KA640 CPU System Maintenance

Order Number EK–179AA–MG–001

The READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	MicroVAX	ULTRIX
DECmate	MicroVMS	UNIBUS
DECnet	PDP	VAX
DECUS	P/OS	VAXBI
DECwriter	Professional	VAXELN
DELNI	Q-bus	VAXcluster
DEQNA	Rainbow	VAXstation
DESTA	RSTS	VMS
DIBOL	RSX	VT
MASSBUS	RT	Work Processor
MicroPDP–11	ThinWire	digital™

ML–S970

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.

Contents

Chapter 3   KA640 Firmware

Chapter 4   Troubleshooting and Diagnostics

Appendix A Address Assignments

Appendix B Related Documentation

Index

Examples

Figures

Tables

Preface

This guide describes the base system, configuration, ROM-based diagnostics, and troubleshooting procedures for systems containing the KA640 CPU.

Intended Audience

This guide is intended for use by DIGITAL Field Service personnel and qualified self-maintenance customers.

Organization

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

Chapter 1 describes the KA640/MS650 CPU and memory subsystem, and the RF30 disk drive.

Chapter 2 contains system configuration guidelines, and provides a table listing current, power, and bus loads for supported options. It also describes the DIGITAL Small Storage Interconnect (DSSI) bus interface cabling between the CPU, the CPU I/O panel, the operator console panel (OCP), and the RF30 disk drives.

Chapter 3 describes the firmware that resides in ROM on the KA640, and provides a list of console error messages and their meaning.

Chapter 4 describes the KA640 diagnostics, including an error message and FRU cross-reference table. It also describes diagnostics that reside on the RF30.

Appendix A lists the KA640 address space.

Appendix B is a list of related documentation. It contains the order numbers for all manuals mentioned in this manual.

Warnings, Cautions, and 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.

Chapter 1
KA640 CPU and Memory Subsystem

1.1 Introduction

This chapter describes the KA640 CPU (Figure 1–1). The KA640 is a quad-height VAX processor module for the Q22-bus (extended LSI–11 bus). It is designed for use in high-speed, real-time applications and for multiuser, multitasking environments. The KA640 employs a cache memory to maximize performance.

There are two variants: the KA640–AA, which runs multiuser software; and the KA640–BA, which runs single-user software.

The KA640 is used in two systems, the MicroVAX 3300 and the MicroVAX 3400. The MicroVAX 3300 is housed in a BA215 enclosure. The MicroVAX 3400 is housed in a BA213 enclosure. Refer to *BA215 Enclosure Maintenance* and *BA213 Enclosure Maintenance* for a detailed description of each enclosure.

CAUTION: *Static electricity can damage integrated circuits. Always use a grounded wrist strap (part no. 29–11762–00) and grounded work surface when working with the internal parts of a computer system.*

The KA640 CPU module and MS650 memory modules combine to form a VAX CPU and memory subsystem that uses the Q22-bus to communicate with I/O devices. The KA640 and MS650 modules mount in standard Q22-bus backplane slots that implement the Q22-bus in the AB rows and the CD interconnect in the CD rows. The KA640 can support up to three MS650 modules, if enough Q22/CD slots are available.

The KA640 communicates with the console device through the H3602–SA CPU I/O panel, which also contains configuration switches and an LED display. The H3602–SA is described in Section 1.3.

The KA640–AA module number (M7624) stamped on the handle varies slightly, depending on the vendor used for the RAM chips:

M7624–AL   KA640–AA, Hitachi chips
M7624–AF   KA640–AA, Toshiba chips
M7624–BL   KA640–BA, Hitachi chips
M7624–BF   KA640–BA, Toshiba chips

Figure 1–1:   KA640 CPU Module
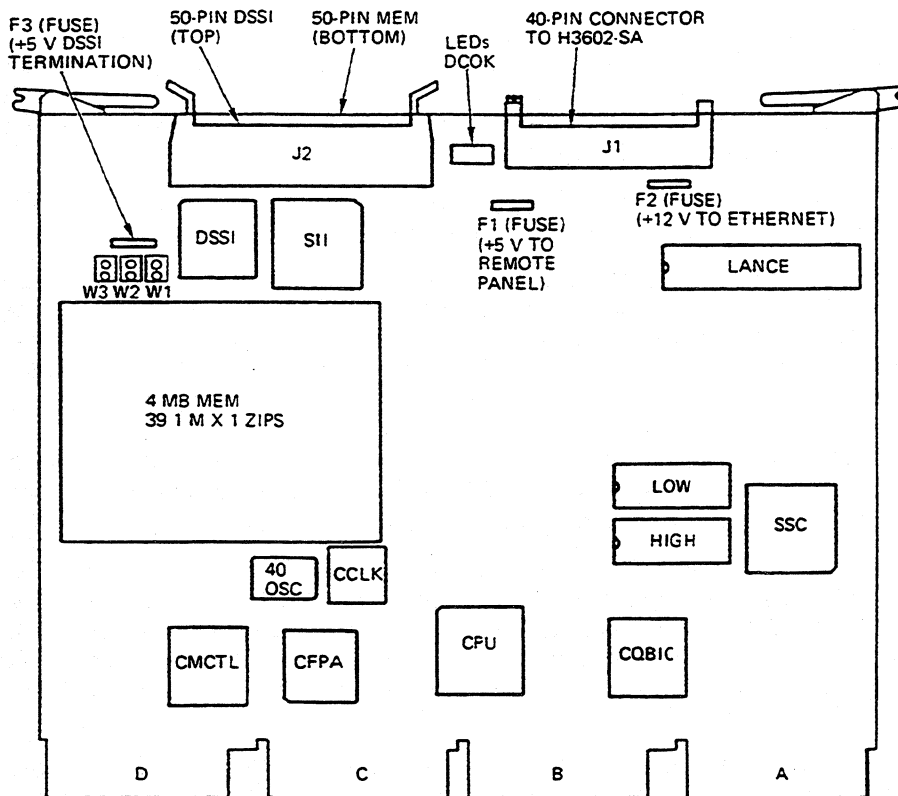
MLO–001280

1.2 KA640 Features

The major features of the KA640 CPU are listed below.

- The VAX central processor, which is implemented in a single VLSI chip called the CVAX. It achieves a 100 nanosecond (ns) microcycle and a 200 ns bus cycle at an operating frequency of 20 megahertz (MHz). It supports full VAX memory management with demand paging and a 4-Gbyte virtual address space.

- A floating point accelerator with the MicroVAX chip subset of the VAX floating point instruction set and data types.

- A 4-Mbyte, 400 ns, 39 bit-wide array (32-bit data and 7-bit ECC) implemented with 1 Mbit dynamic RAMs in zig-zag in-line packages (ZIPs).

- A console port compatible with the VAX processor whose baud rate can be set through an external switch on the H3602–SA.

- A set of processor clock registers that support:

  - A VAX standard time-of-year (TOY) clock with support for battery backup. (Batteries are located in the H3602–SA.)

  - An interval timer with 10 millisecond (ms) interrupts.

  - Two programmable timers, similar in function to the VAX standard interval timer.

- A boot and diagnostic facility with four on-board LEDs. This facility supports an external 4-bit display and configuration switches on the H3602–SA.

- 128 Kbytes of 16 bit-wide ROM.

- A Q22-bus interface.

- A DSSI bus interface.

- An Ethernet interface.

1.2.1 CVAX Chip

The CVAX chip contains all general purpose registers (GPRs) visible to the VAX processor, several system registers such as MSER, CADR, SCBB, the cache memory (1 Kbyte), and all memory management hardware, including a 28-entry translation buffer.

The CVAX chip supports the MicroVAX chip subset of the VAX instruction set and data types, plus the following string instructions:

CMPC3
CMPC5
LOCC
SCANC
SKPC
SPANC

The CVAX chip provides the following subset of the VAX data types:

Byte
Word
Longword
Quadword
Character string
Variable-length bit field

Support for the remaining VAX data types can be provided through macrocode emulation.

1.2.2 Clock Functions

Clock functions are implemented by the CVAX clock chip (CCLK). The CVAX clock chip is a 44-pin CERQUAD surface mount chip that contains approximately 350 transistors. It provides the following functions:

- Generates two MOS clocks for the CPU, the floating point accelerator, and the main memory controller

- Generates three auxiliary clocks for other TTL logic

- Synchronizes reset signal for the CPU, the floating point accelerator, and the main memory controller

- Synchronizes data ready and data error signals for the CPU, floating point accelerator, and the main memory controller

1.2.3 Floating Point Accelerator

The floating point accelerator is implemented by a chip called the CFPA. The CFPA chip contains approximately 60,000 transistors in a 68-pin CERQUAD surface mount package. It executes the VAX f_, d_, and g_floating point instructions (except for CLRx, MOVx, and TSTx), and accelerates the execution of MULL, DIVL, and EMUL integer instructions. The CFPA chip receives opcode information from the CVAX chip, and receives operands directly from memory or from the CVAX chip. The floating point result is always returned to the CVAX chip.

1.2.4 Cache Memory

The KA640 module incorporates a cache memory to maximize CPU performance. The cache is implemented within the CVAX chip. The cache is a 1-Kbyte, two-way associative, write-through cache memory, with a 100 nanosecond (ns) cycle time.

1.2.5 Memory Controller

The main memory controller is implemented by a VLSI chip called the CMCTL. The CMCTL contains approximately 25,000 transistors in a 132-pin CERQUAD surface mount package. It supports ECC (error correction code) memory, with a 400 ns cycle time for longword read transfers and a 600 ns cycle time for quadword transfers. It has 200 ns cycle time for unmasked longword writes and a 500 ns cycle time for masked longword writes.

The maximum amount of main memory supported by KA640 systems is 28 Mbytes. This memory resides on the KA640 module (4 Mbytes) and on one to three MS650–AA 8-Mbyte memory modules, depending on the system configuration. The MS650 modules communicate with the KA640 through the MS650 memory interconnect, which utilizes the CD interconnect and a 50-pin ribbon cable.

1.2.6 MicroVAX System Support Functions

System support functions are implemented by the System Support Chip (SSC). The SSC contains approximately 83,000 transistors in an 84-pin CERQUAD surface mount package. The SSC provides console and boot code support functions; operating system support functions; timers; and many extra features, including the following:

- Word-wide ROM unpacking

- 1-Kbyte battery backed-up RAM

- Halt arbitration logic

- A console serial line

- An interval timer with 10 millisecond (ms) interrupts

- A VAX standard time-of-year (TOY) clock with support for battery backup

- An IORESET register

- Programmable CDAL bus timeout

- Two programmable timers

- A register for controlling the diagnostic LEDs

1.2.7 Resident Firmware

The resident firmware consists of 128 Kbytes of 16 bit-wide ROM, located on two 27512 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 KA640 and MS650 modules

- Emulation of a subset of the VAX standard console (automatic or manual bootstrap, automatic or manual restart, and a simple command language for examining or altering the state of the processor)

- Booting from supported Q22-bus devices

- Multilingual capability

The firmware is described in detail in Chapter 3.

1.2.8 Q22-bus Interface

The Q22-bus interface is implemented by the CQBIC chip. The CQBIC chip contains approximately 40,870 transistors in a 132-pin CERQUAD surface mount package. It supports up to 16-word block mode transfers between a Q22-bus DMA device and main memory, and up to 2-word block mode transfers between the CPU and Q22-bus devices. It has a 500 ns cycle time for longword read transfers and an 800 ns cycle time for quadword read transfers. It has a 400 ns cycle time for unmasked longword writes and a 600 ns cycle time for masked longword writes. The Q22-bus interface contains the following:

- A 16-entry map cache for the 8,192-entry, scatter/gather map that resides in main memory, used for translating 22-bit Q22-bus addresses into 26-bit main memory addresses

- Interrupt arbitration logic that recognizes Q22-bus interrupt requests BR7–BR4

The Q22-bus interface handles programmed and power-up resets, and CPU halts (deassertion of DCOK).

The KA640–AA module contains 240 ohm termination for the Q22-bus.

1.2.9 KA640 Network Interface

The KA640 features an on-board network interface implemented through a LANCE chip, a 32K x 8 bit-wide static ROM, and two 32K x 8 bit static RAMs. This interface allows the KA640 to be connected to either a ThinWire or standard Ethernet cable, through the H3602–SA I/O panel.

The network interface includes four registers for control and status reporting, a 24-word transmit silo, and a 24-word receive silo. It also includes a word-wide 64-Kbyte buffer (two 32K x 8 static RAM chips). The DMA controller reads control information and writes status information to and from main memory. It also transfers data (one word per memory reference) between main memory and either the transmit or receive silo. The DMA controller can perform up to eight masked longword references before giving up the CDAL bus. Each reference takes 600 ns and contains either a byte or word of data. The minimum time between bus requests is 8 μsec.

1.2.10 KA640 DSSI Interface

The KA640 contains an SII chip, a DXX chip, and four 32K x 8-bit static RAMs that implement the DIGITAL Small Storage Interconnect (DSSI) bus interface. The DSSI interface allows the KA640 to transmit packets of data to, and receive packets of data from, up to seven other DSSI devices (RF-series disk drives or a second KA640 module). The DSSI bus improves system performance for two reasons:

- It is faster than the Q22-bus.

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

The physical characteristics of the DSSI bus are as follows:

    4 Mbytes per second bandwidth
    Distributed arbitration
    Synchronous operation
    Parity checking
    Six meter total bus length (includes internal and external cabling)
    Single-ended bus transceivers
    Maximum of eight nodes (KA640 counts as one)
    Eight data lines
    One parity line
    Eight control lines

Refer to the following sections for more information about the DSSI bus and disk drives:

Section 2.4   Setting and changing DSSI node names, addresses and unit numbers, dual host configuration rules.

Section 3.9.14   Console SET HOST command.

Section 4.4   DSSI drive acceptance testing.

Section 4.8   RF30 drive resident diagnostics and local programs.

1.3 H3602–SA I/O Panel

The H3602–SA (Figure 1–2) contains the console serial line connector, console baud rate switch, two Ethernet connectors and LEDs, hex LED display, and power-up mode switch. The switches are read by the firmware when the processor halts. For this reason changing the baud rate on the H3602–SA does not take effect until the next power-up or system reset. (By contrast, on the KA630, the switches are hardwired into the hardware.) The switches are also read when the power-up mode switch is in the test position. The H3602–SA has the following switches, connectors, and indicators:

- Baud rate select switch.

- Power-up mode select switch.

- Halt enable/disable switch from the console keyboard BREAK key or CTRL/P, depending on the state of SSCCR <15>. Break is the default.

  If this switch is set to the enable position, the system does not autoboot on power-up. It enters console I/O mode and displays the >>> prompt.

- Ethernet connector select. The H3602–SA has two connectors for Ethernet cable: a 15-conductor connector for standard Ethernet cable, and a male BNC connector for a ThinWire Ethernet coaxial cable. The H3602–SA contains a switch to select the Ethernet connector, and LEDs to indicate the selected connector and valid +12 vdc for that connector.

- Hex LED display, which provides a countdown of the system power-up self tests. See Table 4–6 for the meaning of this display.

Figure 1–2: H3602–SA I/O Panel

FRONT

REAR

HEX DISPLAY

POWER-UP MODE

BREAK ENABLE

CONSOLE CONNECTOR

ETHERNET CONNECTOR SELECT

BAUD RATE SELECT

CABLE TO KA640

MLO-001283
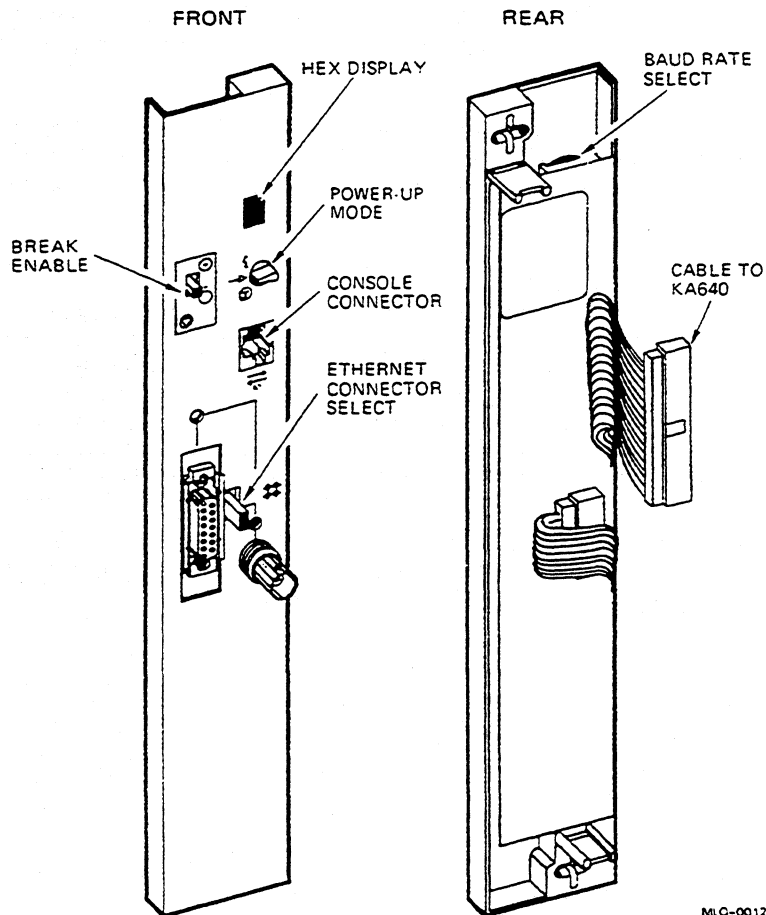
1.4 MS650–AA Memory Module

The MS650–AA memory module is a quad-height, Q22-bus module (Figure 1–3). The MS650–AA is an 8-Mbyte, 400 ns, 39 bit-wide array (32-bit data and 7-bit ECC) implemented with 256-Kbyte dynamic RAMs in zig-zag in-line packages (ZIPs).

