

KA655 CPU System Maintenance

Order Number EK-306AA-MG-001

**digital equipment corporation
maynard, massachusetts**

March 1989

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.


The software, if any, described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license. No responsibility is assumed for the use or reliability of software or equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

© Digital Equipment Corporation 1989. All rights reserved.

Printed in U.S.A.

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

ML-S1123

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.

This document was prepared using VAX DOCUMENT, Version 1.1.

Contents

Preface

xi

Chapter 1 KA655 CPU and Memory Subsystem

1.1	Introduction	1-1
1.2	KA655 CPU Features	1-7
1.2.1	Central Processing Unit (CPU)	1-8
1.2.2	Clock Functions	1-9
1.2.3	Floating Point Accelerator	1-9
1.2.4	Cache Memory	1-9
1.2.5	Memory Controller	1-10
1.2.6	MicroVAX System Support Functions	1-10
1.2.7	Resident Firmware	1-11
1.2.8	Q22-Bus Interface	1-11
1.3	KA655 Connectors	1-12
1.4	H3600-SA CPU I/O Panel	1-12
1.5	MS650-BA Memory	1-16

Chapter 2 Configuration

2.1	Introduction	2-1
2.2	General Module Order	2-1
2.2.1	Module Order Rules for KA655 Systems	2-2
2.2.2	Recommended Module Order for KA655 Systems	2-2
2.3	Module Configuration	2-3
2.4	DSSI Configuration	2-4
2.4.1	Changing RF-Series ISE Parameters	2-6
2.4.2	Changing the Unit Number	2-7
2.4.3	Changing the Allocation Class	2-9

2.4.4	DSSI Cabling	2-10
2.4.4.1	DSSI Bus Termination and Length	2-12
2.4.5	Dual-Host Capability	2-13
2.4.6	Limitations to Dual-Host Configurations	2-14
2.5	Configuration Worksheet	2-14

Chapter 3 KA655 Firmware

3.1	Introduction	3-1
3.2	KA655 Firmware Features	3-1
3.3	Halt Entry, Exit, and Dispatch Code	3-2
3.4	External Halts	3-3
3.5	Power-Up Sequence	3-4
3.5.1	Mode Switch Set to Test	3-4
3.5.2	Mode Switch Set to Language Inquiry	3-5
3.5.3	Mode Switch Set to Normal	3-6
3.6	Bootstrap	3-6
3.6.1	Bootstrap Initialization Sequence	3-7
3.6.2	VMB Boot Flags	3-8
3.6.3	Supported Boot Devices	3-8
3.6.4	Autoboot	3-9
3.7	Operating System Restart	3-11
3.7.1	Restart Sequence	3-11
3.7.2	Locating the RPB	3-12
3.8	Console I/O Mode	3-13
3.8.1	Command Syntax	3-13
3.8.2	Address Specifiers	3-14
3.8.3	Symbolic Addresses	3-14
3.8.4	Console Command Qualifiers	3-17
3.8.5	Console Command Keywords	3-19
3.9	Console Commands	3-21
3.9.1	BOOT	3-21
3.9.2	CONFIGURE	3-23
3.9.3	CONTINUE	3-25
3.9.4	DEPOSIT	3-26
3.9.5	EXAMINE	3-27

3.9.6	FIND	3-29
3.9.7	HALT	3-30
3.9.8	HELP	3-31
3.9.9	INITIALIZE	3-33
3.9.10	MOVE	3-34
3.9.11	NEXT	3-36
3.9.12	REPEAT	3-38
3.9.13	SEARCH	3-39
3.9.14	SET	3-41
3.9.15	SHOW	3-44
3.9.16	START	3-47
3.9.17	TEST	3-48
3.9.18	UNJAM	3-49
3.9.19	X—Binary Load and Unload	3-50
3.9.20	! (Comment)	3-52

Chapter 4 Troubleshooting and Diagnostics

4.1	Introduction	4-1
4.2	General Procedures	4-1
4.3	KA655 ROM-Based Diagnostics	4-2
4.3.1	Diagnostic Tests	4-3
4.3.2	Scripts	4-6
4.3.3	Script Calling Sequence	4-8
4.3.4	Creating Scripts	4-10
4.3.5	Console Displays	4-14
4.3.6	System Halt Messages	4-24
4.3.7	Console Error Messages	4-25
4.3.8	VMB Error Messages	4-26
4.4	Acceptance Testing	4-26
4.5	Troubleshooting	4-31
4.5.1	FE Utility	4-31
4.5.2	Isolating Memory Failures	4-34
4.5.3	Additional Troubleshooting Suggestions	4-37
4.6	Loopback Tests	4-38
4.6.1	Testing the Console Port	4-38

4.7	Module Self-Tests	4-38
4.8	RF-Series ISE Troubleshooting and Diagnostics	4-40
4.8.1	DRVTST	4-42
4.8.2	DRVEXR	4-43
4.8.3	HISTORY	4-44
4.8.4	ERASE	4-45
4.8.5	PARAMS	4-46
4.8.5.1	EXIT	4-47
4.8.5.2	HELP	4-47
4.8.5.3	SET	4-47
4.8.5.4	SHOW	4-48
4.8.5.5	STATUS	4-48
4.8.5.6	WRITE	4-48
4.9	Diagnostic Error Codes	4-49

Appendix A Configuring the KFQSA

A.1	KFQSA Overview	A-1
A.1.1	Dual-Host Configuration	A-2
A.2	Configuring the KFQSA at Installation	A-2
A.2.1	Entering Console I/O Mode	A-4
A.2.2	Displaying Current Addresses	A-5
A.2.3	Running the Configure Utility	A-6
A.3	Programming the KFQSA	A-8
A.4	Reprogramming the KFQSA	A-15
A.5	Changing the ISE Allocation Class and Unit Number	A-17

Appendix B KA655 CPU Address Assignments

B.1	General Local Address Space Map	B-1
B.2	Detailed Local Address Space Map	B-2
B.3	Internal Processor Registers	B-6
B.3.1	KA655 VAX Standard IPRs	B-9
B.3.2	KA655 Unique IPRs	B-10
B.4	Global Q22-Bus Address Space Map	B-11

Appendix C Related Documentation

Index

Examples

2-1	Changing a DSSI Node Name	2-6
2-2	Changing a DSSI Unit Number	2-8
2-3	Changing a DSSI Allocation Class	2-9
3-1	Language Selection Menu	3-6
3-2	Selecting a Boot Device	3-10
4-1	Creating a Script with Utility 9F	4-11
4-2	Listing and Repeating Tests with Utility 9F	4-13
4-3	Console Display (No Errors)	4-15
4-4	Sample Output with Errors	4-15
4-5	FE Utility Example	4-32
4-6	Isolating Bad Memory Using T 9C	4-36
4-7	9C—Conditions for Determining a Memory FRU	4-37
A-1	KFQSA (M7769) Service Mode Switch Settings	A-4
A-2	Entering Console Mode Display	A-5
A-3	SHOW QBUS Display	A-5
A-4	Configure Display	A-7
A-5	Display for Programming the First KFQSA	A-10
A-6	Display for Programming the KFQSA in a Dual-Host Configuration (Second System)	A-11
A-7	SHOW QBUS Display	A-13
A-8	SHOW DEVICE Display	A-14
A-9	Reprogramming the KFQSA Display	A-16
A-10	Display for Changing Allocation Class and Unit Number ...	A-18

Figures

1-1	KA655 CPU Module (M7625-AA)	1-2
1-2	KA655 CPU Functional Block Diagram	1-3
1-3	KA655 System-Level Block Diagram, Part I	1-5
1-4	KA655 System-Level Block Diagram, Part II	1-6
1-5	KA655 Pin Orientation	1-12
1-6	H3600-SA CPU I/O Panel	1-13
1-7	MS650-BA Memory Module (M7622-A)	1-17
2-1	DSSI Cabling, BA213 Enclosure	2-11
2-2	RF-Series ISE Operator Control Panel (OCP)	2-12
2-3	BA213 Configuration Worksheet	2-16
4-1	KA655 CPU Module LEDs	4-18
A-1	KFQSA Module Layout (M7769)	A-3
A-2	Dual-Host Configuration Nodes and Addresses (Example)	A-9

Tables

1-1	H3600-SA Controls and Connectors	1-14
1-2	H3600-SA CPU I/O Panel Switches	1-15
2-1	DSSI Device Order	2-5
2-2	RF-Series ISE Switch Settings	2-5
2-3	Power and Bus Loads for KA655 Options	2-15
3-1	Actions Taken on a Halt	3-3
3-2	Language Inquiry on Power-Up or Reset	3-5
3-3	Virtual Memory Bootstrap (VMB) Boot Flags	3-8
3-4	Boot Devices Supported by the KA655-AA	3-9
3-5	Console Symbolic Addresses	3-15
3-6	Symbolic Addresses Used in Any Address Space	3-17
3-7	Console Command Qualifiers	3-18
3-8	Command Keywords by Type	3-19
3-9	Console Command Summary	3-19
4-1	Test and Utility Numbers	4-4
4-2	Scripts Available to Field Service	4-7
4-3	Commonly Used Field Service Scripts	4-8
4-4	Values Saved, Machine Check Exception During Executive	4-17
4-5	Values Saved, Exception During Executive	4-17

4-6	KA655 Console Displays and FRUs	4-19
4-7	System Halt Messages	4-24
4-8	Console Error Messages	4-25
4-9	VMB Error Messages	4-26
4-10	Hardware Error Summary Register	4-33
4-11	Loopback Connectors for Q22-Bus Devices	4-39
4-12	DRVST Messages	4-42
4-13	DRVEXR Messages	4-43
4-14	HISTORY Messages	4-44
4-15	ERASE Messages	4-46
4-16	RF-Series ISE Diagnostic Error Codes	4-49
B-1	VAX Memory Space	B-1
B-2	VAX Input/Output Space	B-2
B-3	Detailed VAX Memory Space	B-2
B-4	Detailed VAX Input/Output Space	B-3
B-5	KA655 Internal Processor Registers	B-7
B-6	IPRs Implemented According to Standard VAX Architecture	B-9
B-7	KA655 Unique IPRs	B-10
B-8	Q22-Bus Memory Space	B-11
B-9	Q22-Bus I/O Space with BBS7 Asserted	B-11

Preface

This guide describes the base system, configuration guidelines, ROM-based diagnostics, and troubleshooting procedures for systems containing the KA655 central processing unit (CPU).

Intended Audience

This document is intended only for Digital Field Service personnel and qualified self-maintenance customers.

Organization

This guide has four chapters and three appendixes.

Chapter 1 describes the KA655 CPU and the MS650-BA memory.

Chapter 2 contains system configuration guidelines and provides a table listing current, power, and bus loads for supported options.

Chapter 3 describes KA655 diagnostic firmware.

Chapter 4 describes the KA655 diagnostics, including an error message and FRU cross-reference table. It also describes diagnostics that reside on RF-series integrated storage elements (ISEs).

Appendix A explains how to configure the KFQSA storage adapter.

Appendix B contains a list of KA655 CPU address assignments.

Appendix C contains a list of related documentation.

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. |

KA655 CPU and Memory Subsystem

1.1 Introduction

This chapter describes the KA655 central processing unit (CPU) and the MS650-BA memory.

The KA655 CPU module (M7625-AA), shown in Figure 1-1, is a quad-height VAX processor 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 KA655 employs a cache memory to maximize performance.

Figure 1-1: KA655 CPU Module (M7625-AA)

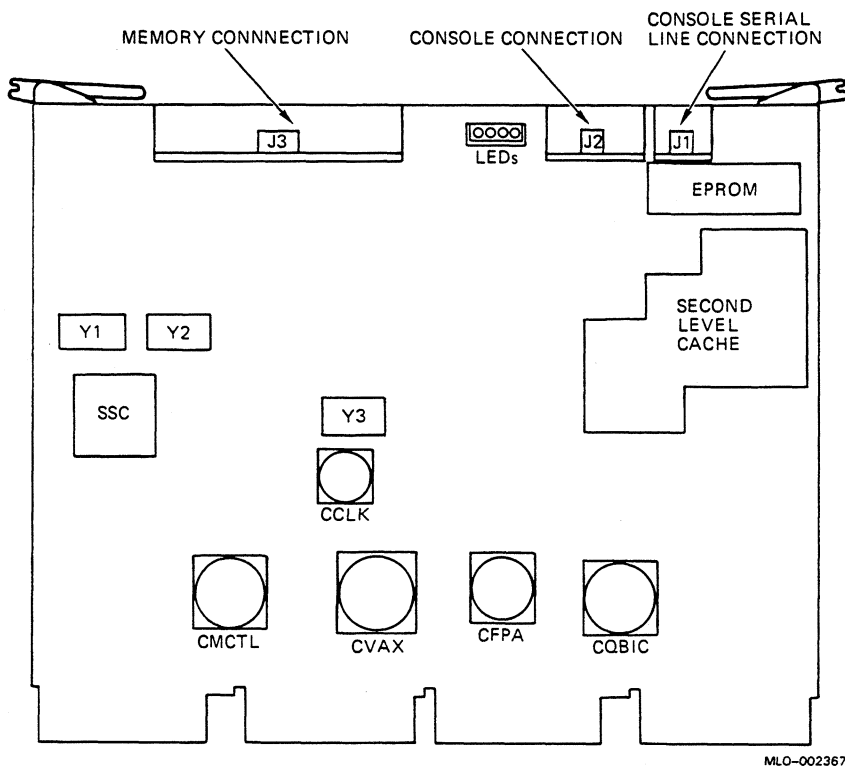
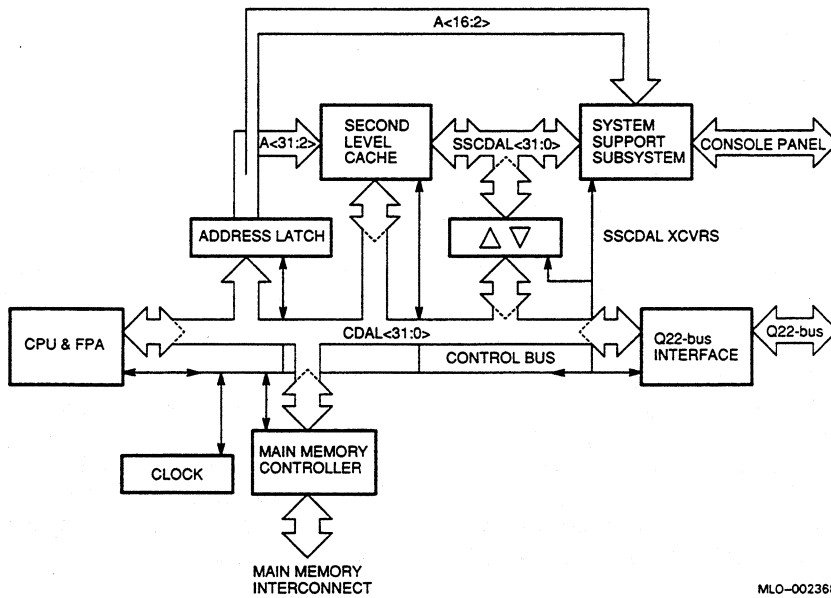


Figure 1-2 shows a functional block diagram of the KA655 CPU.

Figure 1-2: KA655 CPU Functional Block Diagram



MLO-002368

The KA655 has two variants:

- KA655-AA. Runs multiuser software.
- KA655-BA. Runs single-user software.

The KA655 is used in two systems:

- MicroVAX 3800 (BA213 enclosure)
- MicroVAX 3900 (H9644 cabinet)

Refer to *BA213 Enclosure Maintenance* and *H9644 Cabinet Maintenance* for a detailed description of the enclosure and the cabinet.

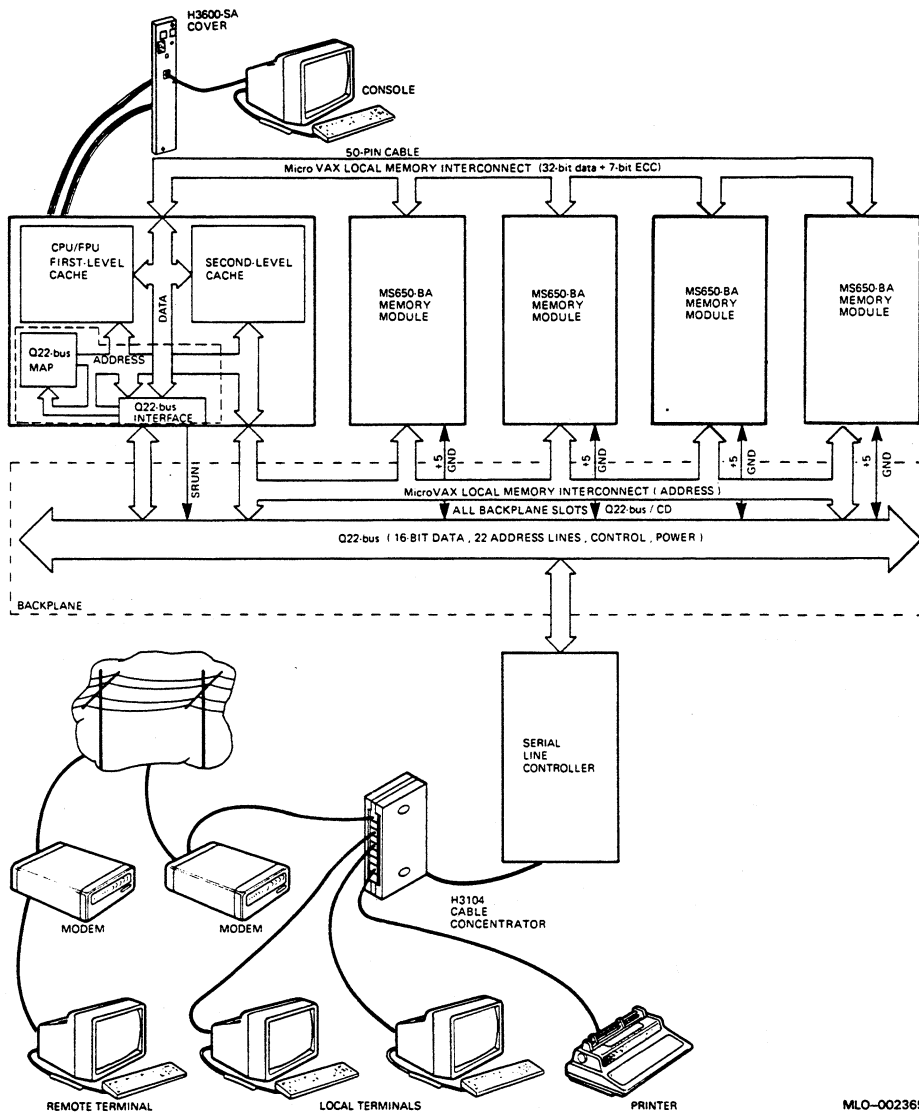
CAUTION: *Static electricity can damage integrated circuits. Always use a grounded wrist strap (part no. 29-11762) and grounded work surface when you work with the internal parts of a computer system.*

The KA655 CPU module and MS650-BA memory modules combine to form a VAX CPU and memory subsystem that uses the Q22-bus to communicate with I/O devices. The KA655 and MS650-BA 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 KA655 can support up to four MS650-BA modules, if enough Q22-bus CD slots are available. Figures 1-3 and 1-4 show the KA655 system-level block diagram.

