAM in surface-mount packages.

The KA670-based system allows for any combination of up to four MS670–BA/MS670–CA memory arrays providing a memory capacity from 32 Mbytes up to 256 Mbytes.

1.4 Console Module

The console module (H3604) allows the KA670 CPU module to interface to a serial line console device, a DSSI bus, and to the Ethernet. The console module is wide enough to cover the five slots dedicated to the KA670 and its four MS670 modules. Five adhesive tags are included for the user to name the modules in the respective slots.

The console module contains the following connectors:

- Console serial line (with baud rate switch)
- Two Ethernet connectors (standard/ThinWire selectable)
- Two DSSI connectors that allow daisy-chaining of one DSSI bus; terminators are shipped for both DSSI connectors

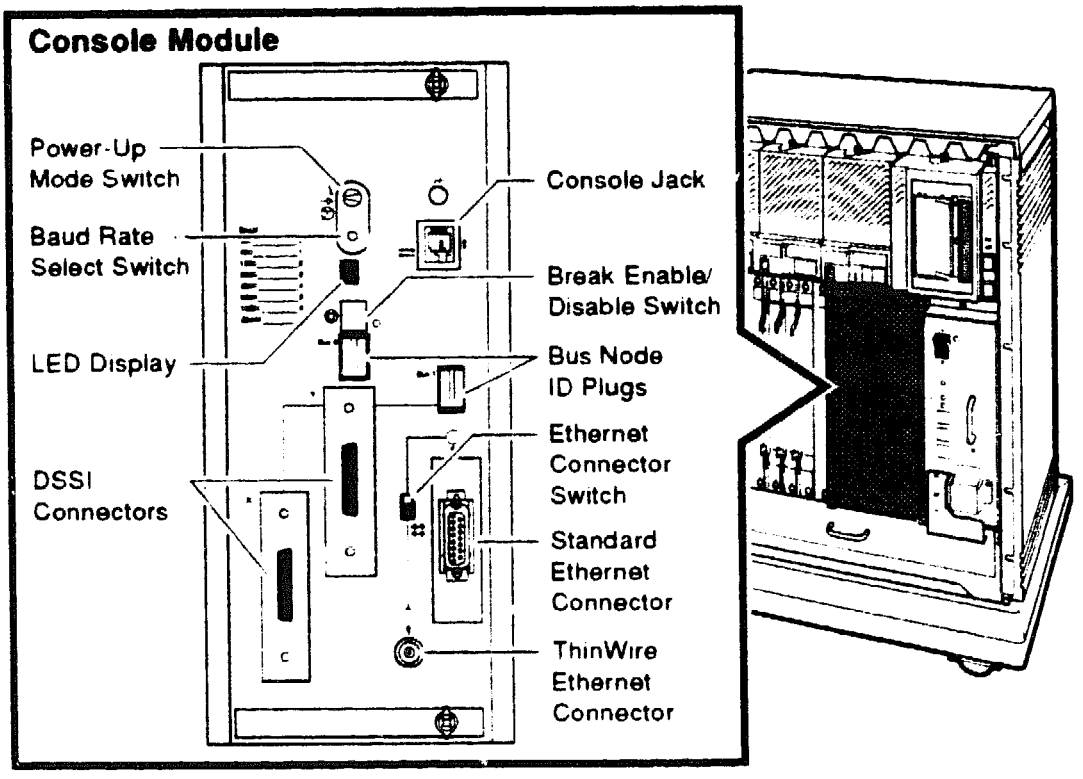
The four switches are:

- Baud rate select switch for the serial console line
- Power-up mode switch (language inquiry, test, or normal)
- Break Enable/Disable switch from the console keyboard BREAK key (default) or CTRL/P, depending on the state of SSCCR<15>, which is set with the console command, SET CONTROLP ENABLED. If the Break Enable/Disable switch is set to the disable position (0), the system attempts to autoboot on power-up.

- Ethernet connector select. The console module contains a switch to select one of two Ethernet connectors. One connector is for 15-conductor cables (standard Ethernet), and the other connector is for a BNC coaxial cable (ThinWire Ethernet). An LED indicates the selected connector and voltage for that connector.

See Figure 1–3.

Figure 1–3: Console Module (Front)

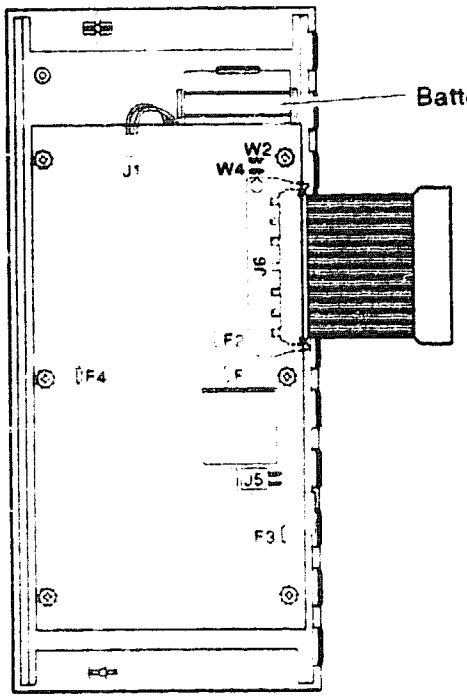


Inside the door of the console module are a DSSI circuit fuse and two jumpers. The fuse is to protect against shorts from the accidental grounding of the DSSI cable power pin. The jumpers are used by Digital personnel only.

There are two connectors from the console module to the internal BA440. One is a 4-pin power connection to a small printed circuit card that inserts into the backplane alongside the KA670. The other is the 100-pin connector to the KA670 CPU module.

See Figure 1–4.

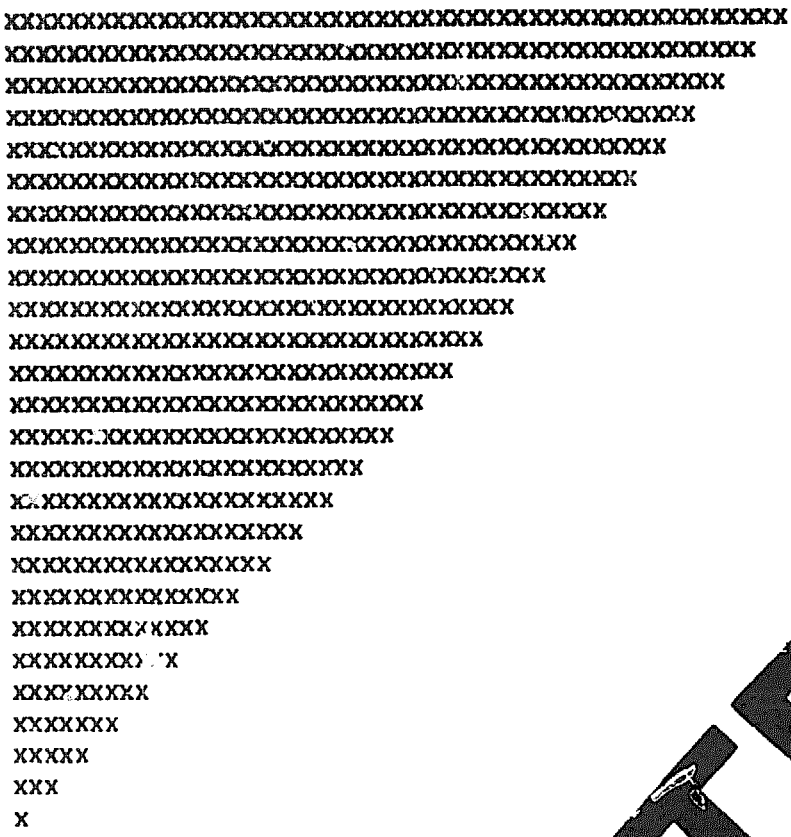
Figure 1–4: Console Module (Back)



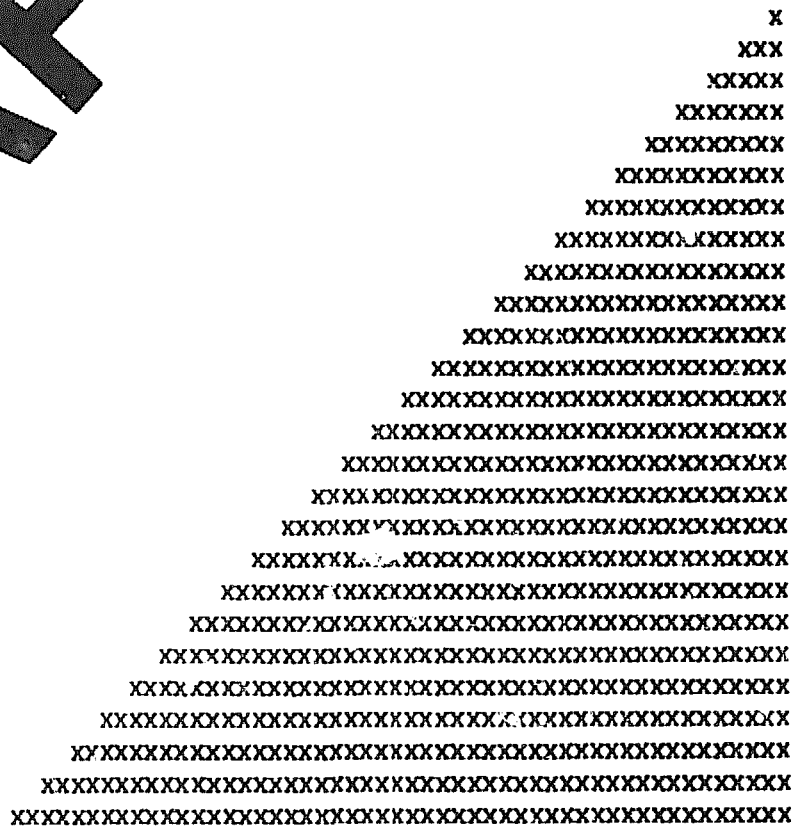
J1 = TOY Clock Battery
J5 = H3604 Power
J6 = CPU Interface
W2 = Remote Boot Enable
W4 = Not Used

F1 = ThinWire Ethernet Power, 5A
PN = 12-09159-00
F2 = -12V Power, .062A
PN = 90-09122-00
F3 = DSSI Terminator Power, 2.0A
PN = 12-10929-06
F4 = Standard Ethernet Power, 1.5A
PN = 12-10929-08

MLO-006351



CHAPTER 2



Chapter 2
Internal and External Configuration

2.1 Introduction

This chapter describes the guidelines for the configuration of a KA670 system, both internal and external. Internal configuration relates to modules and mass storage devices within a system enclosure. External configuration relates to the position or placement of a KA670 system in a single- or dual-host configuration.

2.2 Internal Configuration

Internal configuration includes module order, power consumption, bus loading, and the cabling within the enclosure.

2.2.1 General Module Order

The backplane of the KA670 system is split. The five rightmost backplane slots are dedicated to CPU and memory modules, while the seven slots to the left are for all remaining modules.

The KA670 CPU module and the MS670 memory modules are installed in the five rightmost backplane slots.

- Install the KA670 CPU in slot 5.
- Install MS670 memory modules in slots 1 through 4. Start with slot 4 and add modules from left to right with no gaps.

2.2.2 Installation of Supported Options

The order of the supported options 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

The supported options arrayed by type are:

Communications

CXA16–AA/AF: 16-line DEC423 asynchronous controller
CXB16–AA/AF: 16-line RS422 asynchronous controller
CXY08–AA/AF: 8-line RS232C asynchronous controller with modem
DSV11–SA/SF: Q-bus 2-line synchronous
DEQRA–CA: Token Ring Network Controller
DESQA–SA/SF: ThinWire Ethernet adapter
DFA01–AA/AF: 2400/1200 BPS modem
DPV11–SA/SF: Q-bus synchronous programmable interface
DRV1W–SA/SF: General purpose 16-bit parallel DMA interface
DRV1J–SA/SF: Q-bus parallel interface
DIV32–SA/SF: Q-bus ISDN basic rate access interface
KMV1A–SA/SF: Single-line programmable controller with DMA

General

KWV11–SA/SF: Programmable real-time clock
IBQ01–SA/SF: DECscan/BITBUS controller
IEQ11–SA/SF: Dual-bit DMA serial Q-bus controller
AAV11–SA/SF: 2-channel DAC module
ADV11–SA/SF: 16-channel ADC module
AXV11–SA/SF: 16-channel ADC, 2-channel DAC module
ADQ32–SA/SF: 32-channel ADC module
LPV11–SA/SF: Line printer controller
DRQ3B–SA/SF: Q-bus parallel I/O interface
MRV11: Q-bus, universal socket, 32-Kbyte EPROM

Mass Storage, Tape, Pedestal Expansions

RF72E–AA/AF: 1.0-Gbyte full-height integrated storage element
RF71E–AA/AF: 400-Mbyte full-height integrated storage element
RF31E–AA/AF: 381-Mbyte half-height integrated storage element
TK70E–AA/AF/TQK70–SA/SF: 5.25-inch cartridge, 296-Mbyte tape drive, tape controller
TF85: 5.25-inch cartridge, 2.6-Gbyte DSSI tape drive with built-in controller
TLZ04–JA/JF/GA: 1.2-Gbyte cassette (DAT) tape drive (requires KZQSA storage adapter)
KLESI–SA: Q-bus to LESI adapter
KFQSA–SE/SG: DSSI Q-bus adapter
KZQSA–SA/SF: Storage adapter for TLZ04 tape drive and RRD42 compact disc drive
RA70: Storage expansion (separate cabs only)

RA81/82: Storage array (separate cabs only)
RA90: Storage array (separate cabs only)
KDA50–SE/SC: SDI Q-bus adapter
KRQ50–SA/SF: Q-bus controller for RRD40–DC
TU81E–SA/SB: Magnetic tape
TSV05–SE/SF/SH/SJ: Q-bus TS05 magnetic tape controller
B400X: Expansion box with 10 Q-bus slots and up to 4 ISEs
R400X–B9: Expansion box with up to 7 RF-series ISEs
RRD40: 600-Mbyte CDROM table-top drive
RRD42: 600-Mbyte tabletop compact disc drive (requires KZQSA storage adapter)
RSV20–A: WORM optical drive subsystem (KLESI controller)
ESE20: Electronic storage element (KDA50 controller)

2.2.2.1 Recommended Order of Q-bus Options

The modules, listed in recommended order of placement on the Q22-bus, are:

MRV11 (Placement not critical)
AAV11
ADV11
AXV11
KWV11
DRV1J
DESQA
DPV11
DIV32
VCB02
DFA01
CXA16
CXY08
CXB16
LPV11
DRV1W
KRQ50
IEQ11
ADQ32
DRQ3B
DSV11
KLESI
IBQ01
TSV05 (M7530 controller)
KDA50–SE
KFQSA–SE
KZQSA
TQK50

TQK70
KDA50–SA/KFQSA–SA
M9060–YA

2.2.3 Module Configuration and Naming

Each 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 and interrupt vector values. 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 console I/O prompt (>>>). The CONFIG utility eliminates the need to boot the VMS operating system to determine CSRs and interrupt vectors. Enter the CONFIGURE command, then HELP for the list of supported devices:

```
>>>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? cxa16,1
Device,Number? desqa,1
Device,Number? tqk70
Device,Number? qza
Device,Number? kfqsa-disk
De   e,Number? exit
```

```
Address/Vector Assignments
-774440/120 DESQA
-772150/154 KFQSA-DISK
-774500/260 TQK70
-760440/300 CXA16
-761300/310 KZQSA
```

NOTE: *Of the devices listed in the CONFIG display, not all are supported on the VAX 4000 Model 300 systems. See Section 2.2.2 for supported options.*

The LPV11–SA has two sets of CSR address and interrupt vectors. To determine the correct values for an LPV11–SA, enter LPV11,2 at the DEVICE prompt for one LPV11–SA or enter LPV11,4 for two LPV11–SA modules.

2. See *Microsystems Options* for switch and CSR and interrupt vector jumper settings for supported options.

2.3 Mass Storage Configuration

There is space for four mass storage devices—three ISEs and one tape drive or else four ISEs (integrated storage elements) The ISEs are part of the Digital Storage Systems Interconnect (DSSI) bus.

DSSI bus 0 is part of the backplane. The ISEs are of the RF-series and they plug into the backplane to become part of the bus. Each ISE must have its own unique DSSI bus node ID. The ISE receives its bus node ID from a plug on the ISE front panel. Bus node ID plugs on the console panel provide bus node IDs for the built-in DSSI host adapters.

The VMS operating system creates DSSI disk device names according to the following scheme:

```
nodename $ DIA unit number
```

For example,

```
SUSAN$DIA3
```

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

```
>>>BOOT SUSAN$DIA3
```

You can access local programs in the RF-series ISE through the MicroVAX Diagnostic Monitor (MDM), through the VMS operating system, or console I/O mode, using the SET HOST/DUP command. The SET HOST/DUP command creates a virtual terminal connection to the storage device and the designated local program, using the Diagnostic and Utilities Protocol (DUP)

standard dialog. Section 2.3.3 describes the procedure for accessing DUP through the VMS operating system. Section 3.9.14 describes the console I/O mode SET HOST/DUP command.

2.3.1 Changing the DSSI Node Name of an ISE

Each ISE has a node name that is maintained in EEPROM on board 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 DSSI device to something more meaningful by following the procedure in Example 2–1. In the example, the node name for the ISE at DSSI node 0, bus 0, is changed from R7CZZC t[illegible]SDSK.

See Section 4.8.5 for further information about the PARAMS local program.

Example 2–1: Changing a DSSI Node Name

```
>>>SHOW DSSI
DSSI Bus 0 Node 0 (R7CZZC) !The node name for this drive will
-DIA0 (RF71)    !be changed from R7CZZC to SYSDSK.
DSSI Bus 0 Node 1 (R7ALUC)
-DIA1 (RF71)
DSSI Bus 0 Node 2 (R7EB3C)
-DIA2 (RF71)
DSSI Bus 0 Node 6 (*)

DSSI Bus 1 Node 0 (SNEEZY)
-DIB0 (RF71)
DSSI Bus 1 Node 1 (DOPEY)
-DIB1 (RF71)
DSSI Bus 1 Node 2 (SLEEPY)
-DIB2 (RF71)
DSSI Bus 1 Node 3 (GRUMPY)
-DIB3 (RF71)
DSSI Bus 1 Node 4 (BASHFUL)
-DIB4 (RF71)
DSSI Bus 1 Node 5 (HAPPY)
-DIB5 (RF71)
DSSI Bus 1 Node 6 (DOC)
-DIB6 (RF71)
DSSI Bus 1 Node 7 (*)
 >>
 >>SET HOST/DUP/DSSI/BUS:0 0 PARAMS
 tarting DUP server...
opyright (c) 1990  igital Equipment Corporation
PARAMS>
```

Example 2–1 (continued on next page)

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

```
PARAMS>SHOW NODENAME
Parameter      Current            Default          Type      Radix
---------  ----------------  ----------------  --------  -----
NODENAME          R7CZZC             RF71         String  Ascii   B

PARAMS>SET NODENAME SYSDSK

PARAMS>SHOW NODENAME

Parameter      Current            Default          Type      Radix
---------  ----------------  ----------------  --------  -----
NODENAME          SYSDSK             RF71         String  Ascii   B

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

Stopping DUP server...
>>>SHOW DSSI
DSSI Bus 0 Node 0 (SYSDSK)    !The node name has changed from
-DIA0 (RF71)      !R7CZZC to SYSDSK.
DSSI Bus 0 Node 1 (R7ALUC)
-DIA1 (RF71)
DSSI Bus 0 Node 2 (R7EB3C)
-DIA2 (RF71)
DSSI Bus 0 Node 6 (*)

DSSI Bus 1 Node 0 (SNEEZY)
-DIB0 (RF71)
DSSI Bus 1 Node 1 (DOPEY)
-DIB1 (RF71)
DSSI Bus 1 Node 2 (SLEEPY)
-DIB2 (RF71)
DSSI Bus 1 Node 3 (GRUMPY)
-DIB3 (RF71)
DSSI Bus 1 Node 4 (BASHFUL)
-DIB4 (RF71)
DSSI Bus 1 Node 5 (HAPPY)
-DIB5 (RF71)
DSSI Bus 1 Node 6 (DOC)
-DIB6 (RF71)
DSSI Bus 1 Node 7 (*)
>>>
```

2.3.2 Changing the DSSI Unit Number

By default, the ISE assigns its unit number to the same value as the DSSI node ID. When devices are added to the second bus (bus 1), and the system is using a nonzero allocation class, you need to assign new unit numbers for the ISEs on one of the busses, as the unit numbers for ISEs throughout the system must be unique.

Example 2–2 shows how to change the unit number of a DSSI device. This example changes the unit number for the RF71 at DSSI node ID 0, on bus 1, from 1 to 10 (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 4.8.5 for further information about the PARAMS local program.

Example 2–2: Changing a DSSI Unit Number

```
>>>SHOW DSSI
DSSI Bus 0 Node 0 (SYSDSK)    'The unit number for this drive will
-DIA0 (RF71)                  'be changed from 0 to 10 (DIA0 to DAI10).
DSSI Bus 0 Node 1 (R7ALUC)
-DIA1 (RF71)
DSSI Bus 0 Node 2 (R7EB3C)
-DIA2 (RF71)
DSSI Bus 0 Node 6 (*)

DSSI Bus 1 Node 0 (SNEEZY)
-DIB0 (RF71)
DSSI Bus 1 Node 1 (DOPEY)
-DIB1 (RF71)
DSSI Bus 1 Node 2 (SLEEPY)
-DIB2 (RF71)
DSSI Bus 1 Node 3 (GRUMPY)
-DIB3 (RF71)
DSSI Bus 1 Node 4 (BASHFUL)
-DIB4 (RF71)
DSSI Bus 1 Node 5 (HAPPY)
-DIB5 (RF71)
DSSI Bus 1 Node 6 (DOC)
-DIB6 (RF71)
DSSI Bus 1 Node 7 (*)
>>>
>>>SET HOST/DUP/DSSI/BUS:0 0 PARAMS
Starting DUP server...
Copyright (c) 1990  Digital Equipment Corporation
PARAMS>
```

Example 2–2 (continued on next page)

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

```
PARAMS>SHOW UNITNUM
Parameter     Current           Default         Type      Radix
--------- ---------------  ---------------  --------  -----
UNITNUM                 0                0      Word     Dec   U

PARAMS>SET UNITNUM 10
PARAMS>SET FORCEUNI 0
PARAMS>SHOW UNITNUM

Parameter     Current           Default         Type      Radix
--------- ---------------  ---------------  --------  -----
UNITNUM                10                0      Word     Dec   U

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

PARAMS>WRITE 'This command writes the changes to EEPROM.
PARAMS>EX
Exiting...
Stopping DUP server...
>>>
>>>SHOW DSSI
DSSI Bus 0 Node 0 (SYSDSK)     !The unit number for this drive has
-DIA10 (RF71)          !changed and the bus node ID remains at 0.
DSSI Bus 0 Node 1 (R7ALUC)
-DIA1 (RF71)
DSSI Bus 0 Node 2 (R7EB3C)
-DIA2 (RF71)
DSSI Bus 0 Node 6 (*)

DSSI Bus 1 Node 0 (SNEEZY)
-DIB0 (RF71)
DSSI Bus 1 Node 1 (DOPEY)
-DIB1 (RF71)
DSSI Bus 1 Node 2 (SLEEPY)
-DIB2 (RF71)
DSSI Bus 1 Node 3 (GRUMPY,
-DIB3 (RF71)
DSSI Bus 1 Node 4 (BASHFUL)
-DIB4 (RF71)
DSSI Bus 1 Node 5 (HAPPY)
-DIB5 (RF71)
DSSI Bus 1 Node 6 (DOC)
-DIB6 (RF71)
DSSI Bus 1 Node 7 (*)
```

Example 2–2 (continued on next page)

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

```
>>>
```

2.3.3 Access to RF-Series Firmware in VMS Through DUP

You can access the RF-series ISE firmware utilities from the VMS operating system as well as through the console commands described in Section 4.8.

Access the ISE firmware through the VMS operating system to look up or to view parameter settings. To change ISE parameter settings, enter the ISE firmware through the console I/O mode SET HOST/DUP command. These parameters must be stable while VMS is booted. Reboot VMS for changes to take effect.

The CONNECT command does an implicit load. Load the FYDRIVER, using the following commands in SYSGEN:

```
$ MCR SYSGEN
SYSGEN> CONNECT FYA0/NOADAPTER
SYSGEN> EXIT
$
```

2.3.3.1 Allocation Class

There are two methods of assigning allocation class: VMS SYSGEN and console methods.

When a KA670 system containing ISEs is configured in a cluster, either as a boot node or a satellite node, you can assign the allocation class, using VMS SYSGEN, which matches the nonzero allocation class of the ISE.

To change the allocation class of the ISE, use the following commands:

```
$ MCR SYSGEN
SYSGEN> SET ALLOCLASS <allocation class value>
SYSGEN> WRITE CURRENT
```

You can also access the ISE firmware utilities, using the following VMS command:

```
$ SET HOST/DUP/SERVER=MSCP$DUP/TASK=PARAMS nodename
```

You can then use the following console commands to change the allocation class:

```
>>>SET HOST/DUP/DSSI/BUS:<bus number> <DSSI node number> PARAMS
Starting DUP server...

PARAMS>SET ALLCLASS <allocation class value>
```

```
PARAMS>WRITE
Changes require controller initialization, ok? [Y/N] Y

Stopping DUP server..
>>>
```

2.4 DSSI Cabling, Device Identity, and Bus Termination

The ISEs within the BA440 enclosure are connected to the system backplane and communicate internally over the backplane. There are no internal DSSI cables. Externally, a 50-pin round cable connects the DSSI bus to other devices, either hosts or expanders.

There are two DSSI storage busses in the KA670 system. One DSSI storage bus, Bus 0, is routed along the backplane and exits the enclosure at the lower left front. The near-end termination is contained on the backplane for the internal DSSI, Bus 0. The far-end termination for Bus 0 is provided by the removable terminators (PN 12–29258–01).

The other DSSI storage bus, Bus 1, is configured by means of the DSSI connectors on the console module panel. If unused, both DSSI connectors must be terminated.

All DSSI devices on the same bus must have unique node IDs. You can see on the face of the console module the two DSSI node ID plugs (Figure 1–3). These node ID plugs provide an identity for the DSSI adapter on each DSSI bus. The DSSI busses are separate, so that when shipped as a single-host configuration, both DSSI adapters have the number 7 on their ID plugs.

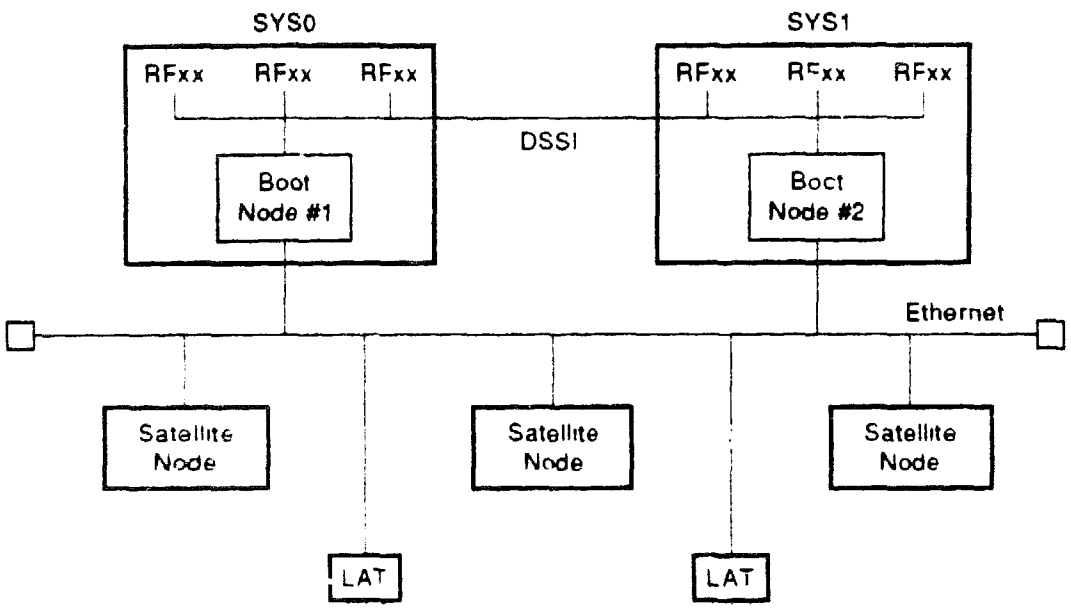
2.5 External Configuration: DSSI VAXcluster (Dual-Host) Capability

A DSSI VAXcluster or dual-host configuration is one in which two systems can access the same DSSI devices. Some failures of either system can be tolerated, in which case the remaining system continues to access all the DSSI devices and assure continued processing.

The KA670 can be combined with other VAX 4000 or MicroVAX systems to make a dual-host configuration, as shown in Figure 2–1. The CPU of the other host is as follows:

- Another KA670 or KA660
- A KA640
- Any other supported CPU enhanced with a KFQSA module

Figure 2–1: DSSI Cabling for a Generic Dual-Host Configuration

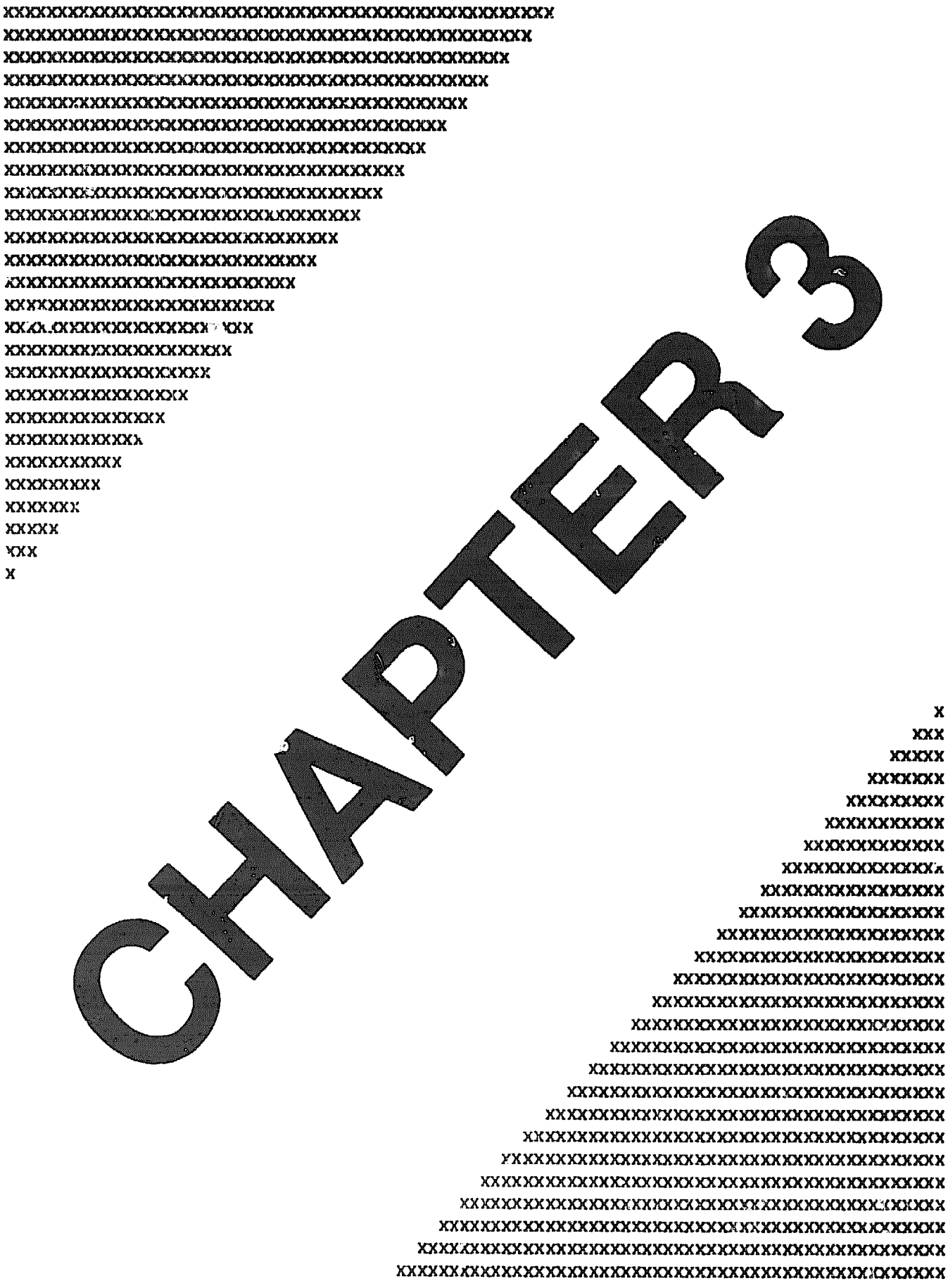


Two KA670 systems may be connected through an external DSSI cable (BC21M). Each KA670 system is a boot member for a number of satellite nodes. The system disk may reside in either enclosure. The KA670 in each enclosure has equal access to the system disk and to any other DSSI device in either enclosure.