The KA640 and MS650–AA memory modules are connected through the CD rows of backplane slots 1 through 4, and through a 50-conductor cable. The part number of this cable varies depending on the number of connectors, as follows:

Number of Connectors	CPU/Memory Configuration	Part Number
3	KA640 + 2 MS650–AA modules	17–01898–01
4	KA640 + 3 MS650–AA modules	17–01898–02[1]
5	KA640 + 3 MS650–AA modules	17–01898–03

[1]Recommended cable. Use five-connector cable only if this cable is not available.

The cable is keyed so that it is installed in the correct connector on the KA640 (the connector next to the module). The DSSI cable is attached to the connector "piggy backed" to the memory connector.

Figure 1–3: MS650–AA Memory Module
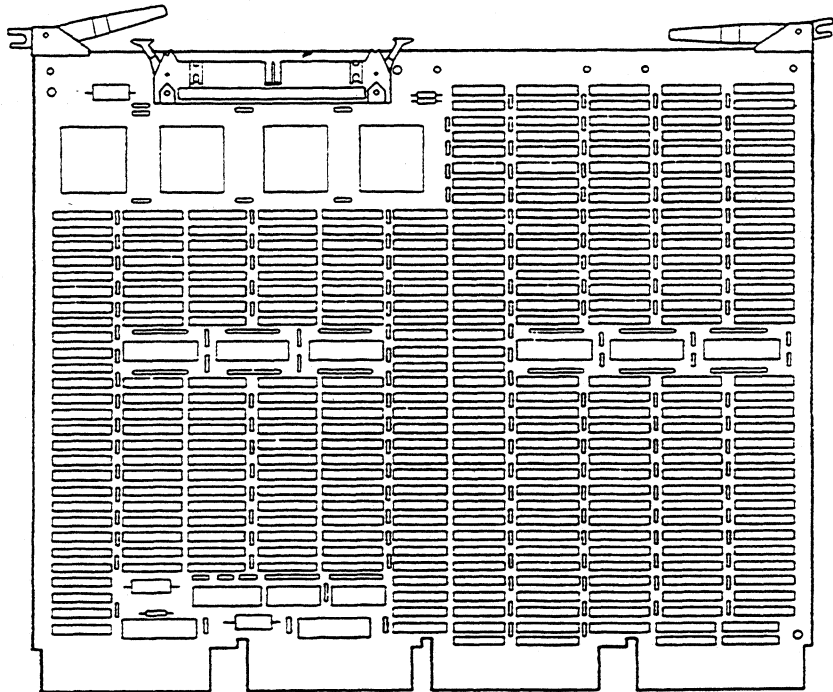
MLO–001284

1.5 RF30 Disk Drive

The RF30 is a half-height, 13.3-cm (5.25-in) fixed-disk drive for BA200-series enclosures. Table 1–1 lists the specifications for the RF30 drive.

Table 1–1: RF30 Specifications

Specifications	
Average seek time	22 milliseconds
Average rotational latency	8.33 milliseconds
Average access time	30.33 milliseconds
Peak transfer rate	12 Mbits/second
User capacity	150 Mbytes
User capacity (blocks)	293,040
Width	14.60 cm (5.75 in)
Depth	20.45 cm (8.25 in)
Height	4.40 cm (1.75 in)
Form factor	Standard 5.25-in footprint
Power requirements	+5 Vdc, 1.10 A
	+12 Vdc, 0.80 A
Power consumption	15.1 W
VMS support	Version 5.0–2A and later
ULTRIX–32 support	Version 3.0 and later
VAXELN support	Version 3.2 and later
MicroVAX Diagnostic Monitor support	Revision 2.3 and later

The RF30 disk drive is based on the DIGITAL Small Storage Interconnect (DSSI) architecture. DSSI supports up to seven storage devices, daisy-chained to the host system through the KA640 CPU or a host adapter module.

The disk drive controller is built into the RF30 drive, rather than being a separate module. This feature enables many drive functions to be handled without host-system or adapter intervention, resulting in improved I/O performance and throughput rates.

DSSI node ID switches are located on the electronics controller module. Set these switches to assign a unique node ID number to each drive on the DSSI bus. Refer to Table 2–2 for the correct DIP switch settings.

The RF30 disk drive contains a Ready indicator and a fault indicator.

The Ready indicator displays the activity status of the drive. It lights on power-up. After successful completion of the power-up diagnostics, the

indicator goes out, until the media heads are on the requested cylinder and the drive is read/write ready.

The Fault indicator lights at power-up. After successful completion of the power-up diagnostics, this indicator goes out. If the Fault indicator lights again after going out, a read/write safety error or a drive error condition has occurred.

Chapter 2

Configuration

2.1 Introduction

This chapter describes the guidelines for changing the configuration of a KA640 system, and for configuring a multihost system.

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

Module order in the backplane
Module configuration
Mass storage device configuration

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

Backplane
I/O panel
Power supply
Mass storage devices

2.2 General Module Order

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

- Relative use of devices in the system

- Expected performance of each device relative to other devices

- The ability of a device to tolerate delays between bus requests and bus grants (called delay tolerance or interrupt latency)

- The tendency of a device to prevent other devices farther from the CPU from accessing the bus

2.2.1 Module Order for KA640 Systems

Observe the following rules about module order:

- Install the KA640 CPU in slot 1.

- Install MS650 memory modules in slots 2, 3, and 4.

- Do not install dual-height modules in the CD rows.

The Q22-bus does not pass through the CD rows of the backplane in a BA200-series enclosure. Install all Q22-bus modules in the AB rows. Install dual-height grant cards in the AB rows only, or single-height grant cards in the A row only.

Here is the recommended module order in a KA640 system:

    KA640
    MS650
    AAV11–SA
    ADV11–SA
    AXV11–SA
    KWV11–SA
    TSV05–SA
    DELQA–SA
    DPV11–SA
    KMV1A–SA, –SB, –SC
    DFA01
    CXY08–AA
    CXB16–M
    CXA16–M
    LPV11–SA
    DRV1W–SA
    IEQ11–SA
    ADQ32–M
    DRQ3B–SA
    IBQ01–SA
    KLESI–SA
    TQK50–SA
    TQK70–SA
    M9060–YA

2.3 Module Configuration

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:

   ```
   >>> config
   Enter device configuration, HELP, or EXIT
   Device, Number? help
   Devices:
   ```

LPV11	KXJ11	DLV11J	DZQ11	DZV11	DFA01
RLV21	TSV05	RXV21	DRV11W	DRV11B	DPV11
DMV11	DELQA	DEQNA	RQDX3	KDA50	RRD50
RQC25	KXXXX-DISK	TQK50	TQK70	TU81E	RV20
KXXXX-TAPE	KMV11	IEQ11	DHQ11	DHV11	CXA16
CXB16	CXY08	VCB02	QDSS	DRV11J	DRQ3B
VSV21	IBQ01	IDV11A	IDV11B	IDV11C	IDV11D
IAV11A	IAV11B	MIRA	ADQ32	DTC04	DESQA
IGQ11					

   See the description of the CONFIGURE command in Chapter 3 (Section 3.9.2) for an example of obtaining the correct CSR addresses and interrupt vectors using this command.

   The LPV11–SA, which is the LPV11 version compatible with the BA200-series enclosures, has two sets of CSR address and interrupt vectors. To determine the correct values for an LPV11–SA, enter LPV11,2 at the DEVICE 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.

2.4 DSSI Configuration

Each device must have a unique DIGITAL Small Storage Interconnect (DSSI) node ID. The RF30 receives its node ID from a plug on the operator control panel (OCP) on the front panel. By convention, DSSI drives are mounted in the BA213 or BA215 enclosures from right to left, as listed in Table 2–1.

Table 2–1: DSSI Disk Drive Order

Device	Position	Node ID[1]
BA213 enclosure		
First	Right side	0
Second	Center	1
Third	Left side	2
BA215 enclosure[2]		
First	Right side	0
Second	Center	1

[1] KA640 node ID = 7
[2] BA215 OCP has three drive plugs, but only two drives. The third plug is blank.

If the cable between the RF30 and the OCP is disconnected, the RF30 reads the node ID from three DIP switches on its electronics controller module (ECM).

NOTE: *Pressing the system reset button on the front of a BA213 or BA215 power supply has no effect on the RF30 drives. You must perform a power cycle.*

The node ID switches are located behind the 50-pin connector on the ECM. Switch 1 (the MSB) is nearest to the connector. Switch 3 (the LSB) is farthest from the connector. Refer to the RF30 section in *Microsystems Options* for an illustration and further information. Table 2–2 lists the switch settings for the eight possible node addresses.

Table 2-2: RF30 DIP Switch Settings

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

The 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 RF30 through the MicroVAX Diagnostic Monitor (MDM), or through the VMS operating system (version 5.0) and console I/O mode SET HOST/DUP command. This 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 2.4.3 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.

2.4.1 Changing the Node Name

Each RF30 drive 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 RF30 drive at DSSI node address 1 is changed from R3YBNE to DATADISK.

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

Example 2-1: Changing a DSSI Node Name

```
>>> sho dssi
DSSI Node 0 (MDC)
-DIA0 (RF30)

DSSI Node 1 (R3YBNE)        !The node name for this drive will be
-DIA1 (RF30)                !changed from R3YBNE to DATADISK.

DSSI Node 7 (*)
>>>
>>> set host/dup/dssi 1
Starting DUP server...
Copyright  1988  Digital Equipment Corporation
DRVEXR V1.0  D   5-NOV-1988 15:33:06
DRVTST V1.0  D   5-NOV-1988 15:33:06
HISTRY V1.0  D   5-NOV-1988 15:33:06
ERASE  V1.0  D   5-NOV-1988 15:33:06
PARAMS V1.0  D   5-NOV-1988 15:33:06
DIRECT V1.0  D   5-NOV-1988 15:33:06
End of directory
Task Name? params
Copyright  1988  Digital Equipment Corporation

PARAMS> sho nodename

Parameter       Current           Default           Type      Radix
---------   ----------------   ----------------   --------   -----
NODENAME        R3YBNE            RF30              String    Ascii  B

PARAMS> set nodename datadisk

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

Stopping DUP server...
>>> sho dssi
DSSI Node 0 (MDC)
-DIA0 (RF30)

DSSI Node 1 (DATADISK)      !The node name has changed from
-DIA1 (RF30)                !R3YBNE to DATADISK.

DSSI Node 7 (*)
```

2.4.2 Changing the Unit Number

By default, the RF30 disk drive assigns the disk's unit number to the same value as the DSSI node address for that drive. This occurs whether the DSSI node address is determined from the OCP unit ID plugs or from the three DIP switches on the RF30 controller module.

RF30 drives conform to the DIGITAL Storage Architecture (DSA). Each drive can be assigned a unit number from 0 to 16,383 (decimal). The unit number does not have to be the same as the DSSI node address.

Example 2–2 shows how to change the unit number of a DSSI device. This example changes the unit number for the RF30 drive at DSSI node address 2 from 1 to 50 (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

```
>>> sho dssi
DSSI Node 0 (MDC)
-DIA0 (RF30)

DSSI Node 1 (R3QJNE)        !The unit number for this drive will be
-DIA1 (RF30)                !changed from 1 to 50 (DIA1 to DIA50).

DSSI Node 7 (*)
>>>
>>> set host/dup/dssi 1
Starting DUP server...
Copyright  1988  Digital Equipment Corporation
DRVEXR V1.0  D  5-NOV-1988 15:33:06
DRVTST V1.0  D  5-NOV-1988 15:33:06
HISTRY V1.0  D  5-NOV-1988 15:33:06
ERASE  V1.0  D  5-NOV-1988 15:33:06
PARAMS V1.0  D  5-NOV-1988 15:33:06
DIRECT V1.0  D  5-NOV-1988 15:33:06
End of directory
```

Example 2–2 Cont'd. on next page

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

```
Task Name? params
Copyright 1988  Digital Equipment Corporation

PARAMS> sho unitnum

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

PARAMS> sho forceuni

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

PARAMS> set unitnum 50

PARAMS> set forceuni 0

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

PARAMS> ex
Exiting...

Task Name?

Stopping DUP server...
>>>
>>>sho dssi
DSSI Node 0 (MDC)
-DIA0 (RF30)

DSSI Node 1 (R3QJNE)          !The unit number has changed
-DIA50 (RF30)                 !and the node ID remains at 1.

DSSI Node 7 (*)
```

2.4.3 Access to RF30 Firmware in VMS Through DUP

You can also access the RF30 firmware utilities from the VMS operating system as well as through the console commands described in Section 4.8.

NOTE: *Access the RF30 firmware through the VMS operating system to look up or to view parameter settings, but not to change them. To change RF30 parameter settings, enter the RF30 firmware through the console I/O mode SET HOST/DUP command.*

Load the FYDRIVER using the following commands in SYSGEN:

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

You can then access the RF30 firmware utilities using the following VMS command:

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

2.4.4 DSSI Cabling

A 50-conductor ribbon cable connects the RF30 drive to the DSSI bus (Figure 2–1). A separate 5-conductor cable carries +5 Vdc and +12 Vdc to the drive from the enclosure power supply.

A 2-conductor cable connects the fifth pin on the RF30 power connector to the operator control panel (OCP, Figure 2–2). In the BA213 enclosure, one of these (two) cables is for an RF30 connected to the right side power supply, and the other is for an RF30 connected to the left side supply.

These cables carry the ACOK signal (same as POK) to the RF30. The OCP delays this signal to one RF30 for each power supply to stagger the start-up of one of two possible devices attached to each supply. This delay prevents excessive current draw at power-up. The BA215 enclosure has only one power supply, but implements this signal delay in the same way.

The 50-conductor DSSI ribbon cable connects to a 50-conductor round cable that is routed through the bottom of the mass storage area to the DSSI connector on the KA640.

CAUTION: *When removing or installing new drives, be sure to connect the rightmost connector of the DSSI ribbon cable to the round cable connected to the KA640. Do not "T" the bus by connecting the round connector to any of the ribbon cable's center connectors.*

Figure 2–1: DSSI Cabling, BA213 Enclosure

FROM POWER
SUPPLY TO
RF30

FROM POWER
SUPPLY TO
RF30

TO RF2

DSSI
CABLE

TO RF3

TO RF2

TO
KA640
DSSI

TO RF1

TO RF1

'ACOK' SIGNAL,
STAGGERS RF30
POWER-UP

DSSI BUS
TERMINATION   TO RF3

TO
BACKPLANE

FROM
POWER
SUPPLY
TO RF30

DSSI
CABLE

KA640 DSSI
CONNECTOR
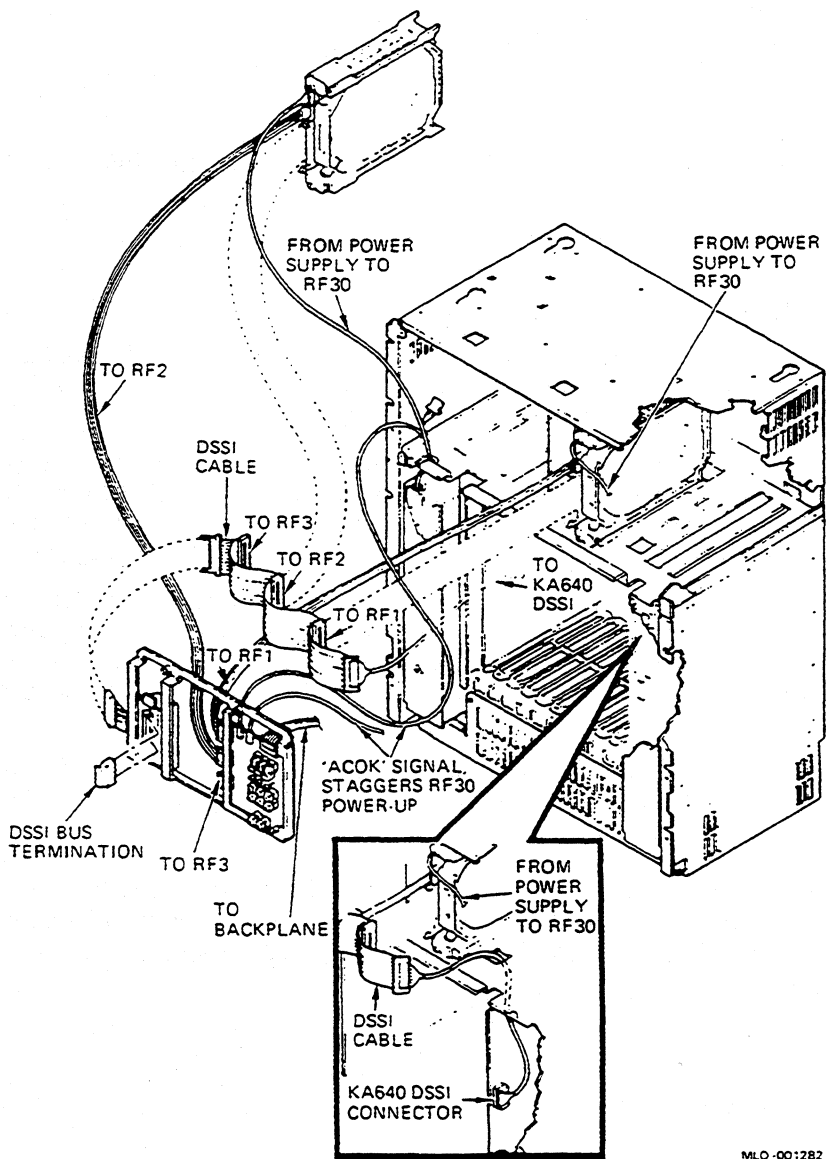
MLO-001282

Figure 2–2: RF30 OCP

FRONT

TO POK LEAD
POWER
SUPPLY
RESERVED FOR
FUTURE USE
10-PIN TO
BACKPLANE

10-PIN
TO RF1

DRIVE-SELECT
PLUGS

DRIVE FAULTS
(RED)

WRITE-PROTECT
BUTTONS

10-PIN
TO RF2

READY BUTTONS

10-PIN
TO RF3

DC OK (GREEN)

RESTART/RUN   CPU HALT

MLO-001281
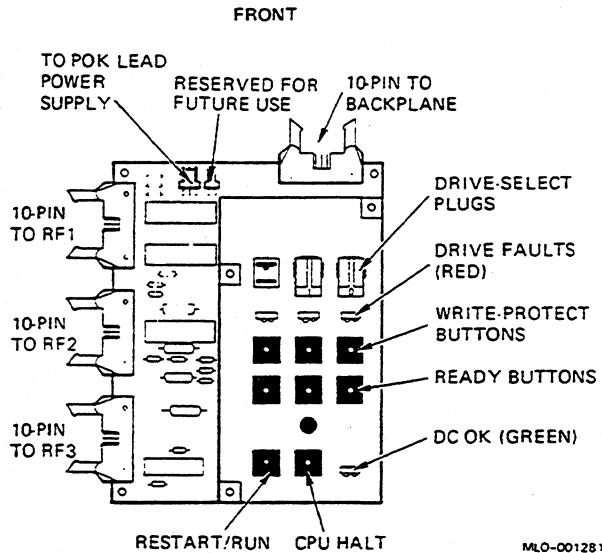
2.4.4.1 DSSI Bus Termination and Length

The DSSI bus must be terminated at both ends. The KA640 module terminates the DSSI bus at one end. A 50-conductor Honda connector on the left side of the media faceplate terminates the bus at the other end. This connector can be removed if you need to expand the bus.