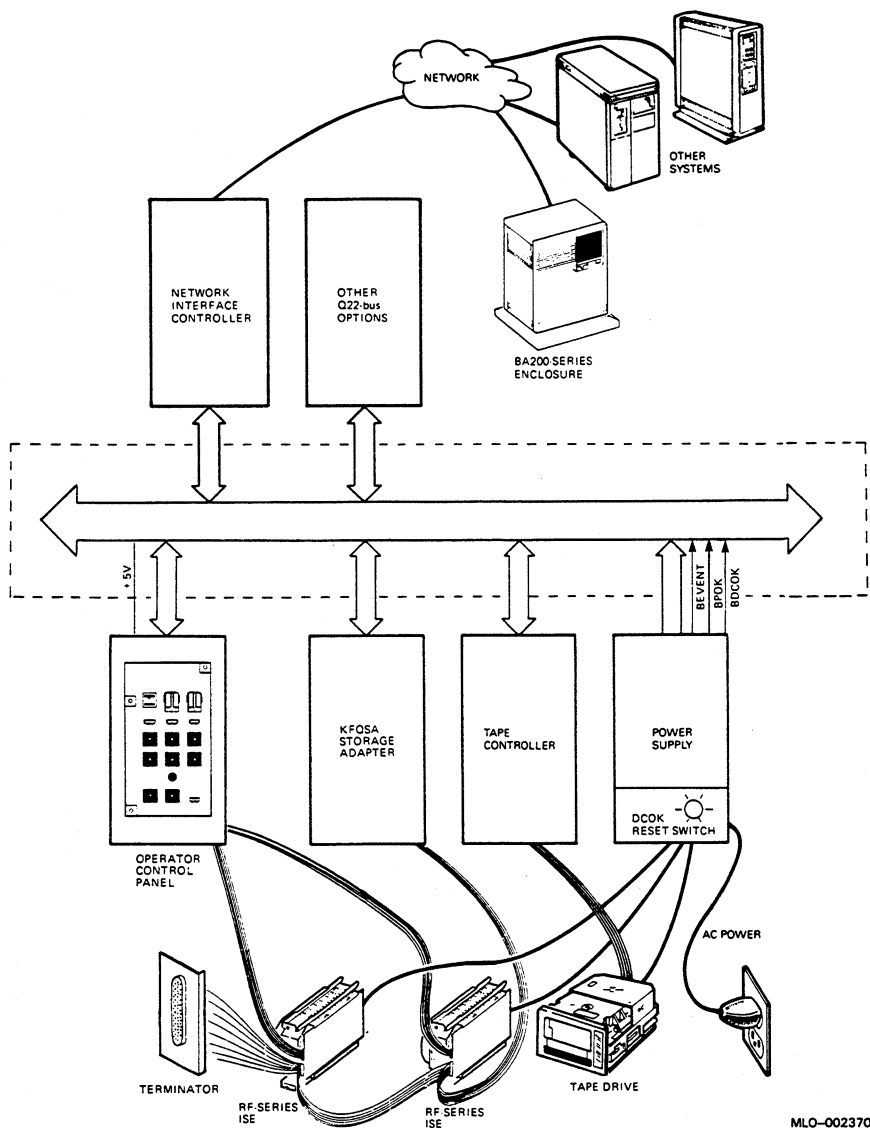
NOTE: *The KA655 CPU supports only the MS650-BA (16 Mbyte) memory module. The MS650-AA (8 Mbyte) is not supported because of its slower access speed.*

The KA655 CPU communicates with the console device through the H3600-SA CPU I/O panel, which contains configuration switches and an LED display. The H3600-SA is described in Section 1.4.

Figure 1-3: KA655 System-Level Block Diagram, Part I



1-6 KA655 CPU System Maintenance



1.2 KA655 CPU Features

The KA655 CPU provides the functionality of the KA650 CPU, but reduces the cycle time from 90 to 60 nanoseconds (ns), placing the KA655 performance at about 3.8 VAX Units of Performance (VUPs).

The major features of the KA655 CPU are as follows:

- A VAX central processor with a 33-MHz clock rate that supports the MicroVAX chip subset of the VAX instruction set and data types, plus the following string instructions: CMPC3, CMPC5, LOCC, SCANC, SKPC, and SPANC. The processor also 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 two-level cache consisting of a 1-Kbyte, 60-ns, first-level cache and a 64-Kbyte, 120-ns, second-level cache. Both caches provide parity protection on the tag and data stores.
- A main memory controller that supports up to 64 Mbytes of 450-ns ECC memory. The controller resides on the CPU module and supports up to four MS650-BA memory modules, depending on the system configuration.
- A console port compatible with VAX processors. The console port has an external baud-rate switch located on the CPU I/O panel (H3600-SA).
- A set of processor clock registers that support:
 - A VAX standard time-of-year (TOY) clock with support for battery backup. (The batteries are located on the inside of the H3600-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 located on the H3600-SA. It also supports 128 Kbytes of 16 bit-wide ROM containing programs for:
 - Board initialization
 - Emulation of a subset of the VAX standard console
 - Power-up self-testing of the KA655 and MS650-BA modules
 - Booting from supported Q22-bus devices

- Help utility
- KFQSA programming utility
- A Q22-bus interface that 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. This interface contains:
 - A 16-entry map cache for the scatter-gather map, which resides in main memory. This 8192-entry map is used for translating 22-bit, Q22-bus addresses into 26-bit, main memory addresses.
 - Interrupt arbitration logic that recognizes Q22-bus interrupt requests BR7 through BR4.
 - An interprocessor communications facility that supports communication between the Q22-bus arbiter and up to three auxiliary processors by means of doorbell interrupts.

1.2.1 Central Processing Unit (CPU)

The central processing unit (CPU) is implemented by the CVAX chip. The CVAX chip contains approximately 180,000 transistors in an 84-pin CERQUAD surface mount package. The chip achieves a 60-ns microcycle and a 120-ns bus cycle at an operating frequency of 33 MHz. The chip also supports full VAX memory management and a 4-Gbyte virtual address space.

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

The CVAX chip performs the following functions:

- Fetches all VAX instructions
- Executes 181 VAX instructions
- Assists in the execution of 21 additional instructions
- Passes 70 floating-point instructions to the CFPA60 chip

The remaining 32 VAX instructions (including `h_floating` and `octaword`) are emulated in macrocode.

In addition, 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 the 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 the CFPA60 chip. The CFPA60 chip contains approximately 60,000 transistors in a 68-pin CERQUAD surface mount package. It processes `f_`, `d_`, and `g_` floating format instructions and accelerates the execution of `MULL`, `DIVL`, and `EMUL` integer instructions. The CFPA60 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 KA655 CPU module contains a two-level cache memory to maximize CPU performance.

The first-level cache is implemented within the CVAX chip. This cache is a 1-Kbyte, two-way associative, write-through cache memory. It has a 60-ns cycle time.

The second-level cache is implemented using 16K x 4-bit static RAMs. This cache is a 64-Kbyte, direct-mapped, write-through cache memory. It has a 120-ns cycle time for longword transfers and a 180-ns cycle time for quadword transfers.

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 up to 64 Mbytes of 450-ns ECC memory.

The ECC memory resides on from one to four 16-Mbyte memory modules (MS650-BA), depending on the system configuration. The MS650-BA modules communicate with the KA655 through the MS650 memory interconnect, which utilizes the CD interconnect by means of 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-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 one 27510 EPROM. The firmware gains control when the processor halts. It contains programs that provide the following services:

- Board initialization
- Power-up self-testing of the KA655 and MS650-BA 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 KA655 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. The CQBIC chip supports block mode transfers as follows:

- Sixteen words on memory write
- Four words on memory read
- Longword transfers between CPU and Q22-bus

The Q22-bus interface contains the following:

- A 16-entry map cache for the 8192-entry scatter-gather map that resides in main memory. The map is used for translating 22-bit, Q22-bus addresses into 26-bit, main memory addresses.
- Interrupt arbitration logic that recognizes Q22-bus interrupt requests BR7 through BR4.

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

The KA655-AA module contains 240-ohm termination for the Q22-bus.

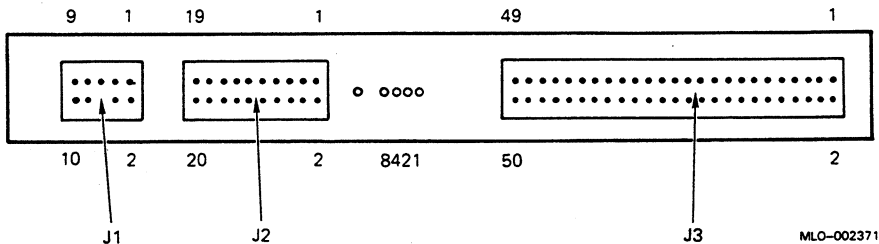
1.3 KA655 Connectors

The KA655 CPU module has three connectors:

- J1. For the cable from the internal console SLU connector on the H3600-SA CPU I/O panel.
- J2. For the cable from the configuration and display connector on the H3600-SA. (Connects the CPU to the three switches and LED display on the I/O panel.)
- J3. For a cable from the first MS650-BA memory module.

The orientation of connectors J1, J2, and J3 is shown in Figure 1-5.

Figure 1-5: KA655 Pin Orientation

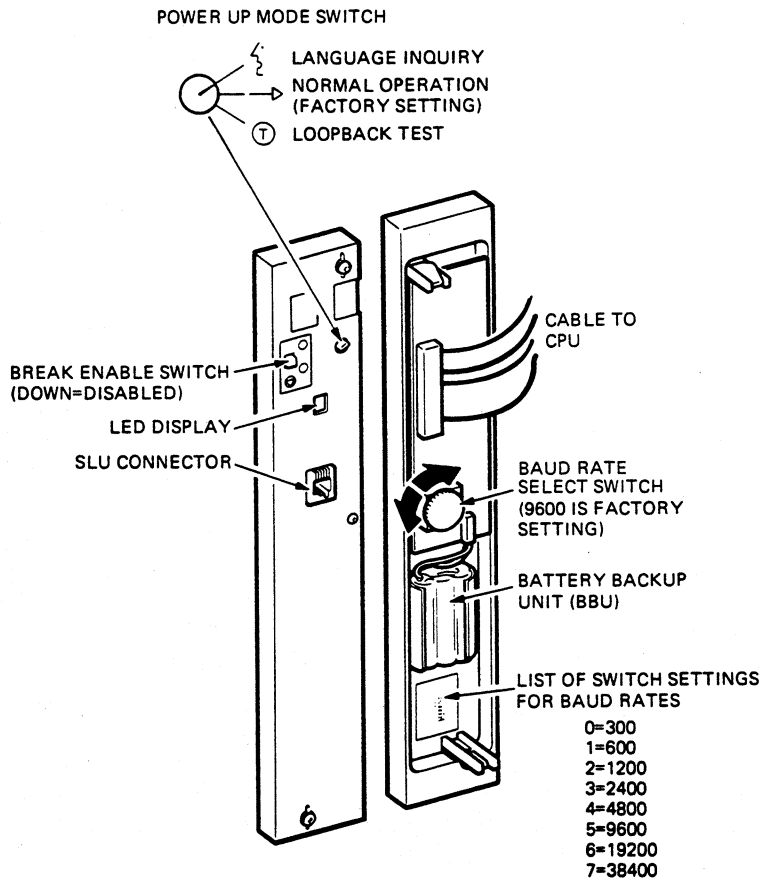


1.4 H3600-SA CPU I/O Panel

The H3600-SA CPU I/O panel, shown in Figure 1-6, has a ribbon cable with two connectors that plug into the KA655 module console SLU and console control connectors. The H3600-SA fits over backplane slots 1 and 2, covering both the KA655 module and the first of up to four possible MS650-BA memory modules.

The controls and connectors on the H3600-SA are shown in Figure 1-6 and listed in Table 1-1.

Figure 1-6: H3600-SA CPU I/O Panel



MLO-002372

Table 1-1: H3600-SA Controls and Connectors

Outside	Inside
Modified modular jack (MMJ) SLU connector	Cable to SLU connector on KA655
Power-up mode switch	Battery backup unit (BBU)
Hex LED display	Cable to console control connector on KA655
Enable/disable switch	Baud rate select switch

Enable/Disable Switch

Although the KA630, KA650, and KA655 CPU modules all use the H3600-SA, the function of the enable/disable switch is slightly different in each case:

- When the H3600-SA is connected to a KA630 CPU, the switch enables and disables halts from the **[BREAK]** key on the console keyboard and from the halt button on the system front panel. In MicroVAX II systems, the switch is referred to as the *halt* enable switch.
- When the H3600-SA is connected to a KA650 or KA655 CPU, the switch enables and disables halts from the **[BREAK]** key on the console keyboard *only*. You cannot disable halts initiated from the halt button on the system front panel. In MicroVAX 3500, 3600, 3800, and 3900 systems, the switch is referred to as the *break* enable switch.

You also select the power-up mode and console terminal baud rate using switches on the H3600-SA. Release the H3600-SA quarter-turn fasteners and turn the H3600-SA around (do not disconnect the ribbon cable) to change the baud rate of the console serial line, or to change the batteries for the backup unit. The factory setting for the baud rate is 9600. Figure 1-6 shows the possible baud rate settings.

Hexadecimal LED Display

The LEDs display a hexadecimal number for each power-up test and stage of the bootstrap process. Chapter 4 lists the meanings of these numbers.

Modified Modular Jack SLU Connector (Outside)

The modified modular jack (MMJ) is a six-pin connector for a cable that connects to the console terminal.

Battery Backup Unit (Inside)

When the system is turned off, the battery backup unit (BBU) provides power to the KA655 time-of-year logic (25.6-kHz oscillator, TODR register, and 1 Kbyte of RAM in the SSC). The 1 Kbyte of RAM stores the code for the language that is displayed in the console messages. If the BBU fails, the code is lost.

Cable

The cable contains two connectors that plug into the console SLU and console control connectors on the KA655 module. The 20-conductor end connects the baud rate select switch, the power-up mode switch, and the hexadecimal LED display to the console control connector (J2) of the KA655. The 10-conductor end connects the console serial line to connector J1 on the KA655.

Table 1-2 lists the H3600-SA switch functions.

Table 1-2: H3600-SA CPU I/O Panel Switches

Switch	Position	Function
Enable/disable (two-position toggle)	Dot outside circle	Breaks are disabled (factory setting). On power-up or restart, the system tries to load software from one of the devices after completing the power-up diagnostics.
	Dot inside circle	Breaks are enabled. On power-up or restart, the system enters console I/O mode after completing the power-up diagnostics.
Power-up mode (three-position rotary)	Arrow	Run (factory setting). If the console terminal supports the DEC multinational character set (MCS), the system prompts the user for the console language if the battery backup has failed and upon initial power-up of a system containing a new CPU. All start-up diagnostics run.
	Face	Language inquiry. If the console terminal supports the DEC MCS, the user is prompted for the console language on every power-up and restart. All power-up diagnostics run.
	T inside circle	Test. ROM programs run the wraparound console serial line unit (SLU) tests.
Baud rate select	300 to 38,400	Sets the baud rate of the console terminal serial line. The factory setting is 9600 baud. The baud rate of this switch must match the rate of the console terminal.

1.5 MS650-BA Memory

The MS650-BA (M7622-A) is a 16-Mbyte memory module that provides memory for the KA655 CPU module. The MS650-BA is a nonintelligent memory array module controlled by a custom memory controller chip (CMCTL) on the KA655 CPU module.

The quad-height MS650-BA, shown in Figure 1-7, has a 450 ns, 39 bit-wide array (32-bit data and 7-bit ECC), implemented with 1 Mbit dynamic RAMs in surface mounted SOJs.