If one of the KA670 modules fails, all satellite nodes booted through that KA670 module lose connections to the system disk. However, the dual-host capability enables each satellite node to know that the system disk is still available through a different path—that of the functioning KA670 module. A connection through that KA670 is then established, and the satellite nodes are able to continue operation. The entire cluster will run slower, since one KA670 is now serving the satellite nodes of both KA670s. Processing can continue, however, until Customer Services can repair the problem.

A dual-host system cannot recover from the following conditions:

- System disk failure, which can be caused by such factors as a power supply failure in the enclosure containing the disk.
- DSSI cabling failure, which must be repaired to continue operation.


CHAPTER 3

Chapter 3
KA670 Firmware

3.1 Introduction

The KA670 provides a maximum of 256 Kbytes of ROM in two 128-Kbyte EPROMs, which are arranged as words and located at the CPU restart location in VAX I/O space at physical address 20040000.

3.2 KA670 Firmware Features

The firmware is located in two 128-Kbyte EPROMs on the KA670. The firmware address range (hexadecimal) in the KA670 local I/O space is 20040000 to 2007FFFF, inclusive. Unlike previous Q22-based processors, there is no duplicate decoding of the boot ROM into halt-protected and halt-unprotected spaces. The firmware displays diagnostic progress and error reports on the KA670 LEDs and on the console terminal.

The firmware performs the following functions:

- Automatic or manual bootstrap and restart of an operating system following processor halts.
- An interactive command language that allows you to examine and alter the state of the processor.
- Diagnostics that test all components on the board and verify that the module is working correctly.
- Support of various terminals and devices for the system console.
- Multilingual support. The firmware can issue system messages in several languages.

To allow the console program to operate, the processor must be functioning at a level to enable it to execute instructions from the console program ROM.

3.2.1 General Description

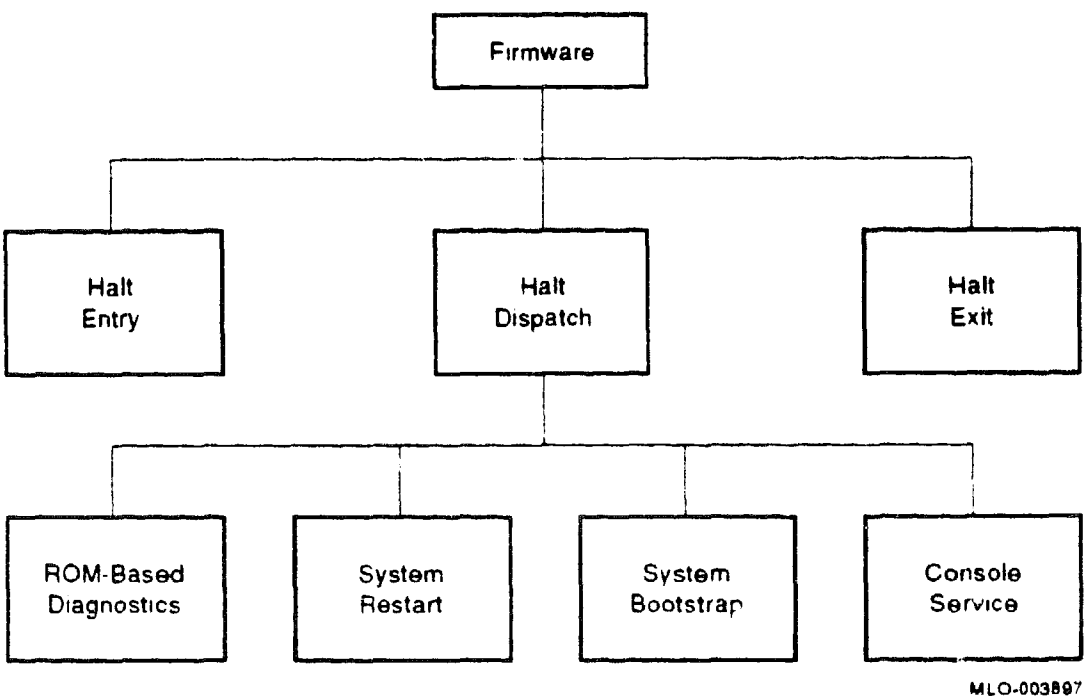
The firmware consists of the following major functional areas:

Halt entry/exit/dispatch code
Bootstrap
Console I/O mode
Diagnostics

The halt entry/exit/dispatch code, bootstrap, and console I/O mode are described in this chapter. Diagnostics are described in Chapter 4.

The KA670 firmware is comprised of several major functional blocks of code, as illustrated in the structural diagram in Figure 3–1.

Figure 3–1: KA670 Firmware Structural Components



The halt entry code is entered following system halts, resets, or severe errors. Basically, this code is responsible for saving the machine state and then transferring control to the firmware dispatcher. Naturally, the halt exit code is entered whenever a transition is desired from halted state to the running state and it performs a restoration of the saved context before the transition. The halt dispatcher determines the nature of the halt, then transfers control to the appropriate code.

The ROM-based diagnostics consist of a series of functional component diagnostics invoked by a diagnostic executive.

Depending on the nature of the halt and the hardware context, the firmware attempts either an operating system restart, a bootstrap operation, or transits to console I/O mode.

3.3 Ha.t Entry/Exit/Dispatch Code

The main purpose of this code is to save the state of the machine on halt entry, invoke the dispatcher, and restore the state of the machine on exit to program I/O mode. Whenever a halt occurs, the processor enters the halt entry code at physical address 20040000. The halt entry code saves the machine state, then transfers control to the firmware halt dispatcher.

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
PR$_SIPL	Interrupt priority level
DLEDR	Diagnostic LED register

To ensure that the console program can run, the halt entry code unconditionally sets the following registers to fixed values on any halt:

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

When the processor exits the firmware and reenters program mode, the user context is restored and any changes become operative only then.

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 processor restart code (PR$_SAVPSL <13:08>), the O /S MailBoX halt action field (CPMBX <01:00>), the Break Enable/Disable switch on the H3604 console module, and the user-defined halt action.

Table 3–1 lists the actions taken, by sequence. The user-defined halt action is used when the O/S MailBoX halt action field is 0 and on power-up if breaks are enabled. This allows users to determine what action should be taken on error halts and at power-up. This user-defined field is set by the console SET HALT command.

Table 3–1: Actions Taken on a Halt

Reset/ Power-Up or Halt	Break Enable Switch	User-Defined Halt Action	O/S MailboX Halt Action	Action(s)
T	1	0,1,3	x	Diagnostics, console
T	1	2,4	x	Diagnostics, if success boot, if either fail console
T	0	x	x	Diagnostics, if success boot, if either fail console
F	1	0	0	Console
F	0	0	0	Restart, if this fails boot, if that fails console
F	x	1	0	Restart, if it fails console
F	x	2	0	Boot, if it fails console
F	x	3	0	Console
F	x	4	0	Restart, if this fails boot, if that fails console
F	x	x	1	Restart, if it fails console
F	x	x	2	Boot, if it fails console
F	x	x	3	Console

"T" indicates that the condition is true.
"F" indicates that the condition is false.
"X" indicates that the condition is "don't care".

Because the KA670 does not support battery backed up main memory, an operating system restart operation is not attempted on power-up.

3.4 External Halts

Power-up diagnostics tests are run in a halt-protected environment and cannot be halted. When the CPU is running or is in program I/O mode, the following conditions can trigger an external halt:

- The Break Enable switch is set to enable, and you press BREAK/CTRL-P on the system console terminal.
- Assertion of the BHALT line on the Q22-bus, if the SCR<14> (BHALT_ ENABLE), default, bit in the CQBIC is set. Halts are initiated by pressing HALT on the operator control panel (OCP).
- Negation of DCOK, if the SCR<7>(DCOK_ACT) bit is set. (The firmware clears this bit.)

- Recognition of a valid MOP BOOT message by an appropriately initialized SGEC, if the REMOTE_BOOT_ENABLE jumper located on the inside of the H3604 console module is in place (BDR<(REM_BOOT_ENB)>(REMOTE_BOOT_ENABLE) = 0). As a result, a bootstrap is attempted and, if that fails, the console is entered.

  NOTE: *The firmware does not initialize the SGEC for this operation. The operating system must set up the SGEC to support this feature.*

The switch labeled Restart negates DCOK. The negation of DCOK may also be asserted by the DESQA sanity timer, or any other Q22-bus module that chooses to implement the Q22-bus restart/reboot protocol. Since on power-up the SCR<7>(DCOK_ACTION) bit is cleared, the default consequence to deasserting DCOK is to generate a processor restart. Hence, pressing the Restart button initiates a power-up sequence and destroys system state.

CAUTION: *Pressing* RESTART *while in console I/O mode will destroy the previously saved system state.*

The action taken by the halt dispatch code on a console BREAK (if breaks are enabled) or the Halt button on the OCP is the same: the firmware enters console I/O mode.

3.5 Power-Up Sequence

On power-up, the firmware identifies the console device, optionally performs a language inquiry, and runs the diagnostics.

Power-up actions differ, depending on the state of the power-up mode switch on the console module. The mode switch has three settings: test, language inquiry, and normal. The differences are described in Sections 3.5.1 through 3.5.3.

The firmware waits for power to stabilize by monitoring SCR<15>(POK). Once power is stable, the firmware verifies that the console battery backup RAM (BBU RAM) is valid (backup battery is charged) by checking SSCCR<31>(BLO). If it is invalid or zero (battery is discharged), BBU RAM is initialized.

After the battery check, the firmware tries to determine the type of terminal attached to the console serial line. It uses this information to determine if multinational support is appropriate.

3.5.1 Mode Switch Set to Test

Use the test position on the H3604 to verify a proper connection between the KA670 and the console terminal.

- To test the console terminal port, insert the H3103 loopback connector into the H3604 console connector and put the switch in the test position. You must install the loopback connector to run the test.
- To test the console cable, install the H8572 connector on the end of the console cable and insert the H3103 into 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 6. The firmware reads the baud rate and mode switch, then transmits and receives a character sequence.
- During the passive state (7 seconds), the LED is set to 3.

If at any time the firmware detects an error (parity, framing, overflow, or no characters), the display hangs at 6. If the configuration switch is moved from the test position, the firmware continues as if on a normal power-up.

3.5.2 Mode Switch Set to Language Inquiry

If the console module mode switch is set to language inquiry, or the firmware detects that the contents of BBU RAM are invalid, the firmware prompts you for the language to be used for displaying the following system messages (if the console terminal supports the multinational character set):

```
Loading system software.
Failure.
Restarting system software.
Performing normal system tests.
Tests completed.
Normal operation not possible.
Bootfile.
Memory configuration error.
No default boot device has been specified.
Available devices.
Device?
Retrying network bootstrap.
```

The language selection menu appears under the conditions listed in Table 3–2. The position of the Break Enable switch has no effect on these conditions. The firmware will not prompt for a language if the console, such as the VT100, does not support the multinational character set (MCS).

Table 3–2: Language Inquiry on Power-Up or Reset

Mode	Language Not Previously Set[1]	Language Previously Set
Language Inquiry	Prompt[2]	Prompt
Normal	Prompt	No Prompt

[1]Action if contents of BBU RAM invalid same as Language Not Previously Set.

[2]Prompt = Language selection menu displayed.

The language selection menu is shown in Example 3–1. If no response is received within 30 seconds, the firmware defaults to English (5).

Example 3–1: Language Selection Menu

```
 1) Dansk
 2) Deutsch (Deutschland/Osterreich)
 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) Portugues
14) Suomi
15) Svenska
 (1..15):
```

NOTE: *The information contained within the parentheses indicates the specific keyboard variant.*

In addition, the console may prompt you for a default boot device following a successful diagnostic countdown. See Section 3.6, Example 3–4.

After the language inquiry, the firmware continues as if on a normal power-up.

3.5.3 Mode Switch Set to Normal

The console displays the language selection menu if the mode switch is set to normal and the contents of BBU RAM are invalid or a language has not yet been selected. The next step in the power-up sequence is to execute the bulk of ROM-based diagnostics. In addition to message text, a countdown is displayed in Example 3–2.

Example 3–2: Normal Diagnostic Countdown

```
Performing normal system tests.
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 console uses the saved console language if the mode switch is set to normal and the contents of BBU RAM are valid.

In the case of diagnostic failure, a diagnostic register dump is performed similar to Example 3–3. The remaining diagnostics execute and the countdown continues. If the diagnostics are successful and halts are enabled, the firmware displays the console prompt.

Example 3–3: Unsuccessful Diagnostic Countdown

```
Performing normal system tests.
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..

?5F 2 0E FF 0000 0000 02        ; SUBTEST_5F_0E, DE_SGEC.LIS

P1=00000000  P2=00000000  P3=5839FF00  P4=00000000  P5=00000000
P6=00000000  P7=00000000  P8=00000000  P9=0000080A P10=00000003
r0=00000054  r1=20084019  r2=00004206  r3=00000000  r4=00000000
r5=1FFFFFFC  r6=C0000003  r7=20008000  r8=00004000 EPC=00000000
10..09..08..07..06..05..04..03..
Normal operation not possible.
>>>
```

3.5.4 LED Codes

In addition to the console diagnostic countdown, a hexadecimal value is displayed on the diagnostic LEDs on the module and the H3604 console module. The purpose of the LED display is to improve fault isolation when there is no console terminal or when the hardware is incapable of communicating with the console. Table 3–3 lists LED codes and associated actions at power-up. The LED code is changed before the corresponding test or action is performed.

Table 3–3: LED Codes

LED Value	Actions
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	CPU/FPA/CACHE CHIP tests
A	Board cache memory tests
9	G-chip tests
8	Memory tests
7	CQBIC (Q22-bus) tests
6	Console loopback tests
5	DSSI subsystem tests
4	Ethernet subsystem tests
3	Console I/O mode
2	Control passed to VMB
1	VMS has loaded the operating system
0	Program I/O mode, control passed to operating system

3.6 Bootstrap

The KA670 supports bootstrap of VAX/VMS, VAXELN, and MDM diagnostics.

The following section shows how the firmware initializes the system to a known state before dispatching to the primary bootstrap, called the virtual memory bootstrap (VMB).

3.6.1 Bootstrap Initialization Sequence

1. Checks CPMBX<2>(BIP), bootstrap in progress. If it is set, bootstrap fails and the console displays the message `Failure.` in the selected console language.
2. If this is an automatic bootstrap, prints the message `Loading system software.` on the console terminal.
3. 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.
4. Writes a form of this boot request, including active boot flags and boot device (BOOT/R5:0 EZA0, for example), to the console terminal.
5. Sets CPMBX<2>(BIP).
6. Initializes the Q22-bus scatter-gather map.
7. Validates the PFN bitmap. If invalid, rebuilds it.
8. 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.
9. Initializes the general purpose registers:

R0	Address of descriptor of boot device name, 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 map enable)
AP	Halt code
SP	Base of 128-Kbyte good memory block + 512
PC	Base of 128-Kbyte good memory block + 512
R1, R6, R7, R8, R9, FP	0

10. Copies the VMB image from EPROM to local memory, beginning at the base of the 128 Kbytes of good memory block + 512 (decimal).

11. Exits from the firmware to VMB residing in memory.

VMB is the primary bootstrap for VAX processors. VMB loads the secondary bootstrap image from the appropriate boot device and transfers control to it.

3.6.2 VMB Boot Flags

The VMB boot flags are listed in Table 3–4.

Table 3–4: Virtual Memory Bootstrap (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, VMS maps the code for the XDELTA debugger 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 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...].

3.6.3 Supported Boot Devices

Table 3–5 lists the boot devices supported by the KA670 CPU. The table correlates the boot device names expected in a BOOT command with the corresponding supported devices. The device name used for the bootstrap operation is one of three:

- EZA0, if no default boot device has been specified
- The default boot device specified at initial power-up or through SET BOOT
- Name explicitly specified in a BOOT command line

Boot device names consist of a device code of at least two letters (A through Z) in length, followed by a single-character controller letter (A through Z), and ending in a device unit number (0 through 16,383).

Table 3–5: Boot Devices Supported by the KA670

Boot Name	Controller Type	Device Type(s)
Disk		
[node$]DIu	On-board DSSI	RFxx
DUcu	KFQSA DSSI	RFxx
	KDA50 MSCP	RA70, RA80, RA81, RA82, RA90
	KRQ50 MSCP	RRD40
DLcn	RLV12	RL02
Compact Disc		
[node$]DKAn	KZQSA SCSI	RRD4x
Tape		
[node$]MImu	On-board DSSI	TF85
MUcu	TQK50 MSCP	TK50
	TQK70 MSCP	TK70
	KLESI	TU81E
MKAn	KZQSA SCSI	TLZ04

Table 3–5 (Cont.): Boot Devices Supported by the KA670

Boot Name	Controller Type	Device Type(s)
Network		
EZA0	On-board Enet	–
XQcu	DESQA	–
PROM		
PRAu	MRV11	–
PRB0	Customer EPROM space	–

3.6.4 Autoboot

IMPORTANT: *Unless you specify otherwise, the KA670 default boot device is the Ethernet adapter, EZA0. See Example 3–4.*

- If the Break Enable/Disable switch is set to disable, the CPU tries to autoboot an operating system upon successful completion of the power-up self-tests.
- The system looks for a previously selected boot device. If you have not yet selected a boot device, the system issues a list of bootable devices and prompts you to select a boot device from the list.

NOTE: *You can also specify a default boot device by typing the SET BOOT command (Section 3.9.1).*

- If you do not type a boot device name within 30 seconds, the system boots from the Ethernet adapter, EZA0.
- If you type a boot device name within 30 seconds, this device becomes the default boot device and the system boots from that device, as shown in Example 3–4.

NOTE: *For diskless and tapeless systems that boot software over the network, select only the Ethernet adapter. All other boot devices are inappropriate.*

Example 3–4: Selecting a Boot Device

```
Performing normal system tests.
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.
Loading system software.
No default boot device has been specified.

Available devices.
-DIB0 (RF30)
-DIB1 (RF30)
-EZA0 (08-00-2B-06-10-42)

Device? [EZA0]: DIB0

(BOOT/R5:0 DIB0)

 2..
-DIB0
 1..0..
```

3.7 Operating System Restart

An operating system restart is the process of bringing up the operating system from a known initialization state following a processor halt. A restart occurs under the conditions listed in Table 3–1.

To restart a halted operating system, 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 an 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.

3.7.1 Restart Sequence

The firmware restarts the operating system in the following sequence:

1. Checks CPMBX<3>(RIP). If it is set, restart fails.
2. Prints this message on the console terminal:

   ```
   Restarting system software.
   ```

3. Sets CPMBX<3>(RIP).

4. Searches for a valid RPB. If none is found, restart fails.
5. Checks the operating system RPB$L_RSTRTFLG<0>(RIP) flag. If it is set, restart fails.
6. Writes a 0 (zero) to the diagnostic LEDs.
7. Dispatches to the restart address, RPB$L_RESTART, with:

   SP = the physical address of the RPB plus 512
   AP = the halt code
   PSL = 041F0000
   PR$_MAPEN = 0

If the restart is successful, the operating system must clear CPMBX<3>(RIP).

If restart fails, the firmware prints this message on the console terminal:

```
Failure.
```

3.7.2 Locating 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.

3.8 Console I/O Mode Control Characters

In console I/O mode, several characters have special meaning:

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 (<CR><LF>).
RUBOUT	When you press RUBOUT, the console deletes the previously typed character. The resulting display differs, depending on whether the console is a video or a hard-copy terminal. For hard-copy terminals, the console echoes a backslash (\), followed by the deletion of the character. 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 RUBOUT RUBOUT 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/A and F14	Toggle insertion/overstrike mode for command line editing. By default, the console powers up to overstrike mode.
CTRL/B or up_arrow (or down_arrow)	Recalls previous command(s). Command recall is only operable if sufficient memory is available. This function may then be enabled and disabled using the SET RECALL command.
CTRL/D and Left Arrow	Move cursor left one position.
CTRL/E	Moves cursor to the end of the line.
CTRL/F and Right Arrow	Move cursor right one position.
CTRL/H, Backspace, and F12	Move cursor to the beginning of the line.
CTRL/U	Echoes ^U<CR> and deletes the entire line. Entered but otherwise ignored if typed on an empty line.
CTRL/S	Stops output to the console terminal until CTRL/Q is typed. Not echoed.
CTRL/Q	Resumes output to the console terminal. 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.
CTRL/C	Echoes ^C<CR> and aborts processing of a command. When entered as part of a command line, deletes the line.
CTRL/O	Ignores transmissions to the console terminal until the next CTRL/O is entered. Echoes ^O when disabling output, not echoed when it reenables output. Output is reenabled if the console prints an error message, or if it prompts for a command from the terminal. Output is also enabled by entering console I/O mode, by pressing the BREAK key, and by pressing CTRL/C.

3.8.1 Command Syntax

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 3–10.

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), but symbolic register names contain decimal register numbers. 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

3.8.2 Address Specifiers

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

Physical memory
Virtual 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.

3.8.3 Symbolic Addresses

The console supports symbolic references to addresses. A symbolic reference defines the address space and the offset into that space. Table 3–6 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 3–6: Console Symbolic Addresses

Symb	Addr	Symb	Addr	Symb	Addr	Symb	Addr
/G—General Purpose Registers							
R0	00	R4	04	R8	08	R12 (AP)	0C
R1	01	R5	05	R9	09	R13 (FP)	0D
R2	02	R6	06	R10	0A	R14 (SP)	0E
R3	03	R7	07	R11	0B	R15 (PC)	0F
/M—Processor Status Longword							
PSL	—						
/I—Internal Processor Registers							
pr$_ksp	00	pr$_pcbb	10	pr$_rxcs	20	—	30
pr$_esp	01	pr$_scbb	11	pr$_rxdb	21	—	31
pr$_ssp	02	pr$_ipl	12	pr$_txcs	22	—	32
pr$_usp	03	pr$_astlv	13	pr$_txdb	23	—	33
pr$_isp	04	pr$_sirr	14	pr$_tbidr	24	—	34
—	05	pr$_sisr	15	pr$_cadr	25	—	35
—	06	—	16	pr$_mcesr	26	—	36
—	07	—	17	pr$_mser	27	pr$_ioreset	37
pr$_p0br	08	pr$_iccr	18	pr$_accs	28	pr$_mapen	38
pr$_p0lr	09	—	19	—	29	pr$_tbia	39
pr$_p1br	0A	pr$_icr	1A	pr$_savpc	2A	pr$_tbis	3A
pr$_p1lr	0B	pr$_todr	1B	pr$_savpsl	2B	pr$_tbdata	3B
pr$_sbr	0C	—	1C	—	2C	—	3C
pr$_slr	0D	—	1D	—	2D	—	3D
—	0E	—	1E	—	2E	pr$_sid	3E

Note: All symbolic values in this table are in hexadecimal.

Table 3–6 (Cont.): Console Symbolic Addresses

Symb	Addr	Symb	Addr	Symb	Addr	Symb	Addr
/I—Internal Processor Registers							
—	0F	—	1F	pr$_tbtag	2F	pr$_tbchk	3F
—	70	pr$_brfr	74	pr$_bcerr	78	pr$_pctag	7C
pr$_bcbts	71	pr$_bcidx	75	pr$_bcfbts	79	pr$_pcidx	7D
pr$_bcp1ts	72	pr$_bcsts	76	pr$_bcfpts	7A	pr$_pcerr	7E
pr$_bcp2ts	73	pr$_bcctl	77	pr$_vinstr	7B	pr$_pcsts	7F
/P—Physical (VAX I/O Space)							
qbio	20000000	qbmem	30000000	qbmbr	20080010	—	—
rom	20040000	cacr	20084000	bdr	20084004	—	—
dscr	20080000	dser	20080004	dmear	20080008	dsear	2008000C
ipcr0	20001f40	ipcr1	20001f42	ipcr2	20001f44	ipcr3	20001f46
ssc_ram	20140400	ssc_cr	20140010	ssc_odal	20140020	ssc_dledr	20140030
ssc_ad0mat	20140130	ssc_ad0msk	20140134	ssc_ad1mat	20140140	ssc_ad1msk	20140144
ssc_tcr0	20140100	ssc_tir0	20140104	ssc_tnir0	20140108	ssc_tivr0	2014010c
ssc_tcr1	20140110	ssc_tir1	20140114	ssc_tnir1	20140118	ssc_tivr1	2014011c
memcsr0	20080100	memcsr1	20080104	memcsr2	20080108	memcsr3	2008010c
memcsr4	20080110	memcsr5	20080114	memcsr6	20080118	memcsr7	2008011c
memcsr8	20080120	memcsr9	20080124	memcsr10	20080128	memcsr11	2008012c
memcsr12	20080130	memcsr13	20080134	memcsr14	20080138	memcsr15	2008013c
memcsr16	20080140	memcsr17	20080144	memcsr18	20080148	memcsr19	2008014c
memcsr20	20080150	memcsr21	20080154	memcsr22	20080158	memcsr23	2008015c
memcsr24	20080160	memcsr25	20080164	memcsr26	20080168	memcsr27	2008016c
memcsr28	20080170	memcsr29	20080174	memcsr30	20080178	memcsr31	2008017c
memcsr32	20080180	memcsr33	20080184	memcsr34	20080188	memcsr35	2008018c
memcsr36	20080190						
nicsr0	20008000	nicsr1	20008004	—	20008008	nicsr3	2000800C
nicsr4	20008010	nicsr5	20008014	nicsr6	20008018	nicsr7	2000801C
—	20008020	nicsr9	20008024	nicsr10	20008028	nicsr11	2000802C
nicsr12	20008030	nicsr13	20008034	nicsr14	20008038	nicsr15	2000803C
sgec_setup	20008000	sgec_poll	20008004	—	20008008	sgec_rba	2000800C
sgec_tba	20008010	sgec_status	20008014	sgec_mode	20008018	sgec_sbr	2000801C
—	20008020	sgec_wdt	20008024	sgec_mfc	20008028	sgec_verlo	2000802C
sgec_verhi	20008030	sgec_proc	20008034	sgec_bpt	20008038	sgec_cmd	2000803C
shac1_sswcr	20004030	shac1_sshms	20004044	shac1_pqbbr	20004048	shac1_par	2000404c

Table 3–6 (Cont.): Console Symbolic Addresses

Symb	Addr	Symb	Addr	Symb	Addr	Symb	Addr
			/P—Physical (VAX I/O Space)				
shac1_psar	20004050	shac1_pfar	20004054	shac1_ppr	20004058	shac1_pmcsr	2000405C
shac1_pcq0cr	20004080	shac1_pcq1cr	20004084	shac1_pcq2cr	20004088	shac1_pcq3cr	2000408C
shac1_pdfqcr	20004090	shac1_pmfqcr	20004094	shac1_parcr	20004098	shac1_pscr	2000409C
shac1_pdcr	200040A0	shac1_picr	200040A4	shac1_pmtcr	200040A8	shac1_pmtscr	200040AC
shac2_sswcr	20004230	shac2_sshma	20004244	shac2_pqbbr	20004248	shac2_psr	2000424c
shac2_psar	20004250	shac2_pfar	20004254	shac2_ppr	20004258	shac2_pmcsr	2000425C
shac2_pcq0cr	20004280	shac2_pcq1cr	20004284	shac2_pcq2cr	20004288	shac2_pcq3cr	2000428C
shac2_pdfqcr	20004290	shac2_pmfqcr	20004294	shac2_parcr	20004298	shac2_pscr	2000429C
shac2_pdcr	200042A0	shac2_picr	200042A4	shac2_pmtcr	200042A8	shac2_pmtscr	200042AC
shac_sswcr	20004230	shac_sshma	20004244	shac_pqbbr	20004248	shac_psr	2000424c
shac_psar	20004250	shac_pfar	20004254	shac_ppr	20004258	shac_pmcsr	2000425C
shac_pcq0cr	20004280	shac_pcq1cr	20004284	shac_pcq2cr	20004288	shac_pcq3cr	2000428C
shac_pdfqcr	20004290	shac_pmfqcr	20004294	shac_parcr	20004298	shac_pscr	2000429C
shac_pdcr	200042A0	shac_picr	200042A	shac_pmtcr	200042A8	shac_pmtscr	200042AC

Table 3–7 lists symbolic addresses that you can use in any address space.

Table 3–7: 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.

3.8.4 Console Numeric Expression Radix Specifiers

By default, the console treats any numeric expression used as an address or a datum as a hexadecimal integer. The user may override the default radix by using one of the specifiers listed in Table 3–8.

Table 3–8: Console Radix Specifiers

Form 1	Form 2	Radix
%b	^b	Binary
%o	^o	Octal
%d	^d	Decimal
%x	^x	Hexadecimal, default

For instance, the value 19 is by default hexadecimal, but it may also be represented as %b11001, %o31, %d25, and %x19 (or in the alternate form as ^b11001, ^o31, ^d25, and ^x19).

3.8.5 Console Command Qualifiers

You can enter console command qualifiers in any order on the command line after the command keyword. The three types of qualifiers are data control, address space control, and command specific. Table 3–9 lists and describes the data control and address space control qualifiers. Command specific qualifiers are listed in the descriptions of individual commands.

Table 3–9: Console Command Qualifiers

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 into 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.
/WRONG	Wrong. On writes, 3 is used as the value of the ECC bits, which always generates double bit errors. Ignores ECC errors on main memory reads.
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.

3.8.6 Console Command Keywords

Table 3–10 lists command keywords by type. Table 3–11 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.

You should not use abbreviations in programs. 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 an updated version of the firmware contains new commands or parameters.

Table 3–10: Command Keywords by Type

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

Table 3–11: Console Command Summary

Command	Qualifiers	Argument	Other(s)
BOOT	/R5:{boot_flags} /{boot_flags}	[{boot_device}[,{boot_device}]...]	—
CONFIGURE	—	—	—
CONTINUE	—	—	—
DEPOSIT	/B /W /L /Q — /G /I /V /P /M /U /N:{count} /STEP:{size} /WRONG	{address}	{data} [{data}]
EXAMINE	/B /W /L /Q — /G /I /V /P /M /U /N:{count} /STEP:{size} /WRONG	[{address}]	—
	/INSTRUCTION		
FIND	/MEM /RPB	—	—
HALT	—	—	—
HELP	—	—	—
INITIALIZE	—	—	—
MOVE	/B /W /L /Q — /V /P /U /N:{count} /STEP:{size} /WRONG	{src_address}	{dest_address}
NEXT	—	[{count}]	—
REPEAT	—	{command}	—
SEARCH	/B /W /L /Q — /V /P /U /N:{count} /STEP:{size} /WRONG	{start_address}	{pattern} [{mask}]
	/NOT		
SET BFLAG	—	{bitmap}	—
SET BOOT	—	[{boot_device}[,{boot_device}]]	—
SET CONTROLP	—	{0/1}	—
SET HALT	—	{halt_action}	—

Table 3–11 (Cont.): Console Command Summary

Command	Qualifiers	Argument	Other(s)
SET HOST	/DUP /DSSI /BUS:{0/1}	{node_number}	[{task}]
SET HOST	/DUP /UQSSP {/DISK | /TAPE} /DUP /UQSSP	{controller_number} {csr_address}	[{task}] [{task}]
SET HOST	/MAINTENANCE /UQSSP /SERVICE /MAINTENANCE /UQSSP	{controller_number} {csr_address}	
SET LANGUAGE	—	{language_type}	—
SET RECALL	—	{0/1}	—
SHOW BFLAG	—	—	—
SHOW BOOT	—	—	—
SHOW CONTROLP	—	—	—
SHOW DSSI	—	—	—
SHOW HALT	—	—	—
SHOW LANGUAGE	—	—	—
SHOW MEMORY	/FULL	—	—
SHOW QBUS	—	—	—
SHOW RECALL	—	—	—
SHOW RLV12	—	—	—
SHOW SCSI	—	—	—
SHOW TRANSLATION	—	{phys_address}	—
SHOW UQSSP	—	—	—
SHOW VERSION	—	—	—
START	—	{address}	—
TEST	—	{test_number}	[{parameters}]
UNJAM	—	—	—
X	—	{address}	{count}

3.9 Console Commands

This section describes the console I/O mode commands. Enter the commands at the console I/O mode prompt (>>>).

3.9.1 BOOT

The BOOT command initializes the processor and transfers execution to VMB. VMB attempts to boot the operating system from the specified device or list of devices, 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] [{boot_device},{boot_device},...]

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.

When specifying a list of boot devices (up to 32 characters, with devices separated by commas and no spaces), the system checks the devices in the order specified and boots from the first one that contains bootable software.

NOTE: *If included in a string of boot devices, the Ethernet device, EZA0, should be placed only as the last device of the string. The system will continuously attempt to boot from EZA0.*

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 Ethernet port, EZA0.

Qualifiers:

Command specific:

/R5:{boot_flags} 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.

/{boot_flags} Same as /R5:{boot_flags}

{device_name} A character string of up to 32 characters. Longer strings cause a VAL TOO BIG error message. When specifying a list of boot devices, the device names should be separated by commas and no spaces. Apart from checking the length, the console does not interpret or validate the device name. The console converts the string to uppercase, then passes VMB a string descriptor to this device name in R0. Use the SET BOOT command to specify a default boot device or list of devices. Use the SHOW BOOT command to display the default boot device. The factory default device is the Ethernet port, EZA0. Table 3–5 lists the boot devices supported by the KA670.

Examples:

```
>>>SHOW BOOT
DUA0
>>>SHOW BFLAG
00000000
>>>B    !Boot using default boot flags and device.
(BOOT/R5:0 DUA0)

  2..
-DUA0
```

```
>>>BO XQA0 !Boot using default boot flags and
(BOOT/R5:0 XQA0)              !specified device.