The DSSI bus has a maximum length of 6 m (19.8 ft), including internal and external cabling.

In a dual-host system, the second KA640 module provides the bus termination.

2.4.5 Dual-Host Capability

A DSSI disk drive such as the RF30 has a multihost capability built into the firmware, which allows the drive to maintain connections with more than one DSSI adapter. Since the KA640 CPU has a built-in DSSI adapter, more than one KA640 CPU can be connected to the same DSSI bus, allowing each KA640 to access all other drives on the bus.

The primary application for such a configuration is a VAXcluster system using Ethernet as the interconnect medium between the boot and the

satellite members. This configuration improves system availability, as described below.

Two KA640 systems are connected through an external DSSI cable (BC21M). Each KA640 system is a boot member for a number of satellite nodes. The system disk resides in the first enclosure, and serves as the system disk for both KA640 systems. The KA640 in each enclosure has equal access to the system disk, and to any other DSSI disk in either enclosure.

If one of the KA640 modules fails, all satellite nodes booted through that KA640 module lose connections to the system disk. However, the multihost capability enables each satellite node to know that the system disk is still available through a different path—that of the remaining good KA640 module. A connection through that KA640 is then established, and the satellite nodes are able to continue operation.

Thus, even if one KA640 module fails, the satellites booted through it are able to continue operation. The entire cluster will run in a degraded condition, since one KA640 is now serving the satellite nodes of both KA640s. Processing can continue, however, until Field Service can repair the problem.

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

- System disk failure. If there is only one system disk, its failure causes the entire cluster to stop functioning until the disk failure is corrected. Disk failure can be caused by such factors as a power supply failure in the enclosure containing the disk.

- DSSI cabling failure. If a failure in one of the DSSI cables renders access to the disks impossible, the cable must be repaired in order to continue operation. Since the DSSI bus cabling is not redundant, a cable failure usually results in a system failure.

2.4.6 Dual-Host Configuration

Dual-host systems have the following configuration limitations:

- A maximum of two systems can be connected, because of cabling and enclosure limitations.

- The DSSI bus supports eight devices or adapters. Since a dual-host system has two KA640 modules, and each has a connection to the DSSI

bus, a maximum of six DSSI devices can be attached to the bus. Two variants are possible:

— Two BA213 enclosures, containing two KA640 CPUs, and six DSSI devices (three in each enclosure). This configuration uses all eight possible DSSI devices.

— Two BA215 enclosures, containing two KA640 CPUs, and four DSSI devices (two in each enclosure). This configuration uses six of eight possible DSSI devices.

• Set DSSI node IDs as follows:

— The first (or only) KA640 is 7.

— The second KA640 in a dual-host system is 6. Section 2.4.6.2 explains how to change the KA640 node ID.

— The remaining devices in a dual-host system are 0–5.

2.4.6.1 Allocation Class

When a KA640 system containing RF-series drives is configured in a cluster, either as a boot node or a satellite node, you must assign the allocation class in VMS SYSGEN and for the RF-series drive to matching nonzero values. To change the allocation class of the RF-series drive, use the following commands:

```
>>> SET HOST/DUP/DSSI <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..
>>>
```

2.4.6.2 Changing the KA640 Node ID

The KA640 node address is configured by three jumpers. Table 2–3 lists the jumper positions and node IDs. Figure 1–1 shows the location of the jumpers.

Table 2–3: Changing the KA640 Node ID

Node ID	W3	W2	W1
0	Out	Out	Out
1	Out	Out	In
2	Out	In	Out
3	Out	In	In
4	In	Out	Out
5	In	Out	In
6	In	In	Out
7	In	In	In

2.5 Configuration Worksheet

This section provides a configuration worksheet of the BA213 system enclosure (Figure 2–3). Use the worksheet to make sure the configuration does not exceed the system's limits for expansion space, I/O space, and power.

For the BA215 enclosure, use the top half of the BA213 enclosure worksheet, and allow for two disk drives instead of one.

Table 2–4 lists power values for supported devices. To check a system configuration, follow these steps:

1. List all the devices to be installed in the system.

2. Fill in the information from Table 2–4 for each device.

3. Add up the columns. Make sure the totals are within the limits for the enclosure.

In a BA213 enclosure, you must install a quad-height load module (M9060–YA) in one of backplane slots 7 through 12 if the continuous minimum current drawn on the second power supply is less than 5 amperes. If the minimum current of 5 amperes is not reached, the power supply enters an error mode and shuts down the system.

Table 2–4: Power and Bus Loads for KA640 Options

Option	Module	Current (Amps) +5 V	+12 V	Power Watts	Bus Loads AC	DC
AAV11–SA	A1009–PA	1.8	0.0	9.0	2.1	0.5
ADV11–SA	A1008–PA	3.2	0.0	16.0	2.3	0.5
AXV11–SA	A026–PA	2.0	0.0	10.0	1.2	0.3
CXA16–AA/–AF	M3118–YA	1.6	0.20	10.4	3.0	0.5
CXB16–AA/–AF	M3118–YB	2.0	0.0	10.0	3.0	0.5
CXY08–AA/–AF	M3119–YA	1.64	0.395	12.94	3.0	0.5
DELQA–SA	M7516–PA	2.7	0.5	19.5	2.2	0.5
DFA01–AA/–AF	M3121–PA	1.97	0.40	14.7	3.0	1.0
DPV11–SA	M8020–PA	1.2	0.30	9.6	1.0	1.0
DRQ3B–SA	M7658–PA	4.5	0.0	22.5	2.0	1.0
DRV1J–SA	M8049–PA	1.8	0.0	9.0	2.0	1.0
DRV1W–SA	M7651–PA	1.8	0.0	9.0	2.0	1.0
DSV11–SA	M3108–PA	5.43	0.69	38.0	3.6	1.0
DZQ11–SA	M3106–PA	1.0	0.36	9.3	1.4	0.5
IBQ01–SA	M3125–PA	5.0	0.0	25.0	4.6	1.0
IEQ11–SA	M8634–PA	3.5	0.0	17.5	2.0	1.0
KA640–AA	M7624–AA/–BA	6.0	0.24	32.88	3.5	1.0
KLESI–SA	M7740–PA	3.0	0.0	15.0	2.3	1.0
KMV1A–SA	M7500–PA	2.6	0.2	15.4	3.0	1.0
KWV11–SA	M4002–PA	2.2	0.13	11.15	1.0	0.3
LPV11–SA	M8086–PA	1.6	0.0	8.0	1.8	0.5
M9060–YA	–	5.3	0.0	26.5	0.0	0.0
MS650–AA	M7621–A	2.7	0.0	13.5	0.0	0.0
RF30	–	1.10	0.80	15.1	–	–
TK50E-EA	–	1.35	2.4	35.6	–	–
TK70E-EA	–	1.5	2.4	36.3	–	–
TQK50	M7546	2.9	0.0	14.5	2.8	0.5
TQK70–SA	M7559	3.5	0.0	17.5	4.3	0.5
TSV05–SA	M7196	6.5	0.0	32.5	3.0	1.0

Figure 2-3: BA213 Configuration Worksheet

RIGHT POWER SUPPLY

SLOT	MODULE	Current (Amps) +5 Vdc	+12 Vdc	Power (Watts)
1				
2				
3				
4				
5				
6				
MASS STORAGE:				
TK Drive				
FIXED DISK				
Total these columns:				
Must not exceed:		33.0 A	7.6 A	230.0 W

LEFT POWER SUPPLY

SLOT	MODULE	Current (Amps) +5 Vdc	+12 Vdc	Power (Watts)
7				
8				
9				
10				
11				
12				
MASS STORAGE:				
FIXED DISK(S) 1.				
2.				
Total these columns:				
Must not exceed:		33.0 A	7.6 A	230.0 W

MLO-001285

Chapter 3
KA640 Firmware

3.1 Introduction

This chapter describes the KA640 firmware, which gains control of the processor whenever the KA640 performs a processor halt. A processor halt transfers control to the firmware. The processor does not actually stop executing instructions.

3.2 KA640 Firmware Features

The firmware is located in two 64-Kbyte EPROMS on the KA640. The firmware address range is 20040000 to 2007FFFF, inclusive (20040000–2005FFFF in halt-protected space and 20060000–2007FFFF in halt-unprotected space) in the KA640 local I/O space. The firmware displays diagnostic progress and error reports on the KA640 LEDs and on the console terminal. It provides the following features:

- Automatic or manual restart or bootstrap of customer application images at power-up, reset, or conditionally after processor halts. (Restart in this context is not the same as restarting or resetting the hardware.)

- Automatic or manual bootstrap 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 as the system console.

- Multilingual support. The firmware can issue system messages in several languages.

The processor must be functioning at a level able to execute instructions from the console program ROM for the console program to operate.

The firmware consists of the following major functional areas:

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

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

3.3 Halt Entry and Dispatch Code

The processor enters the halt entry code at physical address 20040000 whenever a halt occurs. The halt entry code saves machine state, then transfers control to the firmware halt dispatcher.

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

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

The halt entry code unconditionally sets the following registers to fixed values on any halt, to ensure that the console itself can run and to protect the module from physical damage.

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

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

When the processor reenters program mode, the saved registers are restored and any changes become operative only then. References to

processor memory are handled normally. The binary load and unload command (X, Section 3.9.19) cannot reference the console memory pages.

After saving the registers, the halt entry code transfers control to the halt dispatch code. The halt dispatch code determines the cause of the halt by reading the halt field (PR$_SAVPSL <13:08>), the processor halt action field (PR$_CPMBX <01:00>), and the break enable switch on the H3602–SA panel. Table 3–1 lists the actions taken, by sequence. If an action fails, the next action is taken, with the exception of bootstrap, which is not attempted after diagnostic failure.

Table 3–1: Actions Taken on a Halt

Breaks Enabled on H3602–SA	Power-up Halt[1]	Halt Action[2]	Action
T[3]	T	X	Diagnostics, halt
T	F	0	Halt
F	T	X	Diagnostics, bootstrap, halt
F	F	0	Restart, bootstrap, halt
X	F	1	Restart, halt
X	F	2	Bootstrap, halt
X	F	3	Halt

[1]Power-up halt: PR$_SAVPSL<13:08>=3
[2]Halt action: PR$_CPMBX<01:00>
[3]T = condition is true, F = condition is false, X = does not matter

3.4 External Halts

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

- The break enable switch is set to enable, and you press ⌷BREAK⌷ on the system console terminal.

- Assertion of the BHALT line on the Q-bus.

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

The KA640 cannot detect the deassertion of DCOK when in console I/O mode, so no action is taken. More important, however, the deassertion of DCOK destroys system state without notifying the firmware.

CAUTION: *Do not press the Restart button while in console I/O mode. Doing so will destroy system state without notifying the firmware.*

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

The halt dispatch code distinguishes between DCOK deasserted and BHALT by assuming that BHALT must be asserted for at least 10 msec, and that DCOK is deasserted for at most 9 μsec. To determine if the BHALT line is asserted, the firmware steps out into halt-unprotected space after 9 msec. If the processor halts again, the firmware concludes that the halt was caused by the BHALT and not by the deassertion of DCOK. The firmware keeps a halt-in-progress flag to tell if it is halting because of stepping out into halt-unprotected space. This flag is cleared on power-up.

3.5 Power-Up Sequence

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

Power-up actions differ, depending on the state of the power-up mode switch on the H3602–SA (Figure 1–2). The mode switch has three settings: test, language inquiry, and normal. The differences are described in Sections 3.5.0.1 through 3.5.0.3.

The IPT waits for power to stabilize by monitoring SCR<5>(POK). Once power is stable, the IPT verifies that the console private nonvolatile RAM (NVRAM) is valid (backup battery is charged) by checking SSCCR<31>(BLO). If it is invalid or zero (battery is discharged), then the IPT tests and initializes the NVRAM.

After the battery check, the firmware tries to determine the type of terminal attached to the console serial line. If the terminal is a known type, it is treated as the system console.

3.5.0.1 Mode Switch Set to Test

Use the test position on the H3602–SA to verify that the connection between the KA640 and the console terminal is good.

- To test the console terminal, insert the H3103 loopback connector into the H3602–SA 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 (5 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.

3.5.0.2 Mode Switch Set to Language Inquiry

If the H3602–SA mode switch is set to language inquiry, or the firmware detects that the contents of NVRAM are invalid, the firmware prompts you for the language to be used for displaying the following system messages:

```
Loading system software.
Failure.
Restarting system software.
Performing normal system tests.
Tests completed.
Normal operation not possible.
Bootfile.
```

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.

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 NVRAM 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.

Example 3–1: Language Selection Menu

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

In addition, the console may prompt you for a default boot device. See Section 3.6.

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

3.5.0.3 Mode Switch Set to Normal

The console displays the language selection menu if the mode switch is set to normal and the contents of NVRAM are invalid.

The console uses the saved console language if the mode switch is set to normal and the contents of NVRAM are valid.

3.6 Bootstrap

The KA640 supports bootstrap of VAX/VMS, ULTRIX–32, VAXELN, and MDM diagnostics.

The firmware initializes the system to a known state before dispatching to the primary bootstrap (VMB), as follows:

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 ESA0.

4. Writes a form of this boot request, including active boot flags and boot device (BOOT/R5:0 ESA0, 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 the boot device name or 0 if none specified
R2	Length of PFN bitmap in bytes
R3	Address of PFN bitmap
R4	Time-of-day of bootstrap from PR$_TODR
R5	Boot flags
R10	Halt PC value
R11	Halt PSL value (without halt code and mapenable)
AP	Halt code
SP	Base of 128-Kbyte good memory block + 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.

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

Virtual Memory Bootstrap (VMB) is the primary bootstrap for VAX processors. The KA640 VMB resides in the firmware, and is copied into main memory before control is transferred to it. VMB then loads the secondary bootstrap image and transfers control to it.

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, earlier in this chapter.

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 a RIP (restart-in-progress) flag in CPMBX which it uses to avoid repeated attempts to restart a failing operating system. The operating system maintains an additional RIP flag in the RPB.

The firmware restarts the operating system in the following sequence:

1. Checks CPMBX<3>(RIP). If it is set, restart fails.

2. Prints the message `Restarting system software.` on the console terminal.

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 `Failure.` on the console terminal.

3.7.0.1 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

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 the RUBOUT key, 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 character being deleted. If you press additional rubouts, the additional deleted characters are echoed. If you type a non-rubout 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/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 3–6.