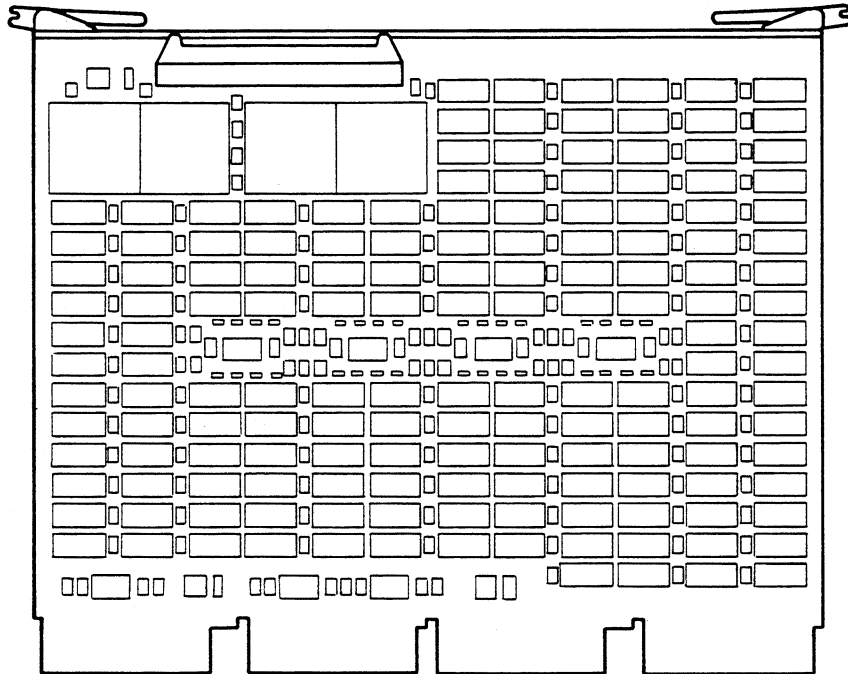
Ordering Information

MS650-BA	16-Mbyte module only (M7622-A).
MS650-BF	Option installation kit for BA200-series enclosures. Includes MS650-BA, filler panel assemblies, blank cover, CPU cable, labels, and installation guide.

Diagnostic Support

MicroVAX Diagnostic Monitor	Release 126 (version 3.01)
Self-test	KA655 self-test

Figure 1-7: MS650-BA Memory Module (M7622-A)



MLO-002373

Chapter 2

Configuration

2.1 Introduction

This chapter describes the guidelines for changing the configuration of a KA655 system, and for configuring a dual-host 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 gaining access to the bus

2.2.1 Module Order Rules for KA655 Systems

Observe the following rules about module order:

- Install the KA655 CPU in slot 1.
- Install a MS650-BA memory module in slot 2. Install any additional MS650-BA memory modules in slots 3 through 5.
- 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.

2.2.2 Recommended Module Order for KA655 Systems

Here is the recommended module order for a KA655 system:

KA655
MS650-BA
AAV11-SA
ADV11-SA
AXV11-SA
KVV11-SA
DRV1J-SA
TSV05-SA
DMV11
LNV21
DEQNA/DELQA/DESQA-SA
DPV11-SA
KMOV1A-SA, -SB, -SC
DZQ11-SA
DFA01-AB
CXY08-AA
CXB16-M/CXA16-M
CSF32-M
LPV11-SA
DRV1W-SA
IEQ11-SA
ADQ32-M
DRQ3B-SA
IBQ01-SA

KLESI-SA
 TQK50-SA/TQK70-SA
 RQDX3-SA
 KDA50-SA
 KFQSA-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:

```

>>> configure
Enter device configuration, HELP, or EXIT
Device,Number? help
Devices:
LPV11    KXJ11    DLV11J   DZQ11    DZV11    DFA01
RLV12    TSV05    RXV21    DRV11W   DRV11B   DPV11
DMV11    DELQA    DEQNA    DESQA    RQDX3    KDA50
RRD50    RQC25    KFQSA-DISK TQK50    TQK70    TU81E
RV20     KFQSA-TAPE KMV11    IEQ11    DHQ11    DHV11
CXA16    CXB16    CXY08    VCB01    QVSS     LNV11
LNV21    QPSS     DSV11    ADV11C   AAV11C   AXV11C
KVV11C   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? exit
  
```

See the CONFIGURE command in Chapter 3 (Section 3.9.2) for an example of how to set the correct CSR addresses and interrupt vectors.

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

2.4 DSSI Configuration

The KA655 CPU and memory subsystem support RF-series integrated storage elements (ISEs) and the KFQSA storage adapter.

The RF-series ISEs are part of a series of mass storage devices based on the Digital Storage Architecture (DSA). These devices use the Digital Storage System Interconnect (DSSI) bus and interface. The term *integrated storage element* is used because each DSSI ISE has an on-board intelligent controller and mass storage control protocol (MSCP) server in addition to the drive and the control electronics.

DSSI supports up to seven ISEs daisy-chained through a single cable to the KFQSA storage adapter. The KFQSA is a protocol converter that supports Q-bus protocols to and from the KA655 CPU and DSSI bus protocols to and from the ISEs. The KFQSA contains the addressing logic required to make a connection between the host and a requested ISE on the DSSI bus. DSSI adapters can also be embedded on a CPU module, such as on the KA640.

Each ISE must have a unique node ID. The ISE receives its node ID from a plug on the operator control panel (OCP) on the front panel of the BA200-series enclosure. By convention, DSSI devices are mounted in the BA200-series enclosures from right to left, as listed in Table 2-1.

Table 2-1: DSSI Device Order

Device	Position	Node ID ¹
First	Right side	0
Second	Center	1
Third	Left side	2

¹KA655 node ID = 7

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

The node ID switches are located behind the 50-pin connector on the ECM. Switch 1 (the MSB) is nearest to the connector. Refer to the RF30 or RF71 section in *Microsystems Options* for more information. Table 2-2 lists the switch settings for the eight possible node addresses.

Table 2-2: RF-Series ISE Switch Settings

DSSI Node ID	Switch ¹		
	1 (MSB)	2	3 (LSB)
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 ²	Up	Up	Up

¹Up = toward the head disk assembly (HDA); Down = toward the drive module

²Normally reserved for the host adapter

NOTE: *Pressing the system reset button on the front of a BA200-series enclosure has no effect on ISEs. If you change a node ID, you must perform a power cycle to enable the new node ID to take effect.*

The VMS operating system creates DSSI device names according to Dlan, where a is the controller letter (A, B, C, and so on) and n is the unit number.

You can gain access to local programs in the ISE through the MicroVAX Diagnostic Monitor (MDM) or through the 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 dialogue. Section 3.9.15 describes the console I/O mode SET/HOST/DUP/UQSSP command and shows an example of how to set host to the RF71 ISE through the KFQSA storage adapter.

2.4.1 Changing RF-Series ISE Parameters

Each ISE has a node name that is maintained in EEPROM on board the controller module. This node name is determined in manufacturing from an algorithm based on the device 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 RF71 ISE at DSSI node address 1 is changed from R3YBNE to DATADISK.

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

Example 2-1: Changing a DSSI Node Name

```
>>> sho uqssp
UQSSP Disk Controller 0 (772150)
-DUA0 (RF71)

UQSSP Disk Controller 1 (760334) !The node name for this drive
-DUB1 (RF71)                    !will be changed from R3YBNE
                                !to DATADISK.

>>> set host/dup/uqssp/disk 0
Starting DUP server...
Copyright 1988 Digital Equipment Corporation
DRVEXR V1.1 D 6-MAR-1989 15:33:06
DRVTST V1.1 D 6-MAR-1989 15:33:06
HISTORY V1.0 D 6-MAR-1989 15:33:06
ERASE V1.3 D 6-MAR-1989 15:33:06
PARAMS V1.2 D 6-MAR-1989 15:33:06
DIRECT V1.0 D 6-MAR-1989 15:33:06
End of directory

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

Example 2-1 Cont'd. on next page

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

```
PARAMS> sho nodename
```

Parameter	Current	Default	Type	Radix
NODENAME	R3YBNE	RF71	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 uqssp
```

```
UQSSP Disk Controller 0 (772150)
```

```
-DUA0 (RF71)
```

```
UQSSP Disk Controller 1 (760334) !The node name has changed
```

```
-DUB1 (RF71)                    !from R3YBNE to DATADISK,  
                                !although the display remains  
                                !unchanged.
```

2.4.2 Changing the Unit Number

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

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

Example 2-2 shows how to change the unit number of a DSSI device. This example changes the unit number for the RF71 ISE at DSSI node address 1 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 more information about the PARAMS local program.

Example 2-2: Changing a DSSI Unit Number

```
>>> sho uqssp
UQSSP Disk Controller 0 (772150)
-DUA0 (RF71)

UQSSP Disk Controller 1 (760334) !The unit number for this
-DUB1 (RF71)                    !drive will be changed from
                                !1 to 50 (DUB1 to DUB50).

>>> set host/dup/uqssp/disk 1
Starting DUP server...
Copyright 1988 Digital Equipment Corporation
DRVEXR V1.1 D 6-MAR-1989 15:33:06
DRVIST V1.1 D 6-MAR-1989 15:33:06
HISTORY V1.0 D 6-MAR-1989 15:33:06
ERASE V1.3 D 6-MAR-1989 15:33:06
PARAMS V1.2 D 6-MAR-1989 15:33:06
DIRECT V1.0 D 6-MAR-1989 15:33:06
End of directory

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 uqssp
UQSSP Disk Controller 0 (772150)
-DUA0 (RF71)
UQSSP Disk Controller 1 (760334) !The unit number has changed
-DUB50 (RF71)                    !from 1 to 50. The node ID
                                !remains at 1.
```


2.4.3 Changing the Allocation Class

If the system is part of a cluster, you must change the default allocation class parameter. The ISEs ship with the allocation class set to zero. Determine the new allocation class for the RF-series ISEs according to the rules on clustering.

NOTE: *In a dual-host configuration, you must assign the same allocation class to both host systems and to the RF-series ISEs. This allocation class must be different from that of other systems or of hierarchical storage controllers (HSCs) in a cluster.*

Change the allocation class and unit number parameters by setting host to the console-based DUP driver utility. Example 2-3 shows how to change the allocation class of a DSSI device. This example displays the existing allocation class, then resets the allocation class to 2.

Example 2-3: Changing a DSSI Allocation Class

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

UQSSP Disk Controller 0 (772150)
Copyright (c) 1988 Digital Equipment Corporation

PARAMS> sho allclass

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

PARAMS> set allclass 2

PARAMS> sho allclass

Parameter      Current      Default      Type      Radix
-----
ALLCLASS              2              0      Byte      Dec      B

PARAMS> write

Changes require controller initialization, ok? [Y/ (N) ] y

Stopping DUP server...
```

2.4.4 DSSI Cabling

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

A 10-conductor cable connects the ISE connector to the operator control panel (OCP, Figure 2-2). In the BA213 enclosure, there are two cables that connect the power supplies to the OCP; one cable connects to the right power supply, and the other connects to the left power supply.

These cables carry the ACOK signal (same as POK) to the ISE. The OCP delays this signal to one ISE for each power supply in order to stagger the start-up of one of two possible devices attached to each supply. This delay prevents the ISEs from drawing excessive current at power-up.

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 KFQSA storage adapter.

CAUTION: *When removing or installing new ISEs, be sure to connect the rightmost connector of the DSSI ribbon cable to the round cable connected to the KFQSA. 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

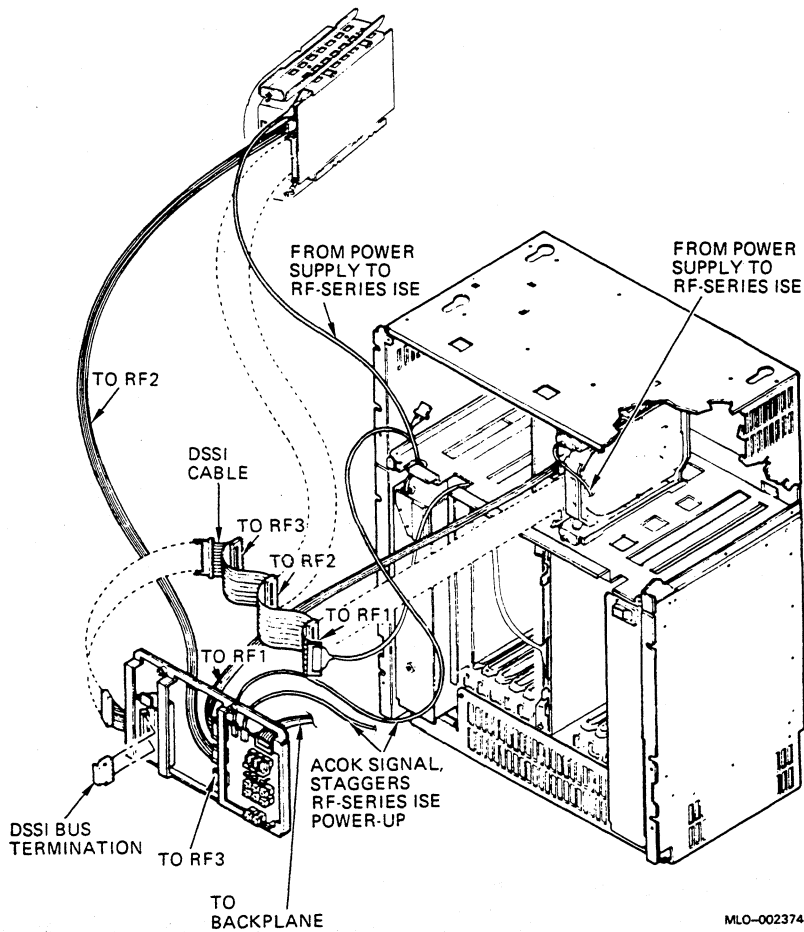
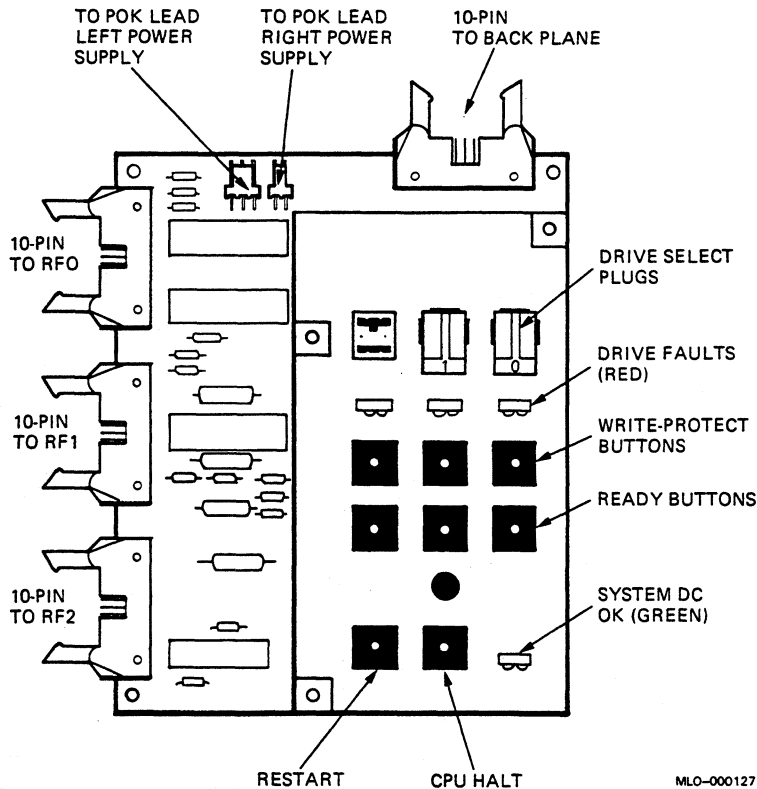


Figure 2-2: RF-Series ISE Operator Control Panel (OCP)



2.4.4.1 DSSI Bus Termination and Length

The DSSI bus must be terminated at both ends. The KFQSA storage adapter terminates the DSSI bus at one end. A terminator connected to the 50-conductor Honda connector on the left side of the media faceplate terminates the bus at the other end. You can remove this terminator if you need to expand the bus.

CAUTION: *Connect the DSSI bus using cables approved by Digital. Use approved configurations only.*

In a dual-host system, described below, the second KFQSA storage adapter provides the bus termination.

2.4.5 Dual-Host Capability

An RF-series ISE has dual-host capability built into the firmware, which allows the device to maintain connections with more than one DSSI adapter. You can connect more than one KFQSA storage adapter to the same DSSI bus to allow each KFQSA access to all other devices on the bus.

The primary application for such a configuration is a VAXcluster system that uses Ethernet as the interconnect medium between the boot node and the satellite members. This configuration improves system availability, as described below.

Two kernel systems are connected through an external DSSI cable (BC21M). For this discussion, a kernel system includes the following:

- KA655 CPU
- MS650-BA memory
- KFQSA storage adapter
- Ethernet adapter
- Q22-bus

If this dual configuration is used to boot a number of satellite nodes, the system disk resides in one of these enclosures and serves as the system disk for both kernel systems. The KFQSA storage adapter in each enclosure has equal access to the system disk and to any other DSSI ISE in either enclosure.

If one of the kernel systems fails, all satellite nodes booted through that kernel system lose connections to the system disk. However, the dual-host capability enables each satellite node to know that the system disk is still available through a different path—that of the remaining operational kernel system. A connection through that kernel system is then established, and the satellite nodes are able to continue operation.

Thus, even if one KFQSA adapter or any other component of the kernel system fails, the satellites booted through that system are able to continue operation. In this case, the entire cluster will run in a degraded condition, since one kernel system is now serving the satellite nodes of both systems. 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. ISE failure can be caused by such factors as a power supply failure in the enclosure containing the ISE.

- DSSI cabling failure. If a failure in one of the DSSI cables renders access to the ISEs 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 Limitations to Dual-Host Configurations

The following limitations apply to dual-host systems:

- Because of cabling and enclosure limitations, you can connect a maximum of two systems.
- The DSSI bus supports eight devices or adapters. Since a dual-host system has two KFQSA adapters, and each has a connection to the DSSI bus, you can attach a maximum of six DSSI devices to the bus.

NOTE: *You can make dual-host connections to the same type of DSSI adapters only. For example, KFQSA to KFQSA or KA640 to KA640.*

2.5 Configuration Worksheet

This section provides a configuration worksheet of the BA213 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.

Table 2-3 lists power values for supported devices. To check a system configuration, follow these steps:

1. List all the devices already installed in the system.
2. List all the devices you plan to install in the system.
3. Fill in the information for each device, using the data listed in Table 2-3.
4. 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 2–3: Power and Bus Loads for KA655 Options

Option	Module	Current (Amps)		Power	Bus Loads	
		+5 V	+12 V	Watts	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
KA655-AA	M7625-AA/-BA	3.7	0.14	21.0	2.2	1.0
KDA50-Q	M7164	6.93	0	34.6	3.0	0.5
KDA50-Q	M6165	6.57	0.03	33.21	–	–
KFQSA-SA	M7769	5.5	0.0	27.0	3.8	0.5
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-SA	–	1.10	0.80	15.1	–	–
RF71E-SA	–	1.25	4.54	26.5	–	–
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)		Power (Watts)
		+5 Vdc	+12 Vdc	
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)		Power (Watts)
		+5 Vdc	+12 Vdc	
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

KA655 Firmware

3.1 Introduction

This chapter describes the KA655 firmware, which gains control of the processor whenever the KA655 performs a processor halt. A processor halt transfers control to the firmware; the processor does not actually stop executing instructions.

3.2 KA655 Firmware Features

The firmware is located in one 128-Kbyte EPROM on the KA655. The firmware address range (hexadecimal) in the KA655 local I/O space is 20040000 to 2007FFFF, inclusive (20040000–2005FFFF is halt-protected space and 20060000–2007FFFF is halt-unprotected space). The firmware displays diagnostic progress and error reports on the KA655 LEDs and on the console terminal.

The firmware performs the following functions:

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

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

The firmware consists of the following major functional areas:

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

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

3.3 Halt Entry, Exit, and Dispatch Code

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

After a halt, the halt entry code saves the current LED code, then writes an E to the LEDs. An E on the LEDs indicates that at least several instructions have been successfully executed, although if the CPU is functioning properly, the E flashes too quickly to be seen.

The halt entry code saves the following registers. The console intercepts any direct reference to these registers and redirects it to the saved copies:

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

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

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

When the processor exits the firmware and reenters program mode, the saved registers are restored and any changes become operative only then. References to processor memory are handled in the normal way. 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 H3600-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 H3600-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

¹Power-up halt: PR\$_SAVPSL<13:08>=3

²Halt action: CPMBX<01:00>

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

3.4 External Halts

The following conditions can trigger an external halt, and different actions are taken depending on the condition:

- The break enable switch is set to enable, and you press **BREAK** on the system console terminal.
- Assertion of the BHALT line on the Q22-bus, if the SCR<14>(BHALT_ENABLE) bit in the CQBIC is set.
- Negation of DCOK, if the SCR<7>(DCOK_ACT) bit is set.

The KA655 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 the previously saved system state.*

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 μ sec. 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.

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 H3600-SA (Figure 1-6). The mode switch has three settings: test, language inquiry, and normal. The differences are described in Sections 3.5.1 through 3.5.3.

The 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.

3.5.1 Mode Switch Set to Test

Use the test position on the H3600-SA to verify a proper connection between the KA655 and the console terminal:

- To test the console terminal port, insert the H3103 loopback connector into the H3600-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.

3.5.2 Mode Switch Set to Language Inquiry

If the H3600-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.
Memory configuration error.
No default boot device has been specified.
Available devices.
Device?
Retrying network bootstrap.

The language selection menu appears under the conditions listed in Table 3-2. The position of the break enable switch has no effect on these conditions.

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

¹Action if contents of NVRAM invalid same as Language Not Previously Set.

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

In addition, the console may prompt you for a default boot device. See Section 3.6, Example 3-2.

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

3.5.3 Mode Switch Set to Normal

The console displays the language selection menu if the mode switch is set to normal and the contents of 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 KA655 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, called the virtual memory bootstrap (VMB), as follows:

3.6.1 Bootstrap Initialization Sequence

1. Checks CPMBX<2>(BIP), bootstrap in progress. If it is set, bootstrap fails and the console displays the message `Failure.` in the selected console language.
2. If this is an automatic bootstrap, prints the message `Loading system software.` on the console terminal.
3. Validates the boot device name. If none exists, supplies a list of available devices and issues a boot device prompt. If you do not specify a device within 30 seconds, uses `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 <code>PR\$_TODR</code>
R5	Boot flags
R10	Halt PC value
R11	Halt PSL value (without halt code and map enable)
AP	Halt code
SP	Base of 128-Kbyte good memory block + 512
PC	Base of 128-Kbyte good memory block + 512
R1, R6, R7, R8, R9, FP	0
10. Copies the VMB image from EPROM to local memory, beginning at the base of the 128 Kbytes of good memory block + 512 (decimal).
11. Exits from the firmware to VMB residing in memory.

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

3.6.2 VMB Boot Flags

The VMB boot flags are listed in Table 3-3.

Table 3-3: Virtual Memory Bootstrap (VMB) Boot Flags

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

3.6.3 Supported Boot Devices

Table 3-4 lists the boot devices supported by the KA655-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 through 16,383).

Table 3—4: Boot Devices Supported by the KA655-AA

Boot Name	Controller Type	Device Type(s)
Disk		
DUcn	KFQSA DSSI	RF30, RF71
	RQDX3 MSCP	RD52, RD53, RD54, RX33, RX50
	KDA50 MSCP	RA70, RA80, RA81, RA82, RA90
	KLESI	RC25
DLcn	RLV21	RL01, RL02
Tape		
MUcn	TQK50 MSCP	TK50
	TQK70 MSCP	TK70
	KLESI	TU81E
Network		
XQcn	DEQNA	—
	DELQA	—
	DESQA	—
PROM		
PRA0	MRV11	—

3.6.4 Autoboot

IMPORTANT: *Unless you specify otherwise, the KA655 default boot device is the Ethernet adapter, XQmn. See Example 3-2.*

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

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

- If you do not type a boot device name within thirty seconds, the system boots from the Ethernet adapter, XQmn.
- If you type a boot device name within thirty seconds, this device becomes the default boot device and the system boots from that device, as shown in Example 3-2.

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

Example 3-2: Selecting a Boot Device