  2..
-XQA0

>>>BOOT I/O !Boot using specified boot flags and
(BOOT/R5:10 DUA0)             !default device.

  2..
-DUA0

>>>BOOT /R5:220 XQA0 !Boot using specified boot
(BOOT/R5:220 XQA0)                    ! flags and device

  2..
-XQA0
```

3.9.2 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 VMS SYSGEN CONFIG 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 console I/O prompt.
2. Enter HELP 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 KA670–AA/BA 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       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? rqdx3,2
Device,Number? dhv11,2
Device,Number? deqna
Device,Number? kfqsa-tape
Device,Number? cxy08
Device,Number? mira
Device,Number? tqk50
Device,Number? tqk70
Device,Number? dhq11
Device,Number? lnv11
Device,Number? exit

Address/Vector Assignments
-774440/120 DEQNA
-772150/154 RQDX3
-760334/300 RQDX3
-774500/260 KFQSA-TAPE
-760444/304 TQK50
-760450/310 TQK70
-760500/320 DHV11
-760520/330 DHV11
-760540/340 CXY08
-760560/350 DHQ11
-776200/360 LNV11
-761260/370 MIRA
>>>!
```

3.9.3 CONTINUE

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. The console enters program I/O mode.

Format:

CONTINUE

Example:

```
>>>CONTINUE
$       !VMS DCL prompt
```

3.9.4 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}, /WRONG

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

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).

Example:

```
>>>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/ST:200 0 8    ! Deposit 8 in the first longword of
                            ! the first 17 pages in physical
                            ! memory.
>>>D/N:200 - 0              ! Starting at previous address, clear
                            ! 513 longwords or 2052 bytes.
```

3.9.5 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. However, the /WRONG qualifier causes EXAMINE to ignore ECC errors on reads from physical memory. 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], /WRONG

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

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:

```
>>>EX PC                              ! Examine the PC.
  G 0000000F FFFFFFFC
>>>EX SP                              ! Examine the SP.
  G 0000000E 00000200
>>>EX PSL                             ! Examine the PSL.
  M 00000000 041F0000
>>>E/M                                ! Examine PSL another way.
  M 00000000 041F0000
>>>E 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

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

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

>>>EX /INS 20040000                   ! Examine 1st byte of ROM.
  P 20040000    11 BRB     20040019

>>>EX /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)

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

>>>
```

3.9.6 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:

/MEMORY 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 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:

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

3.9.7 HALT

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

Format:

HALT

Example:

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

3.9.8 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 specifying a syntactically correct value
..         denotes one of an inclusive range of integers
...        denotes that the previous item may be repeated

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

Valid commands:
    BOOT [[/R5:]<BOOT_FLAGS>] [<BOOT_DEVICE>]
    CONFIGURE
    CONTINUE
    DEPOSIT [<QUALIFIERS>] <ADDRESS> <DATUM> [<DATUM>...]
    EXAMINE [<QUALIFIERS>] [<ADDRESS>]
    FIND [/MEMORY | /RPB]
    HALT
    HELP
    INITIALIZE
    MOVE [<QUALIFIERS>] <ADDRESS> <ADDRESS>
    NEXT [<COUNT>]
    REPEAT
    SEARCH [<QUALIFIERS>] <ADDRESS> <PATTERN> [<MASK>]
    SET BFLG <BOOT_FLAGS>
    SET BOOT <BOOT_DEVICE>
    SET CONTROLP <0..1 |DISABLED|ENABLED>
    SET HALT <0..4 |DEFAULT|RESTART|REBOOT|HALT|RESTART_REBOOT>
    SET HOST/DUP/DSSI/BUS:<0..1> <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 <1..15>
    SET RECALL <0..1 | DISABLED | ENABLED>
    SHOW BFLG
    SHOW BOOT
    SHOW CONTROLP
    SHOW DEVICE
    SHOW DSSI
    SHOW ETHERNET
    SHOW HALT
    SHOW LANGUAGE
    SHOW MEMORY [/FULL]
    SHOW QBUS
    SHOW RECALL
    SHOW RLV12
    SHOW SCSI
```

```
    SHOW TRANSLATION <PHYSICAL_ADDRESS>
    SHOW UQSSP
    SHOW VERSION
    START <ADDRESS>
    TEST [<TEST_CODE> [<PARAMETERS>]]
    UNJAM
    X <ADDRESS> <COUNT>

>>>
```

3.9.9 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
Caches	Flushed
Instruction buffer	Unaffected
Console previous reference	Longword, physical, address 0
TODR	Unaffected
Main memory	Unaffected
General registers	Unaffected
Fault code	Unaffected
Bootstrap-in-progress flag	Unaffected
Internal restart-in-progress flag	Unaffected

The firmware clears all error status bits and initializes the following:

CDAL bus timer
Address decode and match registers
Programmable timer interrupt vectors
SSCCR

Example:

```
>>>INIT
>>>
```

3.9.10 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 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], /WRONG

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:

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

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

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

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

3.9.11 NEXT

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

After the last macro instruction is executed, the console reenters console I/O 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 console enters the "Spacebar Step Mode". In this mode, subsequent spacebar strokes initiate single steps and a carriage return forces a return to the console prompt.

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:

```
>>>DEP 1000 50D650D4                ! Create a simple program.
>>>DEP 1004 125005D1
>>>DEP 1008 00FE11F9
>>>EX /INSTRUCTION /N:5 1000        ! List it.
  P 00001000   D4 CLRL    R0
  P 00001002   D6 INCL    R0
  P 00001004   D1 CMPL    S^#05,R0
  P 00001007   12 BNEQ    00001002
  P 00001009   11 BRB     00001009
  P 0000100B   00 HALT
>>>DEP PR$_SCBB 200                ! Set up a user SCBB...
>>>DEP PC 1000                        ! ...and the PC.
>>>
>>>N                  ! Single step...
  P 00001002   D6 INCL    R0          ! SPACEBAR
  P 00001004   D1 CMPL    S^#05,R0    ! SPACEBAR
  P 00001007   12 BNEQ    00001002    ! SPACEBAR
  P 00001002   D6 INCL    R0          ! CR
>>>N 5                  ! ...or multiple step the program.
  P 00001004   D1 CMPL    S^#05,R0
  P 00001007   12 BNEQ    00001002
  P 00001002   D6 INCL    R0
  P 00001004   D1 CMPL    S^#05,R0
  P 00001007   12 BNEQ    00001002
>>>N 7
  P 00001002   D6 INCL    R0
  P 00001004   D1 CMPL    S^#05,R0
  P 00001007   12 BNEQ    00001002
  P 00001002   D6 INCL    R0
  P 00001004   D1 CMPL    S^#05,R0
  P 00001007   12 BNEQ    00001002
  P 00001009   11 BRB     00001009
>>>N
  P 00001009   11 BRB     00001009
>>>
```

3.9.12 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 EX 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
>>>
```

3.9.13 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 not mask) = (data and not mask), 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}, /WRONG

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 mask of the bits desired in the comparison.

Examples:

```
>>>DEP /P/L/N:1000 0 0                  ! Clear some memory.
>>>
>>>DEP 300 12345678                      ! Deposit some search data.
>>>DEP 401 12345678
>>>DEP 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.  Then try on
>>>SEARCH /N:1000 0 12345678            ! longword boundaries.
  P 00000300 12345678                    ! Search for all non-zero
>>>SEARCH /N:1000 /NOT 0 0              ! longwords.
  P 00000300 12345678
```

```
  P 00000400 34567800
  P 00000404 00000012
  P 00000500 43210000
  P 00000504 00008765
>>>SEARCH /N:1000 /ST:1 0 1 FFFFFFFE    ! Search for odd-numbered
                                         ! longwords on any boundary.
  P 00000502 87654321
  P 00000503 00876543
  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$: brb 10$).
>>>                                     ! Note that none were found.
```

3.9.14 SET

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

Format:

SET {parameter} {value}

Parameters:

BFLAG	Sets the default R5 boot flags. The value must be a hex number of up to eight digits. See Table 3–4 for a list of the boot flags.
BOOT	Sets the default boot device. The value must be a valid device name or list of device names as specified in the BOOT command description in Section 3.9.1.
CONTROLP	Sets Control-P as the console halt condition, instead of a BREAK. Values of 1 or Enabled set Control-P recognition. Values of 0 or Disabled set BREAK recognition. In either case, the setting of the Break Enable /Disable switch.
HALT	Sets the user-defined halt action. Acceptable values are the keywords "default", "restart", "reboot", "halt", "restart_reboot", or a number in the range 0 to 4 inclusive.

HOST

Connects to the DUP or MAINTENANCE driver on the selected node or device. The KA670 DUP driver supports only `send data immediate` messages and those devices that support the messages. It does not support `send data` or `receive data` messages. Note the hierarchy of the SET HOST qualifiers below.

/DUP—Uses the DUP driver to examine or modify parameters of a device on either the DSSI bus or on the Q22-bus.

/BUSn—Selects the desired DSSI bus. A value of 0 selects DSSI bus 0 (internal backplane bus). A value of 1 selects DSSI bus 1 (external console module bus).
/DSSI node—Selects the DSSI node, where "node" is a number from 0 to 7.
/UQSSP—Attaches 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 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 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 the KFQSA EEPROM configuration values. Does not accept a task value.

/UQSSP—

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

RECALL

Sets command recall state to either ENABLED (1) or DISABLED (0).

Qualifiers: Listed in the parameter descriptions above.

Examples:

```
>>>
>>>SET BFLAG 220
>>>
>>>SET BOOT DUA0
>>>
>>>SET HOST/DUP/DSSI 0
Starting DUP server...

DSSI Node 0 (SUSAN)
Copyright © 1990  Digital Equipment Corporation
DRVEXR V1.0  D  5-JUL-1990 15:33:06
DRVTST V1.0  D  5-JUL-1990 15:33:06
HISTRY V1.0  D  5-JUL-1990 15:33:06
ERASE  V1.0  D  5-JUL-1990 15:33:06
PARAMS V1.0  D  5-JUL-1990 15:33:06
DIRECT V1.0  D  5-JUL-1990 15:33:06
End of directory

Task Name?PARAMS
Copyright © 1990  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
    6 PB  FF811ED0  KFQSA  KFX V1.0         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  FF8122DC  BETTY  RFX V101         0      0       0       0
    2 PB  FF8123E0  DSSI1  VMS V5.0         0      0   14328   14328
    3 PB  FF8124E4  3      VMB BOOT         0      0      61      61

PARAMS>EXIT
Exiting...

Task Name?

Stopping DUP server...
>>>
>>>SET HOST/DUP/DSSI/BUS:0 0 PARAMS
Starting DUP server...

DSSI Node 0 (SUSAN)
Copyright © 1990  Digital Equipment Corporation

PARAMS>SHOW NODE

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

```
PARAMS>SHOW ALLCLASS

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

PARAMS>EXIT
Exiting...

Stopping DUP server...
>>>
>>>SET HOST/MAINT/UQSSP 20001468
UQSSP Controller (772150)

Enter SET, CLEAR, SHOW, HELP, EXIT, or QUIT
Node   CSR Address   Model
 0       772150        21
 1       760334        21
 4       760340        21
 5       760344        21
 7     ------ KFQSA ------
? help
Commands:
    SET <node> /KFQSA                        set KFQSA DSSI node number
    SET <node> <CSR_address> <model>         enable a DSSI device
    CLEAR <node>                             disable a DSSI device
    SHOW                                     show current configuration
    HELP                                     print this text
    EXIT                                     program the KFQSA
    QUIT                                     don't program the KFQSA
Parameters:
    <node>                                   0 to 7
    <CSR_address>                            760010 to 777774
    <model>                                  21 (disk) or 22 (tape)
? set 6 /kfqsa
? show
Node   CSR Address   Model
 0       772150        21
 1       760334        21
 4       760340        21
 5       760344        21
 6     ------ KFQSA ------
? exit
Programming the KFQSA...
>>>
>>>SET LANGUAGE 5
>>>
>>>SET HALT RESTART
>>>
```

3.9.15 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.
CONTROLP	Shows the current state of Control-P halt recognition, either Enabled or Disabled.
DEVICE	Displays all devices in the system.
HALT	Shows the user-defined halt action.
DSSI	Shows the status of all nodes that can be found on the DSSI bus. For each node on the DSSI bus, the console displays the node number, the node name, and the boot name and type of the device, if available. The command does not indicate the "bootability" of the device. The node that issues the command reports a node name of "*". The device information is obtained from the media type field of the MSCP command GET UNIT STATUS. In the case where the 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. 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 /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 speculative 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. This command may take several minutes to complete. Press CTRL/C to terminate the command. During execution, the command disables the scatter-gather map.
RECALL	Shows the current state of command recall, either ENABLED or DISABLED.
RLV12	Displays all RL01 and RL02 disks that appear on the Q22-bus.
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.
SCSI	Shows any SCSI devices in the system (TLZ04 or RRD40-series.)
TRANSLATION	Shows any virtual addresses that map to the specified physical address. The firmware uses the current values of page table base and length registers to perform its search; it is assumed that page tables have been properly built.
VERSION	Displays the current firmware version.

Qualifiers: Listed in the parameter descriptions above.

Examples:

```
>>>
>>>SHOW BFLAG
00000220
>>>
>>>SHOW BOOT
DUA0
>>>SHOW CONTROLP
>>>
>>>SHOW DEVICE

KA670-A Vn.n VMBn.n
```

```
DSSI Bus 0 Node 0 (R7CZZC)
-DIA0 (RF71)
DSSI Bus 0 Node 1 (R7ALUC)
-DIA1 (RF71)
DSSI Bus 0 Node 2 (R7EB3C)
-DIA2 (RF71)
DSSI Bus 0 Node 6 (*)
DSSI Bus 1 Node 7 (*)

SCSI Adapter 0 (761300), SCSI ID 7
-DKA100 (DEC TLZ04)

Ethernet Adapter
-EZA0 (08-00-2B-0B-29-14)
>>>
>>>SHOW DSSI
DSSI Bus 0 Node 0 (R7CZZC)
-DIA0 (RF71)
DSSI Bus 0 Node 1 (R7ALUC)
-DIA1 (RF71)
DSSI Bus 0 Node 2 (R7EB3C)
-DIA2 (RF71)
DSSI Bus 0 Node 6 (*)
DSSI Bus 1 Node 7 (*)
>>>
>>>SHOW ETHERNET
Ethernet Adapter
-EZA0 (08-00-2B-0B-29-14)
>>>
>>>SHOW HALT
restart
>>>
>>>SHOW LANGUAGE
English (United States/Canada)
>>>
>>>SHOW MEMORY
Memory 0: 00000000 to 01FFFFFF, 32MB, 0 bad pages
Memory 0: 02000000 to 03FFFFFF, 32MB, 0 bad pages

Total of 64MB, 0 bad pages, 128 reserved pages
>>>
>>>SHOW MEMORY/FULL
Memory 0: 00000000 to 01FFFFFF, 32MB, 0 bad pages
Memory 0: 02000000 to 03FFFFFF, 32MB, 0 bad pages

Total of 64MB, 0 bad pages, 128 reserved pages

Memory Bitmap
-00FF3C00 to 00FF3FFF, 8 pages

Console Scratch Area
-00FF4000 to 00FF7FFF, 32 pages
```

```
Q-bus Map
-0FF8000 to 0FFFFFF, 64 pages

Scan of Bad Pages
>>>

>>>SHOW QBUS
Scan of Qbus I/O Space
-20001920 (774440) = FF08 DELQA/DESQA
-20001922 (774442) = FF00
-20001924 (774444) = FF2B
-20001926 (774446) = FF08
-20001928 (774450) = FFD7
-2000192A (774452) = FF41
-2000192C (774454) = 0000
-2000192E (774456) = 1030
-20001F40 (777500) = 0020 IPCR

Scan of Qbus Memory Space
>>>
>>>SHOW RLV12
>>>
>>>SHOW SCSI
SCSI Adapter 0 (761300), SCSI ID 7
-DKA100 (DEC TLZ04)
>>>
>>>SHOW TRANSLATION 1000
 V 80001000
>>>
>>>SHOW UQSSP
UQSSP Disk Controller 0 (772150)
-DUA0 (RF30)

UQSSP Disk Controller 1 (760334)
-DUA1 (RF30)

UQSSP Disk Controller 2 (760340)
-DUC4 (RF30)

UQSSP Disk Controller 3 (760344)
-DUD5 (RF30)
>>>
>>>
>>>SHOW VERSION
KA670-A Vn.n VMBn.n
>>>
```

3.9.16 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
```

3.9.17 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
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..
```

3.9.18 UNJAM

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

Format:

UNJAM

Example:

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

3.9.19 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 are 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.

You can control the console serial line during a binary unload using control characters (CTRL/C, CTRL/S, CTRL/O, and so on). 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 are not met, then 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.

3.9.20 ! (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

Chapter 4

Troubleshooting and Diagnostics

4.1 Introduction

This chapter contains a description of KA670 ROM-based diagnostics, acceptance test procedures, and power-up self-tests for common options.

4.2 General 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 common problems 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.

If you change the system configuration, run the CONFIGURE utility at the console I/O prompt (>>>) to determine the CSR addresses and interrupt vectors recommended by Digital. These recommended values simplify the use of the MDM diagnostic package and are compatible with VMS device drivers. You can select nonstandard addresses, but they require a special setup for use with VMS drivers and MDM. See the *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.

When troubleshooting, note the status of cables and connectors before you perform each step. Label cables before you disconnect them. This step saves you time and prevents you from introducing new problems.

If the operating system fails to boot (or appears to fail), check the console terminal screen for an error message. If the terminal displays an error message, see Section 4.3. Check the LEDs on the device you suspect is bad. If no errors are indicated by the device LEDs, run the ROM-based diagnostics described in this chapter.

In addition, check the following:

Table 4–1: First Things to Check

Problem	First Steps
No Console Message	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 lit. Check the cabling to the console terminal. Check the terminal setup. Check the power supply status indicators. See Table 4–2.
H3604 Display Off	Check the CPU module LEDs and the H3604 cabling.
H3604 Displays Error	See Table 4–7 to determine error status.

Table 4–2: Power Supply Status Indicators

AC Present	DC OK	Over Temp	Fan Failure	Probable Cause
Off	Off	Off	Off	System not plugged in, AC source not present, or system circuit breaker tripped.
On	Off	Off	Off	Overcurrent or overvoltage protection circuits activated.
On	Off	On	Off	Excessive ambient temp; air vents blocked
On	Off	Off	On	Failure of one or both system fans
On	On	Off	Off	Normal operation

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.

4.3 KA670 ROM-Based Diagnostics

The KA670 ROM-based diagnostic facility is the primary diagnostic tool for troubleshooting and testing of the CPU, memory, Ethernet, 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.

The ROM-based diagnostics can detect failures in field-replaceable units (FRUs) other than the CPU module. For example, they can isolate one of up to four memory modules as FRUs. (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 H3604 display a hexadecimal countdown of the tests from F to 3 (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 4.3.2). There are several field and manufacturing scripts. 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 KA670 is in a manufacturing environment. The diagnostic executive interprets the script to determine what tests to run, the correct order 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 ensures that when the tests are run, the machine is left in a consistent and well-defined state.

4.3.1 Diagnostic Tests

Example 4–1 shows a list of the ROM-based tests and utilities. To get this listing, enter T 3E at the console prompt (T is the abbreviation of TEST). The column headings have the following meanings:

NOTE: *Base addresses shown in this document may not be the same as the addresses you see when you run T 9E. Run T 9E to get a list of actual addresses. See Example 4–3.*

- Test is the test code or utility code.
- Address is an example of the test or utility's base address in ROM. 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. These parameters are encoded in ROM and are provided by the diagnostic executive. Tests accept up to 10 parameters. The asterisks (*) represent parameters that are used by the tests but that you cannot specify individually. These parameters are displayed in error messages, each one preceded by identifiers P1 through P10.

Example 4–1: Test 9E

```
>>>T 9E

Test
#    Address    Name                 Parameters
-----------------------------------------------------------------------
     20050E00   SCB
     20051CF8   De_executive
30   20059B04   Memory_Init_Bitmap   *** mark_Hard_SBEs ******
31   2005992C   Memory_Setup_CSRs    **********
32   200593C4   G_Chip_registers     **********
33   20059338   G_Chip_powerup       **
34   20053354   SSC_ROM              *
35   2006024C   B_Cache_diag_mode    addr_incr wait_time_secs
                                     extended_test *******
36   20061088   B_Cache_w_memory     addr_incr *********
37   20061430   P_B_Cache_w_memory   addr_incr *********
38   20061727   G_Chip_timeout       ******
3F   2005B7D8   Mem_FDM_Addr_shorts *** cont_on_err ******
40   2005C134   Memory_count_pages   First_board Last_bd
                                     Soft_errs_allowed *******
41   2005C30C   Board_Reset          *
42   20053410   Chk_for_Interrupts   *****
44   20060CD8   P_Cache_w_memory     addr_incr *********
45   20057E74   cache_mem_cqbic      start_addr end_addr addr_incr ***
46   2005F978   P_Cache_diag_mode    addr_incr wait_time_secs
                                     extended_test *******
47   2005BEEC   Memory_Refresh       start_a end incr cont_on_err
                                     time_seconds *****
48   2005B410   Memory_Addr_shorts   start_add end_add * cont_on_err
                                     pat2 pat3 ****
49   2005AFB4   Memory_FDM           *** cont_on_err ******
4A   2005AC88   Memory_ECC_SBEs      start_add end_add add_incr
                                     cont_on_err ******
4B   2005A990   Memory_Byte_Errors   start_add end_add add_incr
                                     cont_on_err ******
4C   2005A52C   Memory_ECC_Logic     start_add end_add add_incr
                                     cont_on_err ******
4D   2005A3A4   Memory_Address       start_add end_add add_incr
                                     cont_on_err ******
4E   2005A1DC   Memory_Byte          start_add end_add add_incr
                                     cont_on_err ******
4F   20059F2C   Memory_Data          start_add end_add add_incr
                                     cont_on_err ******
51   200619A1   FPA                  *******
52   20053893   SSC_Prog_timers      which_timer wait_time_us ***
```

Example 4–1 (continued on next page)

Example 4–1 (Cont.): Test 9E

```
53  20053B60  SSC_TOY_Clock       repeat_test_250ms_ea Tolerance ***
54  200534E2  Virtu l_Mode       *********
55  20053D0F  Interval_Timer     *
56  2005DEE4  SHAC_LPBCK         *******
58  2005E6F0  SHAC_RESET         dssi_bus port_number time_secs
59  2005D028  SGEC_LPBCK_ASSIST  time_secs **
5A  20057D78  R_G_Chip_RDAL      dont_report_memory_bad
                                 repeat_count *
5C  2005D590  SHAC               shac_number *******
5F  2005C3DC  SGEC               loopback_type no_ram_tests ******
60  200579AF  SSC_Console_SLU    start_BAUD end_BAUD ******
62  20054144  Console_QDSS       mark_not_present selftest_r0
                                 selftest_r1 *****
63  200542C0  QDSS_any           input_csr selftest_r0
                                 selftest_r1 ******
80  200572F8  CQBIC_memory       **********
81  20053DAB  Qbus_MSCP          iIP_csr ******
82  20053F6D  Qbus_DELQA         device_num_addr ****
83  20054F4E  QZA_LPBCK1         controller_number ********
84  200565E8  QZA_LPBCK2         controller_number *******
85  20054418  QZA_memory         incr test_pattern
                                 controller_number *******
86  2005488C  QZA_DMA            Controller_number
                                 main_mem_buf ********
87  200572B4  QZA_EXTLPBCK       dummy_parameter
90  20053817  CQBIC_registers    *
91  200537B0  CQBIC_powerup      **
99  20061BB6  Flush_Ena_Caches   dis_flush_primary dis_flush_backup
9A  2005EC80  INTERACTION        pass_count disable_device ****
9B  20061875  Init_memory_8MB    *
9C  200581C0  List_CPU_registers *
9D  2005911F  Utility           Expnd_err_msg get_mode init_LEDs
                                 clr_ps_cnt
9E  20053D80  List_diagnostics   *
9F  20061C96  Create_A0_Script   **********
C1  20053005  SSC_RAM_Data       *
C2  200531D8  SSC_RAM_Data_Addr  *
C5  20059236  SSC_registers      *
C6  20052F4C  SSC_powerup        *********

Scripts
#   Description
```

Example 4–1 (continued on next page)

Example 4–1 (Cont.): Test 9E

```
A0  User defined scripts
A1  Powerup tests, Functional Verify, continue on error, numeric
    countdown
A3  Functional Verify, stop on error, test # announcements
A4  Loop on A3 Functional Verify
A5  Address shorts test, run fastest way possible
A6  Memory tests, mark only multiple bit errors
A7  Memory tests
A8  Memory acceptance tests, mark single and multi-bit errors,
    call A7
A9  Memory tests, stop on error
>>>
```

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

```
54  2004FCE9   Virtual mode          *********
30  200518CF   MEM_init_bitmap       *** mark_hard_SBEs ******
```

The virtual mode test on the first line contains several parameters, but you cannot specify any that appear in the table as asterisks. To run this test individually, enter:

```
>>>T 54
```

The MEM_bitmap test on the second line accepts 10 parameters, but you can specify only mark_hard_SBEs because the rest are asterisks. To map out solid, single-bit ECC memory errors, type:

```
>>>T 30 0 0 0 1
```

Even though you cannot change the first three parameters, you need to enter either zeros (0) or ones (1) as placeholders. Zeros are more common and are shown in this example. The zeros hold a place for parameters 1 through 3, which allows the program to parse the command line correctly. The diagnostic executive then provides the proper value for the test.

You enter 1 for parameter 4 to indicate that the test should map out solid, single-bit as well as multibit ECC memory errors. You then terminate the command line by pressing RETURN. You do not need to specify parameters 5 through 10; placeholders are needed only for parameters that precede the user-definable parameter.

4.3.2 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 the EPROM. Other tests are program independent code, and can be run from EPROM 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.

Additional scripts are included in the ROMs for use in manufacturing and engineering environments. Customer Services personnel can run these 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, use the UNJAM and INITIALIZE commands, described in Chapter 3, before running an individual test. You do not need these commands on system power-up, because the system power-up leaves the machine in a defined state.

Customer Services Engineers (CSE) with a detailed knowledge of the KA670 hardware and firmware can also create their own scripts by using the 9F utility. Table 4–3 lists the scripts available to Customer Services.

Table 4–3: Scripts Available to Customer Services

Script[1]	Enter with TEST Command	Description
A0	A0	Runs user-defined script. Enter T 9F to create.
A1	A1, 0	Primary power-up script; builds memory bitmap; marks hard single-bit errors and multibit errors. Continues on error.
A3	A3	Runs power-up tests, but stops on error and prints test numbers instead of countdown pattern.
A4	A4	Loops on A3.
A5	A5	Runs address shorts test from RAM; invokes tests 3F and 48; quicker than executing from ROM.
A6	A6	Similar to power-up memory testing but marks only multibit errors.
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 memory CSRs and resets caches. Also called by the INIT command.
AF	AF	Console program. Resets busmap and resets caches.

[1]Scripts AD, AE, and AF exist primarily for console program; error displays and progress messages are suppressed (not recommended for CSE use).

In most cases, CSE needs only the scripts shown below for effective troubleshooting and acceptance testing.

```
Scripts
#    Description

A0   User defined scripts
A1   Powerup tests, Functional Verify, continue on error, numeric
     countdown
A3   Functional Verify, stop on error, test # announcements
A4   Loop on A3 Functional Verify
A5   Address shorts test, run fastest way possible
A6   Memory tests, mark only multiple bit errors
A7   Memory tests
A8   Memory acceptance tests, mark single and multi-bit errors,
     call A7
A9   Memory tests, stop on error
>>>
```

4.3.3 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. Checks for POK.
2. Establishes SLU as console device.
3. Prints banner message.
4. Displays language inquiry menu on console if console supports multinational character set (MCS) *and* any of the following are true:
   - Battery is dead.
   - Console module switch is set to language inquiry.
   - Contents of SSC RAM are invalid.
5. Calls the diagnostic executive (DE) with Test Code = 0.
   a. DE determines environment is nonmanufacturing from H3604.
   b. DE executes script A1.
   c. DE passes control back to the CP.
6. Issues end message and >>> prompt.

Running Scripts

Some console countdown messages received when running a few scripts are:

```
>>>T A1
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..