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 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 six address spaces:

Physical memory
Virtual memory
Protected memory
General purpose registers (GPR)
Internal processor registers (IPR)
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 3–3 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 3–3: Console Symbolic Addresses

Symbol	Address	Symbol	Address
GPR Address Space (/G)			
R0	0	R1	1
R2	2	R3	3
R4	4	R5	5
R6	6	R7	7
R8	8	R9	9
R10	0A	R11	0B
R12	0C	R13	0D
R14	0E	R15	0F
AP	0C	FP	0D
SP	0D	PC	0E
PSL	–	–	–
IPR Address Space (/I)			
pr$_ksp	00	pr$_esp	01
pr$_ssp	02	pr$_usp	03
pr$_isp	04	pr$_p0br	08
pr$_p0lr	09	pr$_p1br	0A
pr$_p1lr	0B	pr$_sbr	0C
pr$_slr	0D	pr$_pcbb	10
pr$_scbb	11	pr$_ipl	12
pr$_astlv	13	pr$_sirr	14
pr$_sisr	15	pr$_iccr	18
pr$_nicr	19	pr$_icr	1A
pr$_todr	1B	pr$_rxcs	20
pr$_rxdb	21	pr$_txcs	22
pr$_txdb	23	pr$_tbdr	24
pr$_cadr	25	pr$_mcesr	26
pr$_mser	27	pr$_savpc	2A
pr$_savpsl	2B	pr$_ioreset	37
pr$_mapen	38	pr$_tbia	39
pr$_tbis	3A	pr$_sid	3E
pr$_tbchk	3F	–	–

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

Symbol	Address	Symbol	Address
Physical Memory (/P)			
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_cdal	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	0080118	memcsr7	2008011c
memcsr8	20080120	memcsr9	20080124
memcsr10	20080128	memcsr11	2008012c
memcsr12	20080130	memcsr13	20080134
memcsr14	20080138	memcsr15	2008013c
memcsr16	20080140	memcsr17	20080144
nisarom	20084200	nirdp	20084400
nirap	20084404	nibuf	20120000
msi_sbb	20084600	msi_sc1	20084604
msi_sc2	20084608	msi_csr	2008460C
msi_id	20084610	msi_slcsr	20084614
msi_destat	20084618	msi_dstmo	2008461C
msi_data	20084620	msi_dmctrl	20084624
msi_cmlotc	20084628	msi_dmaddrl	2008462C
msi_dmaddrh	20084630	msi_dmabyte	20084634
msi_stlp	20084638	msi_ltlp	2008463C
msi_ilp	20084640	msi_dsctrf	20084644
msi_cstat	20084648	msi_dstat	2008464C
msi_comm	20084650	msi_dictrl	20084654

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

Symbol	Address	Symbol	Address
Physical Memory (/P)			
msi_clock	20084658	msi_bhdiag	2008465C
msi_sidiag	20084660	msi_dmdiag	20084664
msi_mcdiag	20084668	msi_ram	20100000

Table 3–4 lists symbolic addresses that can be used in any address space.

Table 3–4: 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 Command Qualifiers

You can enter console command qualifiers in any order on the command line after the command keyword. There are three types of qualifiers: data control, address space control, and command specific. Table 3–5 lists and describes the data control and address space control qualifiers. Command specific qualifiers are described in the command descriptions.

Table 3-5: 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. Used to override or set error bits when referencing main memory. On writes, use the complement. On reads, ignore ECC errors.
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.

3.8.5 Console Command Keywords

Table 3–6 lists command keywords by type. Table 3–7 lists the parameters, qualifiers, and arguments for each console command. Parameters, used with the SET and SHOW commands only, are listed in the first column along with the command.

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

Table 3–6: Command Keywords by Type

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

Table 3–7: Console Command Summary

Command	Qualifiers	Argument	Other(s)
BOOT	/R5:{bitmap} /{bitmap}	[device_name]	–
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/INSTRUCTION	[address]	–
FIND	/MEM /RPB	–	–
HALT	–	–	–
HELP	–	–	–
INITIALIZE	–	–	–

Table 3–7 (Cont.): Console Command Summary

Command	Qualifiers	Argument	Other(s)
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/NOT	{start_address}	{pattern} [mask]
SET BFLAG	–	{bitmap}	–
SET BOOT	–	{device_string}	–
SET HOST	/DUP {/DSSI n /UQSSP} {/DISK n /TAPE n csr_address} /MAINTENANCE /UQSSP {/SERVICE n csr_address}	{node} n {controller_number}	[task]
SET LANGUAGE	–	{language_type}	–
SHOW BFLAG	–	–	–
SHOW BOOT	–	–	–
SHOW DEVICE	–	–	–
SHOW DSSI	–	–	–
SHOW ETHERNET	–	–	–
SHOW LANGUAGE	–	–	–
SHOW MEMORY	/FULL	–	–
SHOW QBUS	–	–	–
SHOW RLV12	–	–	–
SHOW UQSSP	–	–	–
SHOW VERSION	–	–	–
START	–	{address}	–
TEST	–	{test_number}	[test_argument]
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 the default boot device if none is specified. The console qualifies the bootstrap operation by passing a boot flags bitmap to VMB in R5.

Format:

BOOT [qualifier-list] [device_name]

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

Set the default boot device and boot flags with the SET BOOT and SET BFLAG commands. If you do not set a default boot device, the processor times out after 30 seconds and attempts to boot from the on-board Ethernet port, ESA0.

Qualifiers:

Command specific:

/R5:{bitmap}	A 32-bit hex value passed to VMB in R5. The console does not interpret this value. Use the SET BFLAG command to specify a default boot flags longword. Use the SHOW BFLAG command to display the longword. Table 3–8 lists the supported R5 boot flags.
/{bitmap}	Same as /R5:{bitmap}
{device_name}	A character string of up to 39 characters. Longer strings cause a VAL TOO BIG error message. Apart from length, the console makes no attempt to interpret or validate the device name. The console converts the string to uppercase, then passes VMB a string descriptor to this device name in R0. Table 3–9 lists the boot devices supported by the KA640–AA.

Table 3–8: VMB Boot Flags

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

3.9.1.1 Supported Boot Devices

Table 3–9 lists the boot devices supported by the KA640–AA CPU. The table correlates the boot device names expected in a BOOT command with the corresponding supported devices.

Boot device names consist of a device code 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 (0–16,383).

DSSI devices names may also include a node prefix, consisting of either a node number (0–7) or a node name (a string of up to eight characters), ending in a dollar sign ($).

Table 3–9: Boot Devices Supported by the KA640–AA

	Boot Name	Controller Type	Device Type(s)
Disk			
	[node$]DIAn	On-board DSSI	RF30, RF71
	DUcn	RQDX3 MSCP	RD52, RD53, RD54, RX33, RX50
		KDA50 MSCP	RA70, RA80, RA81, RA82, RA90
		KLESI	RC25
	DLcn	RLV21	RL01, RL02
Tape			
	[node$]MIcn	On-board DSSI	TF70
	MUcn	TQK50 MSCP	TK50
		TQK70 MSCP	TK70
		KLESI	TU81E
Network			
	ESA0	On-board Ethernet	–
	XQcn	DEQNA	–
		DELQA	–
PROM			
	PRA0	MRV11	–

Examples:

```
>>> show boot
0
>>> show bflag
ESA0
>>> b                    ! Boot using default boot flags and device.
(BOOT/R5:0 ESA0)

  2..
-ESA0

>>> b xqa0               ! Boot from XQA0 using default boot flags.
(BOOT/R5:0 XQA0)

  2..
-XQA0

>>> b/10                 ! Boot using supplied boot flag (4)
(BOOT/R5:10 ESA0)        ! and default device.

  2..
-ESA0

>>> boot /r5:220 xqa0    ! Boot using supplied boot flags
(BOOT/R5:220 XQA0)       ! (5 and 9) 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 KA640–AA CPU.

Format:

CONFIGURE

Example:

```
>>> configure
Enter device configuration, HELP, or EXIT
Device,Number? help
Devices:
  LPV11         KXJ11         DLV11J        DZQ11         DZV11         DFA01
  RLV21         TSV05         RXV21         DRV11W        DRV11B        DPV11
  DMV11         DELQA         DEQNA         RQDX3         KDA50         RRD50
  RQC25         KXXXX-DISK    TQK50         TQK70         TU81E         RV20
  KXXXX-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
Numbers:
  1 to 255, default is 1
Device,Number? rqdx3,2
Device,Number? dhv11
Device,Number? qdss
Device,Number? tqk50
Device,Number? tqk70
Device,Number? exit

Address/Vector Assignments
-772150/154 RQDX3
-760334/300 RQDX3
-774500/260 TQK50
-760444/304 TQK70
-760500/310 DHV11
-777400/320 QDSS
>>>
```

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
```

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, /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).

Examples:

```
>>> D/P/B/N:1FF 0 0      ! Clear first 512 bytes of physical memory.
>>> D/V/L/N:3 1234 5     ! Deposit 5 into four longwords starting
                         ! at virtual memory address 1234.
>>> D/N:8 R0 FFFFFFFF    ! Loads GPRs R0 through R8 with -1.
>>> D/N:200 - 0          ! Starting at previous address, clear 513
                         ! bytes.
>>> D/L/P/N:10/S:200 0 8  ! Deposit 8 in the first longword of
                          ! the first 17 pages in physical
                          ! memory.
```

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 examines 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, /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 of physical memory for an RPB. The search does not use the bitmap to qualify which pages are looked at. The command leaves the contents of memory unchanged.

Examples:

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

3.9.7 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 that must be filled in
                   with a syntactically correct value

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

Valid commands:
  DEPOSIT [qualifiers] <ADDRESS> [datum [datum]]
  EXAMINE [qualifiers] [address]
  MOVE [qualifiers] <ADDRESS> <ADDRESS>
  SEARCH [qualifiers] <ADDRESS> <PATTERN> [mask]
  SET BFLAG <BOOT_FLAGS>
  SET BOOT <BOOT_DEVICE>
  SET HOST/DUP/DSSI <NODE_NUMBER> [task]
  SET HOST/DUP/UQSSP </DISK  /TAPE> <CONTROLLER_NUMBER> [task]
  SET HOST/DUP/UQSSP <PHYSICAL_CSR_ADDRESS> [task]
  SET HOST/MAINTENANCE/UQSSP/SERVICE <CONTROLLER_NUMBER> [task]
  SET HOST/MAINTENANCE/UQSSP <PHYSICAL_CSR_ADDRESS> [task]
  SET LANGUAGE <LANGUAGE_NUMBER>
```

```
SHOW BFLAG
SHOW BOOT
SHOW DEVICE
SHOW DSSI
SHOW ETHERNET
SHOW LANGUAGE
SHOW MEMORY [/FULL]
SHOW QBUS
SHOW RLV12
SHOW UQSSP
SHOW VERSION
HALT
INITIALIZE
UNJAM
CONTINUE
START <ADDRESS>
REPEAT
X <ADDRESS> <COUNT>
FIND [/MEMORY or /RPB]
TEST [test_code [parameters]]
BOOT [/R5:<BOOT_FLAGS> or /<BOOT_FLAGS>] [boot_device]
NEXT [count]
CONFIGURE
HELP
>>>
```

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
CVAX cache	Disabled, all entries invalid
Instruction buffer	Unaffected
Console previous reference	Longword, physical, address 0
TODR	Unaffected
Main memory	Unaffected
General registers	Unaffected
Halt code	Unaffected
Bootstrap-in-progress flag	Unaffected
Internal restart-in-progress flag	Unaffected

The firmware clears all error status bits and initializes 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, /W, /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 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:

```
>>> ex pc
  G 0000000F 00000200
>>> next
    PC = 00000202
>>> next 4
    PC = 00000213
>>>
```

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

>>> dep pc 0

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

3.9.12 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 mask complement) = (data AND mask complement), where:

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

SEARCH reports the address under the following conditions:

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

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

Qualifiers:

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

```
>>> search /w/step:1/n:ffff 20040000 fe11
  P 20040002 FE11        ! Find all two-byte sequences in the
  P 200403C7 FE11        ! ROM that could be interpreted as a
  P 20040ECB FE11        ! "branch to self" (10$: brb 10$)
                         ! (brb assembles to FE11)

>>> d/n:10000 0 0
>>> d/l 555 aaaaaaaa
>>> search/p/b/not/n:ffff 0 0
  P 00000555 AA
  P 00000556 AA
  P 00000557 AA
  P 00000558 AA

>>> search /w/step:1/n:ffff 20040000 fe11
  P 20040002 FE11
  P 200403C7 FE11
  P 20040ECB FE11

>>> dep 1000 87654321 /1
>>> search /p/b/n:ffff 0 1 fe
  P 00001000 21
  P 00001001 43
  P 00001002 65
  P 00001003 87

>>> search /p/b/n:ffff/not 0 0 fe
  P 00001000 21
  P 00001001 43
  P 00001002 65
  P 00001003 87
```

3.9.14 SET

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

Format:

SET {parameter} {value}

Parameters:

BFLAG Set the default R5 boot flags. The value must be a hex number of up to 8 digits. See Table 3–8 under the BOOT command description for a list of the boot flags.

BOOT Set the default boot device. The value must be a valid device name as specified in the BOOT command description Section 3.9.1.

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

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

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

/DISK n—Specifies the disk controller number, where n is a number from 0 to 255. The resulting fixed address for n=0 is 20001468 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 DSSI controller module configuration values. Does not accept a task value.

/UQSSP—

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

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

Qualifiers: Listed in the parameter descriptions above.

Examples:

```
>>> set bflag 220     ! Sets boot flags 5 and 9 (See boot flag
                      ! table in the BOOT command description.)
>>> set boot dua0

>>> set host/dup/dssi 0
Starting DUP server...

DSSI Node 0 (SUSAN)
DRVEXR V1.0  D 25-APR-1988 10:01:35
DRVTST V1.0  D 25-APR-1988 10:01:35
HISTRY V1.0  D 25-APR-1988 10:01:35
ERASE  V1.0  D 25-APR-1988 10:01:35
PARAMS V1.0  D 25-APR-1988 10:01:35
DIRECT V1.0  D 25-APR-1988 10:01:35
Copyright © 1988  Digital Equipment Corporation

Task Name? params
Copyright © 1988  Digital Equipment Corporation

PARAMS> stat path
```

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

```
PARAMS> exit
Exiting...

Task Name?

Stopping DUP server...
```

```
>>> set host/dup/dssi 0 params
Starting DUP server...

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

PARAMS> show node

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

PARAMS> show allclass

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

PARAMS> exit
Exiting...

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

3.9.15 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.

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

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

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

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

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

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

MEMORY   Displays main memory configuration board by board.

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

QBUS   Displays all Q22-bus I/O addresses that respond to an aligned word read, and vector and device name information. For each address, the console displays the address in the VAX I/O space in hex, the address as it would appear in the Q22-bus I/O space in octal, and the word data that was read in hex.

             This command may take several minutes to complete. Press CTRL/C to terminate the command. During execution, the command disables the scatter/gather map.

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 if the device contains a bootable image.

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

VERSION   Displays the current firmware version.

Qualifiers: Listed in the parameter descriptions above.

Examples:

```
>>> show bflag
00000220

>>> show boot
XQA0

>>> show device
DSSI Node 0 (SUSAN)
-DIA0 (RF30)

DSSI Node 1 (KAREN)
-DIA1 (RF30)

DSSI Node 4 (WILMA)
-DIA4 (RF30)

DSSI Node 5 (BETTY)
-DIA5 (RF30)

DSSI Node 7 (*)

UQSSP Disk Controller 0 (772150)
-DUA4 (RD53)
-DUA5 (RX50)
-DUA6 (RX50)

UQSSP Tape Controller 0 (774500)
-MUA0 (TK50)

Ethernet Adapter
-ESA0 (AA-00-03-01-2E-3F)
```

```
>>> show dssi
DSSI Node 0 (SUSAN)
-DIA0 (RF30)

DSSI Node 1 (KAREN)
-DIA1 (RF30)

DSSI Node 4 (WILMA)
-DIA4 (RF30)

DSSI Node 5 (BETTY)
-DIA5 (RF30)

.DSSI Node 7 (*)

>>> show ether
Ethernet Adapter
-ESA0 (AA-00-03-01-2E-3F)

>>> show lang
English (United States/Canada)

>>> show memory
Memory 0: 00000000 to 003FFFFF, 4MB, 0 bad pages
Memory 1: 00400000 to 00BFFFFF, 8MB, 0 bad pages
Memory 2: 00C00000 to 013FFFFF, 8MB, 0 bad pages

Total of 20MB, 0 bad pages, 106 reserved pages

>>> show memory/full
Memory 0: 00000000 to 003FFFFF, 4MB, 0 bad pages
Memory 1: 00400000 to 00BFFFFF, 8MB, 0 bad pages
Memory 2: 00C00000 to 013FFFFF, 8MB, 0 bad pages

Total of 20MB, 0 bad pages, 106 reserved pages

Memory Bitmap
-013F2C00 to 013F3FFF, 10 pages

Console Scratch Area
-013F4000 to 013F7FFF, 32 pages

Qbus Map
-013F8000 to 013FFFFF, 64 pages

Scan of Bad Pages
```

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

Scan of Qbus Memory Space

>>> show uqssp
UQSSP Disk Controller 0 (772150)
-DUA4 (RD53)
-DUA5 (RX50)
-DUA6 (RX50)

UQSSP Tape Controller 0 (774500)
-MUA0 (TK50)

>>> show version
KA640-A V4.0, VMB 2.4
>>>
```

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.

Examples:

```
>>> 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
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