```
Performing normal system tests.
40..39..38..37..36..35..34..33..32..31..30..29..28..27..26..25..
24..23..22..21..20..19..18..17..16..15..14..13..12..11..10..09..
08..07..06..05..04..03..
Tests completed.
Loading system software.
No default boot device has been specified.

Available devices.
-DUA0 (RF71)
-DUB1 (RF30)
-MUA0 (TK70)
-XQA0 (08-00-2B-09-95-21)

Device? [XQA0]: dua0

(BOOT/R5:0 DUA0)

2..
-DUA0
1..0..
```

3.7 Operating System Restart

An operating system restart is the process of bringing up the operating system from a known initialization state following a processor halt. A restart occurs under the conditions listed in Table 3-1, 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 an RIP (restart-in-progress) flag in CPMBX, which it uses to avoid repeated attempts to restart a failing operating system. The operating system maintains an additional RIP flag in the RPB.

3.7.1 Restart Sequence

The firmware restarts the operating system in the following sequence:

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

Restarting system software.
3. Sets CPMBX<3>(RIP).
4. Searches for a valid RPB. If none is found, restart fails.
5. Checks the operating system RPB\$L_RSTRTFLG<0>(RIP) flag. If it is set, restart fails.
6. Writes a 0 (zero) to the diagnostic LEDs.
7. Dispatches to the restart address, RPB\$L_RESTART, with :
 SP = the physical address of the RPB plus 512
 AP = the halt code
 PSL = 041F0000
 PR\$_MAPEN = 0

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

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

Failure.

3.7.2 Locating the RPB

The RPB is a page-aligned control block that can be identified by its signature in the first three longwords:

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

The firmware finds a valid RPB as follows:

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

3.8 Console I/O Mode

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

RETURN

Also <CR>. The carriage return ends a command line. No action is taken on a command until after it is terminated by a carriage return. A null line terminated by a carriage return is treated as a valid, null command. No action is taken, and the console prompts for input. Carriage return is echoed as carriage return, line feed (<CR><LF>).

RUBOUT

When you press **RUBOUT**, the console deletes the previously typed character. The resulting display differs, depending on whether the console is a video or a hard-copy terminal.

For hard-copy terminals, the console echoes a backslash (\), followed by the deletion of the character. If you press additional rubouts, the additional deleted characters are echoed. If you type a 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.

CTRLU

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

CTRLS

Stops output to the console terminal until **CTRLQ** is typed. Not echoed.

CTRLQ

Resumes output to the console terminal. Not echoed.

CTRLR

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.

CTRLC

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 3–8.

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

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

The following symbols are qualifier and argument conventions:

- [] An optional qualifier or argument
- { } A required qualifier or argument

3.8.2 Address Specifiers

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

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

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

3.8.3 Symbolic Addresses

The console supports symbolic references to addresses. A symbolic reference defines the address space and the offset into that space. Table 3–5 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 3–5: 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	0E	PC	0F
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 3–5 (Cont.): Console Symbolic Addresses

Symbol	Address	Symbol	Address
Physical Memory (/P)			
qbio	20000000	qbmemb	30000000
qbmbbr	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	20080118	memcsr7	2008011C
memcsr8	20080120	memcsr9	20080124
memcsr10	20080128	memcsr11	2008012C
memcsr12	20080130	memcsr13	20080134
memcsr14	20080138	memcsr15	2008013C
memcsr16	20080140	memcsr17	20080144

Table 3–6 lists symbolic addresses that you can use in any address space.

Table 3–6: 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 3–7 lists and describes the data control and address space control qualifiers. Command specific qualifiers are listed in the descriptions of individual commands.

Table 3–7: 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, uses a value of 3, which forces a double bit error. On reads, ignores 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 3–8 lists command keywords by type. Table 3–9 lists the parameters, qualifiers, and arguments for each console command. Parameters, used with the SET and SHOW commands only, are listed in the first column along with the command.

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

Table 3–8: 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 3–9: 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 3–9 (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 /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 ETHERNET	–	–	–
SHOW LANGUAGE	–	–	–
SHOW MEMORY	/FULL	–	–
SHOW QBUS	–	–	–
SHOW RLV12	–	–	–
SHOW UQSSP	–	–	–
SHOW VERSION	–	–	–
START	–	{address}	–
TEST	–	{test_number}	[[parameters]]
UNJAM	–	–	–
X	–	{address}	{count}

3.9 Console Commands

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

3.9.1 BOOT

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

Format:

BOOT [qualifier-list] [device_name]

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

Set the default boot device and boot flags with the SET BOOT and SET BFLAG commands. If you do not set a default boot device, the processor times out after 30 seconds and attempts to boot from the Ethernet port, XQA0.

Qualifiers:

Command specific:

/R5:{boot_flags} A 32-bit hex value passed to VMB in R5. The console does not interpret this value. Use the SET BFLAG command to specify a default boot flags longword. Use the SHOW BFLAG command to display the longword. Table 3-3 lists the supported R5 boot flags.

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

[device_name] A character string of up to 17 characters. Longer strings cause a VAL TOO BIG error message. Apart from checking the length, the console does not interpret or validate the device name. The console converts the string to uppercase, then passes VMB a string descriptor to this device name in R0. Use the SET BOOT command to specify a default boot device. Use the SHOW BOOT command to display the default boot device. The factory default device is the Ethernet port, XQA0. Table 3-4 lists the boot devices supported by the KA655-AA.

Examples:

```
>>> show boot
DUA0
>>> show bflag
0
>>> b                ! Boot using default boot flags and device.
(BOOT/R5:0 DUA0)

  2..
-DUA0

>>> bo xqa0           ! Boot using default boot flags and
(BOOT/R5:0 XQA0)      ! specified device.

  2..
-XQA0

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

  2..
-DUA0

>>> boot /r5:220 xqa0 ! Boot using specified boot flags and
(BOOT/R5:220 XQA0)   ! 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 KA655-AA CPU.

Format:

CONFIGURE

Example:

```
>>> configure
Enter device configuration, HELP, or EXIT
Device,Number? help
Devices:
LPV11   KXJ11       DLV11J   DZQ11   DZV11   DFA01
RLV12   TSV05       RXV21   DRV11W  DRV11B  DPV11
DMV11   DELQA       DEQNA   DESQA   RQDX3   KDA50
RRD50   RQC25       KFQSA-DISK  TQK50   TQK70   TU81E
RV20    KFQSA-TAPE  KMV11   IEQ11   DHQ11   DHV11
CXA16   CXB16       CXY08   VCB01   QVSS    LNV11
LNV21   QPSS       DSV11   ADV11C  AAV11C  AXV11C
K WV11C 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? ridx3,2
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, /M, /P, /V, /U

Arguments:

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

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

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

Examples:

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

>>> D/V/L/N:3 1234 5           ! Deposit 5 into four longwords
                                 ! starting at virtual memory address
                                 ! 1234.

>>> D/N:8 R0 FFFFFFFF          ! Loads GPRs R0 through R8 with -1.

>>> D/L/P/N:10/ST:200 0 8      ! Deposit 8 in the first longword of
                                 ! the first 17 pages in physical
                                 ! memory.

>>> D/N:200 - 0                ! Starting at previous address, clear
                                 ! 513 longwords or 2052 bytes.
```

3.9.5 EXAMINE

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

EXAMINE uses the same qualifiers as DEPOSIT. However, the /WRONG qualifier causes 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, /M, /P, /V, /U

Command specific:

/INSTRUCTION Disassembles and displays the VAX Macro-32 instruction at the specified address.

Arguments:

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

Examples:

```
>>> ex pc                            ! Examine the PC.
   G 0000000F FFFFFFFC
>>> ex sp                            ! Examine the SP.
   G 0000000E 00000200
>>> ex psl                           ! Examine the PSL.
   M 00000000 041F0000
>>> e/m                              ! Examine PSL another way.
   M 00000000 041F0000
>>> e r4/n:5                         ! Examine R4 through R9.
   G 00000004 00000000
   G 00000005 00000000
   G 00000006 00000000
   G 00000007 00000000
   G 00000008 00000000
   G 00000009 801D9000
```

```

>>> ex pr$_scbb                ! Examine the SCBB, IPR 17
    I 00000011 2004A000        ! (decimal).

>>> e/p 0                      ! Examine local memory 0.
    P 00000000 00000000

>>> ex /ins 20040000           ! Examine 1st byte of ROM.
    P 20040000 11 BRB 20040019

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

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

```

3.9.6 FIND

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

Format:

FIND [qualifier-list]

Qualifiers:

Command specific:

/MEMORY Searches memory for a page-aligned block of good memory, 128 Kbytes in length. The search looks only at memory that is deemed usable by the bitmap. This command leaves the contents of memory unchanged.

/RPB Searches all 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 Kbytes.
>>> 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/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 ETHERNET
SHOW LANGUAGE
SHOW MEMORY [/FULL]
SHOW QBUS
SHOW RLV12
SHOW UQSSP
SHOW VERSION
HALT
INITIALIZE
UNJAM
```

```
CONTINUE
START <ADDRESS>
REPEAT <COMMAND>
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
Caches	Flushed
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, /Q, /N:{count}, /STEP:{size}, /WRONG

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

Arguments:

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

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

Examples:

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

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

>>> mov/n:4 200 0                    ! Move the data.
```

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

3.9.11 NEXT

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

After the last macro instruction is executed, the console reenters console I/O mode.

Format:

NEXT {count}

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

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

Arguments:

{count} A value representing the number of macro instructions to execute.

Examples:

```
>>> dep 1000 50D650D4           ! Create a simple program.
>>> dep 1004 125005D1
>>> dep 1008 00FE11F9
>>> ex /instruction /n:5 1000    ! List it.
P 00001000  D4 CLRL    R0
P 00001002  D6 INCL    R0
P 00001004  D1 CMPL    S^#05,R0
P 00001007  12 BNEQ    00001002
P 00001009  11 BRB     00001009
P 0000100B  00 HALT
>>> dep pr$_scbb 200            ! Set up a user SCBB
>>> dep pc 1000                 ! and the PC.
>>>
>>> n                           ! Single.
P 00001002  D6 INCL    R0
```

```

>>> n
P 00001004 D1 CMPL S^#05,R0
>>> n
P 00001007 12 BNEQ 00001002
>>> n
P 00001002 D6 INCL R0
>>> n 5 ! ...or multiple step
P 00001004 D1 CMPL S^#05,R0 ! the program.
P 00001007 12 BNEQ 00001002
P 00001002 D6 INCL R0
P 00001004 D1 CMPL S^#05,R0
P 00001007 12 BNEQ 00001002
>>> n 7
P 00001002 D6 INCL R0
P 00001004 D1 CMPL S^#05,R0
P 00001007 12 BNEQ 00001002
P 00001002 D6 INCL R0
P 00001004 D1 CMPL S^#05,R0
P 00001007 12 BNEQ 00001002
P 00001009 11 BRB 00001009
>>> n
P 00001009 11 BRB 00001009
>>>

```

3.9.12 REPEAT

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

Format:

REPEAT {command}

Arguments:

{command} A valid console command other than REPEAT.

Examples:

```
>>> repeat ex pr$_todr          ! Watch the clock.
I 0000001B 5AFE78CE
I 0000001B 5AFE78D1
I 0000001B 5AFE78FD
I 0000001B 5AFE7900
I 0000001B 5AFE7903
I 0000001B 5AFE7907
I 0000001B 5AFE790A
I 0000001B 5AFE790D
I 0000001B 5AFE7910
I 0000001B 5AFE793C
I 0000001B 5AFE793F
I 0000001B 5AFE7942
I 0000001B 5AFE7946
I 0000001B 5AFE7949
I 0000001B 5AFE794C
I 0000001B 5AFE794F
I 0000001B 5^C
>>>
```

3.9.13 SEARCH

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

Format:

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

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

A match occurs if (pattern and not mask) = (data and not mask), where:

pattern is the target data

mask is the optional don't care bitmask (which defaults to 0)

data is the data at the current address

SEARCH reports the address under the following conditions:

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

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

Qualifiers:

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

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

Command specific:

/NOT Inverts the sense of the match.

Arguments:

{start_address} A longword address that specifies the first location subject to the search. This address can be an actual address or a symbolic address. If no address is specified, + is assumed.

{pattern} The target data.

[{mask}] A mask of the bits desired in the comparison.

Examples:

```
>>> dep /p/l/n:1000 0 0          ! Clear some memory.
>>>
>>> dep 300 12345678             ! Deposit some search data.
>>> dep 401 12345678
>>> dep 502 87654321
>>>
>>> search /n:1000 /st:1 0 12345678 ! Search for all occurrences
    P 00000300 12345678           ! of 12345678 on any byte
    P 00000401 12345678           ! boundary. Then try on
>>> search /n:1000 0 12345678      ! longword boundaries.
    P 00000300 12345678           ! Search for all non-zero
>>> search /n:1000 /not 0 0        ! longwords.
    P 00000300 12345678
    P 00000400 34567800
    P 00000404 00000012
    P 00000500 43210000
    P 00000504 00008765
>>> search /n:1000 /st:1 0 1 FFFFFFFE ! Search for odd-numbered
                                     ! longwords on any boundary.
    P 00000502 87654321
    P 00000503 00876543
    P 00000504 00008765
    P 00000505 00000087
>>> search /n:1000 /b 0 12         ! Search for all occurrences
    P 00000303 12                 ! of the byte 12.
    P 00000404 12
>>> search /n:1000 /st:1 /w 0 FE11 ! Search for all words that
>>>                                ! could be interpreted as
>>>                                ! a spin (10$: brb 10$).
>>>                                ! Note that none were found.
```


3.9.14 SET

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

Format:

SET {parameter} {value}

Parameters:

BFLAG Set the default R5 boot flags. The value must be a hex number of up to eight digits. See Table 3-3 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. The KA655 DUP driver supports only send data immediate messages and those devices that support the messages. Note the hierarchy of the SET HOST qualifiers below.

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

/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 the KFQSA EEPROM configuration values. Does not accept a task value.

/UQSSP—

/SERVICE n—Specifies service for KFQSA controller module n where n is a value from 0 to 3. (The resulting fixed address of a KFQSA controller module in maintenance mode is 20001910+4*n.)

/csr_address—Specifies the Q22-bus I/O page CSR address for the KFQSA controller module.

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

Qualifiers: Listed in the parameter descriptions above.

Examples:

```
>>>
>>> set bflag 220
>>>
>>> set boot dua0
>>>
>>> show qbus
Scan of Qbus I/O Sapce
-200000DC (760334)=0000 (300) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000DE (760336)=0AA0
-200000E0 (760340)=0000 (304) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000E2 (760342)=0AA0
-200000E4 (760344)=0000 (310) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000E6 (760346)=0AA0
-200000E8 (760350)=0000 (314) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000EA (760352)=0AA0
-20001468 (772150)=0000 (154) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-2000146A (772152)=0AA0
-20001920 (774440)=FF08 (120) DELQA/DEQNA/DESQA
-20001922 (774442)=FF00
-20001924 (774444)=FF2B
-20001926 (774446)=FF09
-20001928 (774450)=FFA3
-2000192A (774452)=FF96
-2000192C (774454)=0050
-2000192E (774456)=1030
-20001940 (774500)=0000 (260) TQK50/TQK70/TU81E/RV20/KFQSA-TAPE
-20001942 (774502)=0BC0
-20001F40 (777500)=(004) IPCR

>>> set host/maint/uqssp 20001468
UQSSP Controller (772150)

Enter SET, CLEAR, SHOW, HELP, EXIT, or QUIT
Node   CSR Address   Model
0       772150        21
1       760334        21
4       760340        21
5       760344        21
7       ----- KFQSA -----
```

? help

Commands:

SET <NODE> /KFQSA

! Set KFQSA DSSI node
! number.

SET <NODE> <CSR_ADDRESS> <MODEL>

! Enable a DSSI device.

CLEAR <NODE>

! Disable a DSSI device.

SHOW

! Show current
! configuration.

HELP

! Print text.

EXIT

! Program the KFQSA.

QUIT

! Do not program the
! KFQSA.

Parameters:

<NODE>

! 0 to 7

<CSR_ADDRESS>

! 760010 to 777774

<MODEL>

! 21 (disk) or 22 (tape)

? set 6 /KFQSA

? show

Node	CSR Address	Model
0	772150	21
1	760334	21
4	760340	21
5	760344	21
6	----- KFQSA -----	

? exit

Programming the KFQSA...

>>>

>>>set language 5

>>>

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 in the system. |
| ETHERNET | Displays hardware Ethernet address for all Ethernet adapters that can be found. Displays as blank if no Ethernet adapter is present. |
| LANGUAGE | Displays console language and keyboard type. Refer to the corresponding SET LANGUAGE command for the meaning. |
| MEMORY | <p>Displays main memory configuration board by board.</p> <p>/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.</p> |
| QBUS | <p>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.</p> <p>This command may take several minutes to complete. Press CTRL/C to terminate the command. During execution, the command disables the scatter-gather map.</p> |
| RLV12 | Displays all RL01 and RL02 disks that appear on the Q22-bus. |
| UQSSP | <p>Displays the status of all disks and tapes that can be found on the Q22-bus that support the UQSSP protocol. For each such disk or tape on the Q22-bus, the firmware displays the controller number, the controller CSR address, and the boot name and type of each device connected to the controller. The command does not indicate whether the device contains a bootable image.</p> <p>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.</p> |
| VERSION | Displays the current firmware version. |

Qualifiers: Listed in the parameter descriptions above.

Examples:

```
>>>
>>> show bflag
00000220
>>>
>>> show boot
DUA0
>>>
>>> show device
UQSSP Disk Controller 0 (772150)
-DUA0 (RF71)

UQSSP Disk Controller 1 (760334)
-DUB1 (RF71)

UQSSP Disk Controller 2 (760340)
-DUC4 (RF71)

UQSSP Tape Controller 0 (774440)
-MUA0 (TK70)

Ethernet Adapter
-XQA0 (08-00-2B-0B-82-29)
>>>
>>> show ethernet
Ethernet Adapter
-XQA0 (08-00-2B-0B-82-29)
>>>
>>> show language
English (United States/Canada)
>>>
>>> show memory
Memory 0: 00000000 to 00FFFFFF, 16MB, 0 bad pages

Total of 16MB, 0 bad pages, 104 reserved pages
>>>
>>> show memory/full
Memory 0: 00000000 to 00FFFFFF, 16MB, 0 bad pages

Total of 16MB, 0 bad pages, 104 reserved pages

Memory Bitmap
-00FF3C00 to 00FF3FFF, 8 pages

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

Q-bus Map
-00FF8000 to 00FFFFFF, 64 pages

Scan of Bad Pages
>>>
```

```

>>> show Q-bus
Scan of Q-bus I/O Space
-200000DC (760334) = 0000 (300) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000DE (760336) = 0AA0
-200000E0 (760340) = 0000 (304) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000E2 (760342) = 0AA0
-20001468 (772150) = 0000 (154) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-2000146A (772152) = 0AA0
-20001920 (774440) = FF08 (120) DELQA/DEQNA/DESQA
-20001922 (774442) = FF00
-20001924 (774444) = FF2B
-20001926 (774446) = FF09
-20001928 (774450) = FF00
-2000192A (774452) = FFE1
-2000192C (774454) = 8400
-2000192E (774456) = 1030
-20001940 (774500) = 0000 (260) TQK50/TQK70/TU81E/RV20/KFQSA-TAPE
-20001942 (774502) = 0BC0
-20001F40 (777500) = 0020 (004) IPCR

Scan of Qbus Memory Space
>>>
>>> show RLV12
>>>
>>> show UQSSP
UQSSP Disk Controller 0 (772150)
-DUA0 (RF71)

UQSSP Disk Controller 1 (760334)
-DUB1 (RF71)

UQSSP Disk Controller 2 (760340)
-DUC4 (RF71)

UQSSP Tape Controller 0 (774500)
-MUA9 (TK70)
>>>show version
KA655-A V5.3, VMB 2.7
>>>

```

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
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 KA655 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. You can select nonstandard addresses, 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. This step saves you time and prevents you from introducing new problems.

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

In addition, check the following connections:

- If no message appears, make sure the console terminal and the system are powered 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 lit.
- 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 H3600-SA. If the display is off, check the CPU module LEDs and the CPU cabling. If a hex error message appears on the H3600-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 KA655 ROM-Based Diagnostics

The KA655 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 four 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 H3600–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 KA655 is in a manufacturing environment. The diagnostic executive interprets the script to determine what tests to run, the correct order to run the tests, and the correct parameters to use for each test.

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

4.3.1 Diagnostic Tests

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:

NOTE: *Base addresses change. The addresses in Table 4–1 are for V5.3; for other versions, use as examples only.*

- Test is the test code or utility code.
- Address is an example of the test or utility's base address in ROM. If a test fails, entering T FE displays diagnostic state to the console. You can subtract the base address of the failing test from the `last_exception_pc` to find the index into the failing test's diagnostic listing (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 ¹	Name	Parameters
	2004BC00	De_SCB	
C6	2004D2F0	SSC_powerup	*****
C7	2004D3B2	CBTCR timeout	***
34	2004D46C	ROM logic test	*
33	2004D534	CMCTL_powerup	*
32	2004D57C	CMCTL regs	MEMCSR0_addr *****
91	2004D6A0	CQBIC_powerup	**
90	2004D730	CQBIC regs	*
80	2004D7B2	CQBIC-memory	*****
60	2004DDD9	Console serial	start_baud end_baud *****
62	2004E128	Console QDSS	mark_not_present selftest_r0 selftest_r1 *****
63	2004E2A8	QDSS self-test	input_csr selftest_r0 selftest_r1 *****
51	2004E40E	CFPA	*****
52	2004E5FA	Prog timer	which_timer wait_time_us ***
53	2004E8C0	TOY clock	repeat_count_250ms_ea ****
55	2004EA6B	Interval timer	*
5A	2004EAE0	VAX CMCTL CDAL	dont_report_memory_bad repeat_count *
45	2004EBF0	cache_mem_cqbic	start_addr end_addr addr_incr ****
46	2004EED8	Cache1_diag_md	addr_incr *****
9E	2004F502	List diags	*
81	2004F528	MSCP-QBUS test	IP_csr *****
82	2004F6EA	DELQA	device_num_addr ****
C1	2004F8C5	SSC RAM	*
C2	2004FA8C	SSC RAM ALL	*
C5	2004FBF8	SSC regs	*
54	2004FCE9	Virtual mode	****
36	2004FF68	Cache2_memory	start_addr end_addr addr_incr *****
35	200504DC	Cache2 integrty	start_addr end_addr addr_incr *****
44	20050CF4	Cache_memory	addr_incr *****
43	20050D4D	Cache1_cache2	addr_incr *****
41	2005107C	Board reset	***
42	20051269	Check_for_intrs	***
31	200512AC	MEM_setup_CSRs	*****

¹V5.3 addresses; use as examples only.

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

Test	Address ¹	Name	Parameters
30	200518CF	MEM_bitmap	*** mark_Hard_SBEs *****
4F	20051D0A	MEM_data	start_add end_add add_incr cont_on_err *****
4E	20051EC8	MEM_byte	start_add end_add add_incr cont_on_err *****
4D	20051FDD	MEM_address	start_add end_add add_incr cont_on_err *****
4C	2005216F	MEM_ECC_error	start_add end_add add_incr cont_on_err *****
4B	20052630	MEM_maskd_errs	start_add end_add add_incr cont_on_err *****
4A	2005280A	MEM_correction	start_add end_add add_incr cont_on_err *****
49	20052A1D	MEM_FDM_logic	*** cont_on_err *****
48	20052FEC	MEM_addr_shrts	start_add end_add * cont_on_err pat1 pat2 *****
47	20053643	MEM_refresh	start end incr cont_on_err time_seconds *****
40	200537CE	MEM_count_errs	First_board Last_board ***** Soft_errs_ allowed
9D	20053B14	Utilities	Expnd_err_msg get_mode init_LEDs clr_ps_ cnt
9C	20053C18	List CPU regs	*
9F	200541D4	Create script	*****

¹V5.3 addresses; use as examples only.

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

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

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

```
>>> T 54
```

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

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

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

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

4.3.2 Scripts

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

- The parameters and flags that need to be passed to the test.
- Where the tests can be run from. For example, certain tests can be run only from the EPROM. Other tests are program independent code, and can be run from EPROM, 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 KA655 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. Also referred to as user script. Enter T 9F to create.
A1	A1, AA, AB, AC, 0, 3	Common section of power-up script. Script AC invokes this script at power-up. This script does not directly invoke any tests, but calls script BD to run the tests.
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. Also called by the INIT command.
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.
BD	BD, A1, AA, AB, AC, 0, 3	Common section of power-up script. Script A1 invokes this script at power-up.

¹Scripts 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. Scripts BE, BF, B4, and B6–B9 are not used.

In most cases, Field Service needs only the scripts 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 environment is nonmanufacturing from H3600–SA. (Manufacturing sets a jumper on the H3600–SA for testing.)
 - b. DE selects script sequence for console SLU.
 - c. DE executes Script BA.
 - Script BA directs DE to invoke script BD. Script BD directs DE to execute tests (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.
 - H3600-SA switch is 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 invoke script BD. Script BD directs DE to execute tests (console announcements are on).
 - b. DE passes control back to CP.
 8. 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 H3600-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 invoke script BD, which then directs DE to execute tests (console announcements are on).
 - c. DE passes control back to the CP.
2. Issues 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 BD) run both times.

4.3.4 Creating Scripts

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

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

Run test 9F first to build the user script (see Example 4-1). Press <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, diagnostic RAM (DIAG_RAM), or in main memory. A script is limited by the size of the data structure that contains it. A larger script can be developed in main memory than in DIAG_RAM, and a larger script can be built in DIAG_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 = DIAG_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=DIAG_RAM,2=RAM] :1
Script starts at 2011FC00
1024 bytes left
Next test number :51
CFPA >>Run from ?[0=ROM,1=DIAG_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=DIAG_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.

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

Example 4-2: Listing and Repeating Tests with Utility 9F

```
>>> T 9F
SP=20140604
Create script in ?[0=SSC,1=DIAG_RAM,2=RAM] :0
Script starts at 20140758
Script starts at 20140758
24 bytes left
24 bytes left
Next test number :? ! Displays available tests and utility
Test
#   Address   Name           Parameters
-----
      2004BC00 De_SCB
C6  2004D2F0  SSC_powerup    *****
C7  2004D3B2  CBTCR timeout  ***
34  2004D46C  ROM logic test *
33  2004D534  CMCTL_powerup  *
32  2004D57C  CMCTL regs     MEMCSR0_addr *****
91  2004D6A0  CQBIC_powerup  **
90  2004D730  CQBIC regs     *
80  2004D7B2  CQBIC_memory   *****
60  2004DDD9  Console serial start_baud end_baud *****
62  2004E128  console QDSS   mark_not_present selftest_r0 selftest_r1 *****
63  2004E2A8  QDSS self-test input_csr selftest_r0 selftest_r1 *****
51  2004E40E  CFPA           *****
52  2004E5FA  Prog timer     which_timer wait_time_us ***
53  2004E8C0  TOY clock      repeat_test_250ms_ea Tolerance ***
55  2004EA6B  Interval timer *
5A  2004EAE0  VAX CMCTL CDAL dont_report_memory bad repeat_count *
45  2004EBF0  cache_mem_cqbic start_addr end_addr addr_incr ****
46  2004EED8  Cachel_diag_md addr_incr *****
9E  2004F502  List diags     *
81  2004F528  MSCP-QBUS test IP_csr *****
82  2004F6EA  DELQA         device_num_addr ****
C1  2004F8C5  SSC RAM        *
C2  2004FA8C  SSC RAM ALL    *
C5  2004FBF8  SSC regs       *
54  2004FCE9  Virtual mode   *****
36  2004FF68  cache2_memory  start_addr end_addr addr_incr *****
35  200504DC  Cach2_integrty start_addr end_addr addr_incr *****
44  20050CF4  Cache_memory   addr_incr *****
43  20050D4D  Cachel_Cache2  addr_incr *****
41  2005107C  Board Reset    ***
42  20051269  Check_for_intrs ***
31  200512AC  MEM_Setup_CSRs *****
30  200518CF  MEM_Bitmap     *** mark_Hard_SBEs *****
```

Example 4-2 Cont'd. on next page

Example 4-2 (Cont.): Listing and Repeating Tests with Utility 9F

```
4F 20051D0A MEM_Data      start_add end_add add_incr cont_on_err *****
4E 20051EC8 MEM_Byte      start_add end_add add_incr cont_on_err *****
4D 20051FDD MEM_Address   start_add end_add add_incr cont_on_err *****
4C 2005216F MEM_ECC_Error  start_add end_add add_incr cont_on_err *****
4B 20052630 MEM_Maskd_Errs start_add end_add add_incr cont_on_err *****
4A 2005280A MEM_Correction start_add end_add add_incr cont_on_err *****
49 20052A1D MEM_FDM_Logic *** cont_on_err *****
48 20052FEC MEM_Addr_shrts start_add end_add * cont_on_err pat2 pat3 ****
47 20053643 MEM_Refresh  start end incr cont_on_err time_seconds *****
40 200537CE MEM_Count_Errs First board last_board Soft_errs_allowed *****
9D 20053B14 Utilities   Expnd_err_msg get_mode init_LEDs clr_ps_cnt
9C 20053C18 List CPU regs *
9F 200541D4 Create script *****
24 bytes left
Next test number: 40
MEM_Count_Errs>>Run from ?[0=ROM,1=DIAG_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..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)

```
KA655-A V5.3 VMB 2.7
Performing Normal System Tests
40..39..38..37..36..35..34..33..32..31..30..29..28..27..26..25..
24..23..22..21..20..19..18..17..16..15..14..13..12..11..10..09..
08..07..06..05..04..03..
Tests completed
>>>
```

The first line contains the firmware revision (V5.3 in this example) and the virtual memory bootstrap (VMB) revision (V2.7 in this example).