>>>T A9
4F..4E..4D..4C..4B..4A..3F..48..48..48..48..48..48..48..48..48..
48..48..48..48..48..48..48..47..40..41..

>>>T A8
31..30..49..4F..4E..4D..4C..4B..4A..3F..48..48..48..48..48..48..
48..48..48..48..48..48..48..48..48..48..47..40..80..41..

>>>T A7
4F..4E..4D..4C..4B..4A..3F..48..48..48..48..48..48..48..48..48..
48..48..48..48..48..48..48..47..40..80..41..
```

4.3.4 Test 9F: 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 4–2). Press RETURN to use the default parameters or flags, which are shown in parentheses. Test 9F prompts you for the following information:

- Script in? The script can be located in the 1-Kbyte SSC 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 SSC RAM.

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

- Test number or script number.

- Modifying existing scripts?

- Run from? The choices are 0 = ROM, 2 = Main Memory, and 3 = Fastest Possible.
- Repeat code? This sets the number of times a test within a script executes before proceeding to the next test.
- Addressing mode? 0 = physical, 1 = virtual.
- Error severity?
- Console error report?
- Stop script on error?
- Repeat?
- LED on entry?
- Console announcement on entry?

Example 4–2 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 test number:` prompt to end the script-building session. Then type T A0 RETURN to run the new script.

You can review or modify (edit) a script you have created. See Example 4–4.

Example 4–2: Creating a Script

```
>>>T 9F

SP=201406A8
Script in ?[0=SSC, 2=RAM] :
Script starts at 20140794
 40 bytes left
Test number (? for list) or script number :51
FPA>> Run from ?[0=ROM, 2=RAM, 3=fastest possible] (0):
FPA>> Addressing mode? [0=physical,1=virtual] (0):
FPA>> Error severity ? [0,1,2,3] (02):
FPA>> Console error report? [0=none,1=full] (01):
FPA>> Stop script on error? [0=NO,1=YES] (01):
FPA>> Repeat? [0=no,1=forever,>1=count<FF] (00):
FPA>> LED on entry (01):
FPA>> Console Announcement on entry (51):
 35 bytes left
Test number (? for list) or script number :
>>>T A0
51..
>>>
```

Example 4–3 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 (40 in Example 4–3) at the `Next test number:` prompt.
2. Enter the UNJAM and INITIALIZE commands.
3. Enter A0 at the `Next test number:` prompt to loop the script indefinitely.
4. Press RETURN at the `Next test number:` prompt.
5. Enter T A0 to begin running the script repeatedly.
6. Press CTRL/C to stop the test.

This sequence 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 4–3: Listing and Repeating Tests with Utility 9F

```
>>>T 9F

SP=201406A8
Script in ?[0=SSC, 2=RAM] :
Script starts at 20140794
Edit existing script ? [0=No, 1=Yes]  (0):
512 bytes left
Test number (? for list) or script number : ?

Test
#   Address    Name                 Parameters
-------------------------------------------------------------------------
    20050E00   SCB
    20051D04   De_executive
30  2005A3AC   Memory_Init_Bitmap   *** mark_Hard_SBEs ******
31  2005A1D4   Memory_Setup_CSRs    **********
32  20059C6C   G_Chip_registers     **********
33  20059BE0   G_Chip_powerup       **
34  20053360   SSC_ROM              *
35  20060AF4   B_Cache_diag_mode    addr_incr wait_time_secs
                                    extended_test *******
36  20061930   B_Cache_w_memory     addr_incr *********
37  20061CD8   P_B_Cache_w_memory   addr_incr *********
38  20061FCF   G_Chip_timeout       ******
3F  2005C080   Mem_FDM_Addr_shorts *** cont_on_err ******
40  2005C9DC   Memory_count_pages   First_board Last_bd
                                    Soft_errs_allowed *******
41  2005CBB4   Board_Reset          *
42  2005341C   Chk_for_Interrupts   *****
44  20061580   P_Cache_w_memory     addr_incr *********
45  2005871C   cache_mem_cqbic      start_addr end_addr
                                    addr_incr *******
46  20060220   P_Cache_diag_mode    addr_incr wait_time_secs
                                    extended_test *******
47  2005C794   Memory_Refresh       start_a end_incr cont_on_err
                                    time_seconds *****
48  2005BCB8   Memory_Addr_shorts   start_add end_add * cont_on_err
                                    pat2 pat3 ****
49  2005B85C   Memory_FDM           *** cont_on_err ******
4A  2005B530   Memory_ECC_SBEs      start_add end_add add_incr
                                    cont_on_err ******
4B  2005B238   Memory_Byte_Errors   start_add end_add add_incr
                                    cont_on_err ******
4C  2005ADD4   Memory_ECC_Logic     start_add end_add add_incr
                                    cont_on_err ******
4D  2005AC4C   Memory_Address       start_add end_add add_incr
```

Example 4–3 (continued on next page)

Example 4–3 (Cont.): Listing and Repeating Tests with Utility 9F

```
                                    cont_on_err ******
4E  2005AA84  Memory_Byte         start_add end_add add_incr
                                    cont_on_err ******
4F  2005A7D4  Memory_Data         start_add end_add add_incr
                                    cont_on_err ******
51  20062249  FPA                 *******
52  200538A2  SSC_Prog_timers     which_timer wait_time_us ***
53  20053B6C  SSC_TOY_Clock       repeat_test_250ms_ea Tolerance ***
54  200534EE  Virtual_Mode        *********
55  20053D1B  Interval_Timer      *
56  2005E78C  SHAC_LPBCK          *******
58  2005EF98  SHAC_RESET          dssi_bus port_number time_secs
59  2005D8D0  SGEC_LPBCK_ASSIST   time_secs **
5A  20058620  R_G_Chip_RDAL       dont_report_memory_bad
                                    repeat_count *
5C  2005DE38  SHAC                shac_number *******
5F  2005CC84  SGEC                loopback_type no_ram_tests ****
60  20058255  SSC_Console_SLU     start_BAUD end_BAUD ******
62  20054150  console_QDSS        mark_not_present selftest_r0
                                    selftest_r1 *****
63  200542CC  QDSS_any            input_csr selftest_r0
                                    selftest_r1 ******
80  20057B9E  CQBIC_memory        **********
81  20053DB7  Qbus_MSCP           IP_csr ******
82  20053F79  Qbus_DELQA          device_num_adc ****
83  20054F5A  QZA_LPBCK1          controller_number ********
84  200565F4  QZA_LPBCK2          controller_number *********
85  20054424  QZA_memory          incr test_pattern
                                    controller_number *******
86  20054898  QZA_DMA             Controller_number main_mem_buf ***
87  2005782C  QZA_EXTLPBCK        controller_number ***
90  20053823  CQBIC_registers     *
91  200537BC  CQBIC_powerup       **
99  2006245E  Flush_Ena_Caches    dis_flush_primary dis_flush_backup
9A  2005F528  INTERACTION         pass_count disable_device ****
9B  2006211D  Init_memory_8MB     *
9C  20058A68  List_CPU_registers  *
9D  200599C7  Utility             Expnd_err_msg get_mode init_LEDs
                                    clr_ps_cnt
9E  20053D8C  List_diagnostics    *
9F  2006253E  Create_A0_Script    **********
C1  20053011  SSC_RAM_Data        *
C2  200531E4  SSC_RAM_Data_Addr   *
C5  20059ADE  SSC_registers       *
C6  20052F58  SSC_powerup         *********
```

Example 4–3 (continued on next page)

Example 4–3 (Cont.): Listing and Repeating Tests with Utility 9F

```
Scripts
#    Description

A0   User defined scripts
A1   Powerup tests, Functional Verify, continue on error, numeric
     countdown
A3   Functional Verify, stop on error, test # announcements
A4   Loop on A3 Functional Verify
A5   Address shorts test, run fastest way possible
A6   Memory tests, mark only multiple bit errors
A7   Memory tests
A8   Memory acceptance tests, mark single and multi-bit errors,
     call A7
A9   Memory tests, stop on error

 512 bytes left
Test number (? for list) or script number :40
Memory_count_pages>> Run from ?[0=ROM, 2=RAM, 3=fastest possible]
(0):
Memory_count_pages>> Error severity ? [0,1,2,3] (02):
Memory_count_pages>> Console error report? [0=none,1=full] (01):
Memory_count_pages>> Stop script on error? [0=NO,1=YES] (01):
Memory_count_pages>> Repeat? [0=no,1=forever,>1=count<FF] (00): 1
Memory_count_pages>> LED on entry (01):
Memory_count_pages>> Console Announcement on entry (40):
Memory_count_pages>> First_board : 00000001 - 00000004 ?(00000001)
Memory_count_pages>> Last_bd : 00000001 - 00000004 ?(00000004)
Memory_count_pages>> Soft_errs_allowed : 00000000 - FFFFFFFF
?(FFFFFFFF) 2
 495 bytes left
Test number (? for list) or script number : A0
>>>T A0
Test number (? for list) or script number: <CR>

40..40..40..40..40..40..40..40..40..40..40..40..40..40..40..40..
40..40..40..40..40..40..40..40..40..40..40..40..40..40..40..40..
40..40..40..40..40..40..40..40..40..40..40..40..40..40..40..40..
^C
>>>
```

4.3.5 Modifying an Existing Script

To modify the script just created, enter the T 9F command and make the changes, as in Example 4–4.

Example 4–4: Modifying an Existing Script

```
>>>T 9F

SP=201406A8
Script in ?[0=SSC, 2=RAM] :
Script starts at 20140794
Edit existing script ? [0=No, 1=Yes] (0):1
 40 bytes left
Test number (? for list) or script number (51) :
FPA>> Run from ?[0=ROM, 2=RAM, 3=fastest possible] (00):2
FPA>> Addressing mode? [0=physical,1=virtual] (00):
FPA>> Error severity ? [0,1,2,3] (00):
FPA>> Console error report? [0=none,1=full] (00):
FPA>> Stop script on error? [0=NO,1=YES] (00):
FPA>> Repeat? [0=no,1=forever,>1=count<FF] (00):
FPA>> LED on entry (01):
FPA>> Console Announcement on entry (51):
 35 bytes left
Test number (? for list) or script number :
>>>T A0
51..
```

In the example above, the only change to the test was to change its location from SSC (default 0) to RAM (2) and to also run the test from RAM. The system will run the script from its last known entry in the script, either SSC or RAM. In other words, if a script has half of its tests located in SSC and half the tests located in RAM, the system will not run all the tests in the script—only those from SSC or from RAM, but not both. In the example above, the last modification of this script was to locate and run the test from RAM, therefore the system will run all the tests located in RAM. The system will run no tests located in SSC.

4.3.6 Console Displays

Example 4–5 shows a typical console display during execution of the ROM-based diagnostics. The numbers on the console display do not refer to actual test numbers. Table 4–7 shows the correspondence between the numbers displayed (listed in the Normal Console Display column) and the actual tests being run (listed in the Error Console Display column).

Example 4–5: Console Display (No Errors)

```
Performing Normal System Tests
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 firmware revision (V5.3 in this example) and the virtual memory bootstrap (VMB) revision (V2.7 in this example).

Diagnostic test failures, if specified in the firmware script, produce an error display in the format shown in Example 4–6.

Example 4–6: Sample Output with Errors

```
KA670-A T2.6, VMB 2.11
Performing normal system tests.
66..65..64..63..62..61..60..59..58..57..56..55..54..53..52..
51..50..

?34 2 08 FF 0000 0000 00        ; SUBTEST_34_08, DE_SSC_ROM.LIS

P1=00000000  P2=0000003C  P3=00000005  P4=00D05570  P5=00000002
P6=00000000  P7=00000000  P8=00000000  P9=00000000 P10=20058A58
r0=FFFFFF00  r1=FFFFFFFF  r2=20066434  r3=55555555  r4=AAAAAAAA
r5=00000000  r6=AAAAAAAA  r7=00000000  r8=0000000C EPC=00000000
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.
>>>
```

Errors are printed in a five-line display. The first line has eight column headings:

```
Test|Severity Error|De_error|Vector Count|Loop_subtest_log|<ascii
msgs>
```

- *Test* identifies the diagnostic test.
- *Severity* is the severity level of a test failure, as dictated by the script. 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.
- *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 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:

Error Code	Description
FF	Normal error exit from diagnostic

Error Code	Description
FE	Unanticipated interrupt
FD	Interrupt in cleanup routine
FC	Interrupt in interrupt handler
FB	Script requirements not met
FA	No such diagnostic
EF	Unanticipated exception in executive

- *Vector* identifies the SCB vector (10 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).
- *Count* is four hex digits. It shows the number of previous errors that have occurred (two in Example 4–6).
- *Loop_subtest_log* is an additional log generated out of the current test specified by the current test number and subtestlog. Usually these logs occur in common subroutines called from a diagnostic test.
- *Ascii messages* contain unique symbols that are terminated by the comma in the ascii field. These symbols identify the most recent subtestlog entry in the listing file. The characters to the right of the comma give the name of the listing file that contains the failed diagnostic.

Lines 2 and 3 of the error printout are parameters 1 through 10. When the diagnostics are running normally, these parameters are the same parameters that are listed in Example 4–1.

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 Table 4–4, Table 4–5, and Table 4–6.

Table 4–4: Machine Check Exception During Executive

Parameter	Value
P1	Contents of stack pointer, points to vector in P2
P2	Vector = 004, machine check
P3	Machine check code

Table 4–4 (Cont.): Machine Check Exception During Executive

Parameter	Value
P4	Contents of VA register
P5	Contents of VIBA register
P6	ICCS register bit <6> and SISR <15:0>
P7	Internal state information
P8	Contents of shift count (SC) register
P9	PC
P10	PSL

Table 4–5: Exception During Executive with No Parameters

Parameter	Value
P1	Contents of stack pointer, points to vector in P2
P2	Vector = nnn (000-3FC), 200 3FC = Q-bus
P3	PC
P4	PSL
P5	Contents of stack
P6	Contents of stack
P7	Contents of stack
P8	Contents of stack
P9	Contents of stack
P10	Contents of stack

Table 4–6: Other Exceptions with Parameters, No Machine Check

Parameter	Value
P1	Contents of stack pointer, points to vector in P2
P2	Vector = nnn (20, 24, 34, 40, 44, 48, 4C, C8)
P3	Optional parameters, could be more than one LW (20, 24, C8)
P4	PC
P5	PSL
P6	Contents of stack
P7	Contents of stack
P8	Contents of stack
P9	Contents of stack
P10	Contents of stack

Lines 4 and 5 of the error printout are general registers R0 through R8 and the error program counter.

In general, the machine check exceptions can provide a clue to the cause of the problem. Machine check codes 01–05, 08–10, 13, 0A, 0B, 0C, and 0D are probably due to CPU fault. Machine check codes 11 and 12 could be a memory problem or a CPU problem. In the case of exceptions with or without parameters (Table 4–5 and Table 4–6) the vector can provide a clue to the fault.

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.

Table 4–7 lists the hex LED display, the default action on errors, and the most likely unit that needs replacing.

The Default 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 4–3). 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 shows the various LED values as they point to problems in field-replaceable units (FRUs).

Table 4–7: KA670 Console Displays As Pointers to FRUs

Hex LED	Normal Console Display	Default on Error	Error Console Display	Description	FRU[1]
Power-Up Tests (Script A1)					
F	None	Loop	None	Power up	5, 1
E	None	Loop	None	Wait for power	5, 1
D	None	Loop	None	–	–
C	66	Cont	?9D	Utility	1
B	65	Cont	?42	Check for interrupts	1, 4
8	64	Cont	?33	G_chip power-up	1
8	63	Cont	?32	G_chip registers	1
8	62	Cont	?31	Memory CSR setup	1, 2
8	61	Halt	?30	Memory bitmap	2, 1
B	60	Cont	?54	Memory management	1
8	59	Cont	?49	Memory counter logic	2, 1, 3
6	58	Cont	?60	Console SLU	1, 6
7	57	Cont	?91	Q-bus interface	1, 4, 3
7	56	Cont	?90	Q-bus interface	1, 4, 3
C	55	Cont	?C6	SSC power-up state	1, 6
C	54	Cont	?52	SSC programmable timer 0	1
C	53	Cont	?52	SSC programmable timer 1	1
C	52	Cont	?53	SSC TOY clock	7, 1
C	51	Cont	?C1	SSC RAM test	1
C	50	Cont	?34	SSC ROM test	1
C	49	Cont	?C5	SSC registers test	1
B	48	Cont	?55	Interval timer	1
8	47	Cont	?38	G_chip timeout counters	1
B	46	Cont	?46	Primary cache test	1
9	45	Cont	?35	Backup cache test	1
8	44	Cont	?4F	Memory data tests	2, 1, 3
8	43	Cont	?4E	Memory byte mask test	2, 1, 3
8	42	Cont	?4D	Memory address tests	2, 1, 3
8	41	Cont	?4C	Memory ECC logic	2, 1, 3

[1]Field-replaceable unit key:

1 = KA670
2 = MS670–BA or MS670–CA
3 = Backplane
4 = Q22 device
5 = System power supply
6 = H3604 I/O panel
7 = Battery

Table 4–7 (Cont.): KA670 Console Displays As Pointers to FRUs

Hex LED	Normal Console Display	Default on Error	Error Console Display	Description	FRU[1]
Power-Up Tests (Script A1)					
8	40	Cont	?4B	Memory masked write test	2, 1, 3
8	39	Cont	?4A	Memory single-bit ECC test	2, 1
8	38	Cont	?3F	Memory address test	2, 1, 3
8	37	Cont	?48[2]	Memory address shorts (0)	2, 1, 3
8	36	Cont	?48	Memory address shorts (1)	2, 1, 3
8	35	Cont	?48	Memory address shorts (2)	2, 1, 3
8	34	Cont	?48	Memory address shorts (3)	2, 1, 3
8	33	Cont	?48	Memory address shorts (4)	2, 1, 3
8	32	Cont	?48	Memory address shorts (5)	2, 1, 3
8	31	Cont	?48	Memory address shorts (6)	2, 1, 3
8	30	Cont	?48	Memory address shorts (7)	2, 1, 3
8	29	Cont	?48	Memory address shorts (8)	2, 1, 3
8	28	Cont	?48	Memory address shorts (9)	2, 1, 3
8	27	Cont	?48	Memory address shorts (10)	2, 1, 3
8	26	Cont	?48	Memory address shorts (11)	2, 1, 3
8	25	Cont	?48	Memory address shorts (12)	2, 1, 3
8	24	Cont	?48	Memory address shorts (13)	2, 1, 3
8	23	Cont	?48	Memory address shorts (14)	2, 1, 3
8	22	Cont	?48	Memory address shorts (15)	2, 1, 3
8	21	Cont	?47	Memory refresh test	2, 1
8	20	Cont	?40	Count bad pages	2
9	19	Cont	?37	P_cache and B_cache/memory interaction	1, 2
8	18	Cont	?44	Primary cache/ memory interaction	1, 2
9	17	Cont	?36	Backup cache/ memory interaction	1, 2
C	16	Cont	?C2	SSC RAM data	1
7	15	Cont	?80	CQBIC interface map registers	1, 2

[1]Field-replaceable unit key:

1 = KA670
2 = MS670–BA or MS670–CA
3 = Backplane
4 = Q22 device
5 = System power supply
6 = H3604 I/O panel
7 = Battery

[2]In the case of multiple FRUs, refer to Section 4.5.3 for further information.

Table 4–7 (Cont.): KA670 Console Displays As Pointers to FRUs

Hex LED	Normal Console Display	Default on Error	Error Console Display	Description	FRU[1]
Power-Up Tests (Script A1)					
9	14	Cont	?45	Cache/CQBIC interaction	1, 2
8	13	Cont	?5A	CPU test	1
A	12	Cont	?51	Floating-point test	1
4	11	Cont	?5F	NI functional test	1, 6
5	10	Cont	?5C	DSSI functional test, bus 1	1, 3
5	9	Cont	?5C	DSSI functional test, bus 0	1, 6
8	8	Cont	?9A	Interaction test	1, 2, 3, 4
0	7	Cont	?83	QZA loopback	4
0	6	Cont	?84	QZA loopback	4
0	5	Cont	?85	QZA memory	4
0	4	Cont	?86	QZA DMA	4
C	3	Cont	?41	Board reset	1, 4
Script A5					
8	3F	Halt	?3F	Memory address tests	2, 1, 3
8	48	Halt	?48[2]	Memory address shorts test	2, 1, 3
Script A6					
8	30	Halt	?30	Memory bitmap	2, 1, 3
8	4F	Halt	?4F	Memory data test	2, 1, 3
8	4E	Halt	?4E	Memory byte mask test	2, 1, 3
8	4D	Halt	?4D	Memory address test	2, 1, 3
8	4C	Halt	?4C	Memory ECC logic	2, 1, 3
8	4B	Halt	?4B	Memory masked write test	2, 1, 3
8	4A	Halt	?4A	Memory SBE test	2, 1, 3
8	3F	Halt	?3F	Memory address test	2, 1, 3

[1]Field-replaceable unit key:

1 = KA670
2 = MS670–BA or MS670–CA
3 = Backplane
4 = Q22 device
5 = System power supply
6 = H3604 I/O panel
7 = Battery

[2]In the case of multiple FRUs, refer to Section 4.5.3 for further information.

Table 4–7 (Cont.): KA670 Console Displays As Pointers to FRUs

Hex LED	Normal Console Display	Default on Error	Error Console Display	Description	FRU[1]
Script A6					
8	48	Halt	?48[2]	Memory address shorts	2, 1, 3
8	47	Halt	?47	Memory refresh test	2, 1, 3
8	40	Halt	?40	Count bad pages	2, 1, 3
7	80	Halt	?80	Q-bus interface map registers	2, 1, 3
Script A8					
8	31	Halt	?31	Memory csr setup	2, 1, 3
8	30	Halt	?30	Memory bitmap	2, 1, 3
8	49	Halt	?49	Memory counter logic	2, 1, 3
Invoke script A7.					
Script A7					
8	4F	Halt	?4F	Memory data tests	2, 1, 3
8	4E	Halt	?4E	Memory byte mask test	2, 1, 3
8	4D	Halt	?4D	Memory address tests	2, 1, 3
8	4C	Halt	?4C	Memory ECC logic	2, 1, 3
8	4B	Halt	?4B	Memory masked write test	2, 1, 3
8	4A	Halt	?4A	Memory SBE test	2, 1, 3
8	3F	CONT	?3F	Memory address test	2, 1, 3
8	48	Halt	?48[2]	Memory address shorts test	2, 1, 3
8	47	Halt	?47	Memory refresh test	2, 1, 3
8	40	Halt	?40	Count bad pages	2, 1, 3
7	80	Halt	?80	Q-bus interface map register	2, 1, 3
C	41	Halt	?41	Board reset	2, 1, 3

[1]Field-replaceable unit key

1 = KA670
2 = MS670–BA or MS670–CA
3 = Backplane
4 = Q22 device
5 = System power supply
6 = H3604 I/O panel
7 = Battery

[2]In the case of multiple FRUs, refer to Section 4.5.3 for further information.

Table 4–7 (Cont.): KA670 Console Displays As Pointers to FRUs

Hex LED	Normal Console Display	Default on Error	Error Console Display	Description	FRU[1]
Script A9					
8	4F	Halt	?4F	Memory data tests	2, 1, 3
8	4E	Halt	?4E	Memory byte mask test	2, 1, 3
8	4D	Halt	?4D	Memory address test	2, 1, 3
8	4C	Halt	?4C	Memory ECC logic	2, 1, 3
8	4B	Halt	?4B	Memory masked write test	2, 1, 3
8	4A	Halt	?4A	Memory SBE test	2, 1, 3
8	48[2]	Halt	?48	Memory address shorts test	2, 1, 3
8	47	Halt	?47	Memory refresh test	2, 1, 3
8	40	Halt	?40	Count bad pages	2, 1, 3
C	41	Halt	?41	Board reset	2, 1, 3
End of script					

[1]Field-replaceable unit key:

1 = KA670
2 = MS670–BA or MS670–CA
3 = Backplane
4 = Q22 device
5 = System power supply
6 = H3604 I/O panel
7 = Battery

[2]In the case of multiple FRUs, refer to Section 4.5.3 for further information.

4.4 Acceptance Testing

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

KA670 module
MS670 memory array
Backplane
DSSI device
H3604

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

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

Press CTRL/C to terminate the scripts. Refer to Section 4.5 if failures occur.

2. Double-check the memory configuration, since test 31 can check for only a few invalid configurations. For example, test 31 cannot report that a memory board is missing from the configuration, since it has no way of knowing if the board should be there or not.

   To check the memory configuration, enter the following command line:

```
>>>SHOW MEMORY/FULL
Memory 0: 00000000 to 01FFFFFF, 32 Mbytes, 0 bad pages

Total of 32 Mbytes, 0 bad pages, 112 reserved pages

Memory Bitmap
-01FF2000 to 01FF3FFF, 16 pages

Console Scratch Area
-01FF4000 to 01FF7FFF, 32 pages

Qbus Map
-01FF8000 to 01FFFFFF, 64 pages

Scan of Bad Pages

>>>
```

   Memories 0 through 3 are the MS670 memory modules. The Q22-bus map always spans the top 32 Kbytes of good memory. The memory bitmap always spans two pages (1 Kbyte) for each 4 Mbytes of memory configured.

   Use utility 9C to compare the contents of configuration registers MCSR –15 with the memory installed in the system:

```
>>>T 9C
```

```
MEM_FRU 3       MCSR_8=00000006      9=00000006      10=00000006      11=00000006
MEM_FRU 4       MCSR12=00000006     13=00000006      14=00000006      15=00000006
RMESR=00440044  RMEAR=00000000 RIOEAR=00080188   CEAR=00000000    MCDSR=152A1700
>>>
```

To identify registers and register bit fields, see the *KA670 CPU Technical Manual*.

Examine MCSR 0–15 to verify the memory configuration. One memory bank is enabled for each 4 Mbytes of memory. The MCSRs map the modules as follows:

MCSR0–3	First MS670; slot 4, closest to CPU
MCSR4–7	Second MS670; slot 3
MCSR8–11	Third MS670; slot 2
MCSR12–15	Fourth MS670; slot 1, farthest from CPU

Verify the following:

- The bank enable bit (<31>) in all four MCSRs for each memory module is set to (8xxx xxxh), which indicates that the base address for the banks contained on the module is valid.
- MCSR bits <2:1> are the signature field and contain the following value, in relation to the size of the array.

Table 4–8: Signature Field Values

MCSR 0–15 <2:1>	Hex Equiv	Configuration
00	0	Unassigned
01	2	RAM size 1 Mbit
10	4	RAM size 4 Mbits
11	6	Bank no response

- MCSR bits <28:23> increment starting at zero in any group of four MCSRs whose bit <31> is set.
- MCSRs display 0000 0006 if no array is present; there should be no gaps in the memory configuration.

3. Check the Q22-bus and the Q22-bus logic in the KA670 CQBIC chip and the configuration of the Q22-bus, as follows:

```
>>>SHOW QBUS
Scan of Qbus I/O Space
-200000DC (760334)=0000 RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000DE (760336)=0AA0
-20001468 (772150)=0000 RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-2000146A (772152)=0AA0
-20001920 (774440)=FF08 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 TQK50/TQK70/TU81E/RV20/KFQSA-TAPE
-20001942 (774502)=0BC0
-20001F40 (777500)=0020 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 speculated device name (IPCR, the KA670 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. This action may be useful to isolate a problem with a controller, the KA670, or the backplane. Use the CVAX 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

KA670-A Vn.n VMBn.n

DSSI Bus 0 Node 7 (*)

DSSI Bus 1 Node 0 (CLAUDE)
-DIB0 (RF71)

DSSI Bus 1 Node 1 (BEANIE)
-DIB1 (RF71)

DSSI Bus 1 Node 4 (*)

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

Ethernet Adapter
-EZA0 (08-00-2B-06-10-42)

Ethernet Adapter 0 (774440)
-XQA0 (08-00-2B-08-D7-41)
```

In the example, the console displays the node numbers of two RF71 controllers it recognizes. The line below each node name and number is the logical unit number of any attached devices, DIB0 and DIB1 in this case.

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 KA670 Ethernet adapter.

The last two lines refer to a DESQA Ethernet controller, its Q22-bus CSR address, its logical name (XQA0), and its station address.

6. 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 the operating system software can be loaded.

NOTE: *In the past, the default action of the ROM-based tests was to check for and mark only double-bit errors. The KA670 checks for and marks single-bit errors by default. Therefore, it is not necessary to run the tests for single-bit errors.*

4.5 Troubleshooting

This section contains suggestions for determining the cause of ROM-based diagnostic test failures.

4.5.1 FE Utility

If any of the tests fail, the major test code is displayed on the LEDs. You may also be able to acquire more information with the FE utility.

Running the FE utility is useful if the message, `Normal operation not possible`, is displayed after the tests have ompleted 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 diagnostic state to the console (Example 4–7). This state indicates the major and minor test code of the test that failed, the 10 parameters associated with the test, and the hardware error summary register.

Example 4–7: FE Utility Example

```
>>>T FE

 Bitmap=01FF2000, Length=00002000, Checksum=FFFF, Busmap=01FF8000
 Test_number=41, Subtest=00, Loop_Subtest=00, Error_type=00
 Error_vector=0000, Last_exception_PC=00000000, Severity=02
 Total_error_count=0005, Led_display=0C, Console_display=03,
 save_mchk_code=11
 parameter_1=00000000  2=00000000  3=00000000  4=00000000
 5=00000000
 parameter_6=00000000  7=00000000  8=00000002  9=0000080A
 10=00000003
 previous_error=000E5F02, FE02 100, 001B5F02, 00064200
 Flags=0FFFFC00400E
 Return_stack=201406A4, Subtest_pc=2005988C, Timeout=00030D40

>>>
```

The most useful fields displayed above are as follows:

- Error_vector, which is the SCB vector through which the unexpected interrupt or exception trapped if de_error equals FE or EF.

- **Total_error_count. Four hex digits showing the number of previous errors that have occurred.**
- **Parameters 1 through 10. Valid only if the test halts on error.**
- **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). (00=FF in the de_error.)**
- **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.**

4.5.2 Overriding Halt Protection

The ROM-based diagnostics run in halt-protected space. When you want to halt diagnostic execution, if the diagnostic program hangs during execution or if the runtime of the diagnostic program is so long you want to suspend it, enter the following commands:

```
>>>E 20140010       !Examine the SSCCR
P 20140010   00D55570

>>>D * 00D05570      !Clear halt-protected space
>>>T 0         !Tests can now be halted
```

This state is in effect only until the first break or a restart.

4.5.3 Isolating Memory Failures

This section describes procedures for isolating memory subsystem failures, particularly when the system contains more than one MS670–BA and/or MS670–CA memory module.

1. SHOW MEMORY/FULL

   Use the SHOW MEMORY/FULL command to examine failures detected by the memory tests. Use this command if test 40 fails, which indicates that pages have been marked bad in the bitmap.