Examples:

```
>>> 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 non-zero, 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: !

Examples

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

Chapter 4
Troubleshooting and Diagnostics

4.1 Introduction

This chapter contains a description of KA640 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. *Microsystems Options* lists address and vector values for most options.

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. Nonstandard addresses can be selected, but they require a special setup for use with VMS drivers and MDM. See *MicroVAX Diagnostic Monitor User's Guide* for information about the CONNECT and IGNORE commands, which are used to set up MDM for testing nonstandard configurations.

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

If the system fails (or appears to fail) to boot the operating system, check the console terminal screen for an error message. If the terminal displays an error message, see Section 4.3. 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 connections:

- If no message appears, make sure the console terminal and the system are on. Check the on/off power switch on both the console terminal and the system. If the terminal has a DC OK LED, be sure it is on.

- Check the cabling to the console terminal.

- If you cannot get a display of any kind on the console terminal, try another terminal.

- If the system DC OK LED remains off, check the power supply and power supply cabling.

- Check the hex display on the H3602–SA. If the display is off, check the CPU module LEDs and the CPU cabling. If a hex error message appears on the H3602–SA or the module, see Section 4.3.

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

CPU
Memory
Mass storage
Communications devices

4.3 KA640 ROM-Based Diagnostics

The KA640 ROM-based diagnostic facility, rather than the MicroVAX Diagnostic Monitor (MDM), is the primary diagnostic tool for troubleshooting and testing of the CPU, memory, Ethernet, and DSSI subsystems. ROM-based diagnostics have significant advantages:

- Load time is virtually nonexistent.

- The boot path is more reliable.

- Diagnosis is done in a more primitive state. (MDM requires successful loading of the VAXELN operating system.)

The ROM-based diagnostics can indicate several different FRUs, not just the CPU module. For example, they can isolate one of up to three memory modules as FRUs. (Table 4–6 lists the FRUs indicated by ROM-based diagnostic error messages.)

The diagnostics run automatically on power-up. While the diagnostics are running, the LEDs on the H3602–SA 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. Qualified Field Service 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 KA640 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 also ensures that when the tests are run, the machine is left in a consistent and well-defined state.

4.3.1 Diagnostic Tests

Table 4–1 shows a list of the ROM-based tests and utilities. To get this listing, enter T 9E at the console prompt (T is the abbreviation of TEST). The column headings have the following meanings:

- Test is the test code or utility code.

- Address is the test or utility's base address in ROM. This address varies. The addresses shown are examples only. If a test fails, entering T FE displays diagnostic state to the console. You can subtract the base address of the failing test from the last_exception_pc to find the index into the failing test's diagnostic listing (available on microfiche).

- Name is a brief description of the test or utility.

- Parameters shows the parameters for each diagnostic test or utility. Tests accept up to ten parameters. The asterisks (*) represent parameters that are used by the tests but that you cannot specify individually. These parameters are encoded in ROM and are provided by the diagnostic executive.

Table 4–1: Test and Utility Numbers

Test	Address	Name	Parameters
C1	2004C96B	SSC RAM	*
C2	2004CB32	SSC RAM ALL	*
C5	2004CCA2	SSC regs	*
C6	2004CD9C	SSC_powerup	**********
C7	2004CE60	CBTCR timeout	***
34	2004CF1C	ROM logic test	*
33	2004CFE4	CMCTL_powerup	*
32	2004D02C	CMCTL regs	MEMCSR0_addr *********
91	2004D150	CQBIC_powerup	**
90	2004D1E2	CQBIC regs	*
80	2004D23B	CQBIC-memory	*********
60	2004D63D	Console serial	start_baud end_baud ******
61	2004D98A	Console QVSS	mark_not_present ***
62	2004DA38	Console QDSS	mark_not_present selftest_r0 selftest_r1 *****
63	2004DCCC	QDSS self-test	input_csr selftest_r0 selftest_r1 ******
51	2004DE33	CFPA	*****
52	2004E01F	Prog timer	which_timer wait_ time_us ***
53	2004E2EC	TOY clock	repeat_count_250ms_ea ****
54	2004E557	Virtual mode	****
55	2004E834	interval timer	*
56	2004_900	SII_ext_loopbck	***
5C	2004EC05	SII_initiator	******
5D	2004F8DA	SII target	*******
58	2005109A	DSSI reset	port_no time_secs *
5A	20051484	VAX CMCTL CDAL	dont_report_memory_bad repeat_count *
57	2005159C	SII_memory	incr test_pattern *****
5B	20051954	SII_registers	****
5E	20051A9C	NI_memory	incr data_pattern ***
5F	20051BEC	NI_test	do_extl ********
41	20052768	Board reset	**
42	200527EC	Check-for_intrs	***
44	200528E8	Cache_memory	addr_incr *********
45	20052C3C	Cache_mem_cqbic	start_addr end_addr addr_incr ****
46	20052F20	Cache1_diag_md	addr_incr *********
31	2005356C	MEM_setup_CSRs	**********
30	20053C6D	MEM_bitmap	*** mark_Hard_SBEs ******
4F	20053D69	MEM_data	start_add end_add add_incr cont_on_err ******
4E	20053F2E	MEM_byte	start_add end_add add_incr cont_on_err ******
4D	20054050	MEM_address	start_add end_add add_incr cont_on_err ******
4C	200541F9	MEM_ECC_error	start_add end_add add_incr cont_on_err ******

Table 4–1 (Cont.): Test and Utility Numbers

Test	Address	Name	Parameters
4B	20054595	MEM_maskd_errs	start_add end_add add_incr cont_on_err ******
4A	20054779	MEM_correction	start_add end_add add_incr cont_on_err ******
49	20054995	MEM_FDM_logic	*** cont_on_err ******
48	20054FCE	MEM_addr_shrts	start_add end_add * cont_on_err pat1 pat2 ****
47	2005540A	MEM_refresh	start end incr cont_on_err time_seconds *****
40	200555AC	MEM_count_errs	First_board Last_board ******* Soft_errs_allowed
9C	200557BD	List CPU regs	*
9D	20055FCC	Utilities	Expnd_err_msg get_mode init_LEDs clr_ps_cnt
9E	200560C6	List diags	*
9F	200560C6	Create script	******
81 .	200567E4	MSCP-QBUS test	IP_csr ******
82	200569AB	DELQA	device_num_addr ****

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

```
54  2004E557   Virtual mode   ******
30  20053C6D   MEM_bitmap     *** mark_Hard_SBEs ******
```

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

```
>>> T 54
```

The MEM_bitmap test on the second line accepts ten parameters, but you
can specify only the fourth one. To mark pages bad in the bitmap for single-
bit or multi-bit errors, enter a 1 in the fourth parameter field:

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

You must enter a value of either 0 (zero) or 1 (one) for the first three
parameters. (0 is used in this example.) The values have no effect on
the test; they are simply place holders for the first three parameters. You
do not have to specify a value for parameters that follow the user-defined
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, cache diagnostic space, or main memory, to enhance execution speed.

- What is to be shown, if anything, on the console.

- What is to be shown, if anything, in the LED display.

- What action to take on errors (halt, repeat, continue).

The power-up script runs every time the system is powered on. You can also invoke the power-up script at any time by entering T 0.

Additional scripts are included in the ROMs for use in manufacturing and engineering environments. Field Service 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 you should use the UNJAM and INITIALIZE commands, described in Chapter 3, before running an individual test. You do not need to use these commands on system power-up, however, because system power-up leaves the machine in a defined state.

Field Service personnel with a detailed knowledge of the KA640 hardware and firmware can also create their own scripts, by using the 9F utility. (See Section 4.3.4.)

Table 4–2 lists the scripts.

Table 4–2: Scripts Available to Field Service

Script[1]	Enter with TEST Command	Description
A0	A0	Soft script created by de_test9f. Enter T 9F to create.
A1	A1, AA, AB, AC, 0, 3	Common section of power-up script. Scripts AA, AB, and AC invoke this script at power-up.
A7	A7, A8	Memory test portion invoked by script A8. Reruns the memory tests without rebuilding and reinitializing the bitmap. Run script A8 once before running script A7 separately to allow mapping out of both single-bit and double-bit main memory ECC errors.
A8	A8	Memory acceptance. Running script A8 with script A7 tests main memory more extensively. It enables hard single-bit and multi-bit 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.
AA	AA, 0	Console SLU. Invokes scripts BA, BC and A1. Does not invoke any tests directly.
AC	AC, 3	Power-up. Invokes scripts BC and A1. Does not invoke any tests directly. Invoked at power-up.
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.
AF	AF	Console program. Resets busmap and resets caches.
BA	BA, 2, AA	Initial power-up script for console SLU before first console announcement. Invoked at power-up.
BC	BC, AA, AC, 0, 3	Called by scripts AA and AC. Provides console announcements. Invoked at power-up.

[1]Scripts A2–A6, B0–B3, and B5 are for manufacturing use. They should not be used by Field Service. Scripts AB and BB are used to test the QDSS, which is not supported at this time. Scripts BD, BE, BF, B4, and B6–B9 are not used.

In most cases, Field Service needs only the commands shown in Table 4–3 for effective troubleshooting and acceptance testing.

Table 4–3: Commonly Used Field Service Scripts

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

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. Runs the IPT.

2. Assumes console device is SLU.

3. Calls the diagnostic executive (DE) with Test Code = 2.

   a. DE determines that the environment is nonmanufacturing from H3602–SA. (Manufacturing sets a jumper on the H3602–SA for testing.)

   b. DE selects script sequence for console SLU.

   c. DE executes Script BA.

   — Script BA directs DE to execute test. (Console announcements are off.)

   d. DE passes control back to the CP.

4. Establishes SLU as console device (whether or not SLU test passed).

5. Prints banner message.

6. Displays language inquiry menu on console if console supports MCS *and* any of the following are true:

- Battery is dead.

- H3602–SA switch set to language inquiry.

- Contents of SSC NVRAM are invalid.

7. Calls DE with Test Code = 3

   a. DE executes Script AC.

   Script AC directs DE to execute scripts BC and A1.

   — Script BC directs DE to execute tests. (Console announcements are on.)

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

   b. DE passes control back to CP.

8. CP issues end message and >>> prompt.

Actions After You Enter T 0

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

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

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

   b. DE executes script AA.

   Script AA directs DE to execute scripts BA, BC, and A1.

   — Script BA directs DE to execute tests. (Console announcements are off.)

   — Script BC directs DE to execute tests. (Console announcements are on.)

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

   c. DE passes control back to the CP.

2. CP prints end message and >>> prompt.

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

4.3.4 User Created Scripts

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

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

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

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

    A script cannot, however, always be located in main memory. For example, a script that runs memory tests will overwrite the user script, since the diagnostic executive cannot relocate the user script to another

area. The diagnostic executive notifies you if you have violated this type of restriction by issuing a script incompatibility message.

- Test number

- Run environment. This defines where the actual diagnostic test can be run from. The choices are 0 = ROM, 1 = MSI RAM, 2 = Main Memory, and 3 = Fastest Possible. Choose number 3 to select the fastest possible data structure to run from that will not overwrite the test.

- Repeat code

- Error severity level

- Console error report

- Script error treatment

- LED display

- Console display

- Parameters

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

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

You cannot review or edit a script you have created.

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

```
>>>T 9F
SP=20140604
Create script in ?[0=SSC,1=SII_RAM,2=RAM] :1
Script starts at 2011FC00
1024 bytes left
Next test number :51
CFPA   >>Run from ?[0=ROM,1=SII_RAM,2=RAM,3=fastest possible] (0):
CFPA   >>Repeat? [0=no,1=on error,2=forever,>2=count<FF] (0):
CFPA   >>Error severity ? [0,1,2,3] (2):
CFPA   >>Console error report? [0=none,1=full] (1):
CFPA   >>Stop script on error? [0=NO,1=YES] (1):
CFPA   >>LED on entry (05):
CFPA   >>Console on entry (51):
1017 bytes left

Next test number :52
Prog timer >>Run from ?[0=ROM,1=SII_RAM,2=RAM,3=fastest possible] (0):
Prog timer >>Repeat? [0=no,1=on error,2=forever,>2=count<FF] (0):
Prog timer >>Error severity ? [0,1,2,3] (2):
Prog timer >>Console error report? [0=none,1=full] (1):
Prog timer >>Stop script on error? [0=NO,1=YES] (1):
Prog timer >>LED on entry (05):
Prog timer >>Console on entry (52):
Prog timer >> which_timer : 00000000 - 00000001 ?(00000000) 1
Prog timer >> wait_time_us : 00000001 - FFFFFFFF ?(0000000A)
1002 bytes left
Next test number :
>>>T A0
51..52..
>>>
```

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

1. Enter the test number (40 in Example 4–2) at the **Next test number:** prompt.

2. Enter A0 at the next **Next test number:** prompt.

3. Press <CR> at the next **Next test number:** prompt.

4. Enter T A0 to begin running the script repeatedly.

5. Press CTRL/C to stop the test.

The above 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 4–2: Listing and Repeating Tests with Utility 9F

```
>>> T 9F
SP=20140604
Create script in ?[0=SSC,1=SII_RAM,2=RAM] :0
Script starts at 20140758
24 bytes left
Next test number :?

Test
#   Address   Name            Parameters
C1  2004D587  SSC RAM          *
C2  2004D74E  SSC RAM ALL      *
C5  2004D8BE  SSC regs         *
"      "        "              "
"      "        "              "
"      "        "              "
47  20056166  MEM_Refresh      start end incr cont_on_err time_seconds *****
40  20056308  MEM_Count_Errs   First_board Last_board Soft_errs_allowed *******
9C  20056519  List CPU regs    *
9D  20056D28  Utilities        Expnd_err_msg get_mode init_LEDs clr_ps_cnt
9E  20056DFC  List diags       *
9F  20056E22  Create script    ******
81  20057540  MSCP-QBUS test   IP_csr ******
82  20057707  DELQA            device_num_addr ****
24 bytes left
 Next test number :40
MEM_Count_Errs>>Run from ?[0=ROM,1=SII_RAM,3=fastest possible] (0):
MEM_Count_Errs>>Repeat? [0=no,1=on error,2=forever,>2=count<FF] (0):
MEM_Count_Errs>>Error severity ? [0,1,2,3] (2):
MEM_Count_Errs>>Console error report? [0=none,1=full] (1):
MEM_Count_Errs>>Stop script on error? [0=NO,1=YES] (1):
MEM_Count_Errs>>LED on entry (04):
MEM_Count_Errs>>Console on entry (40):
MEM_Count_Errs>> First_board : 00000001 - 00000004 ?1
MEM_Count_Errs>> Last_board : 00000001 - 00000004 ?(00000004) 4
MEM_Count_Errs>> Soft_errs_allowed : 00000000 - FFFFFFFF ?2
5 bytes left
Next test number :A0 - script
4 bytes left
Next test number :

>>> T A0
40..40..40..40..40..40..40..40..40..40..40..40..40..40..40..4u..
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
>>>
Next test number :

>>> T A0
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..40..40..40..40..40..40..40..40..40..40..40..40..
^C
>>>
```

4.3.5 Console Displays

Example 4–3 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. Refer to Table 4–6 to see 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 4–3: Console Display (No Errors)

```
KA640-A V4.1, VMB 2.5
Performing Normal System Tests
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 (V4.1 in this example) and the virtual memory bootstrap (VMB) revision (V2.5 in this example).

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

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

Example 4–4: Sample Output with Errors

```
?46 2 07 FE 10 0002
P1=002F0000 P2=00000000 P3=00000000 P4=00FF0000   P5=00000000
P6=00000000 P7=00000000 P8=00000000 P9=00FF0000  P10=00000000
r0=00000000 r1=00010000 r2=55555555 r3=00000080   r4=AAAAAAAA
r5=00000080 r6=01EF0000 r7=20080144 r8=00010000  ERF=20140770

Tests completed
```

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

`Test  Severity  Error  De_error  Vector  Count`

- 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:

      FF—Normal error exit from diagnostic
      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 4–4).

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 Table 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 Tables 4–4 and 4–5.

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

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

Table 4–5: Values Saved, Exception During Executive

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

Lines 4 and 5 of the error printout are general registers R0 through R8 and the hardware error summary register.

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 4-6 lists the hex LED display, the default action on errors, and the most likely FRUs. It is divided into IPTs and scripts.

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

• The diagnostic executive detects an unexpected exception or interrupt.

• A test fails and that failure is reported to the diagnostic executive.

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

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

Figure 4-1 shows the LEDs on the KA640 CPU. They correspond to the hex display on the H3602-SA.
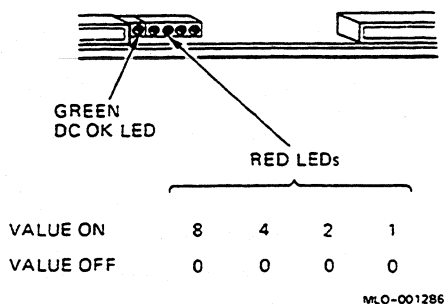
Figure 4-1: KA640 CPU Module LEDs


```
                        GREEN
                        DC OK LED

                                        RED LEDs

        VALUE ON          8      4      2      1
        VALUE OFF         0      0      0      0
                                        MLO-001286
```

Table 4–6: KA640 Console Displays and FRUs

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU
Initial Power-Up Tests					
F	None	None	Loop on test	Power-up	6, 1, 4, 5[1, 2]
D	None	None	Loop on test	WAIT_POK	1
4	None	None	Loop on self	Entering IPT	1
6	None	None	Loop on test	SLU_EXT_LOOPBACK[3]	7, 1
Script BA					
C	None	?9D	Continue	Utilities	1
B	None	?42	Continue	Check_for_intrs	1
C	None	?C6	Continue	SSC_power-up	1
6	None	?60	Continue	CONSOLE_SERIAL	1
End of script					

Script AC

Invoke script BC.
Invoke script A1.
End of script.

Script AA

Invoke script BA.
Invoke script BC.
Invoke script A1.
End of script.

[1]In the case of multiple FRUs, refer to Section 4.5.2 for further information.

[2]If a problem recurs with the same FRU, check that the tolerance for system power supply +5 Vdc, +12 Vdc, and AC ripple are within specification.

[3]This test runs only if the power-up mode switch on the H3602–SA is set to TEST mode. See Section 4.6.1.