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=AAAAAAA
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 on Error column refers to the action taken by the diagnostic executive under the following circumstances:

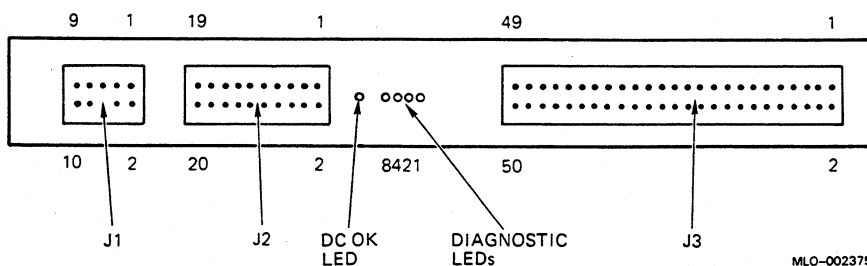
- The diagnostic executive detects an unexpected exception or interrupt.
- A test fails and that failure is reported to the diagnostic executive.

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 KA655 CPU. They correspond to the hex display on the H3600-SA.

Figure 4-1: KA655 CPU Module LEDs



MLO-002375

Table 4–6: KA655 Console Displays and FRUs

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU ¹
Initial Power-Up Tests					
F	None	None	Loop on self	Power-up	6, 1, 4, 5 ^{2, 3}
D	None	None	Loop on self	WAIT_POK	1
4	None	None	Loop on self	Entering IPT	1
6	None	None	Loop on self	SLU_EXT_LOOPBACK ⁴	7, 1

Script AA

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

Script BA

C	None	?9D	Continue	Utilities	1
B	None	?42	Continue	Check_for_intrs	1
C	None	?C6	Continue	SSC_powerup	1
6	None	?60	Continue	CONSOLE_SERIAL	1

End of script

Script AC

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

¹FRU key:

- 1 = KA655
- 2 = MS650–BA
- 3 = Memory interconnect cable
- 4 = Q22-bus device
- 5 = Q22/CD backplane
- 6 = System power supply
- 7 = H3600–SA I/O panel

²In the case of multiple FRUs, refer to Section 4.5.2 for further information.

³If a problem reccurs with the same FRU, check that the tolerance for system power supply +5 Vdc, +12 Vdc, and AC ripple are within specification.

⁴This test only runs if the power-up mode switch on the H3600–SA is set to TEST mode. See Section 4.6.1.

Table 4–6 (Cont.): KA655 Console Displays and FRUs

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU ¹
Script BC					
7	40	?91	Continue	CQBIC_power-up	1
7	39	?90	Continue	CQBIC_registers	1
9	38	?33	Continue	CMCTL_power-up	1
9	37	?32	Continue	CMCTL_registers	1
8	36	?31	Continue	CMCTL_setup_CSRs	1, 2, 3, 5
8	35	?49	Continue	MEMORY_FDM_logic	2, 1, 3, 5
8	34	?30	Halt	MEMORY_bitmap	1, 2, 3, 5
End of script.					
Script A1					
Invoke script BD.					
Script BD					
C	33	?52	Continue	PROG_TIMER_0	1
C	32	?52	Continue	PROG_TIMER_1	1
C	31	?53	Continue	TOY_CLOCK	1
C	30	?C1	Continue	SSC_RAM	1
C	29	?34	Continue	ROM_logic	1
C	28	?C5	Continue	SSC_registers	1
B	27	?55	Continue	INTERVAL_TIMER	1
A	26	?51	Continue	CFPA	1
C	25	?C7	Continue	CBTCR_timeout	1
B	24	?46	Continue	CACHE_DIAG_MODE	1
5	23	?35	Continue	Cache2_integrity	1
C	22	?C2	Continue	SSC_RAM_ALL	1
B	21	?43	Continue	Cache1_cache2	1

¹FRU key:

- 1 = KA655
- 2 = MS650–BA
- 3 = Memory interconnect cable
- 4 = Q22-bus device
- 5 = Q22/CD backplane
- 6 = System power supply
- 7 = H3600–SA I/O panel

Table 4–6 (Cont.): KA655 Console Displays and FRUs

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU ¹
Script BD					
8	20	?4F	Continue	MEMORY_data	2, 1, 3, 5
8	19	?4E	Continue	MEMORY_byte	2, 1, 3, 5
8	18	?4D	Continue	MEMORY_addr	2, 1, 3, 5
8	17	?4C	Continue	MEMORY_ECC_error	2, 1, 3, 5
8	16	?4B	Continue	MEMORY_masked_errors	2, 1, 3, 5
8	15	?4A	Continue	MEMORY_correction	2, 1, 3, 5
8	14	?48	Continue	MEMORY_address_shorts	2, 1, 3, 5
8	13	?47	Continue	MEMORY_refresh	2, 1, 3, 5
8	12	?40	Continue	MEMORY_count_errors	2, 1, 3, 5
B	11	?44	Continue	CACHE1_MEMORY	1, 2, 3, 5
5	10	?36	Continue	Cache2_memory	1, 2, 3, 5
7	09	?80	Continue	CQBIC_MEMORY	1, 2, 4, 3, 5
B	08	?54	Continue	VIRTUAL_MODE	1, 2, 3, 5
B	07	?C5	Continue	SSC_registers	1
B	06	?34	Continue	ROM logic test	1
7	05	?45	Continue	CACHE_MEM_CQBIC	1, 2, 4, 3, 5
9	04	?5A	Continue	CVAX CMCTL CDAL	1
C	03	?41	Continue	Board reset	1, 4
End of script.					
Script A8					
8	31	?31	Halt	CMCTL_setup_CSRs	1, 2, 3, 5
8	49	?49	Halt	MEMORY_FDM_logic	2, 1, 3, 5
8	30	?30	Halt	MEMORY_bitmap	1, 2, 3, 5
Invoke script A7.					
End of script.					
¹ FRU key:					
1 = KA655					
2 = MS650–BA					
3 = Memory interconnect cable					
4 = Q22-bus device					
5 = Q22/CD backplane					
6 = System power supply					
7 = H3600–SA I/O panel					

Table 4–6 (Cont.): KA655 Console Displays and FRUs

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU ¹
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_errors	2, 1, 3, 5
7	80	?80	Cont	CQBIC_memory	1, 2, 4, 3, 5
C	41	?41	Halt	Board reset	1, 4
End of script.					
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	Halt	MEMORY_address_shorts	2, 1, 3, 5
8	47	?47	Halt	MEMORY_refresh	2, 1, 3, 5
8	40	?40	Continue	MEMORY_count_bad pages	2, 1, 3, 5
C	41	?41	Continue	KA655_RESET	1, 4
End of script.					

¹FRU key:

- 1 = KA655
- 2 = MS650–BA
- 3 = Memory interconnect cable
- 4 = Q22-bus device
- 5 = Q22/CD backplane
- 6 = System power supply
- 7 = H3600–SA I/O panel

Table 4-6 (Cont.): KA655 Console Displays and FRUs

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU ¹
Script AD					
8	None	?30	Continue	MEM_bitmap	1, 2, 3, 5
8	None	?4F	Continue	MEM_data	2, 1, 3, 5
8	None	?4E	Continue	MEM_byte	2, 1, 3, 5
8	None	?4D	Continue	MEMORY_addr	2, 1, 3, 5
8	None	?4C	Continue	MEMORY_ECC_error	2, 1, 3, 5
8	None	?4B	Continue	MEMORY_masked_errors	2, 1, 3, 5
8	None	?4A	Continue	MEMORY_correction	2, 1, 3, 5
8	None	?48	Continue	MEMORY_address_shorts	2, 1, 3, 5
8	None	?40	Continue	MEMORY_count_errors	2, 1, 3, 5
7	None	?80	Continue	CQBIC_MEMORY	1, 2, 4, 3, 5
C	None	?41	Continue	Board reset	1, 4
End of script.					
Script AE					
8	None	?31	Continue	CMCTL_setup_CSRx	1, 2, 3, 5
C	41	?41	Continue	Board reset	1, 4
End of script.					
Script AF					
7	None	?80	Continue	CQBIC_MEMORY	1, 2, 4, 3, 5
C	None	?41	Continue	Board reset	1, 4
End of script.					
¹ FRU key:					
1 = KA655					
2 = MS650-BA					
3 = Memory interconnect cable					
4 = Q22-bus device					
5 = Q22/CD backplane					
6 = System power supply					
7 = H3600-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 EXC7	PSL <26:24> = 7 on interrupt or exception.
?1D	PSL RE15	PSL <26:24> = 5 on an rei instruction.
?1E	PSL RE16	PSL <26:24> = 6 on an rei instruction.
?1F	PSL RE17	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

4.4 Acceptance Testing

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

- KA655 module
- MS650–BA module
- Memory data interconnect cable
- Backplane
- DSSI device
- H3600–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 thorough 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. Double-check the memory configuration, since test 31 can check for only a few invalid configurations. For example, test 31 cannot report that a memory board is missing from the configuration, since it has no way of knowing if the board should be there or not.

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

```
>>>SHOW MEMORY/FULL
```

```
Memory 0: 00000000 to 00FFFFFF, 16MB, 0 bad pages
```

```
Total of 16MB, 0 bad pages, 104 reserved pages
```

```
Memory Bitmap
```

```
-00FF3000 to 00FF3FFF, 8 pages
```

```
Console Scratch Area
```

```
-00FF4000 to 00FF7FFF, 32 pages
```

```
Qbus Map
```

```
-00FF8000 to 00FFFFFF, 64 pages
```

```
Scan of Bad Pages
```

```
>>>
```

Memory 0 is the KA655 CPU. Memories 1 through 4 are the MS650-BA 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  =00157FA8  ICCS =00000000
TCR0 =00000000  TIRO =000207E3  TNIR0=00000000  TIVR0=00000078
TCR1 =00000000  TIR1 =00000000  TNIR1=00000000  TIVR1=0000007C
RXCS =00000000  RXDB =0000000D  TXCS =00000000  TXDB =00000030
MSER =00000000  CADR =0000000C  CACR =F5B40040
BDR  =FFFFFFF5  DLED R=0000000C  SSCCR=00D45577  CBTCR=00000004
SCR  =0000C000  DSER =00000000  QBEAR=0000000A  DEAR =00000000
QBMR =00FF8000  IPCRn=0020
```

```
MEM_FRU 1 MEMCSR_0=80000017  1=80400017  2=80800017  3=80C00017
MEM_FRU 2 MEMCSR_4=00000000  5=00000000  6=00000000  7=00000000
MEM_FRU 3 MEMCSR_8=00000000  9=00000000  10=00000000  11=00000000
MEM_FRU 4 MEMCSR12=00000000  13=00000000  14=00000000  15=00000000
MEMCSR16=00000044  17=0000003C
```

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

```
MEMCSR 0-3      First MS650-BA memory module
MEMCSR 4-7      Second MS650-BA memory module
MEMCSR 8-11     Third MS650-BA memory module
```


Verify the following:

- The bank enable bit (<31>) in all four MEMCSRs for each memory module is set, which indicates that the memory bank is enabled. MEMCSRs <6:0> should equal 17 hex.
 - 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 4-15 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 0 and 1 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 KA655 CQBIC chip, and the configuration of the Q22-bus, as follows:

```
>>> show qbus
Scan of Qbus I/O Space
-200000DC (760334)=0000 (300) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000DE (760336)=0AA0
-20001468 (772150)=0000 (154) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-2000146A (772152)=0AA0
-20001920 (774440)=FF08 (120) DELQA/DEQNA/DESQA
-20001922 (774442)=FF00
-20001924 (774444)=FF2B
-20001926 (774446)=FF09
-20001928 (774450)=FFA3
-2000192A (774452)=FF96
-2000192C (774454)=0050
-2000192E (774456)=1030
-20001940 (774500)=0000 (260) TQK50/TQK70/TU81E/RV20/KFQSA-TAPE
-20001942 (774502)=0BC0
-20001F40 (777500)=0020 (004) IPCR

Scan of Qbus Memory Space
>>>
```

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 KA655 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 KA655, 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
```

```
UQSSP Disk Controller 0 (772150)
-DUA0 (RF71)
```

```
UQSSP Disk Controller 1 (760334)
-DUB1 (RF71)
```

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

```
Ethernet Adapter 0 (774440)
-XQA0 (08-00-2B-0B-82-29)
```

In the example, the console displays the node numbers of two RF71 controllers it recognizes. The line below each node name and number is the logical unit number of any attached devices, DUA0 and DUB1 in this case.

The UQSSP (TQK70) tape controller and its CSR address are also shown. The line below this display shows a TK70 connected.

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

6. 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. You may also be able to acquire more information with the FE utility.

Running the FE utility is useful if the message, Normal operation not possible, is displayed after the tests have 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 KA655 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 ckck 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-BA 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 KA655 console takes effect only after the entire script completes.) After terminating the script, enter SHOW MEMORY/FULL to see if the tests have marked any pages bad up to that point. See Section 4.4 for an example of this command.

2. T A9

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

Script A9 runs only the memory tests and halts on the first error detected. Unlike the power-up script, it does not continue. Since the script does not rerun the test, it detects memory-related failures that are not hard errors. You should then run the individual test that failed on each memory module one MS650-BA 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-BA 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 four memory modules. Use it after running memory tests individually or within a script. If test 40 fails with subtestlog = 6, examine R5–R8 to determine how many errors have been detected on the CPU memory and the three memory modules, respectively.

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

2..

? 7 SCB error 3 PC = 424

>>> T 9C

```
TOY  =00157FA8  ICCS =00000000
TCR0 =00000000  TIR0 =000207E3  TNIR0=00000000  TIVR0=00000078
TCR1 =00000000  TIR1 =00000000  TNIR1=00000000  TIVR1=0000007C
RXCS =00000000  RXDB =0000000D  TXCS =00000000  TXDB =00000030
MSER =00000000  CADR =0000000C  CACR =F5B40040
BDR  =FFFFFFF50  DLED R=0000000C  SSSCCR=00D45577  CBTCCR=00000004
SCR  =0000C000  DSER =00000000  QBEAR=0000000A  DEAR =00000000
QBMBR=00FF8000  IPCRn=0020
```

```
MEM_FRU 1 MEMCSR_0=80000017  1=80400017  2=80800017  3=80C00017
MEM_FRU 2 MEMCSR_4=81000017  5=81400017  6=81800017  7=81C00017
MEM_FRU 3 MEMCSR_8=00000000  9=00000000  10=00000000  11=00000000
MEM_FRU 4 MEMCSR12=00000000  13=00000000  14=00000000  15=00000000
MEMCSR16=8154000F  17=0000003C
```

>>>

In Example 4-6, 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, which indicates 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 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 = 8154000F Hex = 1000 0001 0101 0100 0000 0000 0000 1111
                        ||  ||
MEMCSR_5 = 81400017 Hex = 1000 0001 0100 0000 0000 0000 0001 0111
                        ^      ^
                    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 KA655 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 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-bit and multi-bit ECC failures as shown in step 2 of acceptance testing (Section 4.4), 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 KA655 ROM-based diagnostics to see if it is an MS650 problem, or if it is related to the KA655, 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.

4.6 Loopback Tests

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

Check that the dc power and the pico fuse on the KA655 are functioning correctly. The 1.5 ampere pico fuse (12-10929-08) is located near the handle on the component side and is numbered F1. The fuse is shown in Figure 1-1. If the fuse is bad, the H3600-SA hex LED display will not light.

4.6.1 Testing the Console Port

To test the console port at power-up, set the power mode switch on the H3600-SA to the test position, and install an H3103 loopback connector into the MMP of the H3600. 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 panel should alternate between 6 and 3. A value of 6 latched in the display indicates a test failure. If the test fails, one of the following parts is faulty: the KA655, the H3600-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 H3600-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 4–11 lists loopback connectors for common KA655 system modules. Refer to *Microsystems Options* for a description of specific module self-tests.

Table 4–11: 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
KA655/H3600–SA	H3103	H3103 + H8572
KMV1A	H3255	H3251

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

²For DSSI to KFQSA or RF-series connector, use 17–02216–01 plus H3281 loopback.
For connection to end of bus, use DSSI loopback.

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

4.8 RF-Series ISE Troubleshooting and Diagnostics

An RF-series ISE may fail either during initial power-up or during normal operation. In both cases the failure is indicated by the lighting of the red fault LED on the enclosure's OCP. The ISE also has a red fault LED, but it is not visible from the outside of the system enclosure.

If the drive is unable to execute the Power-On Self Test (POST) successfully, the red fault LED remains lit and the ready LED does not come on, or both LEDs remain on.

POST is also used to handle two types of error conditions in the drive:

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

Three configuration errors also commonly occur:

- More than one node with the same node number
- Identical node names
- Identical unit numbers

The first error cannot be detected by software. Use the SHOW DSSI command to display the second and third errors. This command lists each device connected to the DSSI bus by node name and unit number.

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

The RF-series ISE contains the following local programs (described in the following sections):

DIRECT	A directory, in DUP specified format, of available local programs
DRVTST	A comprehensive drive functionality verification test
DRVEXR	A utility that exercises the ISE
HISTORY	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 dialogue of each program. The table also indicates the type of messages contained in the dialogue, although the screen display does not indicate the message type. Message types are abbreviated as follows:

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

To access these local programs, use the console SET HOST/DUP command, which creates a virtual terminal connection to the storage device and the designated local program using the Diagnostic and Utilities Protocol (DUP) standard dialogue.

Once the connection is established, the local program is in control. When the program terminates, control is returned to the KA655 console.

To abort or prematurely terminate a program and return control to the KA655 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 4-12.