   You can also use SHOW MEMORY/FULL after terminating a script that is taking an unusually long time to run. Press CTRL/C to terminate the script after the completion of the current test. (CTRL/C on the KA670 console takes effect only after the entire script completes.) After terminating the script, enter SHOW MEMORY/FULL to see if the tests have marked any pages bad up to that point. See Section 4.4 for an example of this command.

2. T A9

```
>>>T [memory test] starting_board_number ending_board_number
    adr_incr
```

Script A9 runs only the memory tests and halts on the first error detected. Unlike the power-up script, it does not continue. Since the script does not rerun the test, it detects memory-related failures that are not hard errors. You should then run the individual test that failed on each memory module one MS670 module at a time. You can input parameters 1 and 2 of tests 40, 47, 48, and 4A through 4F as the starting and ending address for testing. It is easier, however, to input the memory module numbers 1–4. For example, if test 4F fails, test the second memory module as follows:

```
>>>T 4F 2 2
```

If a failure is detected for a second of three MS670–BA modules, for example, repeat this procedure for all memory modules to isolate the MS670 module that is the FRU, using the process of elimination.

You can also specify the address increment. For example, to test the third memory module on each page boundary, type:

```
>>>T 4F 3 3 200
```

Only the numbers 1–4 are FRU numbers. Any other values are assumed to be starting and ending addresses. Out of range values are rejected by the diagnostic. If a memory specified in the test command is not physically present, the test fails. See Table 4–9.

Table 4–9: Memory Module as FRU

MCSR No.	Test FRU Input No.	Show Memory Identification
MCSR0–3	1	Memory 0
MCSR4–7	2	Memory 1
MCSR8–11	3	Memory 2
MCSR12–15	4	Memory 3

By default, most memory tests test one longword on a 256-Kbyte boundary. If an error is detected, the tests start testing on a page boundary. Test 48 (address shorts test) is an exception: it checks every location in memory since it can do so in a reasonable amount of time. Other tests, such as 4F (floating ones and zeros test), can take up to

one hour, depending on the amount of memory, if each location were to be tested. If you do specify an address increment, do not input less than 200 (testing on a page boundary), since almost all hard memory failures span at least one page. For normal servicing, do not specify the address increment, since it adds unnecessary repair time; most memory subsystem failures can be found using the default parameter.

All memory tests, except for 40, save the MEMCSR36 (MCDSR), MEMCSR32 (RMESR), MEMCSR33 (RMEAR) in parameters 7, 8, and 9, respectively.

3. T 9C

   The utility 9C is useful if test 31 or some other memory test failed because memory was not configured correctly.

   To help in isolating an FRU, examine registers MCSR 0–15 by entering T 9C at the console I/O mode prompt.

4. T 40

   Although the SHOW MEMORY/FULL command displays pages that are marked bad by the memory test and is easier to interpret than test 40, there is one instance in which test 40 reports information that SHOW MEMORY/FULL does not report. You can use test 40 as an alternative to running script A9 to detect soft memory errors. Specify the third parameter in test 40 (see Table 4–7) to be the threshold for soft errors. To allow 0 (zero) errors, enter the following:

   ```
   >>>T 40 1 4 0
   ```

   This command tests the memory on four memory modules. Use it after running memory tests individually or within a script. If test 40 fails with subtestlog = 6, examine R5–R8 to determine how many errors have been detected on the CPU memory and the three memory modules, respectively.

4.5.4 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 KA670 module, backplane, or MS670 modules as the cause of failure.

Always check the seating of the module before replacing it. If the seating appears to be improper, rerun the tests.

If you are rotating MS670 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 that you map out both solid single-bit and multibit ECC failures as shown in step 2 of acceptance testing (Section 4.4), since in one slot a board may fail most frequently with multibit ECC failures, and in another slot with single-bit ECC failures.

Be sure to put the modules back in their original positions when you are finished.

If memory errors are found in the operating system error log, use the KA670 ROM-based diagnostics to see if it is an MS670 problem or if it is related to the KA670 or backplane. Follow steps 1–3 of Section 4.4 and Section 4.5.3 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 CONFIGURE command to help with configuration problems or when installing new options onto the Q-bus. See the command descriptions in Chapter 3.

4.6 Loopback Tests

You can use external loopback tests to localize problems with the console.

4.6.1 Testing the Console Port

To test the console port at power-up, set the power mode switch on the H3604 to the test position and install an H3103 loopback connector into the MMJ of the H3604. The H3103 connects the console port transmit and receive lines. At power-up, the SLU_EXT_LOOPBACK test then runs a continuous loopback test.

While the test is running, the LED display on the CPU I/O panel should alternate between 6 and 3. A value of 6 latched in the display indicates a test failure. If the test fails, one of the following parts is faulty: the KA670, the H3604, the cabling, or the CPU module.

To test out to the end of the console terminal cable:

1. Plug the MMP end of the console terminal cable into the H3604.
2. Disconnect the other end of the cable from the terminal.
3. Place an H8572 adapter into the disconnected end of the cable.
4. Connect the H3103 to the H8572.
5. Cycle power and observe LEDs.

4.6.2 SHAC Loopback Testing: 56 and 58

Test 56 tests both SHAC chips (the DSSI adapters). This test can be used to check both SHAC chips, the internal DSSI (bus 0) connectivity, external DSSI cables, and the H3604 DSSI bus interconnect. Complete the following procedures before running test 56.

1. Make sure the system is powered down, then connect DSSI bus 0 to DSSI bus 1 with a standard external DSSI cable (BC21M–09). Place a DSSI terminator on the remaining DSSI connector for bus 1. It is not critical which bus 1 connector is used in connecting the cable.
2. Remove all bus node ID plugs from ISEs on bus 0.
3. Install bus node ID plugs on the console module (H3604) so that bus 0 and bus 1 do not have the same bus node ID. For example, assign bus node ID #6 to bus 0 and bus node ID #7 to bus 1.
4. Power up the system. Note that the red fault light on the ISE front panels will remain lit. This is normal when the bus node ID plugs have been removed.
5. Run test 56. When tests have successfully completed, the console prompt is displayed.

```
>>>T 56
>>>
```

This loopback test is useful for isolating DSSI problems. A probable FRU list in order of priority follows:

1. The external BC21M–09 cable
2. The Vterm dual regulator module (PN 54–20404–01)
3. The internal cable that connects DSSI bus 0 from the backplane to the edge of the enclosure (PN 17–02502–01)
4. The internal cable that connects the CPU to the H3604 (PN 17–02353–01)
5. The 2.0 A Pico fuse on the H3604 (PN 12–10929–06)
6. The KA670

NOTE: *The DSSI bus must be terminated for the tests to execute successfully. Vterm power for DSSI bus 0 is supplied by the Vterm regulator module, which plugs into the BA440 backplane. There are no fuses on this module.*

NOTE: *Loopback tests do not test for termination power. Use the following procedure to check termination power:*

Remove the external DSSI cable and terminate busses 0 and 1. Check the terminator LEDs to see if termination power is present.

- *No termination power at bus 0 indicates a possible problem with the internal cable (PN 17–02502–01) that connects DSSI bus 0 from the backplane.*
- *No termination power at bus 1 indicates a possible problem with the Pico fuse (PN 12–10929–06) on the H3604 console module or the power harness module (PN 54–19789–01) for the console module.*

Test 58 is a SHAC and ISE reset and can be used to verify that ISEs can be accessed on the DSSI storage bus. Test 58 causes data packets to be passed between the ISEs and the adapters, verifying that the ISEs are accessible.

Enter T 58 and specify DSSI bus (0 or 1) and the DSSI node ID of the ISE to be tested.

```
>>>T 58 0 5
```

In the example above, bus 0 node 5 was tested. (Each ISE has to be tested separately.)

4.6.3 SGEC Loopback Testing: 5F and 59

Test 5F is the internal loopback test for SGEC (Ethernet controller).

```
>>>T 5F
```

For an external SGEC loopback, enter "1".

```
>>>T 5F 1
```

Before running test 5F on the ThinWire Ethernet port, connect an H8223 T-connector with two H8225 terminators.

Before running test 5F on the standard Ethernet port, you must have a 12–22196–02 loopback connector installed.

T 59 polls other nodes on Ethernet to verify SGEC functionality. The Ethernet cable must be connected to a functioning Ethernet. A series of MOP messages are generated; look for response messages from other nodes.

```
>>>T 59
```

4.7 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.
- 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:

DFA01
DPV11
DRQ3B
DZQ11
KLESI
LPV11
TSV05

The following modules have one green LED, which indicates that the module is receiving +5 and +12 Vdc and has passed self-tests:

CXA16
CXB16
CXY08

Table 4–10 lists loopback connectors for common devices. Refer to *Microsystems Options* for a description of specific module self-tests.

Table 4–10: Loopback Connectors for Common Devices

Device	Module Loopback	Cable Loopback
CXA16/CXB16	H3103 + H8572[1]	
CXY08	H3046 (50-pin)	H3197 (25-pin)
DPV11	H3259	H3260
DZQ11	12–15336–00 or H325	H329 (12–27351–01)
Ethernet[2]	–	–
LPV11	None	None
KA670/H3604	H3103	H3103 + H8572
KMV1A	H3255	H3251

[1]Use the appropriate cable to connect transmit-to-receive lines. H3101 and H3103 are double-ended cable connectors.

[2]For ThinWire, use H8223–00 plus two H8225–00 terminators. For standard Ethernet, use 12–22196–02.

4.8 RF-Series ISE Troubleshooting and Diagnostics

An RF-series 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 enclosure's OCP. 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 two types of error conditions in the drive:

- *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. The red fault LED lights. If this occurs, replace the drive module.
- *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.

If the ISE is connected to the OCP, you must install a unit ID plug in the corresponding socket on the OCP. If the ISE is not connected to the OCP, it reads the unit ID from the three-switch DIP switch 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 does 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 console 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 KA670 console.

To abort or prematurely terminate a program and return control to the KA670 console, press CTRL/C or CTRL/Y.

4.8.1 DRVTST

DRVTST is a comprehensive functionality test. Errors detected by this test are isolated to the FRU level. The messages are listed in Table 4–11.

Table 4–11: DRVTST Messages

Message Type	Message
I	Copyright © 1991 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	5 minutes to complete.
T	Test passed

Table 4–11 (Cont.): DRVTST Messages

Message Type	Message
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...?") results in a read-only test. DRVTST, however, writes to a diagnostic area on the disk. Answering Yes to the first question causes the 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.

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.

4.8.2 DRVEXR

The DRVEXR local program exercises the ISE. The test is data transfer intensive and indicates the overall integrity of the device. Table 4–12 lists the DRVEXR messages.

Table 4–12: 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 code 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 you press the Write-Protect switch on the OCP (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. First, DRVTST always uses a fixed time of 5 minutes, while you specify the time of the DRVEXR routine. Second, DRVTST determines whether the drive is good or bad. DRVEXR reports the data but does not determine the condition of the drive.

4.8.3 HISTRY

The HISTRY local program displays information about the history of the ISE. Table 4–13 lists the HISTRY messages.

Table 4–13: 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]This displays the last 11 fault codes as information 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 © 1989 Digital Equipment Corporation
RF71
EN01082
SUSAN
0
RFX V101
RF71 PCB-5/ECO-00
617
21
A04F
A0'F
A103
A04F
A404
A04F
A404
A04F
A404
A04F
A404
Complete.
```

If no errors have been logged, no hexadecimal fault codes are displayed.

4.8.4 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 4–14 lists the ERASE messages:

Table 4–14: ERASE Messages

Message Type	Message
I	Copyright © 1989 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 revector to.
0010 : The RAM copy of the bad block table is full.

If a failure is detected, the message that indicates the failure is followed by one or more messages that contain error codes.

4.8.5 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

4.8.5.1 EXIT

Use the EXIT command to terminate the PARAMS local program.

4.8.5.2 HELP

Use the HELP command to display a brief list of available PARAMS commands, as shown in the following example:

```
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>
```

4.8.5.3 SET

Use the SET command to change the value of a given parameter. To abbreviate, use the first matching parameter without regard to uniqueness.

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 through H that corresponds to the DSSI bus ID) instead of using the NODENAME value. False (0) if NODENAME is to be used.

4.8.5.4 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.

4.8.5.5 STATUS

Use the STATUS command to display module configuration, history, or current counters, depending on the type specified. The type is the optional ASCII string that denotes the type of display desired. If you omit the type, all available status information is displayed. If present, the type may be abbreviated. The following types are available.

CONFIG	Displays the module name, node name, power-on hours, power cycles, and other such configuration information. Unit failures are also displayed, if applicable.
LOGS	Displays the last eleven 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.

4.8.5.6 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.

4.9 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. Some error codes are listed in Table 4–15. If you see any error code other than those listed in Table 4–15, replace the module. For a complete list of error codes refer to the device's user guide. Appendix H provides a listing of related documents.

Table 4–15: KF-Series ISE

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 AC Present 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.


APPENDIX A

Appendix A
Address Assignments

A.1 KA670 General Local Address Space Map

Address Range	Contents
VAX Memory Space	
0000 0000 to 1FFF FFFF	Local memory space (512 Mbytes)
VAX I/O Space	
2000 0000 to 2000 1FFF	Local Q22-bus I/O space (8 Kbytes)
2000 2000 to 2003 FFFF	Reserved local I/O space (248 Kbytes)
2004 0000 to 2007 FFFF	Local UVROM space
2008 0000 to 201F FFFF	Local register I/O space (1.5 Mbytes)
2020 0000 to 23FF FFFF	Reserved local I/O space (62.5 Mbytes)
2400 0000 to 27FF FFFF	Reserved local I/O space (64 Mbytes)
2008 0000 to 2BFF FFFF	Reserved local I/O space (64 Mbytes)
2C08 0000 to 2FFF FFFF	Reserved local I/O space (64 Mbytes)
3000 0000 to 303F FFFF	Local Q22-bus memory space (4 Mbytes)
3040 0000 to 33FF FFFF	Reserved local I/O space (60 Mbytes)
3400 0000 to 37FF FFFF	Reserved local I/O space (64 Mbytes)
3800 0000 to 3BFF FFFF	Reserved local I/O space (64 Mbytes)
3C00 0000 to 3FFF FFFF	Reserved local I/O space (64 Mbytes)

A.2 KA670 Detailed Local Address Space Map

Contents	Address
Local memory space (up to 512 Mbytes)	0000 0000 to 1FFF FFFF
Q22-bus map—top 32 Kbytes of main memory	

VAX I/O Space

Contents	Address
Local Q22-bus I/O Space	**2000 0000 to 2000 1FFF**
Reserved Q22-bus I/O space	2000 0000 to 2000 0007
Q22-bus floating address space	2000 0008 to 2000 07FF
User-reserved Q22-bus I/O space	2000 0800 to 2000 0FFF
Reserved Q22-bus I/O space	2000 1000 to 2000 1F3F
Interprocessor communication register	2000 1F40
Reserved Q22-bus I/O space	2000 1F44 to 2000 1FFF
Local Register I/O Space	**2000 2000 to 2003 FFFF**
Reserved local register I/O space	2000 4000 to 2000 402F
SHAC1 SSWCR	2000 4030
Reserved local register I/O space	2000 4034 to 2000 4043
SHAC1 SSHMA	2000 4044
SHAC1 PQBBR	2000 4048
SHAC1 PSR	2000 404C
SHAC1 PESR	2000 4050
SHAC1 PFAR	2000 4054
SHAC1 PPR	2000 4058
SHAC1 PMCSR	2000 405C
Reserved local register I/O space	2000 4060 to 2000 407F
SHAC1 PCQ0CR	2000 4080
SHAC1 PCQ1CR	2000 4084
SHAC1 PCQ2CR	2000 4088
SHAC1 PCQ3CR	2000 408C
SHAC1 PDFQCR	2000 4090
SHAC1 PMFQCR	2000 4094
SHAC1 PSRCR	2000 4098

VAX I/O Space	
Contents	**Address**
Local Register I/O Space	**2000 2000 to 2003 FFFF**
SHAC1 PECR	2000 409C
SHAC1 PDCR	2000 40A0
SHAC1 PICR	2000 40A4
SHAC1 PMTCR	2000 40A8
SHAC1 PMTECR	2000 40AC
Reserved local register I/O space	2000 40B0 to 2000 422F
SHAC2 SSWCR	2000 4230
Reserved local register I/O space	2000 4234 to 2000 4243
SHAC2 SSHMA	2000 4244
SHAC2 PQBBR	2000 4248
SHAC2 PSR	2000 424C
SHAC2 PESR	2000 4250
SHAC2 PFAR	2000 4254
SHAC2 PPR	2000 4258
SHAC2 PMCSR	2000 425C
Reserved local register I/O space	2000 4260 to 2000 427F
SHAC2 PCQ0CR	2000 4280
SHAC2 PCQ1CR	2000 4284
SHAC2 PCQ2CR	2000 4288
SHAC2 PCQ3CR	2000 428C
SHAC2 PDFQCR	2000 4290
SHAC2 PMFQCR	2000 4294
SHAC2 PSRCR	2000 4298
SHAC2 PECR	2000 429C
SHAC2 PDCR	2000 42A0
SHAC2 PICR	2000 42A4
SHAC2 PMTCR	2000 42A8
SHAC2 PMTECR	2000 42AC
Reserved local register I/O space	2000 42B0 to 2000 7FFF
NICSR0—Vector add, IPL, sync/async	2000 8000
NICSR1—Polling demand register	2000 8004
NICSR2—Reserved	2000 8008
NICSR3—Receiver list address	2000 800C
NICSR4—Transmitter list address	2000 8010
NICSR5—Status register	2000 8014

VAX I/O Space

Contents	Address
Local Register I/O Space	**2000 2000 to 2003 FFFF**
NICSR6—Command and mode register	2000 8018
NICSR7—System base address	2000 801C
NICSR8—Reserved	2000 8020*
NICSR9—Watchdog timers	2000 8024*
NICSR10—Reserved	2000 8028*
NICSR11—Revision number and missed frame count	2000 802C*
NICSR12—Reserved	2000 8030*
NICSR13—Breakpoint address	2000 8034*
NICSR14—Reserved	2000 8038*
NICSR15—Diagnostic mode and status	2000 803C
Reserved local register I/O space	2000 8040 to 2003 FFFF
UVROM Space	**2004 0000 to 2007 FFFF**
MicroVAX system type register (in UVROM)	2004 0004
Local UVROM (halt-protected)	2004 0000 to 2007 FFFF
Local Register I/O Space	**2008 0000 to 201F FFFF**
DMA system configuration register	2008 0000
DMA system error register	2008 0004
DMA master error address register	2008 0008
DMA slave error address register	2008 000C
Q22-bus map base register	2008 0010
Reserved local register I/O space	2008 0014 to 2008 00FF
Error status register (Reg. 32)	2008 0180
Memory error address (Reg. 33)	2008 0184
I/O error address (Reg. 34)	2008 0188
DMA memory error address (Reg. 35)	2008 018C
DMA mode control and diagnostic status register (Reg. 36)	2008 0190
Reserved local register I/O space	2008 0194 to 2008 3FFF
Boot and diagnostic register (32 copies)	2008 4000 to 2008 407C
Reserved local register I/O space	2008 4080 to 2008 7FFF
Q22-bus map registers	2008 8000 to 2008 FFFF

*These registers are not fully implemented. Accesses yield unpredictable results.

VAX I/O Space

Contents	Address
Local Register I/O Space	**2008 0000 to 201F FFFF**
Reserved local register I/O space	2009 0000 to 2013 FFFF
SSC base address register	2014 0000
SSC configuration register	2014 0010
CP bus timeout control register	2014 0020
Diagnostic LED register	2014 0030
Reserved local register I/O space	2014 0034 to 2014 006B
The following addresses allow those KA670 internal processor registers that are implemented in the SSC chip (external, internal processor registers) to be accessed using the local I/O page. These addresses are documented for diagnostic purposes only and should not be used by nondiagnostic programs.	
Time-of-year register	2014 006C
Console storage receiver status	2014 0070*
Console storage receiver data	2014 0074*
Console storage transmitter status	2014 0078*
Console storage transmitter data	2014 007C*
Console receiver control/status	2014 0080
Console receiver data buffer	2014 0084
Console transmitter control/status	2014 0088
Console transmitter data buffer	2014 008C
Reserved local register I/O space	2014 0090 to 2014 00DB
I/O bus reset register	2014 00DC
Reserved local register I/O space	2014 00E0
ROM data register	2014 00F0†
Bus timeout counter	2014 00F4†
Interval timer	2014 00F8†
Reserved local register I/O space	2014 00FC to 2014 00FF

Local Register I/O Space	
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

*These registers are not fully implemented. Accesses yield unpredictable results.

†These registers are internal SSC registers used for SSC chip test purposes only. They should not be accessed by the CPU.

VAX I/O Space	
Contents	**Address**
Local Register I/O Space	
Timer 1 interval register	2014 0114
Timer 1 next interval register	2014 0118
Timer 1 interrupt vector	2014 011C
Reserved local register I/O space	2014 0120 to 2014 012F
BDR address decode match register	2014 0130
BDR address decode mask register	2014 0134
Reserved local register I/O space	2014 0138 to 2014 03FF
Battery backed-up RAM	2014 0400 to 2014 07FF
Reserved local register I/O space	2014 0800 to 201F FFFF
Reserved local I/O space	2020 0000 to 2FFF FFFF
Local Q22-bus memory space	3000 0000 to 303F FFFF
Reserved local register I/O space	3040 0000 to 3FFF FFFF

A.3 External, Internal Processor Registers

Several of the internal processor registers (IPRs) on the KA670 are implemented in the C-chip or SSC chip rather than the CPU chip. These registers are referred to as external, internal processor registers and are listed here.

IPR	Register Name	Abbreviation
27	Time-of-year register	TOY
28	Console storage receiver status	CSRS*
29	Console storage receiver data	CSRD*
30	Console storage transmitter status	CSTS*
31	Console storage transmitter data	CSDB*
32	Console receiver control/status	RXCS
33	Console receiver data buffer	RXDB
34	Console transmitter control/status	TXCS
35	Console transmitter data buffer	TXDB

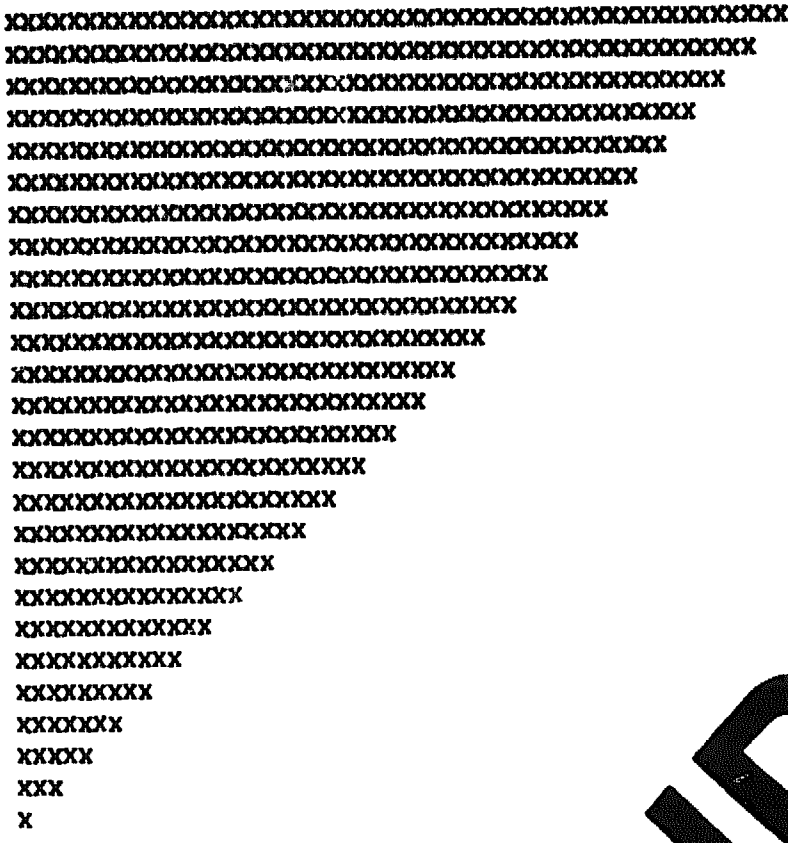
*These registers are not fully implemented. Accesses yield unpredictable results.

IPR	Register Name	Abbreviation
55	I/O system reset register	IORESET
112	Backup cache reserved register	BC112*
113	Backup cache tag store	BCBTS
114	Backup cache P1 tag store	BCP1TS
115	Backup cache P2 tag store	BCP2TS
116	Backup cache refresh register	BCRFR
117	Backup cache index register	BCIDX
118	Backup cache status register	BCSTS
119	Backup cache control register	BCCTL
120	Backup cache error register	BCERR
121	Backup cache flush backup cache tag store	BCFBTS
122	Backup cache flush primary cache tag store	BCPBTS
123	Backup cache reserved register	BC123*

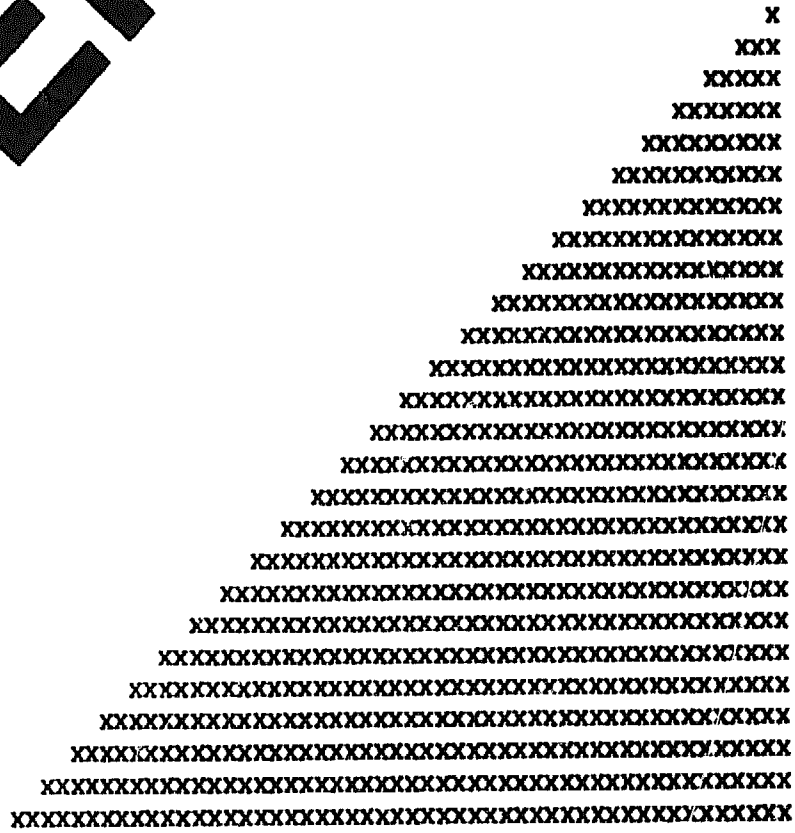
*These registers are not fully implemented. Accesses yield unpredictable results.

A.4 Global Q22-Bus Address Space Map

Q22-bus Memory Space	
Q22-bus memory space (octal)	0000 0000 to 1777 7777
Q22-bus I/O Space (BBS7 Asserted)	
Q22-bus I/O space (octal)	1776 0000 to 1777 7777
Reserved Q22-bus I/O space	1776 0000 to 1776 0007
Q22-bus floating address space	1776 0010 to 1776 3777
User-reserved Q22-bus I/O space	1776 4000 to 1776 7777
Reserved Q22-bus I/O space	1777 0000 to 1777 7477
Interprocessor communication register	1777 7500
Reserved Q22-bus I/O space	1777 7502 to 1777 7777



APPENDIX B



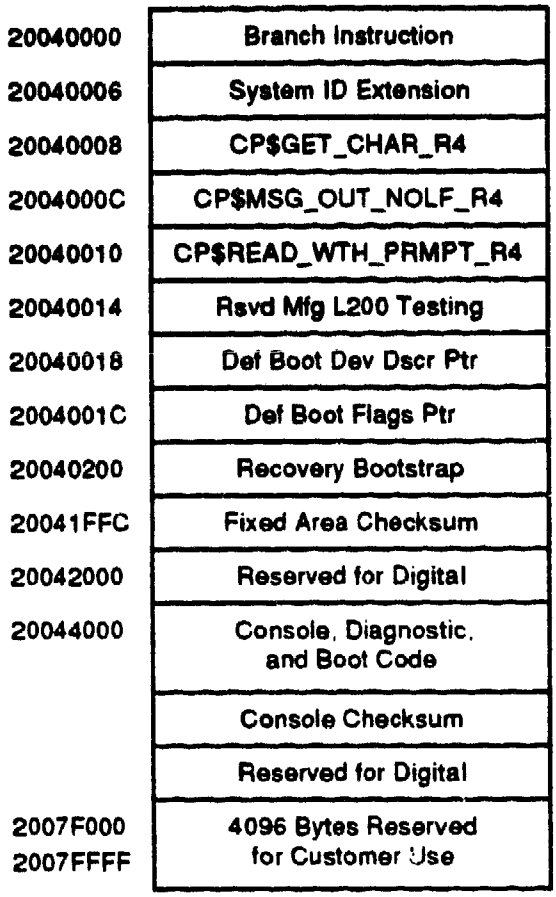
Appendix B
ROM Partitioning

This section describes ROM partitioning and subroutine entry points that are public and are guaranteed to be compatible over future versions of the firmware.

B.1 Firmware EPROM Layout

The KA670 uses two 128-Kbyte EPROMs for a total of 256 Kbytes. Unlike previous Q22-bus based MicroVAX processors, there is no duplicate decoding of the EPROM into halt protected and halt unprotected spaces. The entire EPROM is halt protected. See the EPROM layout in Figure B–1.

Figure B–1: KA670 EPROM Layout



The first instruction executed on halts is a branch around the System Id Extension (SIE) and the callback entry points. This allows these public data structures to reside in fixed locations in the EPROM.

The callback area entry points provide a simple interface to the currently defined console for VMB and secondary bootstraps. This is documented further in the next section.

The fixed area checksum is the sum of longwords from 20040000 to the checksum inclusive. This checksum is distinct from the checksum that the rest of the console uses.

The console, diagnostic and boot code constitute the bulk of the KA670 firmware. This code is field upgradable. The console checksum is from 20044000 to the checksum inclusive.

The memory between the console checksum and the user area at the end of the EPROMs is reserved for Digital for future expansion of the KA670 firmware. The contents of this area is set to FF.

The last 4096 bytes of EPROM are reserved for customer use and are not included in the console checksum. During a PROM bootstrap with PRB0 as the selected boot device, this block is tested for a PROM "signature block".

B.2 System ID Registers

The KA670 firmware and operating system software reference two registers to determine the processor on which they are running. The first, the System IDentification register (SID), is a CVAX internal processor register. The second, the System Identification Extension register (SIE), is a firmware register located in the KA670 EPROM.

B.2.1 PR$_SID (IPR 3E)

The SID longword can be read from IPR 3E (hex), using the MFPR instruction. This longword value is processor specific; however, the layout of this register is shown in Figure B–2.

Figure B–2: SID: System Identification Register

MLO-003898

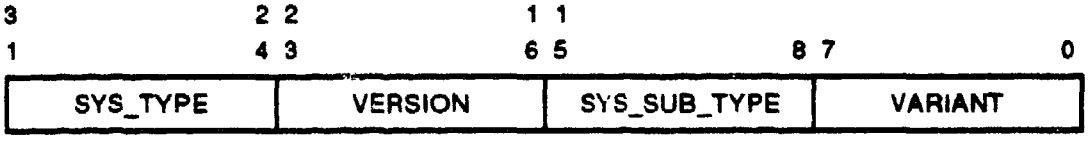
Field	Name	RW	Description
31:24	CPU_TYPE	ro	CPU type is the processor specific identification code. *0A : CVAX* *0E : RVAX*
24:8	reserved	ro	Reserved for future use.
7:0	VERSION	ro	Version of the microcode.

B.2.2 SIE (20040004)

The System Identification Extension register is an extension to the SID and is used to further differentiate between hardware configurations. The SID identifies which CPU and microcode is executing, and the SIE identifies what module and firmware revision are present. Note, the fields in this register are dependent on SID<31:24>(CPU_TYPE).

By convention, all MicroVAX systems implement a longword at physical location 20040004 in the firmware EPROM for the SIE. The layout of the SIE is shown in Figure B–3.

Figure B–3: SIE: System Identification Extension (20040004)



Field	Name	RW	Description
31:24	SYS_TYPE	ro	This field identifies the type of system for a specific processor. *01 : Q22-bus single processor system.*
23:16	VERSION	ro	This field identifies the resident version of the firmware EPROM encoded as two hexadecimal digits. For example, if the banner displays V5.0, then this field is 30 (hex).
15:8	SYS_SUB_TYPE	ro	This field identifies the particular system subtype. *01 : KA650* *02 : KA640* *03 : KA655* *04 : KA670*
7:0	VARIANT	ro	This_field_is_reserved_by_Digital. *01 : KA670-AA* *02 : KA670-BA*

B.2.3 Call-Back Entry Points

The KA670 firmware provides several entry points that facilitate I/O to the designated console device. Users of these entry points do not need to be aware of the console device type, be it a video terminal or workstation.

The primary intent of these routines is to provide a simple console device to VMB and secondary bootstraps, before operating systems load their own terminal drivers.

These are JSB (subroutine as opposed to procedure) entry points located in fixed locations in the firmware. These locations branch to code that in turn calls the appropriate routines.

All the entry points are designed to run at IPL 31 on the interrupt stack in physical mode. Virtual mode is not supported. Due to internal firmware architectural restrictions, users are encouraged to only call into the halt protected entry points. These entry points are listed below.

CP$GET_CHAR_R4	20040008
CP$MSG_OUT_NOLF_R4	2004000C
CP$READ_WTH_PRMPT_R4	20040010

B.2.4 CP$GET_CHAR_R4

This routine returns the next character entered by the operator in R0. A timeout interval can be specified. If the timeout interval is zero, no timeout is generated. If a timeout is specified and if timeout occurs, a value of 18 (CAN) is returned instead of normal input.

Registers R0, R1, R2, R3, and R4 are modified by this routine, all others are preserved.