FRU key:

1 = KA640, 2 = MS650, 3 = memory interconnect cable

4 = Q22-bus device, 5 = Q22/CD backplane, 6 = system power supply

7 = H3602–SA I/O panel

Table 4–6 (Cont.):   KA640 Console Displays and FRUs

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU
Script BC					
7	41	?91	Continue	CQBIC_power-up	1
7	40	?90	Continue	CQBIC_registers	1
9	39	?33	Continue	CMCTL_power-up	1
9	38	?32	Continue	CMCTL_registers	1
5	37	?5B	Continue	SII_registers	1
9	36	?31	Continue	CMCTL_setup_CSRs	1, 2, 3, 5
8	35	?49	Continue	MEMORY_FDM_logic	2, 1, 3, 5
End of script					
Script A1					
C	34	?52	Continue	PROG_TIMER_0	1
C	33	?52	Continue	PROG_TIMER_1	1
C	32	?53	Continue	TOY_CLOCK	1
C	31	?C1	Continue	SSC_RAM	1
C	30	?34	Continue	ROM_logic	1
C	29	?C5	Continue	SSC_registers	1
5	28	?57	Halt	SII_memory	1
C	27	?C2	continue	SSC_RAM_addr_shorts	1
B	26	?55	Continue	INTERVAL_TIMER	1
A	25	?51	Continue	CFPA	1
C	24	?C7	Continue	CBTCR_timeout	1
B	23	?46	Continue	CACHE_DIAG_MODE	1
5	22	?30	Halt	MEMORY_bitmap	1, 2, 3, 5
8	21	?4F	Continue	MEMORY_data	2, 1, 3, 5
8	20	?4E	Continue	MEMORY_byte	2, 1, 3, 5
8	19	?4D	Continue	MEMORY_addr	2, 1, 3, 5

FRU key:

1 = KA640, 2 = MS650, 3 = memory interconnect cable

4 = Q22-bus device, 5 = Q22/CD backplane, 6 = system power supply

7 = H3602–SA I/O panel

Table 4–6 (Cont.): KA640 Console Displays and FRUs

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU
Script A1					
8	18	?4C	Continue	MEMORY_ECC_error	2, 1, 3, 5
8	17	?4B	Continue	MEMORY_masked_errors	2, 1, 3, 5
8	16	?4A	Continue	MEMORY_correction	2, 1, 3, 5
8	15	?48	Continue	MEMORY_address_shorts	2, 1, 3, 5
8	14	?47	Continue	MEMORY_refresh	2, 1, 3, 5
8	13	?40	Continue	MEMORY_count_bad pages	2, 1, 3, 5
B	12	?44	Continue	CACHE1_MEMORY	1, 2, 3, 5
7	11	?80	Continue	CQBIC_MEMORY	1, 2, 4, 3, 5
B	10	?54	Continue	VIRTUAL_MODE	1, 2, 3, 5
7	09	?45	Continue	CACHE_MEM_CQBIC	1, 2, 4, 3, 5
9	08	?5A	Continue	CVAX_CMCTL drivers	1
5	07	?5C	Continue	SII_initiator	1
5	06	?5D	Continue	SII_target	1
4	05	?5E	Continue	NI_memory	1
4	04	?5F	Continue	NI_functional	1
C	03	?41	Continue	KA640_RESET	1, 4

End of script.

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU
Script A9					
8	4F	?4F	Halt	MEMORY_data	2, 1, 3, 5
8	4E	?4E	Halt	MEMORY_byte	2, 1, 3, 5
8	4D	?4D	Halt	MEMORY_addr	2, 1, 3, 5
8	4C	?4C	Halt	MEMORY_ECC_error	2, 1, 3, 5
8	4B	?4B	Halt	MEMORY_masked_errors	2, 1, 3, 5
8	4A	?4A	Halt	MEMORY_correction	2, 1, 3, 5
8	48	?48	Continue	MEMORY_address_shorts	2, 1, 3, 5
8	47	?47	Continue	MEMORY_refresh	2, 1, 3, 5
8	40	?40	Continue	MEMORY_count_bad pages	2, 1, 3, 5

FRU key:

1 = KA640, 2 = MS650, 3 = memory interconnect cable

4 = Q22-bus device, 5 = Q22/CD backplane, 6 = system power supply

7 = H3602–SA I/O panel

Table 4–6 (Cont.):   KA640 Console Displays and FRUs

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU
Script A9					
C	41	?41	Continue	KA640_RESET	1, 4
End of script.					
Script A8					
9	31	?31	Halt	CMCTL_setup_CSRs	1, 2, 3, 5
8	49	?49	Halt	MEMORY_FDM_logic	2, 1, 3, 5
5	30	?30	Halt	MEMORY_bitmap	1, 2, 3, 5
Invoke script A7.					
End of script.					
Script A7					
8	4F	?4F	Halt	MEMORY_data	2, 1, 3, 5
8	4E	?4E	Halt	MEMORY_byte	2, 1, 3, 5
8	4D	?4D	Halt	MEMORY_addr	2, 1, 3, 5
8	4C	?4C	Halt	MEMORY_ECC_error	2, 1, 3, 5
8	4B	?4B	Halt	MEMORY_masked_errors	2, 1, 3, 5
8	4A	?4A	Halt	MEMORY_correction	2, 1, 3, 5
8	48	?48	Halt	MEMORY_address_shorts	2, 1, 3, 5
8	47	?47	Halt	MEMORY_refresh	2, 1, 3, 5
8	40	?40	Cont	MEMORY_count_bad pages	2, 1, 3, 5
7	80	?80	Cont	CQBIC_memory	1, 2, 4, 3, 5
C	41	?41	Halt	KA640_RESET	1, 4
End of script.					

FRU key:

1 = KA640, 2 = MS650, 3 = memory interconnect cable

4 = Q22-bus device, 5 = Q22/CD backplane, 6 = system power supply

7 = H3602–SA I/O panel

4.3.6 System Halt Messages

Table 4–7 lists messages that may appear on the console terminal when a system error occurs.

Table 4–7: System Halt Messages

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

4.3.7 Console Error Messages

Table 4–8 lists messages issued in response to an error or to a console command that was entered incorrectly.

Table 4–8: Console Error Messages

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

4.3.8 VMB Error Messages

If VMB is unable to boot, it returns an error message to the console. Table 4–9 lists the error messages and their descriptions.

Table 4–9: VMB Error Messages

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

4.4 Acceptance Testing

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

KA640 module
MS650 module
Memory data interconnect cable
Backplane
DSSI drive
H3602–SA

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.

2. Make sure no solid single-bit ECC errors are in memory by entering the following commands:

   >>> T 30 0 0 0 1
   >>> T A1

   The first command runs test 30, which enables mapping out of solid single-bit and multi-bit ECC errors in main memory.

   The second command runs script A1, which invokes CPU and memory tests without resetting the bitmap to mark only solid multi-bit ECC errors in main memory. This command gives you a quick memory check, since most tests run on a 256-Kbyte boundary.

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

   >>> T A8
   >>> R T A7

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

   The second command runs script A7 repeatedly. This command runs the memory tests only and does not reset the bitmap. Press [CTRL/C] after two passes to terminate the script. This test takes up to 5 minutes per pass, depending on the amount of memory in the system. Most of the memory diagnostics test memory on a page boundary.

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

3. Check the memory configuration again, 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: 0000000 to 003FFFFF, 4MB, 0 bad pages
Memory 1: 00400000 to 00BFFFFF, 8MB, 0 bad pages

Total of 12MB, 0 bad pages, 102 reserved pages

Memory Bitmap
-00BF3400 to 00BF3FFF, 6 pages

Console Scratch Area
-00BF4000 to 00BF7FFF, 32 pages

Qbus Map
-00BF8000 to 00BFFFFF, 64 pages

Scan of Bad Pages
>>>
```

Memory 0 is the KA640 CPU. Memories 1 through 3 are the MS650 memory modules. The Q22-bus map always spans the top 32 Kbytes of good memory. The memory bitmap always spans two pages (1 Kbyte) per 4 Mbytes of memory configured.

Use utility 9C to compare the contents of configuration registers MEMCSR 0–15 with the memory installed in the system:

```
>>> T 9C

TOY  =018200B8  ICCS =00000000
TCR0 =00000000  TIR0 =00000000  TNIR0=00000000  TIVR0=00000078
TCR1 =00000000  TIR1 =00000000  TNIR1=00000000  TIVR1=0000007C
RXCS =00000000  RXDB =0000000D  TXCS =00000000  TXDB =00000030
CACR =FFB40080  MSER =00000000  CADR =0000000C
BDR  =FFFFFFD0  DLEDR=0000000C  SSCCR=00D45077  CBTCR=00000004
SCR  =0000C000  DSER =00000000  QBEAR=0000000F  DEAR =00000000
QBMBR=007F8000  IPCRn=0020

MEM_FRU 1 MEMCSR_0=80000015  1=00000015  2=00000015  3=00000015
MEM_FRU 2 MEMCSR_4=80400016  5=80800016  6=00000016  7=00000016
MEM_FRU 3 MEMCSR_8=00000000  9=00000000 10=00000000 11=00000000
MEM_FRU 4 MEMCSR12=00000000 13=00000000 14=00000000 15=00000000
          MEMCSR16=8094000F 17=00000000
```

One memory bank is enabled for each 4 Mbytes of memory. The MEMCSRs map modules as follows:

MEMCSR 0	KA640 CPU
MEMCSR 4–7	First MS650 memory module
MEMCSR 8–11	Second MS650 memory module
MEMCSR 12–15	Third MS650 memory module

Verify the following:

- For the KA640 CPU module, the bank enable bit (<31>) in MEMCSR_0 is set, indicating the memory bank on the KA640 is enabled. MEMCSR1-3<31> should always be clear, indicating banks are not enabled. MEMCSRs <6:0> should equal 15 hex for MEMCSR 0-3.

- If MS650-AA modules are installed, the bank enable bits are set in the first two MEMCSRs and cleared in the last two MEMCSRs. MEMCSRs <6:0> should equal 16 hex for all four MEMCSRs. See the values for MEMCSR 4-7 in the example.

- If a memory board is not present, bits <31:0> are all zeros for the corresponding group of four MEMCSRs. See the values for MEMCSR 8-11 in the example.

- Bits <25:22> should increment by one starting at zero in any group of four MEMCSRs whose bit <31> equals 1. In the example above, bits <25:22> of MEMCSR 4 and 5 increment by one, resulting in an increment of four in their longwords. If bit <31> equals 0, <25:22> should equal zero.

4. Check the Q22-bus and the Q22-bus logic in the KA640 CQBIC chip, and the configuration of the Q22-bus, as follows:

```
>>> show qbus
Scan of Qbus I/O Space
-20000120 (760440) = 0080 (300) DHQ11/DHV11/CXA16/CXB16/CXY08
-20000122 (760442) = F081
-20000124 (760444) = DD18
-20000126 (760446) = 0200
-20000128 (760450) = 0000
-2000012A (760452) = 0000
-2000012C (760454) = 8000
-2000012E (760456) = 0000
-20001920 (774440) = FF08 (120) DELQA/DEQNA
-20001922 (774442) = FF00
-20001924 (774444) = FF2B
-20001926 (774446) = FF06
-20001928 (774450) = FF16
-2000192A (774452) = FFF2
-2000192C (774454) = 00F8
-2000192E (774456) = 1030
-20001940 (774500) = 0000 (260) TQK50/TQK70/TU81E/RV20/K-TAPE
-20001942 (774502) = 0BC0
-20001F40 (777500) = 0020 (004) IPCR

Scan of Qbus Memory Space
>>>
```

The columns are described below. The examples listed are from the last line of the example above.

First column = the VAX I/O address of the CSR, in hex (20001F40).
Second column = the Q22-bus address of the CSR, in octal (777500).
Third column = the data, contained at the CSR address, in hex (0020).
Fourth column = the device vector in octal, according to the fixed or floating Q22-bus and UNIBUS algorithm (004).
Fifth column = the device name (IPCR, the KA640 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 Q-22 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 Q-22 bus interrupt logic

If you do not specify the CSR address, the test searches for and runs on the first MSCP device by default. To test the first TMSCP device, you must specify the first parameter:

```
>>> T 81 20001940
```

You can specify other addresses if you have multiple MSCP or TMSCP devices in the first parameter. This action may be useful to isolate a problem with a controller, the KA640, or the backplane. Use the VAX address provided by the SHOW QBUS command to determine the CSR value. If you do not specify a value, the MSCP device at address 20001468 is tested by default.

5. Check that all UQSSP, MSCP, TMSCP, and Ethernet controllers and devices are visible by typing the following command line:

```
>>> show device

DSSI Node 0 (R3YRME)
-DIA0 (RF30)

DSSI Node 1 (R3VBNC)
-DIA1 (RF30)

DSSI Node 7 (*)

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

Ethernet Adapter
-ESA0 (08-00-2B-08-E8-6E)

Ethernet Adapter 0 (774440)
-XQA0 (08-00-2B-06-16-F2)
```

In the above example, the console displays the remote DSSI node names and node numbers of two RF30 controllers it recognizes. The lines below each node name and number are the logical unit numbers of any attached devices, DIA0 and DIA1 in this case.

DSSI Node 7 (*) is the KA640 DSSI adapter. In most cases, the KA640 is the local DSSI node shown by the asterisk and has a node number of 7. DSSI node names, node numbers, and unit numbers should be unique.

The 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 KA640 Ethernet adapter.

The last two lines refer to DELQA and DEQNA controllers, the Q22-bus CSR address, logical name (XQA0), and the station address.

6. Test the DSSI subsystem using the KA640 ROM-based Diagnostics and Utilities Protocol (DUP) facility. This facility allows you to connect to the DUP server in the RF drive controller. Here are some examples:

```
>>> set host/dup/dssi 7
Starting DUP server...

Stopping DUP server...
```

In this example, a DUP connection was made with DSSI node 7, the KA640. The DUP server times out, since no local programs exist and no response packet was received.

```
>>> set host/dup/dssi 1
Starting DUP server...

DSSI Node 1 (R3VBNC)
DRVEXR V1.0  D 21-FEB-1988 21:27:54
DRVTST V1.0  D 21-FEB-1988 21:27:54
HISTRY V1.0  D 21-FEB-1988 21:27:54
ERASE  V1.0  D 21-FEB-1988 21:27:54
PARAMS V1.0  D 21-FEB-1988 21:27:54
DIRECT V1.0  D 21-FEB-1988 21:27:54
End of directory

Task Name? drvtst
Write/read anywhere on medium? [1=Yes/(0=No)]: <CR>
5 minutes for test to complete.
Compare failed on head  1 track 1091.
Compare failed on head  0 track  529.

Task Name? drvexr
Write/read anywhere on medium? [1=Yes/(0=No)]: <CR>
Test time in minutes? [(10)-100]:
10 minutes for test to complete.
R3VBNC::MSCP$DUP 21-FEB-1988 21:37:35 DRVEXR CPU=00:00:01.88 PI=43
R3VBNC::MSCP$DUP 21-FEB-1988 21:37:38 DRVEXR CPU=00:00:03.38 PI=79
Compare failed on head  1 track 1091.
R3VBNC::MSCP$DUP 21-FEB-1988 21:37:40 DRVEXR CPU=00:00:04.97 PI=116
^C
>>>
```

In the above example, the local programs DRVTST and DRVEXR are
run on drive 1. Do not enter 1 in response to the question Write/read
anywhere on medium?. Doing so will destroy data on the disk. Enter
<CR>, which uses the default, allowing reads and writes to the DBNs
only. CTRL/T or CTRL/G displays a message as shown in the DRVEXR
example above (the lines beginning with R3VBNC::). In the example,
CTRL/T has been pressed twice, to show the difference in the time and in
the value of the progress indicator (PI).

Press CTRL/C to terminate the program.

Use the local programs PARAMS (Section 4.8.5) and HISTRY
(Section 4.8.3) to determine the cause of errors displayed during
DRVTST or DRVEXR. DRVTST should run successfully for one pass
on each drive. Field Service can refer to the *RF30 Disk Drive Service
Manual* for details about the DUP local programs and corrective action.

7. If there are one or more DELQA modules in the system, use test 82 to
   invoke the Ethernet option's self-test and receive status from the host
   firmware. Test 82 is useful for acceptance testing if you cannot access
   the system enclosure to see the DELQA LEDs.

8. After the above steps have completed successfully, load MDM and run the system tests from the Main Menu. If they run successfully, the system has gone through its basic checkout and you can load the software.

4.5 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 that run after the IPTs and up to the primary console test fail, the major test code is displayed on the LEDs. Run the FE utility if the message, **Normal operation not possible**, is displayed after the tests have completed and there is no other error indication, or if you need more information than what is provided in the error display.

The FE utility dumps diagnostic state to the console (Example 4–5). This state indicates the major and minor test code of the test that failed, the ten parameters associated with the test, and the hardware error summary register.

Example 4–5: FE Utility Example