Table 4-12: DRVTST Messages

Message Type	Message
I	Copyright © 1988 Digital Equipment Corporation
Q	Write/read anywhere on the medium? [1=yes/(0=no)]
Q	User data will be corrupted. Proceed? [1=yes/(0=no)]
I	5 minutes to complete.
T	Test passed.
Or:	
FE	Unit is currently in use. ¹
FE	Operation aborted by user.
FE	xxxx—Unit diagnostics failed. ²
FE	xxxx—Unit read/write test failed. ²

¹Either the drive is inoperative, in use by a host, or is currently running another local program.

²Refer to the diagnostic error code list at the end of this chapter.

Answering No to the first question (“Write/read...?”) results in a read-only test. DRVTST however, writes to a diagnostic area on the disk. Answering Yes to the first question causes the the second question to be displayed.

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

DRVTST resets the ECC error counters, then calls the timed I/O routine. After the timed I/O routine ends (5 minutes), DRVTST saves the counters again. It computes the uncorrectable error rate and byte (symbol) error rate. If either rate is too high, the test fails and the appropriate error code is displayed.

4.8.2 DRVEXR

The DRVEXR local program exercises the ISE. The test is data transfer intensive, and indicates the overall integrity of the device. Table 4-13 lists the DRVEXR messages.

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)]
Q	Test time in minutes? [(10)-100]
I	ddd minutes to complete.
I	ddddddddd blocks (512 bytes) transferred.
I	ddddddddd bytes in error (soft).
I	ddddddddd uncorrectable ECC errors (recoverable).
T	Complete.
Or:	
FE	Unit is currently in use. ¹
FE	Operation aborted by user.
FE	xxxx—Unit diagnostics failed. ²
FE	xxxx—Unit read/write test failed. ²

¹Either the drive is inoperative, in use by a host, or is currently running another local program.

²Refer to the diagnostic error code list at the end of this chapter.

Answering No to the first question (“Write/read...?”) results in a read-only test. DRVEXR however, writes to a diagnostic area on the disk. Answering Yes to the first question causes the second question to be displayed.

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

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

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

DRVEXR uses the same timed I/O routine as DRVTST, with two exceptions. First, DRVTST always uses a fixed time of five minutes, while you specify the time of the DRVEXR routine. Second, DRVTST determines whether the drive is good or bad. DRVEXR reports the data but does not determine the condition of the drive.

4.8.3 HISTRY

The HISTRY local program displays information about the history of the ISE. Table 4–14 lists the HISTRY messages.

Table 4–14: 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.

¹This displays the last 11 fault codes as information messages.
Refer to the diagnostic error code list at the end of this chapter.

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

```
Copyright © 1988 Digital Equipment Corporation
RF71
EN01082
SUSAN
0
RFX V101
RF71 PCB-5/ECO-00
617
21
A04F
A04F
A103
A04F
A404
A04F
A404
A04F
A404
A04F
A404
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 4–15 lists the ERASE messages:

Table 4–15: ERASE Messages

Message Type	Message
I	Copyright © 1989 Digital Equipment Corporation
Q	Write/read anywhere on the medium? [1=yes/(0=no)]
Q	User data will be corrupted. Proceed? [1=yes/(0=no)]
I	6 minutes to complete.
T	Complete.
Or:	
FE	Unit is currently in use.
FE	Operation aborted by user.
FE	xxxx—Unit diagnostics failed. ¹
FE	xxxx—Operation failed. ²

¹Refer to the diagnostic error code list at the end of this chapter.

²xxxx = one of the following error codes:

000D : Cannot write the RCT.

000E : Cannot read the RCT.

000F : Cannot find an RBN to revector to.

0010 : The RAM copy of the bad block table is full.

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

4.8.5 PARAMS

The PARAMS local program supports modifications to device parameters that you may need to change, such as device node name and allocation class. You invoke it in the same way as the other local programs. However, you use the following commands to make the modifications you need:

EXIT	Terminates PARAMS program
HELP	Prints a brief list of commands and their syntax
SET	Sets a parameter to a value
SHOW	Displays a parameter or a class of parameters
STATUS	Displays module configuration, history, or current counters, depending on the status type chosen
WRITE	Alters the device parameters

4.8.5.1 EXIT

Use the EXIT command to terminate the PARAMS local program.

4.8.5.2 HELP

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

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

4.8.5.3 SET

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

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

The following parameters are useful to 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 RF71x (where x is a letter from A through H that corresponds to the DSSI bus ID) instead of using the NODENAME value. False (0) if NODENAME is to be used.

4.8.5.4 SHOW

Use the SHOW command to display the settings of a parameter or a class of parameters. It displays the full name of the parameter (8 characters or less), the current value, the default value, radix and type, and any flags associated with each parameter.

4.8.5.5 STATUS

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

CONFIG	Displays the module name, node name, power-on hours, power cycles, and other such configuration information. Unit failures are also displayed, if applicable.
LOGS	Displays the last eleven machine and bug checks on the module. The display includes the processor registers (D0-D7, A0-A7), the time and date of each failure (if available; otherwise the date 17 November 1858 is displayed), and some of the hardware registers.
DATALINK	Displays the data link counters.
PATHS	Displays available path information (open virtual circuits) from the point of view of the controller. The display includes the remote node names, DSSI IDs, software type and version, and counters for the messages and datagrams sent and/or received.

4.8.5.6 WRITE

Use the WRITE command to write the changes made while in PARAMS to the drive nonvolatile memory. The WRITE command is similar to the VMS SYSGEN WRITE command. Parameters are not available, but you must be aware of the system and/or drive requirements and use the WRITE command accordingly or it may not succeed in writing the changes.

The WRITE command may fail for one of the following reasons:

- You altered a parameter that required the unit, and the unit cannot be acquired (that is, the unit is not available to the host). Changing the unit number is an example of a parameter that requires the unit.
- You altered a parameter that required a controller initialization, and you replied negatively to the request for reboot. Changing the node name or the allocation class are examples of parameters that require controller initialization.
- Initial drive calibrations were in progress on the unit. The use of the WRITE command is inhibited while these calibrations are running.

Appendix A

Configuring the KFQSA

This appendix describes the KFQSA storage adapter and explains how to:

- Configure the KFQSA storage adapter at installation
- Enter console I/O mode
- Run the Configure utility
- Program the EEROM on the KFQSA
- Reprogram the EEROM on the KFQSA
- Change the ISE's allocation class and unit number

A.1 KFQSA Overview

The KFQSA module is a storage adapter that allows Q-bus host systems that support the KFQSA to communicate with storage peripherals based on the Digital Storage Architecture (DSA), using the Digital Storage System Interconnect (DSSI). In a KA655-based system, one KFQSA module can connect up to six RF-series integrated storage elements (ISEs) to the host system using a single DSSI bus cable.

The KFQSA contains the addressing logic required to make a connection between the host and a requested ISE on the DSSI bus. Each ISE has its own controller, which contains the intelligence and logic necessary to control data transfers over the DSSI bus. The KFQSA presents a mass storage control protocol (MSCP) U/Q port for each ISE.

The EEROM on the KFQSA contains a configuration table. After you install the KFQSA, you program the EEROM with the CSR address for each ISE in the system.

A.1.1 Dual-Host Configuration

An RF-series ISE has dual-host capability built into the firmware, which allows the device to maintain connections with more than one KFQSA storage adapter. You can connect more than one KFQSA to the same DSSI bus, which allows each KFQSA to access all other devices on the bus. For more information on dual-host capability, see Chapter 2, Section 2.4.5.

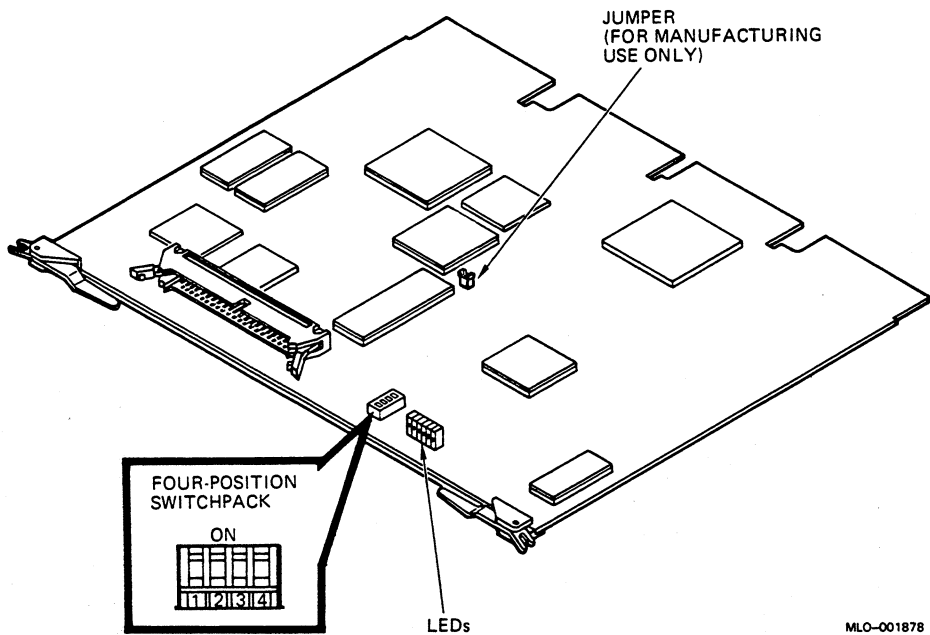
A.2 Configuring the KFQSA at Installation

At installation, configure the KFQSA as follows:

CAUTION: *Static electricity can damage integrated circuits. Use the wrist strap and antistatic mat found in the Antistatic Kit (29-26246) when you work with the internal parts of a computer system.*

1. Check the KFQSA module for the presence of a jumper intended for manufacturing use only. The location of this jumper is shown in Figure A-1. Remove the jumper, if present.
2. Use the four-position DIP switchpack shown in Figure A-1 as follows to set a temporary CSR address that enables you to access the EEROM:
 - a. Set switches 1, 2, 3, and 4 to reflect a fixed CSR address to allow the KFQSA to be programmed. Example A-1 shows the correct switch settings.
 - b. Install the KFQSA adapter module into the backplane according to the procedures in the appropriate enclosure maintenance manual.

Figure A-1: KFQSA Module Layout (M7769)



Example A-1: KFQSA (M7769) Service Mode Switch Settings

KFQSA Four-Position Switchpack

	S/N Mode	Fx/F1	MSB	LSB
Switches:	1	2	3	4
	0	1	0	0

S/N = Service mode/Normal operating mode

Fx/F1 = fixed/floating CSR address

0 = on, 1 = off

A.2.1 Entering Console I/O Mode

After installing the KFQSA, you issue a series of commands to the KA655 system at the console prompt (>>>) in order to program the EEROM on the KFQSA. You may type these commands in either uppercase or lowercase letters. Unless otherwise specified, type each command, then press **Return**.

Enter the console I/O mode as follows:

1. Set the Break Enable/Disable switch on the CPU cover panel to the enable position (up; dot inside circle).
2. Set the on/off power switch to on (1).
3. When the power-up self-tests complete, the console prompt appears, as shown in Example A-2.

Example A-2: Entering Console Mode Display

Performing normal system tests.

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.

>>>

A.2.2 Displaying Current Addresses

Type **SHOW QBUS** to display the current Q22-bus addresses (Example A-3). Note that the KFQSA adapter appears in service mode as KFQSA #0.

Example A-3: SHOW QBUS Display

>>> show qbus

Scan of Qbus I/O Space

-20001910 (774420) = 0000 (000) KFQSA #0
-20001912 (774422) = 0AA0
-20001920 (774440) = FF08 (120) DELQA/DEQNA/DESQA
-20001922 (774442) = FF00
-20001924 (774444) = FF2B
-20001926 (774446) = FF09
-20001928 (774450) = FFA3
-2000192A (774452) = FF96
-2000192C (774454) = 8000
-2000192E (774456) = 1030
-20001940 (774500) = 0000 (260) TQK50/TQK70/TU81E/RV20/KFQSA-TAPE
-20001942 (774502) = 0BC0
-20001F40 (777500) = 0020 (004) IPCR

Scan of Qbus Memory Space

>>>

A.2.3 Running the Configure Utility

Now that you have reconfigured the system by installing the KFQSA storage adapter, you must run the Configure utility to find the correct address for each device and module in the system. The Configure utility uses floating address space rules.

Run the Configure utility as follows. Refer to Example A-4.

1. At the console prompt, type `CONFIGURE`, then type `HELP` at the `Device,Number?` prompt for a list of devices that can be configured.

NOTE: *Some of the devices listed in the `HELP` display are not supported by the KA655-AA CPU.*

2. For each device in the system, type the device name at the `Device,Number?` prompt. If you have more than one of the same type, type a comma followed by the total number of that device. In Example A-4, the system contains one KFQSA with six ISEs.

Be sure you list *all* the devices: those already installed and those you plan to install.

3. Type `EXIT`. The Configure utility displays an address and vector assignment for each device. Example A-4 shows the address and vector assignments and the device input.
4. For all modules except the KFQSA, verify that the CSR addresses are set correctly by comparing the addresses listed in the `SHOW QBUS` command with those listed in the Configure utility display. If necessary, remove modules from the backplane and reset switches or jumpers to the addresses in your Configure display, using module removal and replacement procedures in *BA213 Enclosure Maintenance*.
5. Write down the addresses for the KFQSA devices.

Example A-4: Configure Display

```
>>> configure
Enter device configuration, HELP, or EXIT
Device,Number? help
Devices:
  LPV11  KXJ11      DLV11J  DZQ11  DZV11  DFA01
  RLV12  TSV05      RXV21  DRV11W DRV11B  DPV11
  DMV11  DELQA      DEQNA  DESQA  RQDX3  KDA50
  RRD50  RQC25      KFQSA-DISK TQK50  TQK70  TU81E
  RV20   KFQSA-TAPE KMV11  IEQ11  DHQ11  DHV11
  CXA16  CXB16      CXY08  VCB01  QVSS   LNV11
  LNV21  QPSS       DSV11  ADV11C AAV11C  AXV11C
  KWV11C ADV11D     AAV11D VCB02  QDSS   DRV11J
  DRQ3B  VSV21      IBQ01  IDV11A IDV11B  IDV11C
  IDV11D IAV11A     IAV11B MIRA   ADQ32  DTC04
  DESNA  IGQ11

Numbers:
  1 to 255, default is 1
Device,Number? kfqsa-disk,6
Device,Number? desqa
Device,Number? tqk70
Device,Number? exit

Address/Vector Assignments
-774440/120 DESQA
-772150/154 KFQSA-DISK      ! Node 0 (assigned in order, 0 to n)
-760334/300 KFQSA-DISK      ! Node 1
-760340/304 KFQSA-DISK      ! Node 2
-760344/310 KFQSA-DISK      ! Node 3
-760350/314 KFQSA-DISK      ! Node 4
-760354/320 KFQSA-DISK      ! Node 5
-774500/260 TQK70
```

A.3 Programming the KFQSA

Program the configuration table in the EEROM of the KFQSA to include all ISEs on the DSSI bus, as follows. Refer to Examples A-5 through A-8.

IMPORTANT: *Use this procedure for the first or second KFQSA. In dual-host configurations, both the first and the second KFQSA adapters must have equal access to the system disk and to any DSSI ISEs. Program the KFQSA in the first system, then program the KFQSA in the second system.*

1. Determine the DSSI node plug address for each ISE you are configuring. Start with node 0, then continue up through node 5. In Example A-4, nodes 0, 1, 2, 3, 4, and 5 are used; nodes 6 and 7 are saved for KFQSA adapters. Reserve node 7 for the first KFQSA and node 6 for the second KFQSA.

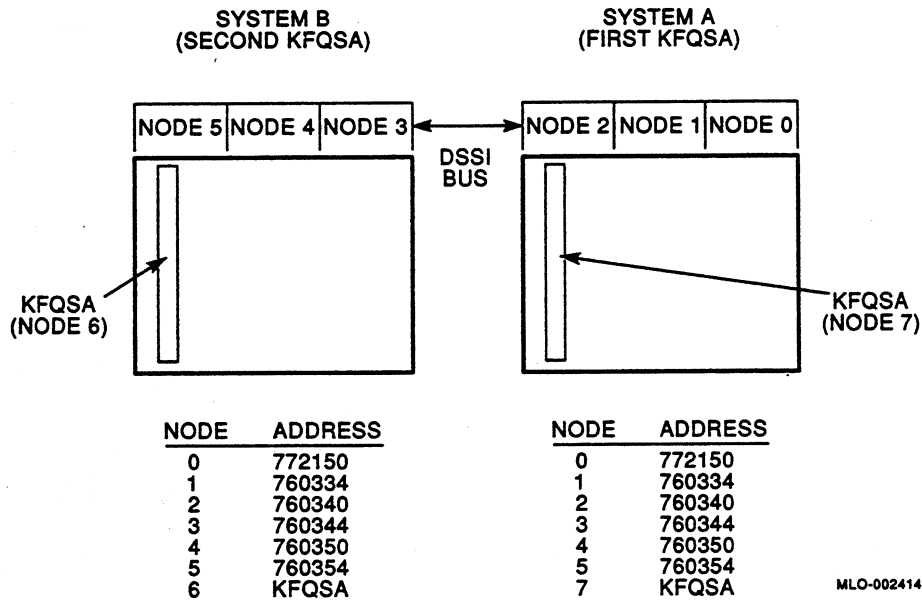
Figure A-2 shows an example of the nodes and addresses in a dual-host configuration.

2. At the console prompt on each system, type SET HOST/UQSSP/MAINT/SERV 0 to set host to the KFQSA.
3. Type HELP to display a list of supported commands.
4. Program the KFQSA to include each DSSI device in the system:
 - a. For each ISE: type SET, followed by the node number, the CSR address (from the list of addresses you obtained from the Configure utility), and the model number (disk ISE's are model 21).

For the KFQSA in the second system of a dual-host configuration, type SET 6 /KFQSA to set the node to 6 (Example A-6).

- b. Type SHOW to display the configuration table you just programmed.
- c. Check the display to make sure the addresses are correct.
- d. Type EXIT to save the configuration table, or QUIT to delete the table.

Figure A-2: Dual-Host Configuration Nodes and Addresses (Example)



Example A-5: Display for Programming the First KFQSA

```
>>> set host/uqssp/maint/serv 0      !0 refers to first KFQSA
                                       !in the system.