```
;-------------------------------------------------------------
; Usage with timeout:

movl    #timeout_in_tenths_of_second,r0 ; Specify timeout.
jsb     @#CP$GET_CHAR_R4                ; Call routine.
cmpb    r0,#^x18                        ; Check for timeout.
beql    timeout_handler                 ; Branch if timeout.
; Input is in R0.

;-------------------------------------------------------------
; Usage without timeout:

clrl    r0                              ; Specify no timeout.
jsb     @#CP$GET_CHAR_R4                ; Call routine.
; Input is in R0.

;-------------------------------------------------------------
```

B.2.5 CP$MSG_OUT_NOLF_R4

This routine outputs a message to the console. The message is specified either by a message code or a string descriptor. The routine distinguishes between message codes and descriptors by requiring that any descriptor be located outside the first page of memory. Hence, message codes are restricted to values between 0 and 511.

Registers R0, R1, R2, R3, and R4 are modified by this routine, all others are preserved.

```
;-------------------------------------------------------------
; Usage with message code:

movzbl  #console_message_code,r0    ; Specify message code.
jsb     @#CP$MSG_OUT_NOLF_R4        ; Call routine.

;-------------------------------------------------------------
; Usage with a message descriptor (position dependent).

movaq   5$,r0                       ; Specify address of desc.
jsb     @#CP$MSG_OUT_NOLF_R4        ; Call routine.
.
.

5$:     .ascid  /This is a message/ ; Message with descriptor.

;-------------------------------------------------------------
; Usage with a message descriptor (position independent).

pushab  5$                          ; Generate message desc.
pushl   #10$-5$                     ; on stack.
movl    sp,r0                       ; Pass desc. addr. in R0.
jsb     @#CP$MSG_OUT_NOLF_R4        ; Call routine.
clrq    (sp)+                       ; Purge desc. from stack.
.
.

5$:     .ascii  /This is a message/ ; Message.
10$:                                ;

;-------------------------------------------------------------
```

B.2.6 CP$READ_WTH_PRMPT_R4

This routine outputs a prompt message and then inputs a character string from the console. When the input is accepted, DELETE, CONTROL-U, and CONTROL-R functions are supported.

As with CP$MSG_OUT_NOLF_R4, either a message code or the address of a string descriptor is passed in R0 to specify the prompt string. A value of zero results in no prompt.

A descriptor of the input string is returned in R0 and R1. R0 contains the length of the string and R1 contains the address. This routine inputs the string into the console program string buffer and therefore the caller need not provide an input buffer. Successive calls, however, destroy the previous contents of the input buffer.

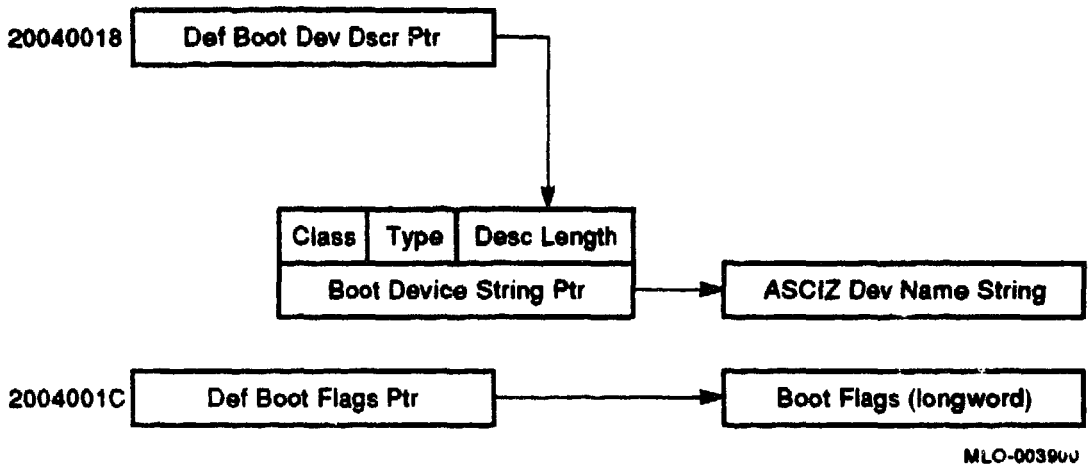
Registers R0, R1, R2, R3, and R4 are modified by this routine, all others are preserved.

```
;-------------------------------------------------------------
; Usage with a message descriptor (position independent).

pushab  10$                             ; Generate prompt desc.
pushl   #10$-5$                         ; on stack.
movl    sp,r0                           ; Pass desc. addr. in R0.
jsb     @#CP$READ_WTH_PRMPT_R4          ; Call routine.
clrq    (sp)+                           ; Purge prompt desc.
.                                       ; Input desc in R0 and R1.
.

5$:     .ascii  /Prompt> /              ; Prompt string.
10$:

;-------------------------------------------------------------
```

B.3 Boot Information Pointers

Two longwords located in ROM are used as pointers to the default boot device descriptor and the default boot flags, since the actual location of this data may change in successive versions of the firmware. Any software that uses these pointers should reference them at the addresses in halt protected space. See Figure B–4.

Figure B–4: Boot Information Pointers



The following macro defines the boot device descriptor format.

```
;-------------------------------------------------------------
; Default Boot Device Descriptor
;
boot_device_descriptor::
        base = .
        . = base + dsc$w_length
        .word   nvr$s_boot_device

        . = base + dsc$b_dtype
        .byte   dsc$k_dtype_z

        . = base + dsc$b_class
        .byte   dsc$k_class_z

        . = base + dsc$a_pointer
        .long   nvr_base + nvr$b_boot_device

        . = base + dsc$s_dscdef1
;-------------------------------------------------------------
```


APPENDIX C

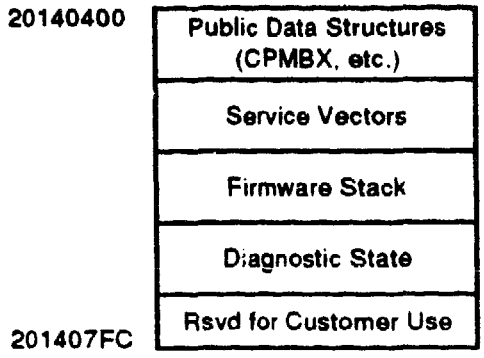
Appendix C
System Support Chip RAM Partitioning

This appendix describes the KA670 firmware partitions of the SSC 1 Kbyte battery backup unit (BBU) RAM.

C.1 SSC RAM Layout

The KA670 firmware uses the 1 Kbyte of RAM on the SSC for storage of firmware specific data structures and other information that must be preserved across power cycles. This RAM resides in the SSC chip starting at address 20140400. The RAM should not be used by the operating systems except as documented below. This RAM is not reflected in the bitmap built by the firmware. See Figure C–1.

Figure C–1: KA670 SSC RAM Layout



C.1.1 Public Data Structures

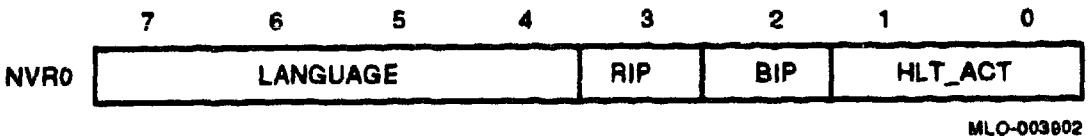
The following is a list of the public data structures in RAM used by the console.

Fields that are designated as reserved and/or internal use should not be written, since there is no protection against such corruption.

C.1.2 Console Program MailBoX (CPMBX)

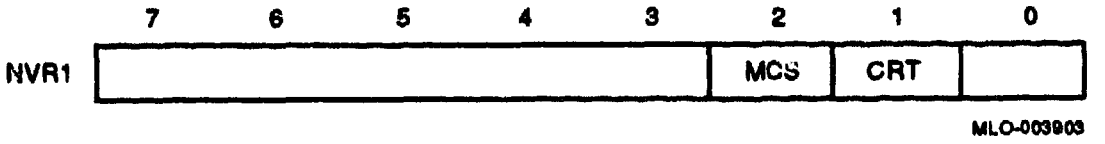
The Console Program MailBoX (CPMBX) is a software data structure located at the beginning of RAM (20140400). The CPMBX is used to pass information between the KA670 firmware and diagnostics, VMB, or an operating system. The CPMBX consists of three bytes referred to here as SSCR0, SSCR1, and SSCR2. See Figures C–2, C–3, and C–4.

Figure C–2: SSCR0 (20140400): Console Program MailBoX (CPMBX)



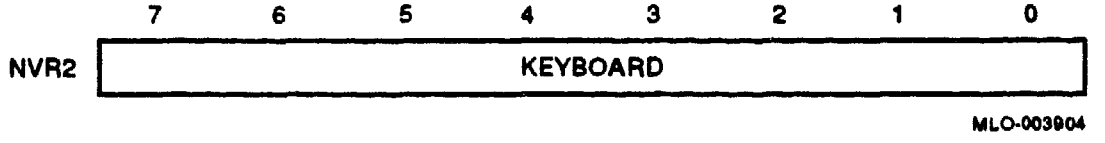
Field	Name	Description
7:4	LANGUAGE	This field specifies the current selected language for displaying halt and error messages on terminals that support MCS.
3	RIP	If set, a restart attempt is in progress. This flag must be cleared by the operating system, if the restart succeeds.
2	BIP	If set, a bootstrap attempt is in progress. This flag must be cleared by the operating system, if the bootstrap succeeds.
1:0	HLT_ACT	Processor halt action—this field is used to control the automatic restart/bootstrap procedure. HLT_ACT is normally written by the operating system. 0 : Restart; if that fails, reboot; if that fails, halt. 1 : Restart; if that fails, halt. 2 : Reboot; if that fails, halt. 3 : Halt.

Figure C–3: SSCR1 (20140401)



Field	Name	Description
2	MCS	If set, indicates that the attached terminal supports Multinational Character Set. If clear, MCS is not supported.
1	CRT	If set, indicates that the attached terminal is a CRT. If clear, indicates that the terminal is hard copy.

Figure C–4: SSCR2 (20140402)



Field	Name	Description
7:0	KEYBOARD	This field indicates the national keyboard variant in use.

C.1.3 Firmware Stack

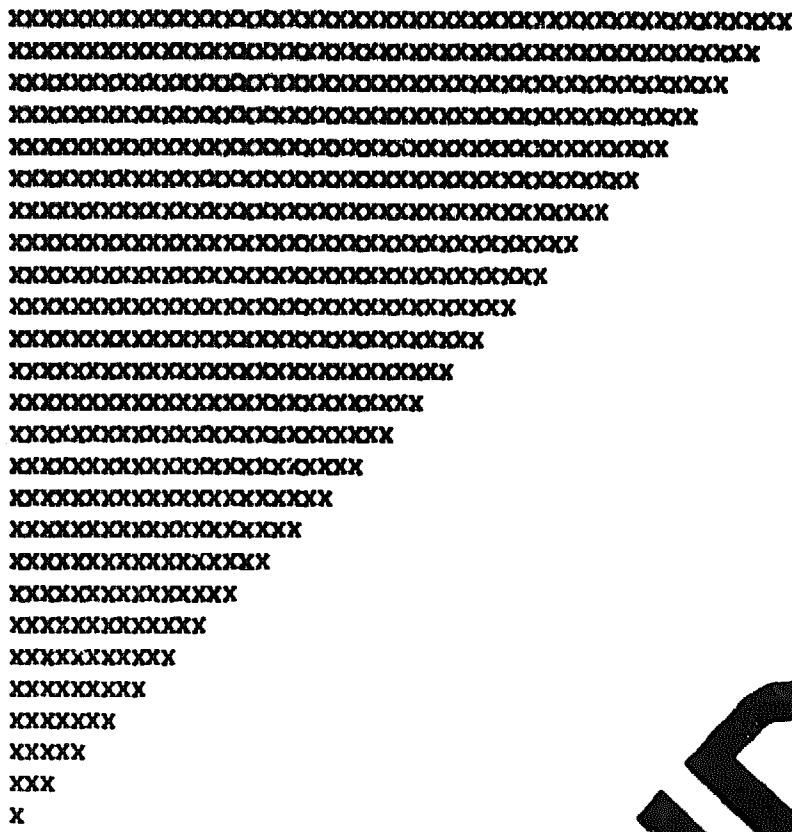
This section contains the stack that is used by all the firmware with the exception of VMB, which has its own built-in stack.

C.1.4 Diagnostic State

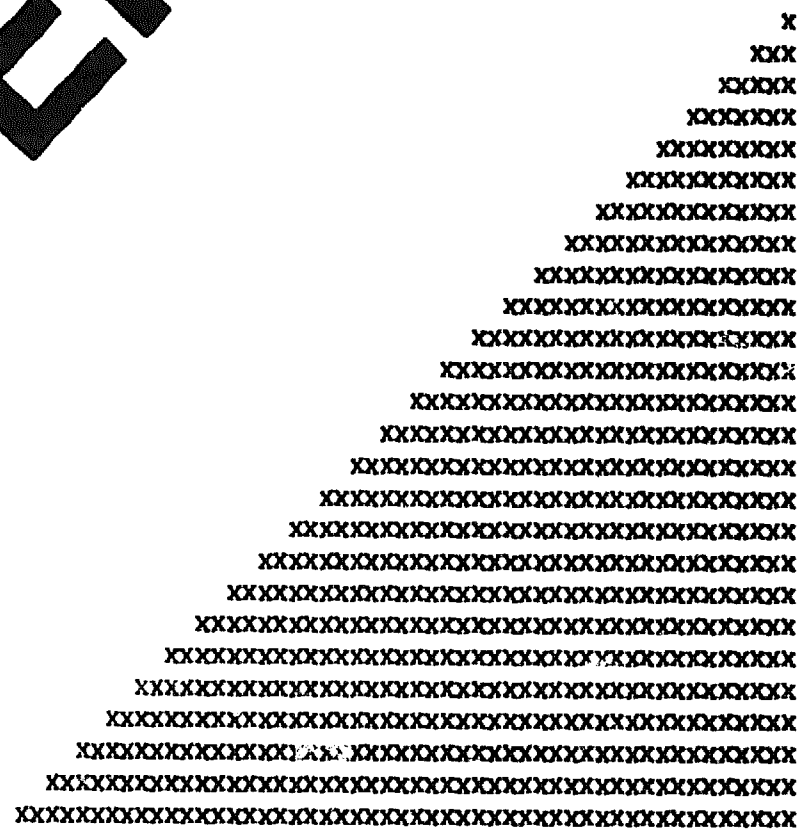
This area is used by the firmware resident diagnostics. This section is not documented here.

C.1.5 USER Area

The KA670 console reserves the last longword (address 201407FC) of the RAM for customer use. This location is not tested by the console firmware. Its value is undefined.



APPENDIX D



Appendix D
Data Structures

This appendix contains definitions of key global data structures that are used by the KA670 firmware.

D.1 Halt Dispatch State Machine

The KA670 halt dispatcher determines what actions the firmware will take on halt entry based on the machine state. The dispatcher is implemented as a state machine, which uses a single bitmap control word and the transition Table D–1 to process all halts. The transition table is sequentially searched for matches with the current state and control word. If there is a match, a transition occurs to the next state.

The control word is comprised of the following information:

- **Halt Type,** used for resolving external halts. Valid only if Halt Code is 00.

  000 : power-up state
  001 : halt in progress
  010 : negation of Q22-bus DCOK
  011 : console BREAK condition detected
  100 : Q22-bus BHALT
  101 : SGEC BOOT_L asserted (trigger boot)

- **Halt Code,** compressed form of SAVPSL<13:8>(RESTART_CODE).

  00 : RESTART_CODE = 2, external halt
  01 : RESTART_CODE = 3, power-up/reset
  10 : RESTART_CODE = 6, halt instruction
  11 : RESTART_CODE = any other, error halts

- **Mailbox Action,** passed by an operating system in CPMBX<1:0>(HALT_ACTION).

  00 : restart, boot, halt
  01 : restart, halt
  10 : boot, halt
  11 : halt

- **User Action,** specified with the SET HALT console command.

  000 : default
  001 : restart, halt
  010 : boot, halt
  011 : halt
  100 : restart, boot, halt

- **HEN (H),** BREAK (halt) enable switch, BDR<<REFERENCE>(HLT_ENB)>
- **ERR (E),** error status
- **TIP (T),** trace in progress
- **DIP (D),** diagnostics in progress
- **BIP (B),** bootstrap in progress CPMBX<2>
- **RIP (R),** restart in progress CPMBX<3>

Table D–1: Firmware State Transition Table

Current State	Next State	Halt Type	Halt Code	Mailbox Action	User Action	H E T D B R
		Perform conditional initialization[1]				
ENTRY	–>RESET INIT	xxx	01	xx	xxx	x - x - x - x - x - x
ENTRY	–>BREAK INIT	011	00	xx	xxx	x - x - x - x - x - x
ENTRY	–>TRACE INIT	xxx	10	xx	xxx	x - 0 - 1 - x - x - x
ENTRY	–>OTHER INIT	xxx	xx	xx	xxx	x - x - x - x - x - x
		Perform common initialization[2]				
RESET INIT	–>INIT	xxx	xx	xx	xxx	x - x - x - x - x - x
BREAK INIT	–>INIT	xxx	xx	xx	xxx	x - x - x - x - x - x
TRACE INIT	–>INIT	xxx	xx	xx	xxx	x - x - x - x - x - x
OTHER INIT	–>INIT	xxx	xx	xx	xxx	x - x - x - x - x - x

[1] Perform a unique initialization routine on entry. In particular, power-ups, BREAKs, and TRACEs require special initialization. Any other halt entry performs a default initialization.

[2] After performing conditional initialization, complete common initialization.

Table D–1 (Cont.): Firmware State Transition Table

Current State	Next State	Halt Type	Halt Code	Mailbox Action	User Action	H E T D B R
		Check for external halts[3]				
INIT	->BOOTSTRAP	010	00	xx	xxx	0 - x - x - x - x - x
INIT	->BOOTSTRAP	101	00	xx	xxx	x - x - x - x - x - x
INIT	->HALT	xxx	00	xx	xxx	x - x - x - x - x - x
		Check for pending (NEXT) trace[4]				
INIT	->TRACE	xxx	10	xx	xxx	x - x - 1 - x - x - x
TRACE	->EXIT	xxx	10	xx	xxx	x - 0 - 1 - x - x - x
TRACE	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x
		Check for bootstrap conditions[5]				
INIT	->BOOTSTRAP	xxx	01	xx	xxx	0 - 0 - 0 - 0 - 0 - 0
INIT	->BOOTSTRAP	xxx	01	xx	010	1 - 0 - 0 - 0 - 0 - 0
INIT	->BOOTSTRAP	xxx	01	xx	100	1 - 0 - 0 - 0 - 0 - 0
INIT	->BOOTSTRAP	xxx	1x	10	xxx	x - 0 - 0 - 0 - 0 - 0
INIT	->BOOTSTRAP	xxx	1x	00	010	x - 0 - 0 - 0 - 0 - 0
INIT	->BOOTSTRAP	xxx	1x	00	100	x - 0 - 0 - 0 - 0 - 1
INIT	->BOOTSTRAP	xxx	1x	00	100	x - 1 - 0 - 0 - 0 - x
INIT	->BOOTSTRAP	xxx	1x	00	000	0 - 0 - 0 - 0 - 0 - 1

[3] Halt on all external halts, except

If DCOK (unlikely) and halts are disabled, bootstrap.
If SGEC remote trigger, bootstrap.

[4] Unconditionally enter the TRACE state, if the TIP flag is set and the halt was due to a HALT instruction. From the TRACE state the firmware exits, if TIP is set and ERR is clear, otherwise it halts.

[5] Bootstrap,

If power-up and halts are disabled.
If power-up and halts are enabled and user action is 2 or 4.
If not power-up and mailbox is 2.
If not power-up and mailbox is 0 and user action is 2.
If not power-up and restart failed and mailbox is 0 and user action is 0 or 4.

Table D–1 (Cont.): Firmware State Transition Table

Current State	Next State	Halt Type	Halt Code	Mailbox Action	User Action	H E T D B R
RESTART	->BOOTSTRAP	xxx	1x	00	000	0 - 1 - 0 - 0 - 0 - x
		Check for restart conditions.[6]				
INIT	->RESTART	xxx	1x	01	xxx	x - 0 - 0 - 0 - 0 - 0
INIT	->RESTART	xxx	1x	00	001	x - 0 - 0 - 0 - 0 - 0
INIT	->RESTART	xxx	1x	00	100	x - 0 - 0 - 0 - 0 - 0
INIT	->RESTART	xxx	1x	00	000	0 - 0 - 0 - 0 - 0 - 0
		Perform common exit processing, if no errors.[7]				
BOOTSTRAP	->EXIT	xxx	xx	xx	xxx	x - 0 - x - x - x - x
RESTART	->EXIT	xxx	xx	xx	xxx	x - 0 - x - x - x - x
HALT	->EXIT	xxx	xx	xx	xxx	x - 0 - x - x - x - x
		Exception transitions, just halt.[8]				
INIT	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x
BOOT	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x
REST	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x
HALT	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x
TRACE	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x
EXIT	->HALT	xxx	xx	xx	xxx	x - x - x - x - x - x

[6] Restart the operating system if not power-up and

If mailbox is 1.
If mailbox is 0 and user action is 1 or 4.
If mailbox is 0 and user action is 0 and halts are disabled.

[7] Exit after halts, bootstrap, or restart. The exit state transitions to program I/O mode.

[8] Guard block that catches all exception conditions. In all cases, just halt.

"x" is used in this table to indicate a "don't care" field.

A transition to a "next state" occurs if a match is found between the control word and a "current state" entry in the table. The firmware does a linear search through the table for a match. Therefore, the order of the entries

in the transition table is important. The control longword is reassembled before each transition from the current machine state. The state machine transitions are shown in Table D–1.

D.2 RPB

VMB typically uses the low portion of memory unless there are bad pages in the first 128 Kbytes. The first page in its block is used for the RPB (Restart Parameter Block), through which it communicates to the operating system. Usually, this is page 0.

VMB will initialize the RPB as shown in Table D–2.

Table D–2: Restart Parameter Block Fields

(R11)+	Field Name	Description
00:	RPB$L_BASE	Physical address of base of RPB.
04:	RPB$L_RESTART	Cleared.
08:	RPB$L_CHKSUM	-1.
0C:	RPB$L_RSTRTFLG	Cleared.
10:	RPB$L_HALTPC	R10 on entry to VMB (HALT PC).
10:	RPB$L_HALTPSL	PR$_SAVPSL on entry to VMB (HALT PSL).
18·	RPB$L_HALTCODE	AP on entry to VMB (HALT CODE).
1C:	RPB$L_BOOTR0	R0 on entry to VMB.
		NOTE: *The field RPB$W_R0UBVEC, which overlaps the high-order word of RPB$L_BOOTR0, is set by the boot device drivers to the SCB offset (in the second page of the SCB) of the interrupt vector for the boot device.*
20:	RPB$L_BOOTR1	VMB version number. The high-order word of the version is the major ID and the low-order word is the minor ID.
24:	RPB$L_BOOTR2	R2 on entry to VMB.
28:	RPB$L_BOOTR3	R3 on entry to VMB.
2C:	RPB$L_BOOTR4	R4 on entry to VMB.
		NOTE: *The 48-bit booting node address is stored in RPB$L_BOOTR3 and RPB$L_BOOTR4 for compatibility with ELN V1.1 (this field is only initialized this way when performing a network boot).*

Table D–2 (Cont.): Restart Parameter Block Fields

(R11)+	Field Name	Description
30:	RPB$L_BOOTR5	R5 on entry to VMB.
34:	RPB$L_IOVEC	Physical address of boot driver's I/O vector of transfer addresses.
38:	RPB$L_IOVECSZ	Size of BOOT QIO routine.
3C:	RPB$L_FILLBN	LBN of secondary bootstrap image.
40:	RPB$L_FILSIZ	Size of secondary bootstrap image in blocks.
44:	RPB$Q_PFNMAP	The PFN bitmap is an array of bits, where each bit has the value "1" if the corresponding page of memory is valid, or has the value "0" if the corresponding page of memory contains a memory error. Through use of the PFNMAP, the operating system can avoid memory errors by avoiding known bad pages altogether. The memory bitmap is always page aligned and describes all the pages of memory from physical page #0 to the high end of memory, but excluding the PFN bitmap itself and the Q-bus map registers. If the high byte of the bitmap spans some pages available to the operating system and some pages of the PFN bitmap itself, the pages corresponding to the bitmap itself will be marked as bad pages. The first longword of the PFNMAP descriptor contains the number of bytes in the PFNMAP; the second longword contains the physical address of the bitmap.
4C:	RPB$L_PFNCNT	Count of "good" pages of physical memory, but not including the pages allocated to the Q22-bus scatter/gather map, the console scratch area, and the PFN bitmap at the top of memory.
50:	RPB$L_SVASPT	0.
54:	RPB$L_CSRPHY	Physical address of CSR for boot device.
58:	RPB$L_CSRVIR	0.
5C:	RPB$L_ADPPHY	Physical address of ADP (really the address of QMRs – ^x800 – 800 (hex) to look like a UBA adapter).
60:	RPB$L_ADPVIR	0.
64:	RPB$W_UNIT	Unit number of boot device.
66:	RPB$B_DEVTYP	Device type code of boot device.

Table D–2 (Cont.): Restart Parameter Block Fields

(R11)+	Field Name	Description
67:	RPB$B_SLAVE	Slave number of boot device.
68:	RPB$T_FILE	Name of secondary bootstrap image. Defaults to [SYS0.SYSEXE]SYSBOOT.EXE. This field (up to 40 bytes) is overwritten with the input string on a solicit boot. **NOTE:** *1: For VAX/VMS, the RPB$T_FILE must contain the root directory string "SYSn." on a nonnetwork bootstrap. This string is parsed by SYSBOOT (that is, SYSBOOT does not use the high nibble of BOOTR5).* *2: The RPB$T_FILE is overwritten to contain the boot node name for compatibility with ELN V1.1 (this field is only initialized this way when performing a network boot).*
90:	RPB$B_CONFREG	Array (16 bytes) of adapter types (NDT$_UB0_UNIBUS).
A0:	RPB$B_HDRPGCNT	Count of header pages.
A1:	RPB$W_BOOTNDT	Boot adapter nexus device type. Used by SYSBOOT and INIADP (OF SYSLOA) to configure the adapter of the boot device (changed from a byte to a word field in Version 12 of VMB).
B0:	RPB$L_SCBB	Physical address of SCB.
BC:	RPB$L_MEMDSC	Count of pages in physical memory including both good and bad pages. The high 8 bits of this longword contain the TR number, which is always zero for KA670.
C0:	RPB$L_MEMDSC+4	PFN of the first page of memory. This field is always zero for KA670, even if page 0 is a bad page. **NOTE:** *No other memory descriptors are used.*
104:	RPB$L_BADPGS	Count of "bad" pages of physical memory.
108:	RPB$B_CTRLLTR	Boot device controller number biased by 1. In VAX/VMS, this field is used by INIT (in SYS) to construct the boot device's controller letter. A zero implies this field has not been initialized. If initialized, A=1, B=2, and so on. (This field was added in Version 13 of VMB.)
nn:	———	The rest of the RPB is zeroed.

D.3 VMB Argument List

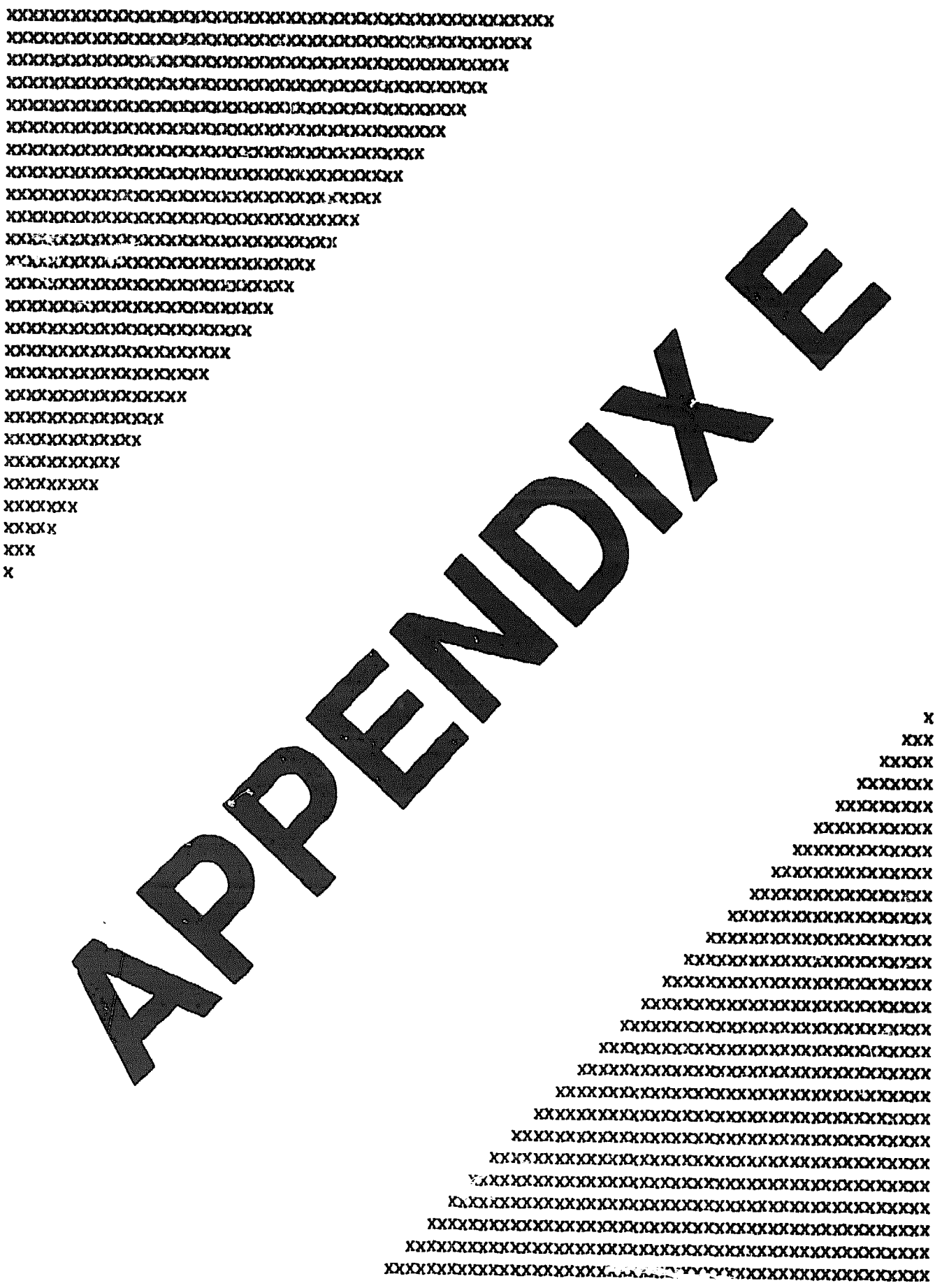
The VMB code will also initialize an argument list as shown in Table D–3 (the address of the argument list is passed in the AP).

Table D–3: VMB Argument List

(AP)+	Field Name	Description
04:	VMB$L_FILECACHE	Quadword file name.
0C:	VMB$L_LO_PFN	PFN of first page of physical memory (always zero, regardless of where 128 Kbytes of "good" memory start).
10:	VMB$L_HI_PFN	PFN of last page of physical memory.
14:	VMB$Q_PFNMAP	Descriptor of PFN bitmap. First longword contains count of bytes in bitmap. Second longword contains physical address of bitmap. (Same rules as for RPB$Q_PFNMAP listed above.)
1C:	VMB$Q_UCODE	Quadword.
24:	VMB$B_SYSTEMID	48-bit (actually a quadword is allocated) booting node address initialized when performing a network boot. This field is copied from the Target System Address parameter of the parameter's message. (The DECnet HIORD value is added if the field was 2 bytes.)
2C:	VMB$L_FLAGS	Set as needed.
30:	VMB$L_CI_HIPFN	Cluster interface high PFN.
34:	VMB$Q_NODENAME	Boot node name, which is initialized when performing a network boot. This field is copied from the Target System Name parameter of the parameter's message.
3C:	VMB$Q_HOSTADDR	Host node address, only initialized when booting over the network. This field is copied from the Host System Address parameter of the parameter's message.
44:	VMB$Q_HOSTNAME	Host node name (this value is only initialized when performing a network boot). This field is copied from the Host System Name parameter of the parameter's message.
4C:	VMB$Q_TOD	Time of day (this value is only initialized when performing a network boot). The time of day is copied from the first 8 bytes of the Host System Time parameter of the parameter's message. (The time differential values are NOT copied.)

Table D–3 (Cont.): VMB Argument List

(AP)+	Field Name	Description
54:	VMB$L_XPARAM	Pointer to data retrieved from request of the parameter file.


APPENDIX E

Appendix E
Error Messages

The error messages issued by the KA670 firmware fall into three categories: halt code messages, VMB error messages, and console messages.

Error messages are in general cryptic to avoid the space requirements of translating a large number of messages.

E.1 Halt Code Messages

The following messages are issued by the firmware whenever the processor halts, except on power-up, which is not treated as an error condition.

For example:

```
?06 HLT INST
PC = 800050D3
```

The number preceding the Halt message is the "halt code." This number is obtained from SAVPSL<13:8>(RESTART_CODE), IPR 43, which is written on any CVAX processor restart operation. See Table E–1.

Table E–1: Halt Messages

Code	Message	Description
?02	EXT HLT	External Halt, caused by either console BREAK condition, Q22-bus BHALT_L, or DBR<AUX_HLT> bit was set while enabled.
03		Power-up, no Halt message is displayed. However, the presence of the firmware banner and diagnostic countdown indicates this Halt reason.
?04	ISP ERR	In attempting to push state onto the interrupt stack during an interrupt or exception, the processor discovered that the interrupt stack was mapped NO ACCESS or NOT VALID.
?05	DBL ERR	The processor attempted to report a machine check to the operating system, and a second machine check occurred.
?06	HLT INST	The processor executed a Halt instruction in kernel mode.
?07	SCB ERR3	The SCB vector had bits <1:0> equal to 3.
?08	SCB ERR2	The SCB vector had bits <1:0> equal to 2.
?0A	CHM FR ISTK	A change mode instruction was executed when PSL<IS> was set.
?0B	CHM TO ISTK	The SCB vector for a change mode had bit <0> set.
?0C	SCB RD ERR	A hard memory error occurred while the processor was trying to read 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 translation not valid occurred during processing of a kernel stack not valid exception.
?12	DBL ERR2	Double machine check error. A machine check occurred while trying to service a machine check.
?13	DBL ERR3	Double machine check error. A machine check occurred while trying to service a kernel stack not valid exception.

Table E–1 (Cont.): Halt Messages

Code	Message	Description
?19	PSL EXC5[1]	PSL<26:24> = 5 on interrupt or exception.
?1A	PSL EXC6[1]	PSL<26:24> = 6 on interrupt or exception.
?1B	PSL EXC7[1]	PSL<26:24> = 7 on interrupt or exception.
?1D	PSL REI5[1]	PSL<26:24> = 5 on an rei instruction
?1E	PSL REI6[1]	PSL<26:24> = 6 on an rei instruction.
?1F	PSL REI7[1]	PSL<26:24> = 7 on an rei instruction.
?3F	MICROVERIFY FAILURE	Microcode power-up self-test failed.

[1]For the last six cases, the VAX architecture does not allow execution on the interrupt stack while in a mode other than kernel. In the first three cases, an interrupt is attempting to run on the interrupt stack while not in kernel mode. In the last three cases, an REI instruction is attempting to return to a mode other than kernel and still run on the interrupt stack.

E.2 VMB Error Messages

Table E–2 lists the VMB error messages.

Table E–2: VMB Error Messages

Code	Message	Description
?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	BADIRECTORY	Bad directory file.
?47	FILNOTCNTG	Invalid program image format.
?48	ENDOFFILE	Premature end-of-file encountered.
?49	BADFILENAME	Bad file name given.

Table E–2 (Cont.): VMB Error Messages

Code	Message	Description
?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	Invalid memory configuration.
?54	RETRY	No devices bootable, retrying.
?55	IVDEVNAM	Invalid device name.
?56	DRVERR	Drive error.

E.3 Console Error Messages

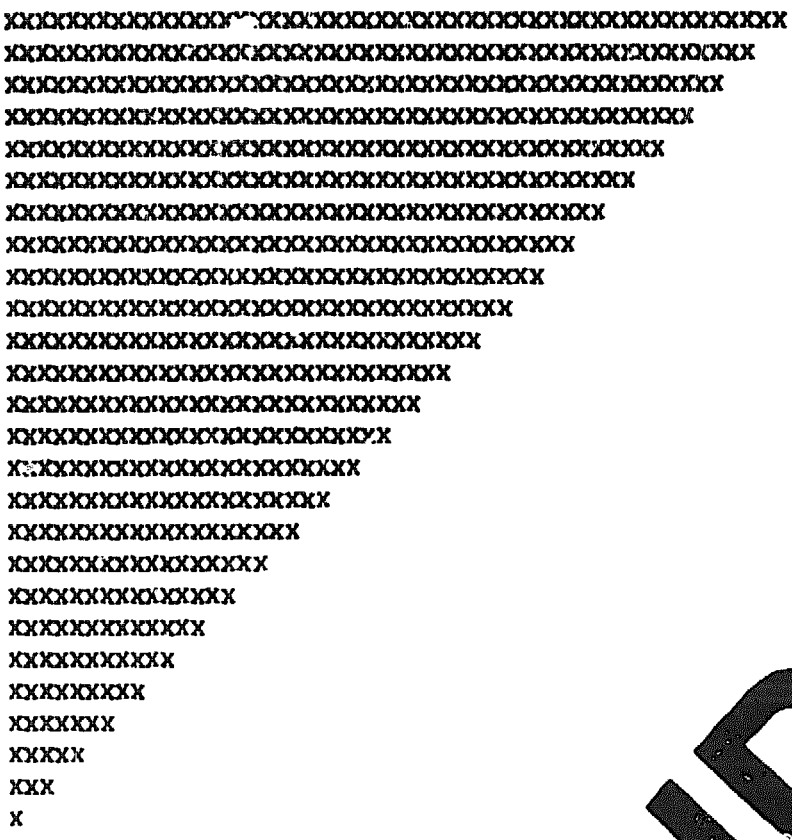
Console error messages, shown in Table E–3, are issued in response to a console command that has an error(s).

Table E–3: Console Error Messages

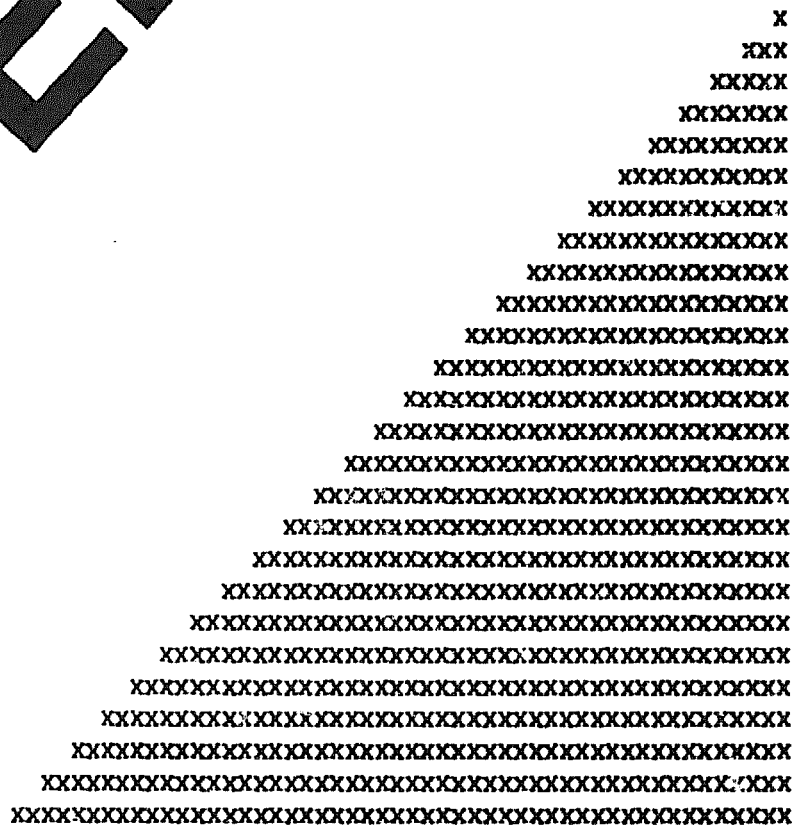
Code	Message	Description
?61	CORRUPTION	The console program database has been corrupted.
?62	ILLEGAL REFERENCE	Illegal reference. The requested reference would violate virtual memory protection, the address is not mapped, the reference is invalid in the specified address space, or the value is invalid in the specified destination.
?63	ILLEGAL COMMAND	The command string cannot be parsed.
?64	INVALID DIGIT	A number has an invalid digit.
?65	LINE TOO LONG	The command was too large for the console to buffer. The message is issued only after receipt of the terminating carriage return.
?66	ILLEGAL ADDRESS	The address specified falls outside the limits of the address space.
?67	VALUE TOO LARGE	The value specified does not fit in the destination.
?68	QUALIFIER CONFLICT	Qualifier conflict; for example, two different data sizes are specified for an EXAMINE command.
?69	UNKNOWN QUALIFIER	The switch is unrecognized.
?6A	UNKNOWN SYMBOL	The symbolic address in an EXAMINE or DEPOSIT command is unrecognized.
?6B	CHECKSUM	The command or data checksum of an X command is incorrect. If the data checksum is incorrect, this message is issued and is not abbreviated to "Illegal command".
?6C	HALTED	The operator entered a Halt command.
?6D	FIND ERROR	A FIND command failed either to find the RPB or 128 Kbytes of good memory.
?6E	TIME OUT	During an X command, data failed to arrive in the time expected (60 seconds).
?70	UNIMPLEMENTED	Unimplemented function.

Table E–3 (Cont.): Console Error Messages

Code	Message	Description
?71	NO VALUE QUALIFIER	Qualifier does not take a value.
?72	AMBIGUOUS QUALIFIER	Not enough unique characters to determine the qualifier.
?73	VALUE QUALIFIER	Qualifier requires a value.
?74	TOO MANY QUALIFIERS	Too many qualifiers supplied for this command.
?75	TOO MANY ARGUMENTS	Too many arguments supplied for this command.
?76	AMBIGUOUS COMMAND	Not enough unique characters to determine the command.
?77	TOO FEW ARGUMENTS	Insufficient arguments supplied for this command.
?78	TYPEAHEAD OVERFLOW	The typeahead buffer overflowed.
?79	FRAMING ERROR	A framing error was detected on the console serial line.
?7A	OVERRUN ERROR	An overrun error was detected on the console serial line.
?7B	SOFT ERROR	A soft error occurred.
?7C	HARD ERROR	A hard error occurred.
?7D	MACHINE CHECK	A machine check occurred.



APPENDIX F



Appendix F
Machine State on Power-Up

This appendix describes the state of the KA670 after a power-up halt.

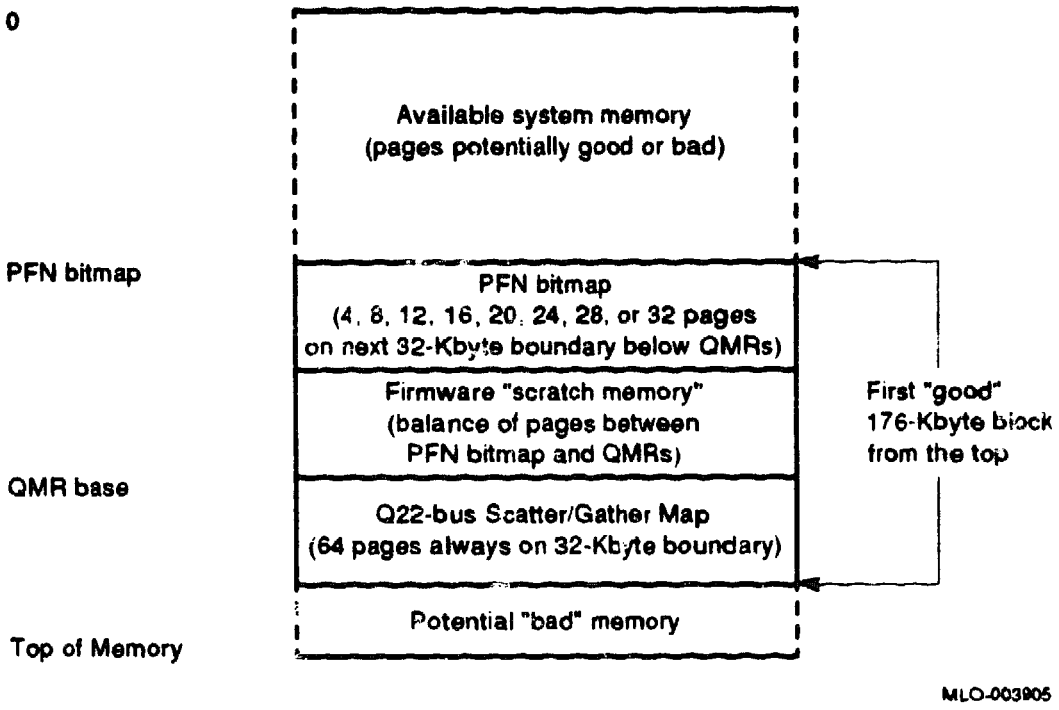
The descriptions in this section assume a machine with no errors, that the machine has just been turned on, and that only the power-up diagnostics have been run. The state of the machine is not defined if individual diagnostics are run or during any other halts other than a power-up halt (SAVPSL<13:8>(RESTART_CODE) = 3).

The following sections describe data structures that are guaranteed to be constant over future versions of the KA670 firmware. Placement and/or existence of any other structure(s) is not implied.

F.1 Main Memory Layout and State

Main memory is tested and initialized by the firmware on power-up. Figure F–1 is a diagram of how main memory is partitioned after diagnostics.

Figure F–1: Memory Layout After Power-Up Diagnostics



F.1.1 Reserved Main Memory

To build the scatter/gather map and the bitmap, the firmware attempts to find a physically contiguous page-aligned 176-Kbyte block of memory at the highest possible address that has no multibit errors. Single-bit errors are tolerated in this section.

Of the 176 Kbytes, the upper 32 Kbytes are dedicated to the Q22-bus scatter /gather map, as shown in Figure F–1. Of the lower portion, up to 128 Kbytes at the bottom of the block are allocated to the PFN (Page Frame Number) bitmap. The size of the PFN bitmap is dependent on the extent of physical memory; each bit in the bitmap maps one page (512 bytes) of memory. The remainder of the block between the bitmap and the scatter/gather map (minimally 16 Kbytes) is allocated for the firmware.

F.1.1.1 PFN Bitmap

The PFN bitmap is a data structure that indicates which pages in memory are deemed usable by operating systems. The bitmap is built by the diagnostics as a side effect of the memory tests on power-up. The bitmap always starts on a page boundary. The bitmap requires 1 Kbyte for every 4 Mbytes of main memory; hence, an 8-Mbyte system requires 2 Kbytes, 16 Mbytes require 4 Kbytes, 32 Mbytes require 8 Kbytes, and 64 Mbytes require 16 Kbytes. The bitmap does not map itself or anything above it. There may be memory above the bitmap that has both good and bad pages.

Each bit in the PFN bitmap corresponds to a page in main memory. There is a one-to-one correspondence between a page frame number (origin 0) and a bit index in the bitmap. A one in the bitmap indicates that the page is "good" and can be used. A zero indicates that the page is "bad" and should not be used. By default, a page is flagged "bad" if a single-bit or a multibit error occurs when referencing the page.

The PFN bitmap is protected by a checksum stored in the NVRAM. The checksum is a simple byte wide, two's complement checksum. The sum of all bytes in the bitmap and the bitmap checksum should result in zero. Operating systems that modify the bitmap are encouraged to update this checksum to facilitate diagnosis by Customer Services personnel.

F.1.1.2 Scatter/Gather Map

On power-up, the scatter/gather map is initialized by the firmware to map to the first 4 Mbytes of main memory. Main memory pages will not be mapped, if there is a corresponding page in Q22-bus memory, or if the page is marked bad by the PFN bitmap.

On a processor halt other than power-up, the contents of the scatter/gather map are undefined and are dependent on operating system usage.

Operating systems should not move the location of the scatter/gather map and should access the map only on aligned longwords through the local I/O space of 20088000 to 2008FFFC, inclusive. The Q22-bus map base register, (QBMBR) is set up by the firmware to point to this area and should not be changed by software.

F.1.1.3 Firmware "Scratch Memory"

This section of memory is reserved for the firmware. However, it is only used after successful execution of the memory diagnostics and initialization of the PFN bitmap and scatter/gather map. This memory is primarily used for diagnostic purposes.

F.1.2 Contents of Main Memory

The contents of main memory are undefined after the diagnostics have run. Typically, nonzero test patterns will be left in memory.

The diagnostics will "scrub" all main memory, so that no power-up induced errors remain in the memory system. On the KA670 memory subsystem, the state of the ECC bits and the data bits is undefined on initial power-up. This can result in single and multibit errors if the locations are read before written, because the ECC bits are not in agreement with their corresponding data bits. An aligned longword write to every location (done by diagnostics) eliminates all power-up induced errors.

F.2 Memory Control Registers

The KA670 firmware assigns bank numbers to the MEMCSRs in ascending order, without attempting to disable physical banks that contain errors. High-order unused banks are set to zero. Error loggers should capture the following bits from each MEMCSR.

MEMCSR<31> (bank enable bit). As the firmware always assigns banks in ascending order, knowing which banks are enabled is sufficient information to derive the bank numbers.

MEMCSR<1:0> (bank usage). This field determines the size of the banks on the particular memory board.

Additional information should be captured from the MEMCSR32 (RMESR), MEMCSR33 (RMEAR), MEMCSR34 (RIOEAR), MEMCSR35 (CEAR), and MEMCSR36 (MCDSR) as needed.

F.2.1 On-Chip Cache

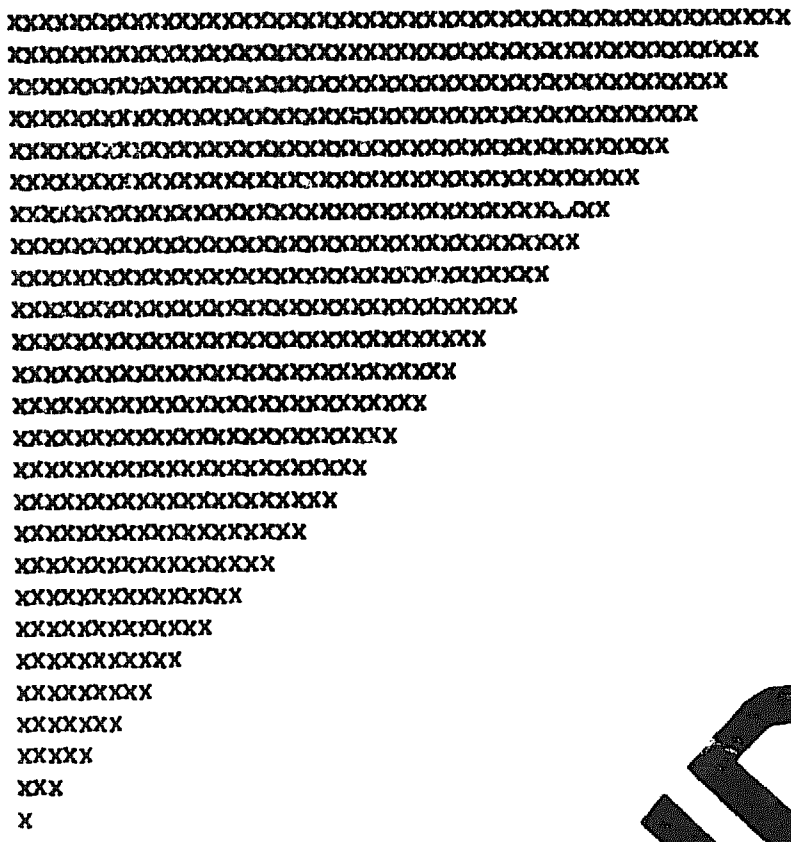
The CPU on-chip cache is tested during the power-up diagnostics, flushed, and then turned off. The cache is again turned on by the BOOT and the INIT commands. Otherwise, the state of the on-chip cache is disabled.

F.2.2 Translation Buffer

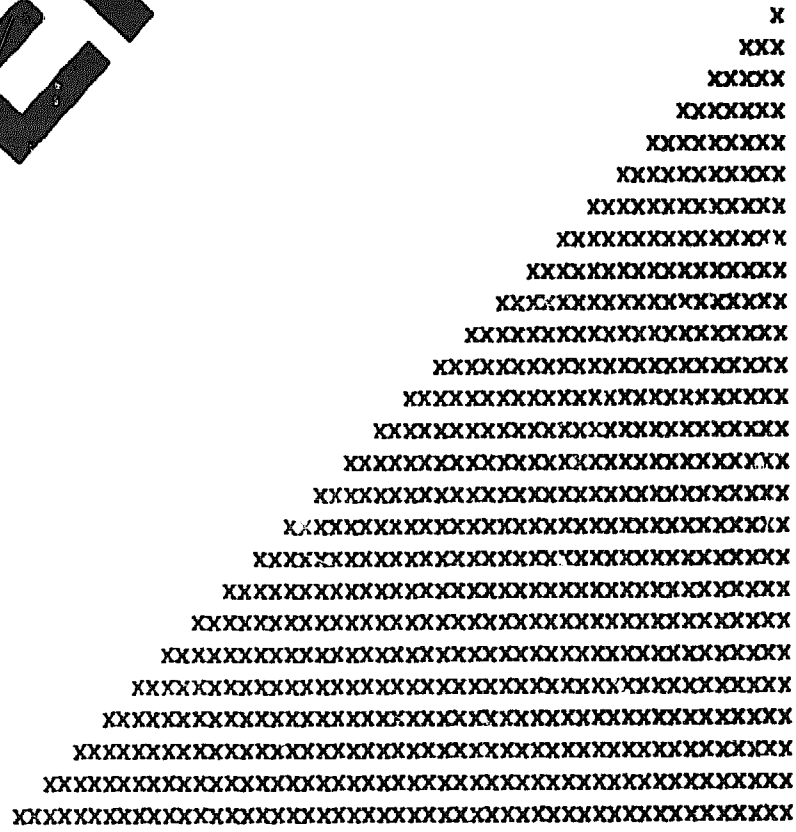
The CPU translation buffer is tested by diagnostics on power-up but not used by the firmware, since the translation buffer runs in physical mode. The translation buffer can be invalidated by using PR$_TBIA, IPR 57.

F.2.3 Halt-Protected Space

The KA670 firmware runs halt-protected.



APPENDIX G



Appendix G
MOP Support

G.1 Network "Listening"

While the KA670 is waiting for a load volunteer during bootstrap, it "listens" on the network for other maintenance messages directed to the node and periodically identifies itself at the end of each 8- to 12-minute interval before a bootstrap retry. In particular, this "listener" supplements the Maintenance Operation Protocol (MOP) functions of the VMB load requester typically found in bootstrap firmware and supports.

- A remote console server that generates COUNTERS messages in response to REQ_COUNTERS messages, unsolicited SYSTEM_ID messages every 8 to 12 minutes, and solicited SYSTEM_ID messages in response to REQUEST_ID messages, as well as recognition of BOOT messages.
- A loopback server that responds to Ethernet LOOPBACK messages by echoing the message to the requester.
- An IEEE 802.2 responder that replies to both XID and TEST messages.

During network bootstrap operation, the KA670 complies with the requirements defined in the "NI Node Architecture Specification" for a primitive node. The firmware listens only to MOP "Load/Dump", MOP "Remote Console", Ethernet "Loopback Assistance", and IEEE 802.3 XID /TEST messages (listed in Table G–4) directed to the Ethernet physical address of the node. All other Ethernet protocols are filtered by the network device driver.

The MOP functions and message types, which are supported by the KA670, are summarized in Tables G–1, G–2, G–3, G–4, and Section G.2.

Table G–1: KA670 Network Maintenance Operations Summary

Function	Role	Transmit		Receive
MOP Ethernet and IEEE 802.2 Messages[1]				
Dump	Requester	—		—
	Server	—		—
Load	Requester	REQ_PROGRAM[2]	to solicit	VOLUNTEER
		REQ_MEM_LOAD	to solicit & ACK	MEM_LOAD
			or	MEM_LOAD_w_XFER
			or	PARAM_LOAD_w_XFER
	Server	—		—
Console	Requester	—		—
	Server	COUNTERS	in response to	REQ_COUNTERS
		SYSTEM_ID[3]	in response to	REQUEST_ID
				BOOT
Loopback	Requester	—		—
	Server	LOOPED_DATA[4]	in response to	LOOP_DATA

[1]All unsolicited messages are sent in Ethernet (MOP V3) and IEEE 802.2 (MOP V4), until the MOP version of the server is known. All solicited messages are sent in the format used for the request.

[2]The initial REQ_PROGRAM message is sent to the dumpload multicast address. If an assistance VOLUNTEER message is received, then the responder's address is used as the destination to repeat the REQ_PROGRAM message and for all subsequent REQ_MEM_LOAD messages.

[3]SYSTEM_ID messages are sent out every 8 to 12 minutes to the remote console multicast address and, on receipt of a REQUEST_ID message, they are sent to the initiator.

[4]LOOPED_DATA messages are sent out in response to LOOP_DATA messages. These messages are actually in Ethernet LOOP TEST format, not in MOP format, and when sent in Ethernet frames omit the additional length field (padding is disabled).

Table G–1 (Cont.): KA670 Network Maintenance Operations Summary

Function	Role	Transmit		Receive
		IEEE 802.2 Messages[5]		
Exchange ID	Requester	—		—
	Server	XID_RSP	in response to	XID_CMD
Test	Requester	—		—
	Server	TEST_RSP	in response to	TEST_CMD

[5]IEEE 802.2 support of XID and TEST is limited to Class 1 operations.

Table G–2: Supported MOP Messages

Message Type	Message Fields						
DUMP/LOAD							
MEM_LOAD_w_XFER	Code 00	Load # nn	Load addr aa-aa-aa-aa		Image data None		Xfer addr aa-aa-aa-aa
MEM_LOAD	Code 02	Load # nn	Load addr aa-aa-aa-aa		Image data dd-...		
REQ_PROGRAM	Code 08	Device 25 LQA 49 KA670	Format 01 V3 04 V4	Program 02 Sys	SW ID[3] C-17[1] C-128[2] If C[1] >00 Len 00 No ID FF OS FE Maint	Processr 00 Sys	Info (see SYSTEM_ID)
REQ_MEM_LOAD	Code 0A	Load # nn	Error ee				
PARM_LOAD_w_XFER	Code 14	Load # nn	Prm typ 01 02 03 04 05 06 00 End	Prm len 1-16 1-06 1-16 1-06 0A 08	Prm val Target name[1] Target addr[1] Host name[1] Host addr[1] Host time[1] Host time[2]		Xfer addr aa-aa-aa-aa
VOLUNTEER	Code 03						
REMOTE CONSOLE							
REQUEST_ID	Code 05	Rsrvd xx	Recpt # nn-nn				
SYSTEM_ID	Code 07	Rsrvd xx	Recpt # nn-nn or 00-00	Info type 01-00 Version 02-00 Functions 07-00 HW addr 64-00 Device 90-01 Datalink 91-01 Bufr size	Info len 03 02 06 01 01 02	Info value 04-00-00 00-59 25 or 49 01 06-04	
REQ_COUNTERS	Code 09	Recpt # nn-nn					

[1]MOP V3.0 only.

[2]MOP x4.0 only.

[3]Software ID field is loaded from the string stored in the 40-byte field, RPB$T_FILE, of the RPB on a solicited boot.

Table G–2 (Cont.): Supported MOP Messages

Message Type	Message Fields						
REMOTE CONSOLE							
COUNTERS	Code 0B	Rcvpt # nn-nn	Counter block				
BOOT[4]	Code 06	Verification vv-vv-vv-vv-vv-vv-vv-vv	Processr 00 Sys	Control xx	Dev ID C-17	SW ID[3] (see REQ_PROGRAM)	Script ID[2] C-128
LOOPBACK							
LOOP_DATA	Skpcnt nn-nn	Skipped bytes bb-...	Function 00-02 Forward data		Forward addr ee-ee-ee-ee-ee-ee		Data dd-...
LOOPED_DATA	Skpcnt nn-nn	Skipped bytes bb-...	Function 00-01 Reply		Rcvpt # nn-nn		Data dd-...
IEEE 802.2							
XID_CMD/RSP	Form 81	Class 01	Rx window size (K) 00				
TEST_CMD/RSP	Optional data						

[2]MOP x4.0 only.

[3]Software ID field is loaded from the string stored in the 40-byte field, RPB$T_FILE, of the RPB on a solicited boot.

[4]A BOOT message is not verified, since in this context, a boot is already in progress. However, a received BOOT message will cause the boot backoff timer to be reset to its minimum value.

Table G–3: Ethernet and IEEE 802.3 Packet Headers

Ethernet MOP Message Format (MOP V3)

Dest_address	Src_address	Prot	Len	MOP msg	Pad	CRC
dd-dd-dd-dd-dd-dd	ss-ss-ss-ss-ss-ss	60-01	nn-nn	dd-...	xx-...	cc-cc
		60-02	nn-nn	dd-...		
		90-00	dd-...			

IEEE 802.3 SNAP SAP MOP Message Format (MOP V4)

Dest_address	Src_address	Len	DSAP	SSAP	Ctl	P_ID	MOP_msg	CRC
dd-dd-dd-dd-dd-dd	ss-ss-ss-ss-ss-ss	nn-nn	AA	AA	03	08-00-2B-60-01 08-00-2B-60-02 08-00-2B-90-00	dd-...	cc-cc

IEEE 802.3 XID/TEST Message Format (MOP V4)

Dest_address	Src_address	Len	DSAP	SSAP	Ctl[1]	Data	CRC
dd-dd-dd-dd-dd-dd	ss-ss-ss-ss-ss-ss	nn-nn	aa	bb	cc	ff-tt-ss (XID) Optional data (TEST)	cc-cc

[1]XID and TEST messages are identified in the IEEE 802.3 control field with binary 101x1111 and 111x0011, respectively. "x" denotes the Poll/Final bit that gets echoed in the response.

Table G–4: MOP Multicast Addresses and Protocol Specifiers

Function	Address	IEEE Prefix[1]	Protocol	Owner
Dump/Load	AB-00-00-01-00-00	08-00-2B	60-01	Digital
Remote Console	AB-00-00-02-00-00	08-00-2B	60-02	Digital
Loopback Assistance	CF-00-00-00-00-00[2]	08-00-2B	90-00	Digital

[1]MOP V4.0 only.

[2]Not used.

G.2 MOP Counters

The following counters are kept for the Ethernet boot channel. All counters are unsigned integers. V4 counters roll over on overflow. All V3 counters "latch" at their maximum value to indicate overflow. Unless otherwise stated, all counters include both normal and multicast traffic. Furthermore, they include information for all protocol types. Frames-received and bytes-received counters do not include frames received with errors. Table G–5 displays the byte lengths and ordering of all the counters in both MOP V3.0 and V4.0.

Table G–5: MOP Counter Block

	Byte Length		
Name	**V3**	**V4**	**Description**
TIME_SINCE_CREATION	2	16	Time since last zeroed
Rx_BYTES	4	8	Bytes received
Tx_BYTES	4	8	Bytes sent
Rx_FRAMES	4	8	Frames received
Tx_FRAMES	4	8	Frames sent
Rx_MCAST_BYTES	4	8	Multicast bytes received
Rx_MCAST_FRAMES	4	8	Multicast frames received
Tx_INIT_DEFFERED	4	8	Frames sent, initially deferred[1]
Tx_ONE_COLLISION	4	8	Frames sent, single collision[1]
Tx_MULTI_COLLISION	4	8	Frames sent, multiple collisions[1]
	2		Send failure[2]
	2		Send failure bitmap[2]
TxFAIL_EXCESS_COLLS		8	Send failure—0 Excessive collisions
TxFAIL_CARIER_CHECK		8	Send failure—1 Carrier check failed
TxFAIL_SHRT_CIRCUIT		8	Send failure—2 Short circuit[3]

[1]Only one of these three counters will be incremented for a given frame.

[2]V3 send/receive failures are collapsed into one counter with bitmap indicating which failures occurred.