```
>>> T FE

bitmap=00BF3400, length=0C00, checksum=007E
busmap=00BF8000
return_stack=201406A8
subtest_pc=2004F4C4
timeout=00000001, error=0B, de_error=FE
de_error_vector=18, severity_code=02, total_error_count=0001
previous_error=FE0B5D5D, 00000000, 00000000, 00000000, 00000000
last_exception_pc=20050807
flags=01FFFD7F, test_flags=20
highest_severity=00
led_display=05
console_display=5D
save_mchk_code=80, save_err_flags=000000
param_1=00000100 2=00000100 3=000000F7 4=00000000  5=00000001
param_6=00000004 7=20050527 8=00000000 9=20140698 10=200521F4
```

The most useful fields displayed above are as follows:

- De_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.

- Previous_error. Contains the history of the last four errors. Each longword contains four bytes of information. From left to right these are the de_error, subtest_log, and test number (copied in both bytes).

- Save machine check code (save_mchk_code). Valid only if the test halts on error. This field has the same format as the hardware error summary register.

- Save error flags (save_err_flags). Valid only if the test halts on error. This field has the same format as the hardware error summary register.

- Parameters 1 through 10. Valid only if the test halts on error. The parameters have the same format as the hardware error summary register.

EF in the previous_error field indicates that an unexpected exception has occurred. If any of the tests that announce to the console fail, and the error code is EF, examine the last longword of the error printout. The last longword is the hardware error summary register and contains the machine check code (<31:24>) and KA640 error status bits (<23:0>). Table 4–10 lists the status bits.

Table 4–10: Hardware Error Summary Register

Bit	Register	Description
31	Machine check code	
30	Machine check code	
29	Machine check code	
28	Machine check code	
27	Machine check code	
26	Machine check code	
25	Machine check code	
24	Machine check code	
23	MSER <6>	; CDAL data parity error
22	MSER <5>	; Mchn chck CDAL parity error
21	MSER <4>	; Machine check cache parity
20	MSER <1>	; Cache data parity error
19	MSER <0>	; Cache tag parity error
18	Unused	
17	MEMCSR16 <31>	; Uncorrectable ECC error
16	MEMCSR16 <30>	; Two or more uncorrectable errors
15	MEMCSR16 <29>	; Correctable single-bit error
14	MEMCSR16 <25>	; Page address bits 25:22 of
13	MEMCSR16 <24>	; Location that caused error.
12	MEMCSR16 <23>	; These four bits point to the
11	MEMCSR16 <22>	; failing 4-Mbyte bank of memory.
10	MEMCSR16 <8>	; DMA read/write error.
9	MEMCSR16 <7>	; CDAL parity error on write.
8	CBCTR <31>	; CDAL bus timeout.
7	CBCTR <30>	; CPU read/write bus timeout.
6	DSER <7>	; Q22-bus NXM.
4	DSER <5>	; Q22-bus parity error.
3	DSER <4>	; Read main memory error.
2	DSER <3>	; Lost error.
1	DSER <2>	; No grant timeout.
0	IPCRn <15>	; DMA Q22-bus memory error.

4.5.2 Isolating Memory Failures

This section describes procedures for isolating memory subsystem failures, particularly when the system contains more than one MS650 memory module.

1. SHO 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 KA650 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
   ```

   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 MS650 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 MS650 modules, for example, repeat this procedure for all memory modules to isolate the MS650 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
   ```

   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 of the memory tests, with the exception of 40, save MEMCSR17 and MEMCSR16 memory status and error registers in parameters 7 and 8, 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 MEMCSR 0–15 by entering T 9C at the console I/O mode prompt (Example 4–6). Utility 9C is also useful for examining the error registers MSER, CACR, DSER, and MEMCSR16, upon a fatal system crash or similar event.

4. T 40

Although the SHOW/MEMORY 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 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 4–1) 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 the CPU and the three 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.

Example 4–6: Isolating Bad Memory Using T 9C

```
>>>T  9C
TOY   =00034283   ICCS  =00000000
TCR0  =00000000   TIR0  =00000000   TNIR0=00000000   TIVR0=00000078
TCR1  =00000000   TIR1  =00000000   TNIR1=00000000   TIVR1=0000007C
RXCS  =00000000   RXDB  =0000000D   TXCS  =00000000   TXDB  =00000030
MSER  =00000000   CADR  =0000000C
BDR   =FFFFFFD0   DLEDR=0000000C    SSCCR=00D45033   CBTCR=00000004
SCR   =0000C000   DSER  =00000000   QBEAR=00000000   DEAR  =00000000
QBMBR=00BF8000    IPCRn=0000

MEM_FRU 1 MEMCSR_0=80000015    1=00000015    2=00000015    3=00000015
MEM_FRU 2 MEMCSR_4=80400016    5=80800016    6=00000016    7=00000016
MEM_FRU 3 MEMCSR_8=00000000    9=00000000   10=00000000   11=00000000
MEM_FRU 4 MEMCSR12=00000000   13=00000000   14=00000000   15=00000000
          MEMCSR16=8094000F   17=00002000

Ethernet  SA = 08-00-2B-08-E8-6E      NICSR0=0004

SII     MSIDR0 =01FF   MSIDR1 =0002   MSIDR2 =0000   MSICSR =0010
        MSIIDR =8007   MSITR  =0000   MSITLP =0000   MSIILP =0000
        MSIDSCR=80FF   MSIDSSR=8500   MSIDCR =0008
>>>
```

In this case, the diagnostics have passed, but the operating system does not boot. One of the console/VMB error messages is displayed. Run utility 9C and examine the error registers. Bit 31 in MEMCSR 16 is set, indicating an uncorrectable ECC error in memory. If any of bits <31:29> are set, there was a memory error. Compare the bits <25:22> against MEMCSR 0–15. If they match and the MEMCSRn <31> is set, then the board that was experiencing the failure (the memory FRU) is the MEM_FRU number on the left.

In Example 4–6, the FRU is the second memory FRU, which is the first MS650–AA module (KA640–AA is the first memory FRU), because both conditions are met by MEMCSR_5 in the MEM_FRU 2 row. The following conditions are shown in Example 4–7:

- MEMCSR_5 matches MEMCSR16, since bits <25:22> (Bank Number) match.

- The Bank Enable bit <31> in MEMCSR_5 is set, indicating that the bank number is valid.

Example 4–7:   9C—Conditions for Determining a Memory FRU

```
                           3      2    2
                           1      5    2
MEMCSR16 = 8094000F Hex = 1000 0000 1001 0100 0000 0000 0000 1111
                                   ||  ||
MEMCSR_5 = 80800016 Hex = 1000 0000 1000 0000 0000 0000 0001 0110
                          ^             ^
              bit 31 set          25:22 match
```

4.5.3 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 KA640 module, CPU/memory cable, backplane, or MS650 modules as the cause of failure.

Always check the seating of the memory cable first before replacing a KA640 or MS650 module. If the seating appears to be improper, rerun the tests. Also remember to leave the middle connector disconnected for a three-connector cable when the system is configured with only one MS650.

If you are rotating MS650 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- and multi-bit ECC failures as shown in step 2 of acceptance testing, since in one slot a board may fail most frequently with multi-bit 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 error log, use the KA640 ROM-based diagnostics to see if it is an MS650 problem, or if it is related to the KA640, CPU/memory interconnect cable, or backplane. Follow steps 1–3 of Section 4.4 and Section 4.5.2 to aid in isolating the failure.

Use the SHOW QBUS, SHOW DEVICE, and SET HOST/DUP commands when troubleshooting I/O subsystem problems.

Use the CONFIG command to help with configuration problems or when installing new options onto the Q-bus. See the command descriptions in Chapter 3.

You can run a DSSI device power-up diagnostic without performing a cold restart or spinning the disk drives down and back up.

Type the following at the console program:

>>>T 58 <NODE_NUMBER>

A CI Reset command is issued to the DSSI device, causing the device to perform its power-up diagnostics.

Parameter·1 is the DSSI node or port number. It must be in the range of 0–7 (0 is the default). Use the default for parameter 2.

You can perform this test repeatedly with the REPEAT command (R T 58 <node_number>). In that case the drive's self-tests run repeatedly until you press CTRL/C to terminate the test.

Once the test has completed successfully, you can examine the DSSI device's internal error logs by running the DUP local programs HISTRY and PARAMS. Refer to Section 4.8.3 and Section 4.8.5 for further information.

4.6 Loopback Tests

You can use external loopback tests to localize problems with the Ethernet, console, and DSSI subsystems.

Check that dc power and pico fuses on the KA640 are functioning correctly. Three 1.5 A pico fuses (12–10929–08) are located near the handle on the component side. The fuses are numbered from left to right as F3, F1, and F2 and are shown in Figure 1–1. Replace the fuse, not the KA640, if a fuse has gone bad. Table 4–11 lists some symptoms of faulty fuses.

Table 4–11: KA640 Fuses

Bad Fuse	Symptom
F1 bad (+5V)	H3602–SA hex LED display is off.
F2 bad (+12V)	Both Thinwire and standard Ethernet LEDs are off on the H3602–SA.
	Ethernet external loopback test 5F fails (for ThinWire only, since the fuse protects +12 V supplied to the DESTA on the H3602–SA).
	The LED on the loopback connector (12–22196–02) for standard Ethernet is off; external loopback tests for standard Ethernet pass, however.
	Console SLU external loopback test fails.
F3 bad (+5V)	LED on DSSI bus terminator or external loopback connectors is off.
	Only local DSSI node (typically node 7 for the KA640) is reported by SHOW DEVICE or SHOW DSSI commands.
	DSSI external loopback test 56 fails.

DSSI Problems

For DSSI problems, run the SII external loopback test (test 56). To check the DSSI bus out to the KA640 connector, plug one end of the cable (17–02216–01) to the H3281 loopback connector and the other end to the KA640 DSSI connector. To test out to the end of the DSSI bus, power down the system, remove all DSSI devices with the exception of the KA640 from the bus, and plug the external DSSI loopback connector in place of the DSSI bus terminator.

Ethernet Problems

For ThinWire Ethernet problems, run the external loopback test (NI test, number 5F) by entering the following:

```
>>> T 5F 1<CR>
```

Set parameter 1 to run this test. Only the external loopback test runs. Be sure to set the ThinWire/standard Ethernet switch on the H3602–SA to the ThinWire position. Use two 50-ohm H8225 terminators connected to an H8223 T-connector.

To test the standard Ethernet connector, use loopback connector 12–22196–02 in conjunction with MDM.

4.6.1 Testing the Console Port

To test the console port at power-up, set the power mode switch on the H3602–SA to the test position, and install an H3103 loopback connector into the MMP of the H3602. The H3103 connects the console port transmit and receive lines. At power-up, the SLU_EXT_LOOPBACK IPT then runs a continuous loopback test.

While the test is running, the LED display on the CPU I/O insert 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 KA640, the H3602–SA, the cabling, or the CPU module.

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

1. Plug the MMJ end of the console terminal cable into the H3602–SA.

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.

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. The test usually does not check the module Q22-bus interface, the line drivers and receivers, or the connector pins—all of which have relatively high failure rates.

A fail by a module self-test is accurate, because the test does not require any other part of the system to be working.

The following modules do not have LED self-test indicators:

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:

CXA16
CXB16
CXY08

Table 4–12 lists loopback connectors for common KA640 system modules. Refer to *Microsystems Options* for a description of specific module self-tests.

Table 4–12: Loopback Connectors for Q22-Bus Devices

Device	Module Loopback	Cable Loopback
CXA16/CXB16	H3103 + H8572[1]	
CXY08	H3046 (50-pin)	H3197 (25-pin)
DELQA	12–22196–02	
DPV11	H3259	H3260
DSSI[2]	–	–
DZQ11	12–15336–00 or H325	H329 (12–27351–01)
Ethernet[3]	–	–
LPV11	None	None
KA640/H3602–SA	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 DSSI to KA640 or RF-series connector, use 17–02216–01 plus H3281 loopback. For connection to end of bus, use the DSSI loopback connector 12–30702–01.

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

4.8 RF30 Troubleshooting and Diagnostics

An RF30 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 OCP on the enclosure front panel. The RF30 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:

1. *Controller errors* are caused by the hardware associated with the controller function of the drive module. A controller error is fatal to the operation of the drive, since the controller cannot establish a logical connection to the host. The red fault LED lights. If this occurs, replace the drive module.

2. *Drive errors* are caused by the hardware associated with the drive control function of the drive module. These errors are not fatal to the drive, since the drive can establish a logical connection and report the error to the host. Both LEDs go out for about 1 second, then the red fault LED lights. In this case, run either DRVTST, DRVEXR, or PARAMS (described in the next sections) to determine the error code.

Here are three common configuration errors:

- 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 RF30 is connected to the OCP, you must install a unit ID plug in the corresponding socket on the OCP. If the RF30 is not connected to the OCP, the RF30 reads its unit ID from the three-switch DIP switch on the side of the drive.

The RF30 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 RF30
HISTRY	A utility that saves information retained by the drive
ERASE	A utility that erases all user data from the disk
PARAMS	A utility that allows you to look at or change drive status, history, and parameters

A description of each local program follows, including a table showing the dialog of each program. The table also indicates the type of messages contained in the dialog, although the screen display will not indicate the message type. Message types are abbreviated as follows:

Q—Question
I—Information
T—Termination
FE—Fatal error

Access these local programs using 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 KA640 console. To abort or prematurely terminate a program and return control to the KA640 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 4–13.

Table 4–13: 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)|
I	5 minutes to complete.
T or	Test passed.
FE	Unit is currently in use.[1]
FE	Operation aborted by user.
FE	xxxx—Unit diagnostics failed.[2]
FE	xxxx—Unit read/write test failed.[2]

[1] Either the drive is inoperative, in use by a host, or is currently running another local program.
[2] Refer to the diagnostic error list at the end of this chapter.

Answering No to the first question ("Write/read...?") results in a read-only test. DRVTST, 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.

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 RF30 disk drive. The test is data transfer intensive, and indicates the overall integrity of the device. Table 4–14 lists the DRVEXR messages.

Table 4–14: 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	dddddddd blocks (512 bytes) transferred.
I	dddddddd bytes in error (soft).
I	dddddddd uncorrectable ECC errors (recoverable).
T	Complete.

Or:

FE	Unit is currently in use.[1]
FE	Operation aborted by user.
FE	xxxx—Unit diagnostics failed.[2]
FE	xxxx—Unit read/write test failed.[2]

[1]Either the drive is inoperative, in use by a host, or is currently running another local program.
[2]Refer to the diagnostic error list at the end of this chapter.

Answering No to the first question ("Write/read...?") results in a read-only test. DRVEXR, however, writes to a diagnostic area on the disk. Answering Yes to the first question results in the second question being asked.

Answering No to the second question ("Proceed?") is the same as answering No to the first question. Answering Yes to the second question permits write and read operations anywhere on the medium.

NOTE: *If the write-protect switch on the OCP is pressed in (LED on) and you answer Yes to the second question, the drive does not allow the test to run. DRVEXR displays the error message 2006—Unit read/write test failed. In this case, the test has not failed, but has been prevented from running.*

DRVEXR saves the error counters, then calls the timed I/O routine. After the timed I/O routine ends, DRVEXR saves the counters again. It then reports the total number of blocks transferred, bits in error, bytes in error, and uncorrectable errors.

DRVEXR uses the same timed I/O routine as DRVTST, with two exceptions. First, DRVTST always uses a fixed time of five minutes, while you specify the time of 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 RF30 disk drive. Table 4–15 lists the HISTRY messages.

Table 4–15: HISTRY Messages

Message Type	Field Length	Field Meaning
I	47 ASCII characters	Copyright notice
I	4 ASCII characters	Product name
I	12 ASCII characters	Drive serial number
I	6 ASCII characters	Node name
I	1 ASCII character	Allocation class
I	8 ASCII characters	Firmware revision level
I	17 ASCII characters	Hardware revision level
I	6 ASCII characters	Power-on hours
I	5 ASCII characters	Power cycles
I[1]	4 ASCII characters	Hexadecimal fault code
T		Complete

[1]Displays the last 11 fault codes as informational messages. Refer to the diagnostic error list at the end of this chapter.

The following example shows a typical screen display when you run HISTRY:

```
Copyright © 1988 Digital Equipment Corporation
RF30
EN01082
SUSAN
0
RFX V101
RF30 PCB-5/ECO-00
617
21
A04F
A04F
A103
A04F
A404
A04F
A404
A04F
A404
A04F
A404
Complete.
```

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

4.8.4 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 4–16 lists the ERASE messages.

Table 4–16: ERASE Messages

Message Type	Message
I	Copyright © 1988 Digital Equipment Corporation
Q	Write/read anywhere on the medium? [1=yes/(0=no)]
Q	User data will be corrupted. Proceed? [1=yes/(0=no)]
I	6 minutes to complete.
T or	Complete.
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 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 indicating the failure will be followed by one or more messages containing 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 example below.

```
PARAMS> help
EXIT
HELP
SET {parameter | .} value
SHOW {parameter | . | /class}
   /ALL     /CONST   /DRIVE
   /SERVO   /SCS     /MSCP
   /DUP
STATUS [type]
   CONFIG   LOGS      DATALINK
   PATHS
WRITE

PARAMS>
```

4.8.5.3 SET

Use the SET command to change the value of a given parameter. *Parameter* is the name or abbreviation of the parameter to be changed. To abbreviate, use the first matching parameter without regard to uniqueness. *Value* is the value assigned to the parameter.

For example, SET NODE SUSAN sets the NODENAME parameter to SUSAN.

The following parameters are useful to Field Service:

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 RF30x (where x is a letter from A to H corresponding to the DSSI bus ID) instead of using the NODENAME value. False (0) if NODENAME is to be used.

4.8.5.4 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. *Type* is the optional ASCII string that denotes the type of display desired. If you omit *Type*, all available status information is displayed. If present, it may be abbreviated. The following types are available.

CONFIG   Displays the module name, node name, power-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 in the available state with respect 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. The exceptions are listed below. The error codes are listed in Table 4–17. If you see any error code other than those listed below, replace the module.

Table 4–17: RF30 Diagnostic Error Codes

Code	Message	Meaning
2032/A032	Failed to see FLT go away	FLT bit of the spindle control status register was asserted for one of the following reasons: 1. Reference clock not present 2. Stuck rotor 3. Bad connection between HDA and module
203A/A03A	Can't spinup, ACLOW is set in WrtFlt	Did not see ACOK signal, which is supplied by the host system power supply for staggered spin-up.
1314/9314	Front panel is broken	Could be either the module or the operator control panel or both.

Appendix A

Address Assignments

A.1 General Local Address Space Map

Table A–1 lists the VAX memory space.

Table A–1: VAX Memory Space

Contents	Address Range
Local memory space (52 Mbytes)	0000 0000–033F FFFF
Reserved memory space (460 Mbytes)	0340 0000–1FFF FFFF

Table A–2 lists the VAX input/output memory space.

Table A–2: VAX Input/Output Space

Contents	Address Range
Local Q22-bus I/O space (8 Kbytes)	2000 0000–2000 1FFF
Reserved local I/O space (248 Kbytes)	2000 2000–2003 FFFF
Local ROM space—halt protected space (128 Kbytes)	2004 0000–2005 FFFF
Local ROM space—halt unprotected space (128 Kbytes)	2006 0000–2007 FFFF
Local register I/O space (1.5 Mbytes)	2008 0000–201F FFFF
Reserved local I/O space (62.5 Mbytes)	2020 0000–23FF FFFF
Reserved local I/O space (64 Mbytes)	2400 0000–27FF FFFF
Reserved local I/O space (64 Mbytes)	2800 0000–2BFF FFFF
Reserved local I/O space (64 Mbytes)	2C00 0000–2FFF FFFF
Local Q22-bus memory space (4 Mbytes)	3000 0000–303F FFFF
Reserved local I/O space (60 Mbytes)	3040 0000–33FF FFFF
Reserved local I/O space (64 Mbytes)	3400 0000–37FF FFFF
Reserved local I/O space (64 Mbytes)	3800 0000–3BFF FFFF
Reserved local I/O space (64 Mbytes)	3C00 0000–3FFF FFFF

A.2 Detailed Local Address Space Map

Table A–3 describes the contents of the VAX memory space.

Table A–3: VAX Memory Space

Contents	Address Range
Local memory space (up to 52 Mbytes) Q22-bus map—top 32 Kbytes of main memory	**0000 0000–033F FFFF**
Reserved memory space	**0340 0000–1FFF FFFF**

Table A–4 describes the contents of the VAX input/output memory space.

Table A–4: VAX Input/Output Space

Contents	Address Range
Local Q22-bus I/O Space	**2000 0000–2000 1FFF**
Reserved Q22-bus I/O space	2000 0000–2000 0007
Q22-bus floating address space	2000 0008–2000 07FF
User reserved Q22-bus address space	2000 0800–2000 0FFF
Reserved and Q22-bus fixed address space	2000 1000–2000 1F3F
Interprocessor communication register (arbiter)	2000 1F40
Reserved Q22-bus I/O space	2000 1F42–2000 1FFF
Reserved Local I/O Space	**2000 2000–2003 FFFF**
Local ROM Space	**2004 0000–2007 FFFF**
Local ROM protected space	2004 0000–2005 FFFF
MicroVAX system type register (in ROM)	2004 0004
Local ROM unprotected space	2006 0000–2007 FFFF
Local Register I/O Space	**2008 0000–201F FFFF**
DMA system configuration register	2008 0000
DMA system error register	2008 0004
Q22-bus error address register	2008 0008
DMA error address register	2008 000C
Q22-bus map base register	2008 0010
Reserved local register I/O space	2008 0014–2008 00FF
Main memory configuration Reg 00	2008 0100
Main memory configuration Reg 01	2008 0104
Main memory configuration Reg 02	2008 0108
Main memory configuration Reg 03	2008 010C
Main memory configuration Reg 04	2008 0110
Main memory configuration Reg 05	2008 0114
Main memory configuration Reg 06	2008 0118
Main memory configuration Reg 07	2008 011C
Main memory configuration Reg 08	2008 0120
Main memory configuration Reg 09	2008 0124
Main memory configuration Reg 10	2008 0128
Main memory configuration Reg 11	2008 012C
Main memory configuration Reg 12	2008 0130
Main memory configuration Reg 13	2008 0134
Main memory configuration Reg 14	2008 0138
Main memory configuration Reg 15	2008 013C

Table A-4 (Cont.): VAX Input/Output Space

Contents	Address Range
Main memory error status register	2008 0140
Main memory control/diagnostic status register	2008 0144
Reserved local register I/O space	2008 0148–2008 3FFF
Reserved (1 copy of BDR)	2008 4000
Boot and diagnostic register	2008 4004
Reserved (126 copies of BDR)	2008 4008–2008 41FF
NI Station Address ROM	2008 4200–2008 427C
Reserved (4 copies of NISA ROM)	2008 4280–2008 43FF
NI Register Data Port	2008 4400
NI Register Address Port	2008 4404
64 copies of NIRDP, NIRAP (reserved)	2008 4408–2008 45FF
MSI Diagnostic Register 0	2008 4600
MSI Diagnostic Register 1	2008 4604
MSI Diagnostic Register 2	2008 4608
MSI Control and Status Register	2008 460C
MSI ID Register	2008 4610
Reserved MSI Register	2008 4614
Reserved MSI Register	2008 4618
MSI Timeout Register	2008 461C
Reserved MSI Register	2008 4620
Reserved MSI Register	2008 4624
Reserved MSI Register	2008 4628
Reserved MSI Register	2008 462C
Reserved MSI Register	2008 4630
Reserved MSI Register	2008 4634
Reserved MSI Register	2008 4638
MSI Long Target List Pointer	2008 463C
MSI Initiator List Pointer	2008 4640
MSI DSSI Control Register	2008 4644
MSI DSSI Status Register	2008 4648
Reserved MSI Register	2008 464C
Reserved MSI Register	2008 4650
MSI Diagnostic Control Register	2008 4654
MSI Clock Control Register	2008 4658
MSI Internal State Register 0	2008 465C
MSI Internal State Register 1	2008 4660
MSI Internal State Register 2	2008 4664
MSI Internal State Register 3	2008 4668
Reserved MSI Register	2008 466C

Table A–4 (Cont.): VAX Input/Output Space

Contents	Address Range
Reserved MSI Register	2008 4670
Reserved MSI Register	2008 4674
Reserved MSI Register	2008 4678
Reserved MSI Register	2008 467C
Reserved (4 copies of MSI reg block)	2008 4680–2008 47FF
Reserved Local Register I/O Space	2008 4800–2008 7FFF
Q22-bus map registers	2008 8000–2008 FFFF
Reserved local register I/O space	2009 0000–200F FFFF
MSI Buffer RAM	2010 0000–2011 FFFF
NI Buffer RAM	2012 0000–2013 FFFF
SSC base address register	2014 0000
SSC configuration register	2014 0010
CDAL bus timeout control register	2014 0020
Diagnostic LED register	2014 0030
Reserved local register I/O space	2014 0034–2014 0068
Diagnostic registers	2014 006C–2014 00FF
Timer 0 control register	2014 0100
Timer 0 interval register	2014 0104
Timer 0 next interval register	2014 0108
Timer 0 interrupt vector	2014 010C
Timer 1 control register	2014 0110
Timer 1 interval register	2014 0114
Timer 1 next interval register	2014 0118
Timer 1 interrupt vector	2014 011C
Reserved local register I/O space	2014 0120–2014 012F
MSIDB address decode match register	2014 0130
MSIDB address decode mask register	2014 0134
Reserved local register I/O space	2014 0138–2014 013F
LIOD address decode match register	2014 0140
LIOD address decode mask register	2014 0144
Reserved local register I/O space	2014 0148–2014 03FF
Battery backed-up RAM	2014 0400–2014 07FF
Reserved local register I/O space	2014 0800–201F FFFF
Reserved local I/O space	2020 0000–2FFF FFFF
Local Q22-bus memory space	3000 0000–303F FFFF
Reserved local register I/O space	3040 0000–3FFF FFFF

A.3 Internal Processor Registers

Table A–5 lists the internal processor registers implemented in the CVAX CPU chip and the SSC.

Table A–5: KA640 IPRs

Dec	Hex	Register Name	Mnemonic	Type	Location
0	0	Kernel Stack Pointer	KSP	r/w	CVAX
1	1	Executive Stack Pointer	ESP	r/w	CVAX
2	2	Supervisor Stack Pointer	SSP	r/w	CVAX
3	3	User Stack Pointer	USP	r/w	CVAX
4	4	Interrupt Stack Pointer	ISP	r/w	CVAX
5	5	Reserved			CVAX
6	6	Reserved			CVAX
7	7	Reserved			CVAX
8	8	P0 Base Register	P0B	r/w	CVAX
9	9	P0 Length Register	P0LR	r/w	CVAX
10	A	P1 Base Register	P1BR	r/w	CVAX
11	B	P1 Length Register	P1LR	r/w	CVAX
12	C	System Base Register	SBR	r/w	CVAX
13	D	System Length Register	SLR	r/w	CVAX
14	E	Reserved			CVAX
15	F	Reserved			CVAX
16	10	Process Control Block Base	PCBB	r/w	CVAX
17	11	System Control Block Base	SCBB	r/w	CVAX
18	12	Interrupt Priority Level	IPL	r/w	CVAX
19	13	AST Level	ASTLVL	r/w	CVAX
20	14	Software Interrupt Request	SIRR	w	CVAX
21	15	Software Interrupt Summary	SISR	r/w	CVAX
22	16	Reserved			CVAX

Table A–5 (Cont.): KA640 IPRs

Dec	Hex	Register Name	Mnemonic	Type	Location
23	17	Reserved			CVAX
24	18	Interval Clock Control Status	ICCS	r/w	CVAX
25	19	Next Interval Count	NICR	w	CVAX
26	1A	Interval Count	ICR	r	CVAX
27	1B	Time-of-year Register	TOY	r/w	SSC
28	1C	Console Storage Receiver Status	CSRS[1]	r/w	SSC
29	1D	Console Storage Receiver Data	CSRD[1]	r	SSC
30	1E	Console Storage Transmitter Status	CSTS[1]	r/w	SSC
31	1F	Console Storage Transmitter Data	CSDB[1]	w	SSC
32	20	Console Receiver Control Status	RXCS	r/w	SSC
33	21	Console Receiver Data Buffer	RXDB	r	SSC
34	22	Console Transmitter Control Status	TXCS	r/w	SSC
35	23	Console Transmitter Data Buffer	TXDB	w	SSC
36	24	Translation Buffer Disable	TBDR	r/w	CVAX
37	25	Cache Disable	CADR	r/w	CVAX
38	26	Machine Check Error Summary	MCESR	r/w	CVAX
39	27	Memory System Error	MSER	r/w	CVAX
40	28	Reserved			CVAX
41	29	Reserved			CVAX
42	2A	Console Saved PC	SAVPC	r	CVAX
43	2B	Console Saved PSL	SAVPSL	r	CVAX
44	2C	Reserved			CVAX
45	2D	Reserved			CVAX
46	2E	Reserved			CVAX
47	2F	Reserved			CVAX
55	47	I/O System Reset Register	IORESET	–	CVAX

A.4 Global Q22-Bus Address Space Map

Table A–6 lists the addresses and memory contents of the Q22-bus memory space.

Table A–6: Q22-Bus Memory Space

Contents	Address
Q22-bus memory space (octal)	0000 0000–1777 7777

Table A–7 lists the contents and addresses of the Q22-bus I/O space with BBS7 asserted.

Table A–7: Q22-Bus I/O Space with BBS7 Asserted

Contents	Address
Q22-bus I/O space (Octal)	1776 0000–1777 7777
Reserved Q22-bus I/O space	1776 0000–1776 0007
Q22-bus floating address space	1776 0010–1776 3777
User reserved Q22-bus address space	1776 4000–1776 7777
Reserved and Q22-bus fixed address space	1777 0000–1777 7477
Interprocessor communication register	1777 7500
Reserved Q22-bus I/O space	1777 7502–1777 7777

Appendix B

Related Documentation

The following documents contain information relating to MicroVAX or
MicroPDP–11 systems.

Document Title	Order Number
Modules	
CXA16 Technical Manual	EK–CAB16–TM
CXY08 Technical Manual	EK–CXY08–TM
DEQNA Ethernet User's Guide	EK–DEQNA–UG
DHV11 Technical Manual	EK–DHV11–TM
DLV11–J User's Guide	EK–DLV1J–UG
DMV11 Synchronous Controller Technical Manual	EK–DMV11–TM
DMV11 Synchronous Controller User's Guide	EK–DMV11–UG
DPV11 Synchronous Controller Technical Manual	EK–DPV11–TM
DPV11 Synchronous Controller User's Guide	EK–DPV11–UG
DRV11–J Interface User's Manual	EK–DRV1J–UG
DRV11–WA General Purpose DMA User's Guide	EK–DRVWA–UG
DZQ11 Asynchronous Multiplexer Technical Manual	EK–DZQ11–TM
DZQ11 Asynchronous Multiplexer User's Guide	EK–DZQ11–UG
DZV11 Asynchronous Multiplexer Technical Manual	EK–DZV11–TM
DZV11 Asynchronous Multiplexer User's Guide	EK–DZV11–UG
IEU11–A/IEQ11–A User's Guide	EK–IEUQ1–UG
KA630–AA CPU Module User's Guide	EK–KA630–UG
KA640–AA CPU Module User's Guide	EK–KA640–UG
KA650–AA CPU Module User's Guide	EK–KA650–UG
KDA50–Q CPU Module User's Guide	EK–KDA5Q–UG
KDJ11–B CPU Module User's Guide	EK–KDJ1B–UG
KDJ11–D/S CPU Module User's Guide	EK–KDJ1D–UG
KDF11–BA CPU Module User's Guide	EK–KDFEB–UG
KMV11 Programmable Communications Controller User's Guide	EK–KMV11–UG
KMV11 Programmable Communications Controller Technical Manual	EK–KMV11–TM

Document Title	Order Number
Modules	
LSI–11 Analog System User's Guide	EK–AXV11–UG
Q-Bus DMA Analog System User's Guide	EK–AV11D–UG
RQDX2 Controller Module User's Guide	EK–RQDX2–UG
RQDX3 Controller Module User's Guide	EK–RQDX3–UG
Disk and Tape Drives	
RA60 Disk Drive Service Manual	EK–ORA60–SV
RA60 Disk Drive User's Guide	EK–ORA60–UG
RA81 Disk Drive Service Manual	EK–ORA81–SV
RA81 Disk Drive User's Guide	EK–ORA81–UG
SA482 Storage Array User's Guide (for RA82)	EK–SA482–UG
SA482 Storage Array Service Manual (for RA82)	EK–SA482–SV
RC25 Disk Subsystem User's Guide	EK–ORC25–UG
RC25 Disk Subsystem Pocket Service Guide	EK–ORC25–PS
RRD50 Subsystem Pocket Service Guide	EK–RRD50–PS
RRD50 Digital Disk Drive User's Guide	EK–RRD50–UG
RX33 Technical Description Manual	EK–RX33T–TM
RX50–D, –R Dual Flexible Disk Drive Subsystem Owner's Manual	EK–LEP01–OM
TK50 Tape Drive Subsystem User's Guide	EK–LEP05–UG
TS05 Tape Transport Pocket Service Guide	EK–TSV05–PS
TS05 Tape Transport Subsystem Technical Manual	EK–TSV05–TM
TS05 Tape Transport System User's Guide	EK–TSV05–UG

Document Title	Order Number
Systems	
MicroVAX Special Systems Maintenance	EK–181AA–MG
630QB Maintenance Print Set	MP–02071–01
630QE Maintenance Print Set	MP–02219–01
630QY Maintenance Print Set	MP–02065–01
630QZ Maintenance Print Set	MP–02068–01
BA23 Enclosure Maintenance	EK–186AA–MG
BA123 Enclosure Maintenance	EK–188AA–MG
BA213 Enclosure Maintenance	EK–189AA–MG
BA214 Enclosure Maintenance	EK–190AA–MG
BA215 Enclosure Maintenance	EK–191AA–MG
H9642–J Cabinet Maintenance	EK–187AA–MG
H9644 Cabinet Maintenance	EK–221AA–MG
KA630 CPU System Maintenance	EK–178AA–MG
KA640 CPU System Maintenance	EK–179AA–MG
KA650 CPU System Maintenance	EK–180AA–MG
KDF11–B CPU System Maintenance	EK–245AA–MG
KDJ11–D/S CPU System Maintenance	EK–246AA–MG
KDJ11–B CPU System Maintenance	EK–247AA–MG
MicroPDP–11 Hardware Information Kit (for BA23)	00–ZYAAA–GZ
MicroPDP–11 Hardware Information Kit (for BA123)	00–ZYAAB–GZ
MicroPDP–11 Hardware Information Kit (for H9642–J)	00–ZYAAE–GZ
MicroPDP–11 Hardware Information Kit (for BA213)	00–ZYAAS–GZ
Microsystems Options	EK–192AA–MG
Microsystems Site Preparation Guide	EK–O67AB–PG
MicroVAX II Hardware Information Kit (for BA23)	00–ZNAAA–GZ
MicroVAX II Hardware Information Kit (for BA123)	00–ZNAAB–GZ
MicroVAX II Hardware Information Kit (for H9642–J)	00–ZNAAE–GZ
MicroVAX 3500 Customer Hardware Information Kit	00–ZNAES–GZ
MicroVAX 3600 Customer Hardware Information Kit (for H9644)	00–ZNAEF–GZ
VAXstation 3200 Owner's Manual (BA23)	EK–154AA–OW
VAXstation 3500 Owner's Manual (BA213)	EK–171AA–OW
VAXstation II/GPX Owner's Manual (BA23)	EK–106AA–OW
VAXstation II/GPX Owner's Manual (BA123)	EK–105AA–OW

Document Title	Order Number
Diagnostics	
DEC/X11 Reference Card	AV–F145A–MC
DEC/X11 User's Manual	AC–FO53D–MC
XXDP User's Manual	AZ–GNJAA–MC
XXDP DEC/X11 Programming Card	EK–OXXDP–MC
MicroVAX Diagnostic Monitor Ethernet Server User's Guide	AA–FNTAC–DN
MicroVAX Diagnostic Monitor Reference Card	AV–FMXAA–DN
MicroVAX Diagnostic Monitor User's Guide	AA–FM7AB–DN
Networks	
Ethernet Transceiver Tester User's Manual	EK–ETHTT–UG
VAX/VMS Networking Manual	AA–Y512C–TE
VAX NI Exerciser User's Guide	AA–HI06A–TE

Index