[3]Always zero.

Table G–5 (Cont.): MOP Counter Block

Name	Byte Length V3	Byte Length V4	Description
TxFAIL_OPEN_CIRCUIT		8	Send failure—3 Open circuit[3]
TxFAIL_LONG_FRAME		8	Send failure—4 Frame too long[3]
TxFAIL_REMOTE_DEFER		8	Send failure—5 Remote failure to defer[3]
	2		Receive failure[2]
	2		Receive failure bitmap[2]
RxFAIL_BLOCK_CHECK		8	Receive failure—Block check failure
RxFAIL_FRAMING_ERR		8	Receive failure—Framing error
RxFAIL_LONG_FRAME		8	Receive failure—Frame too long[3]
UNKNOWN_DESTINATION	2	8	Unrecognized frame destination
DATA_OVERRUN	2	8	Data overrun
NO_SYSTEM_BUFFER	2	8	System buffer unavailable[3]
NO_USER_BUFFER	2	8	User buffer unavailable[3]
FAIL_COLLIS_DETECT		8	Collision detect check failure

[2]V3 send/receive failures are collapsed into one counter with bitmap indicating which failures occurred.
[3]Always zero.

The following list describes each of the counters in more detail.

- **Time since last zeroed** The time that has elapsed since the counters were last zeroed. Provides a frame of reference for the other counters by indicating the amount of time they cover. For MOP V3, this time is the number of seconds. MOP V4 uses the UTC Binary Relative Time format.
- **Bytes received** The total number of user data bytes successfully received. This does not include Ethernet data link headers. This number is the number of bytes in the Ethernet data field, which includes any padding or length fields when they are enabled. These are bytes from frames that passed hardware filtering. When the number of frames received is used to calculate protocol overhead, the overhead plus bytes received provides a measurement of the amount of Ethernet

bandwidth (over time) consumed by frames addressed to the local system.

- **Bytes sent** The total number of user data bytes successfully transmitted. This does not include Ethernet data link headers or data link generated retransmissions. This number is the number of bytes in the Ethernet data field, which includes any padding or length fields when they are enabled. When the number of frames sent is used to calculate protocol overhead, the overhead plus bytes sent provides a measurement of the amount of Ethernet bandwidth (over time) consumed by frames sent by the local system.

- **Frames received** The total number of frames successfully received. These are frames that passed hardware filtering. Provides a gross measurement of incoming Ethernet usage by the local system. Provides information used to determine the ratio of the error counters to successful transmits.

- **Frames sent** The total number of frames successfully transmitted. This does not include data link generated retransmissions. Provides a gross measurement of outgoing Ethernet usage by the local system. Provides information used to determine the ratio of the error counters to successful transmits.

- **Multicast bytes received** The total number of multicast data bytes successfully received. This does not include Ethernet data link headers. This number is the number of bytes in the Ethernet data field. In conjunction with total bytes received, provides a measurement of the percentage of this system's receive bandwidth (over time) that was consumed by multicast frames addressed to the local system.

- **Multicast frames received** The total number of multicast frames successfully received. In conjunction with total frames received, provides a gross percentage of the Ethernet usage for multicast frames addressed to this system.

- **Frames sent, initially deferred** The total number of times that a frame transmission was deferred on its first transmission attempt. In conjunction with total frames sent, measures Ethernet contention with no collisions.

- **Frames sent, single collision** The total number of times that a frame was successfully transmitted on the second attempt after a normal collision on the first attempt. In conjunction with total frames sent, measures Ethernet contention at a level where there are collisions, but the backoff algorithm still operates efficiently.

- **Frames sent, multiple collisions** The total number of times that a frame was successfully transmitted on the third or later attempt after normal collisions on previous attempts. In conjunction with total frames sent, measures Ethernet contention at a level where there are collisions, and the backoff algorithm no longer operates efficiently.

  NOTE: *No single frame is counted in more than one of the above three counters.*

- **Send failures** The total number of times a transmit attempt failed. Each time the counter is incremented, a type of failure is recorded. When Read-counter function reads the counter, the list of failures is also read. When the counter is set to zero, the list of failures is cleared. In conjunction with total frames sent, provides a measure of significant transmit problems. All the problems reflected in this counter are also captured as events. Following are the possible failures. More information on their meanings and use can be found in the section on events.

- **Excessive collisions** Exceeded the maximum number of retransmissions due to collisions. Indicates an overload condition on the Ethernet.

- **Carrier check failed** The data link did not sense the receive signal that is required to accompany the transmission of a frame. Indicates a failure in either the transmitting or receiving hardware. Could be caused by either transceiver, transceiver cable, or a babbling controller that has been cut off.

- **Short circuit** There is a short somewhere in the local area network coaxial cable, or the transceiver or controller/transceiver cable has failed. This indicates a problem either in local hardware or global network. The two can be distinguished by checking to see if other systems are reporting the same problem.

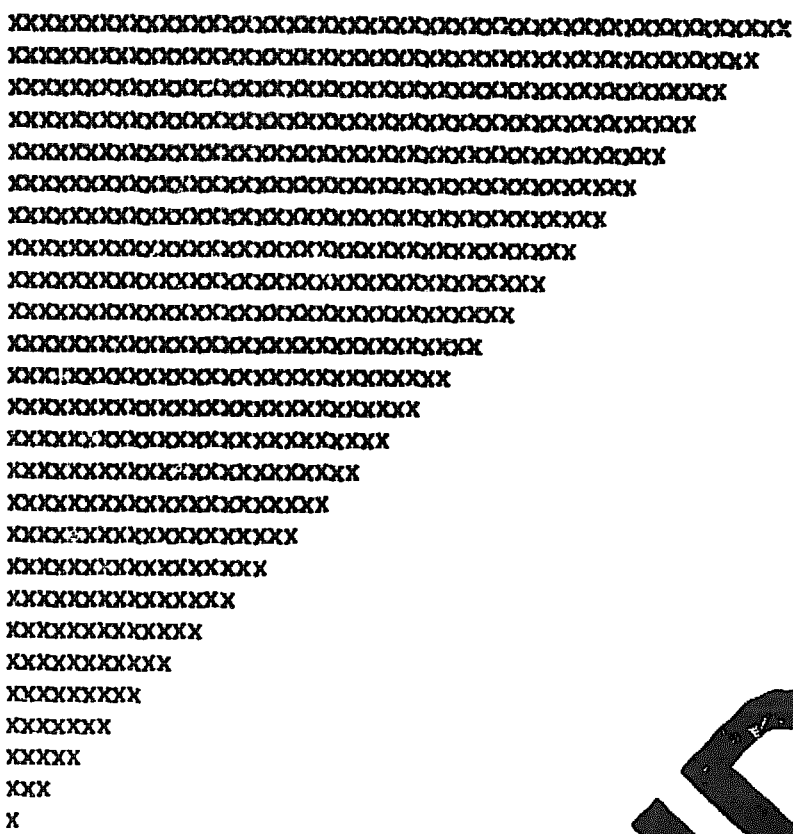
- **Open circuit** There is a break somewhere in the local area network coaxial cable. This indicates a problem either in local hardware or global network. The two can be distinguished by checking to see if other systems are reporting the same problem.

- **Frame too long** The controller or transceiver cut off transmission at the maximum size. This indicates a problem with the local system. Either it tried to send a frame that was too long or the hardware cut off transmission too soon.

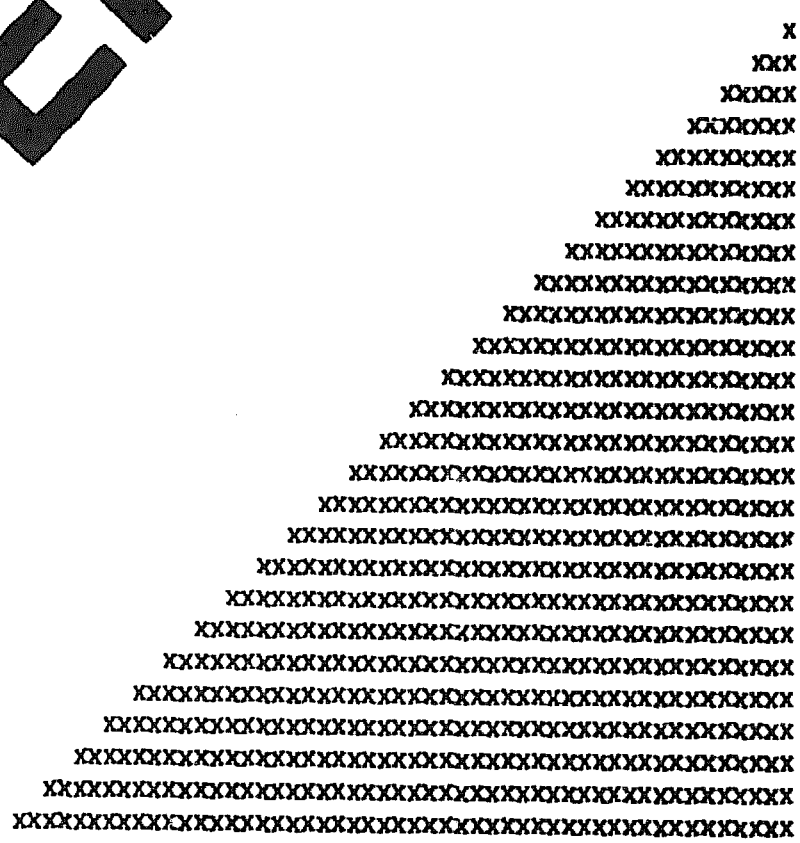
- **Remote failure to defer** A remote system began transmitting after the allowed window for collisions. This indicates either a problem with some other system's carrier sense or a weak transmitter.

- **Receive failures** The total number of frames received with some data error. Includes only data frames that passed either physical or multicast address comparison. This counter includes failure reasons in the same way as the send failure counter. In conjunction with total frames received, provides a measure of data-related receive problems. All the problems reflected in this counter are also captured as events. Following are the possible reasons. More information on their meaning and use can be found in the section on events.
- **Block check failure** A frame failed the CRC check. This indicates several possible failures, such as EMI, late collisions, or improperly set hardware parameters.
- **Framing error** The frame did not contain an integral number of 8-bit bytes. This indicates several possible failures, such as EMI, late collisions, or improperly set hardware parameters.
- **Frame too long** The frame was discarded because it was outside the Ethernet maximum length and could not be received. This indicates that a remote system is sending invalid length frames.
- **Unrecognized frame destination** The number of times a frame was discarded because there was no portal with the protocol type or multicast address enabled. This includes frames received for the physical address, the broadcast address, or a multicast address.
- **Data overrun** The total number of times the hardware lost an incoming frame because it was unable to keep up with the data rate. In conjunction with total frames received, provides a measure of hardware resource failures. The problem reflected in this counter is also captured as an event.
- **System buffer unavailable** The total number of times no system buffer was available for an incoming frame. In conjunction with total frames received, provides a measure of system buffer-related receive problems. The problem reflected in this counter is also captured as an event. This can be any buffer between the hardware and the user buffers (those supplied on Receive requests). Further information as to potential different buffer pools is implementation specific.
- **User buffer unavailable** The total number of times no user buffer was available for an incoming frame that passed all filtering. These are the buffers supplied by users on Receive requests. In conjunction with total frames received, provides a measure of user buffer-related receive problems. The problem reflected in this counter is also captured as an event.

- **Collision detect check failure** The approximate number of times that collision detect was not sensed after a transmission. If this counter contains a number roughly equal to the number of frames sent, either the collision detect circuitry is not working correctly or the test signal is not implemented.



APPENDIX H

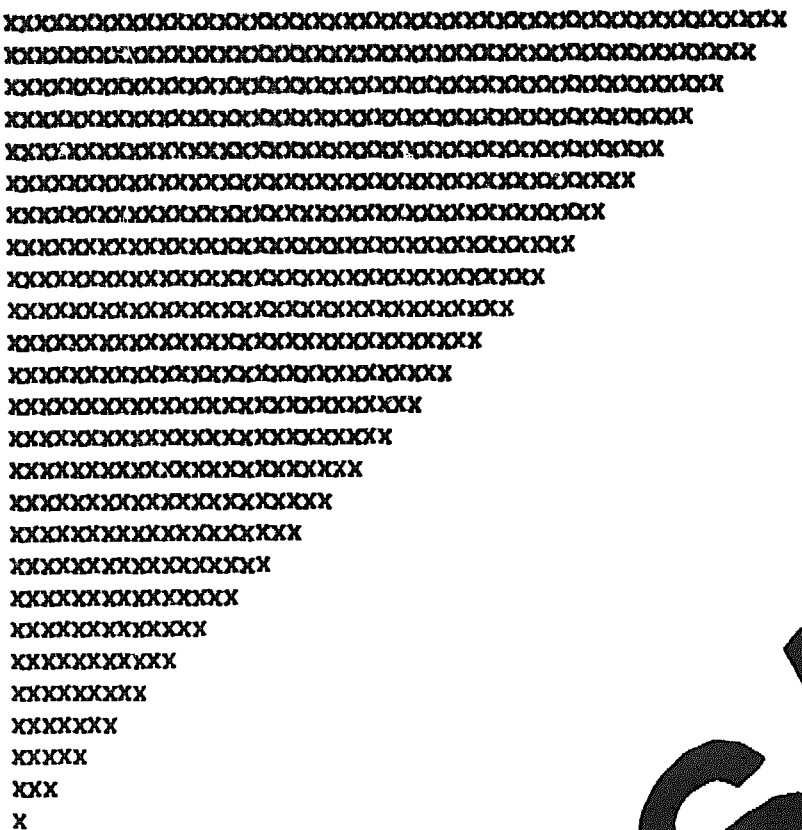


Appendix H
Related Documentation

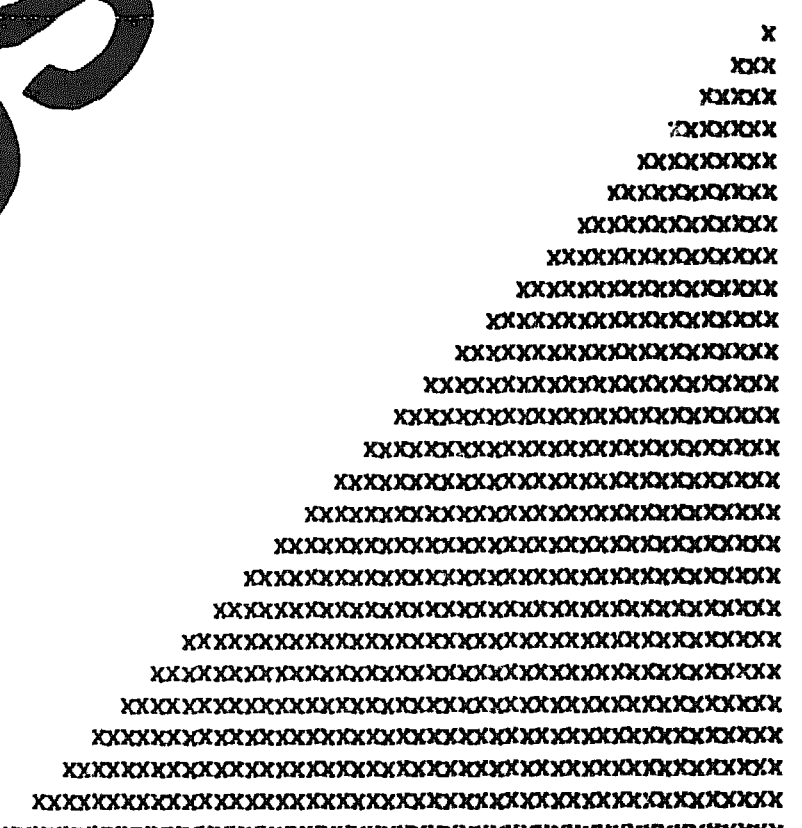
The following documents contain information relating to the maintenance of the KA670 CPU system.

Document Title	Order Number
MicroVAX Diagnostic Monitor User's Guide	AA-FM7A*-DN
Microsystems Options	EK-192A*-MG
KFQSA Installation Guide	EK-KFQSA-IN
KA670 Technical Manual	EK-KA670-TM
BA430/BA440 Enclosure Maintenance	EK-348A*-MG
RF30/RF71 Integrated Storage Element User's Guide	EK-RF71*-UG
RF31/RF72 Integrated Storage Element User's Guide	EK-RF72*-UG
TF85 Cartridge Tape Subsystem Owner's Manual	EK-TF85-OM

NOTE: * *Indicates the revision code. The latest revision will always be shipped when a manual is ordered.*



GLOSSARY



Glossary

BFLAG	Boot FLAGs is the longword supplied in the SET BFLAG and BOOT /R5: commands that qualify the bootstrap operation. SHOW BFLAG displays the current value.
BHALT	Q22-bus Halt signal is usually tied to the front panel Halt switch.
BIP	Boot In Progress flag in CPMBX<2>
CPMBX	Console Program MailBoX is used to pass information between operating systems and the firmware.
CQBIC	CVAX to Q22-Bus Interface Chip
DCOK	Q22-bus signal indicating dc power is stable. This signal is usually tied to the front panel restart switch.
DE	Diagnostic Executive is a component of the ROM-based diagnostics responsible for set-up, execution, and clean-up of component diagnostic tests.
DNA	Digital Network Architecture
EPROM	Erasable Programmable Read-Only Memory is used on some products to store firmware. Commonly used synonyms are PROM or ROM. Erasable by using ultraviolet light.
Firmware	Firmware in this document refers to instruction code residing at physical address 20040000 on the KA670. and diagnostics, bootstraps, console, and halt entry/exit code.
GPR	General Purpose Registers on the KA670 are the sixteen standard VAX longword registers R0 through R15. The last four registers, R12 through R15, are also known by their unique mnemonics AP (Argument Pointer), FP (Frame Pointer), SP (Stack Pointer), and PC (Program Counter), respectively.
IPL	Interrupt Priority Level ranges from 0 to 31 (0 to 1F hex).
IPR	Internal Processor Registers on the KA670 are those implemented by the processor chip set. These longword registers are only accessible with the instructions MTPR (Move To Processor Register) and MFPR (Move From Processor Register) and require kernel mode privileges. This document uses the prefix "PR$_" when referencing these registers.

ISE	Integrated storage element. An intelligent disk drive used on the Digital Storage Systems Interconnect.
KA670	CPU processor module with integral DSSI ports (2), Ethernet adapter, Q-bus interface, and 128-Kbyte backup cache.
LED	Light Emitting Diode
MOP	Maintenance Operations Protocol specifies message protocol for network loopback assistance, network bootstrap, and remote console functions.
MSCP	Mass Storage Control Protocol is used in Digital disks and tapes.
msec	millisecond (10e-3 seconds)
PC	Program Counter or R15
PCB	Process Control Block is a data structure pointed to by the PR$_PCBB register and contains the current process' hardware context.
PFN	Page Frame Number is an index of a page (512 bytes) of local memory. A PFN is derived from the bit field <23:09> of a physical address.
PR$_ICCS	Interval Clock Control and Status, IPR 24
PR$_IPL	Interrupt Priority Level, IPR 18
PR$_MAPEN	Memory management MAPping ENable, IPR 56
PR$_PCBB	Process Control Block Base register, IPR 16
PR$_RXCS	R(X)eceive Console Status, IPR 32
PR$_RXDB	R(X)eceive Data Buffer, IPR 33
PR$_SAVISP	SAVed Interrupt Stack Pointer, IPR 41
PR$_SAVPC	SAVed Program Counter, IPR 42
PR$_SAVPSL	SAVed Program Status Longword, IPR 43
PR$_SCBB	System Control Block Base register, IPR 17
PR$_SISR	Software Interrupt Summary Register, IPR 21
PR$_TODR	Time Of Day Register, IPR 27, is commonly referred to as the Time Of Year register or TOY clock.
PR$_TXCS	T(X)ransmit Console Status, IPR 34
PR$_TXDB	T(X)ransmit Data Buffer, IPR 35

PSL, PSW	Processor Status Longword is the VAX extension of the PSW (Processor Status Word). The PSW (lower word) contains instruction condition codes and is accessible by nonprivileged users; however, the upper word contains system status information and is accessible by privileged users.
QBMBR	Q22-Bus Map Base Register found in the CQBIC determines the base address in local memory for the scatter/gather registers.
QDSS	Q22-bus video controller for workstations
QMR	Q22-bus Map Register
QNA	Q22-bus Ethernet controller module
RAM	Random Access Memory
RIP	Restart In Progress flag in CPMBX<3>
RPB	Restart Parameter Block is a software data structure used as a communication mechanism between firmware and the operating system. Information in this block is used by the firmware to attempt an operating system (warm) restart.
SCB	System Control Block is a data structure pointed to by PR$_SCBB. It contains a list of longword exception and interrupt vectors.
SGEC	Second Generation Ethernet Chip
SHAC	Single Host Adapter Chip
SP	Stack Pointer or R14
SRM	Standard Reference Manual, as in *VAX SRM*
SSC	System Support Chip
SSC RAM	On the KA670, this is 1 Kbyte of battery backup RAM on the SSC.
µsec	microsecond (10e-6 seconds)
VMB	Virtual Memory Boot is the portion of the firmware dedicated to booting the operating system.


INDEX

Index

A

Acceptance testing, 4–27
Autoboot, description of, 3–13

B

Binary load and unload (X command), 3–48
Boot
  flags, 3–11
  naming device for, 3–13
  supported devices, 3–12
BOOT command, 3–24
Bootstrap
  conditions, 3–10
  initialization, 3–10
Break enable/disable switch
  disable setting, description of, 3–13
Bus length (DSSI), 2–11

C

Cabling
  DSSI, 2–11
  ISE, 2–11
Comment command (!), 3–50
! (comment command), 3–50
Configuration, 2–1
  and module order, 2–1
  DSSI, 2–5
CONFIGURE command, 2–4, 3–26
Console commands
  address space control qualifiers, 3–22
  address specifiers, 3–17
  binary load and unload (X), 3–48
Console commands (Cont.)
  BOOT, 3–24
  ! (comment), 3–50
  CONFIGURE, 3–26
  CONTINUE, 3–28
  data control qualifiers, 3–21
  DEPOSIT, 3–28
  EXAMINE, 3–29
  FIND, 3–30
  HALT, 3–31
  HELP, 3–31
  INITIALIZE, 3–33
  keywords, 3–22
  list of, 3–23
  MOVE, 3–34
  NEXT, 3–35
  qualifier and argument conventions, 3–17
  qualifiers, 3–21
  REPEAT, 3–37
  SEARCH, 3–37
  SET, 3–39
  SHOW, 3–43
  START, 3–47
  symbolic addresses, 3–18
  syntax, 3–17
  TEST, 3–47
  UNJAM, 3–48
  X (binary load and unload), 3–48
Console displays, 4–18
Console error messages
  sample of, 4–18
Console I/O mode
  restart caution, 3–5
  special characters, 3–16
Console port, testing, 4–36
CONTINUE command, 3–28

9C utility, 4–28, 4–35

D

DEPOSIT command, 3–28
Diagnostic executive, 4–3
  error field, 4–19
Diagnostics, RF-series, 4–40
Diagnostic tests
  list of, 4–4
  parameters for, 4–4
DRVEXR local program, 4–43
DRVTST local program, 4–41
DSSI
  bus length, 2–11
  bus termination, 2–11
  cabling, 2–11
  configuration, 2–5
  drive order, 2–5
  dual-host, 2–11
  node ID, 2–5
  node name, changing, 2–6
  unit number, changing, 2–8
DSSI VAXcluster, 2–11
Dual-host
  capability, 2–11

E

Entry and dispatch code, 3–3
ERASE local program, 4–45
Error messages
  console, sample of, 4–18
9E utility
  examples, 4–12
EXAMINE command, 3–29

F

FE utility, 4–32
FIND command, 3–30
Firmware, 3–1
  power-up sequence, 3–5
9F utility, 4–11

G

General purpose registers (GPRs)
  in error display, 4–21
  initialization of, 3–10
  symbolic addresses for, 3–18

H

H3103 loopback connector, 4–36
  testing serial line with, 3–6
H3604 I/O panel, 4–36
H3604 mode switch
  set to language inquiry, 3–6
  set to normal, 3–8
  set to test, 3–6
H8572 loopback connector, 4–36
Halt actions
  summary, 3–3
HALT command, 3–31
Halt protection, override, 4–33
Halts
  conditions for external, 3–4
  entry and dispatch code, 3–3
  registers saved, 3–3
  registers set to fixed values, 3–3
HELP command, 3–31
HISTRY local program, 4–44

I

INITIALIZE command, 3–33
Initial power-up test
  See IPR
IPR
  PR$_SID, B–3

K

KA670
  variants, 1–1

L

Language selection menu
  conditions for display of, 3–6
  example of, 3–7

Language selection menu (Cont.)
messages, list of, 3–6
Loopback connectors
H3103, 3–6, 4–36
H8572, 4–36
list of, 4–39
tests, 4–36

M

MCSR0–15, 4–28
Memory
acceptance testing of, 4–28
isolating FRU, 4–29, 4–33
testing, 4–33
Module
configuration, 2–4
order, in backplane, 2–1
self-tests, 4–39
MOP functions, G–2
MOVE command, 3–34

N

Network listening, G–1
NEXT command, 3–35

O

OCP
cabling, 2–11
Operator console panel
See OCP

P

Parameters
for diagnostic tests, 4–7
in error display, 4–20
PARAMS local program, 4–46
commands, 4–46
POST
errors handled by, 4–40
Power-on self test
See POST
Power-up
Power-up (Cont.)
memory layout, F–2
Power-up sequence, 3–5
Power-up tests, 3–5

R

RAM
CPMBX, C–2
partitioning, C–1
REPEAT command, 3–37
Restart caution, 3–5
Restart parameter block (PRB), 3–11
Restart sequence, 3–14
RF-series ISE
access to firmware through DUP, 2–10
cabling, 2–11
diagnostic error codes, 4–49
diagnostics, 4–40
errors, 4–40
RF-series ISE local programs
DRVEXR, 4–43
DRVTST, 4–41
ERASE, 4–45
HISTRY, 4–44
list of, 4–41
PARAMS, 4–46
ROM-based diagnostics, 4–3 to 4–49
and memory testing, 4–33
console displays during, 4–18
list of, 4–4
parameters, 4–4
utilities, 4–4
RPB
initialization, D–5
RPB, locating, 3–15

S

Scripts, 4–3, 4–8 to 4–16
calling sequence for, 4–10
creation of, using 9F utility, 4–11
list of, 4–8
SEARCH command, 3–37

Self-test, for modules, 4–39
Serial line test using H3103, 3–6
SET command, 3–39
SET HOST/DUP command, 3–39
SHOW command, 3–43
SID, B–3
START command, 3–47
Symbolic addresses, 3–18
  for any address space, 3–20
  for GPRs, 3–18

T

Termination power, tests for, 4–38
TEST command, 3–47
Tests, diagnostic
  list of, 4–4
  parameters for, 4–7
Troubleshooting, 4–32 to 4–49
  procedures, general, 4–1
  suggestions, additional, 4–35

U

UNJAM command, 3–48
Utilities, diagnostic, 4–4

V

Virtual memory bootstrap
  See VMB
VMB, 3–11
  boot flags, 3–11

X

X command (binary load and unload), 3–48