UQSSP Controller (772150)

Enter SET, CLEAR, SHOW, HELP, EXIT, or QUIT

Node   CSR Address   Model
  7     ----- KFQSA -----
? help
Commands:
    SET <NODE> /KFQSA                !Sets KFQSA DSSI node
                                       !number
    SET <NODE> <CSR_ADDRESS> <MODEL> !Enables a DSSI device
    CLEAR <NODE>                     !Disables a DSSI device
    SHOW                             !Displays current
                                       !configuration
    HELP                             !Displays this display
    EXIT                             !Saves the KFQSA program
    QUIT                             !Does not save the KFQSA
                                       !program

Parameters:
    <NODE>                           !0 through 7
    <CSR_ADDRESS>                     !760010 to 777774
    <MODEL>                           !21 (disk) or 22 (tape)

? set 0 772150 21
? set 1 760334 21
? set 2 760340 21
? set 3 760344 21
? set 4 760350 21
? set 5 760354 21
? show
Node   CSR Address   Model
  0      762105      21
  1      760334      21
  2      760340      21
  3      760344      21
  4      760350      21
  5      760354      21
  7     ----- KFQSA -----
? exit
Programming the KFQSA...              !Note from the system that
                                       !the KFQSA is
                                       !being programmed.
```

Example A-6: Display for Programming the KFQSA in a Dual-Host Configuration (Second System)

```
>>> set host/uqssp/maint/service 0    !0 refers to the KFQSA in the
                                         !second system of a dual-host
                                         !configuration. You set host
                                         !from the console of the second
                                         !system.
```

UQSSP Controller (772150)

Enter SET, CLEAR, SHOW, HELP, EXIT, or QUIT

```
Node      CSR Address      Model
 7         ----- KFQSA -----
```

? help

Commands:

```
    SET <NODE> /KFQSA                !Sets KFQSA DSSI node
                                         !number
    SET <NODE> <CSR_ADDRESS> <MODEL> !Enables a DSSI device
    CLEAR <NODE>                     !Disables a DSSI device
    SHOW                             !Displays current
                                         !configuration
    HELP                             !Displays this display
    EXIT                             !Saves the KFQSA program
    QUIT                             !Does not save the KFQSA
                                         !program
```

Parameters:

```
    <NODE>                            !0 through 7
    <CSR_ADDRESS>                     !760010 to 777774
    <MODEL>                           !21 (disk) or 22 (tape)
```

```
? set 0 772150 21
? set 1 760334 21
? set 2 760340 21
? set 3 760344 21
? set 4 760350 21
? set 5 760354 21
? set 6 /KFQSA
? show
```

```
Node      CSR Address      Model
 0         762105          21
 1         760334          21
 2         760340          21
 3         760344          21
 4         760350          21
 5         760354          21
 6         ----- KFQSA -----
```

? exit

Programming the KFQSA...

```
!Note from the system that
!the KFQSA is being
!programmed.
```

5. To allow the new program to take effect, turn the system power off by setting the on/off switch to off (0).
6. Remove the KFQSA from the backplane.
7. On the KFQSA, set switch 1 on the four-position switchpack to Off (1). (Figure A-1 shows the location and position of the switchpack.) This action sets the KFQSA to the normal operating mode; switches 2, 3, and 4 are disabled and the DSSI addresses are read from the EEROM.
8. Reinstall the KFQSA in the backplane.
9. Confirm that the unit ID plugs on the enclosure's operator control panel (OCP) match the node ID's you just programmed.
10. Power on the system by setting the on/off switch to on (1). Wait for the self-tests to complete.
11. At the console prompt, type `SHOW QBUS` to verify that all addresses are present and correct, as shown in Example A-7.
12. Type `SHOW DEVICE` to verify that all ISEs are displayed correctly, as shown in Example A-8.

Example A-7: SHOW QBUS Display

```
>>> show qbus
Scan of Qbus I/O Space
-200000DC (760334)=0000 (300) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000DE (760336)=0AA0
-200000E0 (760340)=0000 (304) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000E2 (760342)=0AA0
-200000E4 (760344)=0000 (310) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000E6 (760346)=0AA0
-200000E8 (760350)=0000 (314) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000EA (760352)=0AA0
-200000EC (760354)=0000 (320) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000EE (760356)=0AA0
-20001468 (772150)=0000 (154) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-2000146A (772152)=0AA0
-20001920 (774440)=FF08 (120) DELQA/DEQNA/DESQA
-20001922 (774442)=FF00
-20001924 (774444)=FF2B
-20001926 (774446)=FF09
-20001928 (774450)=FFA3
-2000192A (774452)=FF96
-2000192C (774454)=0050
-2000192E (774456)=1030
-20001940 (774500)=0000 (260) TQK50/TQK70/TU81E/RV20/KFQSA-TAPE
-20001942 (774502)=0BC0
-20001F40 (777500)=(004) IPCR

Scan of Qbus Memory Space
>>>
```

Example A-8: SHOW DEVICE Display

```
>>> show device
UQSSP Disk Controller 0 (772150)
-DUA0 (RF71)

UQSSP Disk Controller 1 (760334)
-DUB1 (RF71)

UQSSP Disk Controller 2 (760340)
-DUC2 (RF71)

UQSSP Disk Controller 3 (760344)
-DUD3 (RF71)

UQSSP Disk Controller 4 (760350)
-DUE4 (RF71)

UQSSP Disk Controller 5 (760354)
-DUF5 (RF71)

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

Ethernet Adapter 0 (774440)
-XQA0 (08-00-2B-09-A3-96)
```

A.4 Reprogramming the KFQSA

When you add a new DSSI device to a system with at least one RF-series ISE that has been programmed correctly, you must reprogram each KFQSA on the DSSI bus to include the new device(s) as follows:

1. Enter console I/O mode, using the procedure in Section A.2.1.
2. At the console prompt, type `SHOW DEVICE` for a display of all devices currently in the system. The display includes tape drives and the Ethernet adapter, as shown in Section A.3, Example A-8.

This display lists the VAX address and port name of the device, such as DUA0 RF71.

3. Type `SHOW QBUS` for a display of the eight-digit VAX address (hex) for each device, as shown in Section A.3, Example A-7.
4. Find the eight-digit VAX address for an ISE attached to the KFQSA that you are reprogramming. Use the `SET HOST` command to enter the KFQSA through an existing port and edit the configuration table, as follows. Refer to Example A-9.
 - a. Type `SET HOST/UQSSP/MAINT` followed by the VAX address.
 - b. Use the `SET` and `CLEAR` commands to reconfigure the KFQSA, as shown in Example A-9.
 - c. Type `SHOW` to display the new KFQSA configuration table setting.
 - d. Type `EXIT` to save the configuration table, or `QUIT` to cancel the reprogramming.

Example A-9: Reprogramming the KFQSA Display

```
>>> set host/uqssp/maint 20001468
UQSSP Controller (772150)
```

Node	CSR Address	Model
0	772150	21
1	760334	21
2	760340	21
3	760344	21
4	760350	21
5	760354	21
7	----- KFQSA -----	

```
? clear 5
```

```
? show
```

Node	CSR Address	Model
0	772150	21
1	760334	21
2	760340	21
3	760344	21
4	760350	21
7	----- KFQSA -----	

```
? set 5 760354 21
```

```
? show
```

Node	CSR Address	Model
0	772150	21
1	760334	21
2	760340	21
3	760344	21
4	760350	21
7	----- KFQSA -----	

```
? exit
```

```
Programming the KFQSA...
```

```
!Note from the system that the
!KFQSA is being programmed.
```

A.5 Changing the ISE Allocation Class and Unit Number

This section describes how to change the ISE allocation class and unit number.

If the system is part of a cluster, you must change the default allocation class parameter. The ISEs ship with the allocation class set to zero.

NOTE: *In a dual-host configuration, you must assign the same allocation class to both host systems and to the RF-series ISEs. This allocation class must be different from that of other systems or of hierarchical storage controllers (HSCs) in a cluster.*

For most configurations, you will not need to change the default unit numbers.

Change the allocation class and unit number parameters using the console-based DUP driver utility, as follows. Refer to Example A-10.

1. Determine the correct allocation class for the RF-series ISEs according to the rules on clustering.
2. Enter console I/O mode, using the procedure in Section A.2.1.
3. At the console prompt, type `SET HOST/DUP/UQSSP/DISK 0 PARAMS (0 through 5 for the ISE to which you want to connect) to start the DUP server.`
4. At the `PARAMS>` prompt, type `SHOW ALLCLASS` to check the current allocation class.
5. Type `SET ALLCLASS 2` to set the new allocation class to 2 (or type the number you desire).
6. Type `SHOW UNITNUM` to check the unit number.
7. To change the ISE's unit number from the default value, type `SET UNITNUM n` (where `n` is the new unit number). For example, type `SET UNITNUM 20` to change the unit number from 0 to 20.
8. Type `SET FORCEUNI 0` to set the forceunit flag to zero in order to use a non-default value. If you do not change the `FORCEUNI` parameter, the drive unit number defaults to the number of the corresponding DSSI plug on the operator control panel (OCP).
9. Type `SHOW ALLCLASS` to check the new allocation class.
10. Type `SHOW UNITNUM` to show the new unit number.

11. Type `SHOW FORCEUNI` to show the new forceunit flag values.
12. Type `WRITE`, then type `Y` to save the new values into the `EEROM`, or `N` to cancel the reprogramming.
13. Type `SHOW DEVICE` to make sure you have programmed the first ISE to have a unit number of 20. When you boot the operating system, the display shows the new allocation class and new unit number.

Example A-10: Display for Changing Allocation Class and Unit Number

```
>>> set host/dup/eqssp/disk 0 params
```

```
Starting DUP server...
```

```
UQSSP Disk Controller 0 (772150)
```

```
Copyright (c) 1988 Digital Equipment Corporation
```

```
PARAMS> show allclass
```

Parameter	Current	Default	Type	Radix	
ALLCLASS	1	0	Byte	Dec	B

```
PARAMS> set allclass 2
```

```
PARAMS> show unitnum
```

Parameter	Current	Default	Type	Radix	
UNITNUM	0	0	Word	Dec	B

```
PARAMS> set unitnum 20
```

```
PARAMS> set forceuni 0
```

```
PARAMS> show allclass
```

Parameter	Current	Default	Type	Radix	
ALLCLASS	2	0	Byte	Dec	B

```
PARAMS> show unitnum
```

Parameter	Current	Default	Type	Radix	
UNITNUM	20	0	Word	Dec	U

```
PARAMS> show forceuni
```

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

Example A-10 Cont'd. on next page

Example A-10 (Cont.): Display for Changing Allocation Class and Unit Number

```
PARAMS> write
Changes require controller initialization, ok? [Y/ (N) ] y
Stopping DUP server...

>>> show device
UQSSP Disk Controller 0 (772150)
-DUA0 (RF71)

UQSSP Disk Controller 1 (760334)
-DUB1 (RF71)

UQSSP Disk Controller 2 (760340)
-DUC2 (RF71)

UQSSP Disk Controller 3 (760344)
-DUD3 (RF71)

UQSSP Disk Controller 4 (760350)
-DUE4 (RF71)

UQSSP Disk Controller 5 (760354)
-DUF5 (RF71)

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

Ethernet Adapter 0 (774440)
-XQA0 (08-00-2B-09-A3-96)
```

Appendix B

KA655 CPU Address Assignments

B.1 General Local Address Space Map

Table B-1: VAX Memory Space

Contents	Address Range
Local memory space (64 Mbytes)	0000 0000-03FF FFFF
Reserved memory space (64 Mbytes)	0400 0000-07FF FFFF
Reserved memory space (64 Mbytes)	0800 0000-0BFF FFFF
Reserved memory space (64 Mbytes)	0C00 0000-0FFF FFFF
Cache diagnostic space (64 Mbytes)	1000 0000-13FF FFFF
Reserved cache diagnostic space (64 Mbytes)	1400 0000-17FF FFFF
Reserved cache diagnostic space (64 Mbytes)	1800 0000-1BFF FFFF
Reserved cache diagnostic space (64 Mbytes)	1C00 0000-1FFF FFFF

Table B-2: VAX Input/Output Space

Address Range	Contents
2000 0000–2000 1FFF	Local Q22-bus I/O space (8 Kbytes)
2000 2000–2003 FFFF	Reserved local I/O space (248 Kbytes)
2004 0000–2005 FFFF	Local ROM space—halt protected space (128 Kbytes)
2006 0000–2007 FFFF	Local ROM space—halt unprotected space (128 Kbytes)
2008 0000–201F FFFF	Local register I/O space (1.5 Mbytes)
2020 0000–23FF FFFF	Reserved local I/O space (62.5 Mbytes)
2400 0000–27FF FFFF	Reserved local I/O space (64 Mbytes)
2800 0000–2BFF FFFF	Reserved local I/O space (64 Mbytes)
2C08 0000–2FFF FFFF	Reserved local I/O space (64 Mbytes)
3000 0000–303F FFFF	Local Q22-bus memory space (4 Mbytes)
3040 0000–33FF FFFF	Reserved local I/O space (60 Mbytes)
3400 0000–37FF FFFF	Reserved local I/O space (64 Mbytes)
3800 0000–3BFF FFFF	Cache tag diagnostic space (64 Mbytes)*
3C00 0000–3FFF FFFF	Reserved cache tag diagnostic space (64 Mbytes)

*Not visible during normal operation.

B.2 Detailed Local Address Space Map

Table B-3: Detailed VAX Memory Space

Contents	Address Range
Local memory space (up to 64 Mbytes)	0000 0000–03FF FFFF
Q22-bus map—top 32 Kbytes of main memory	
Reserved memory space	0400 0000–0FFF FFFF
Cache diagnostic space	1000 0000–13FF FFFF
Reserved cache diagnostic space	1800 0000–1FFF FFFF

Table B-4: Detailed VAX Input/Output Space

Contents	Address Range
Local Q22-bus I/O Space	2000 0000–2000 1FFF
Reserved Q22-bus I/O space (diagnostics)	2000 0000–2000 0007
Q22-bus floating address space	2000 0008–2000 07FF
Q22-bus fixed and user I/O space	2000 0800–2000 0FFF
Q22-bus fixed and reserved I/O space	2000 1000–2000 1F3F
Interprocessor communication register (normal operation)	2000 1F40
Interprocessor communication register (reserved)	2000 1F42
Interprocessor communication register (reserved)	2000 1F44
Interprocessor communication register (reserved)	2000 1F46
Reserved Q22-bus I/O space	2000 1F48–2000 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
DMA master error address register	2008 0008
DMA slave error address register	2008 000C
Q22-bus map base register	2008 0010
Reserved local register I/O space	2008 0014–2008 00FF
Main memory configuration register 00	2008 0100
Main memory configuration register 01	2008 0104
Main memory configuration register 02	2008 0108
Main memory configuration register 03	2008 010C
Main memory configuration register 04	2008 0110
Main memory configuration register 05	2008 0114
Main memory configuration register 06	2008 0118
Main memory configuration register 07	2008 011C
Main memory configuration register 08	2008 0120
Main memory configuration register 09	2008 0124
Main memory configuration register 10	2008 0128
Main memory configuration register 11	2008 012C
Main memory configuration register 12	2008 0130
Main memory configuration register 13	2008 0134
Main memory configuration register 14	2008 0138

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

Contents	Address Range
Main memory configuration register 15	2008 013C
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
Cache control register	2008 4000
Boot and diagnostic register	2008 4004
Reserved local register I/O space	2008 4008–2008 7FFF
Q22-bus map registers	2008 8000–2008 FFFF
Reserved local register I/O space	2009 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
<hr/> The following addresses allow those KA655 IPRs that are implemented in the SSC to be accessed through the local I/O page. These addresses are documented for diagnostic purposes only and should not be used by nondiagnostic programs.	
Time-of-year register	2014 006C
Console storage receiver status	2014 0070*
Console storage receiver data	2014 0074*
Console storage transmitter status	2014 0078*
Console storage transmitter data	2014 007C*
Console receiver control/status	2014 0080
Console receiver data buffer	2014 0084
Console transmitter control/status	2014 0088
Console transmitter data buffer	2014 008C
Reserved local register I/O space	2014 0090–2014 00DB
* These registers are not fully implemented; accesses yield unpredictable results.	
I/O bus reset register	2014 00DC
Reserved local register I/O space	2014 00E0–2014 00EF
ROM data register	2014 00F0**
Bus timeout counter	2014 00F4**

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

Contents	Address Range
Interval timer	2014 00F8**
Reserved local register I/O space	2014 00FC–2014 00FF

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

Local Register I/O Space (Continued)

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
CACR address decode match register	2014 0130
CACR decode mask register	2014 0134
Reserved local register I/O space	2014 0138–2014 013F
BDR address decode match register	2014 0140
BDR 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 0000–37FF FFFF
Cache tag diagnostic space	3800 0000–3BFF FFFF †
Reserved cache tag diagnostic space	3C00 0000–3FFF FFFF

† Not visible during normal operation.

B.3 Internal Processor Registers

Each Internal Processor Register (IPR) falls into one of the categories listed below. You must use the MFPR and MTPR privileged instructions to access the IPRs.

1. Implemented by KA655 (in the CVAX chip) as specified in the *VAX Architecture Reference Manual*.
2. Implemented by KA655 (in the SSC) as specified in the *VAX Architecture Reference Manual*.
3. Implemented by KA655 (and all designs that use the CVAX chip) uniquely.
4. Not implemented, timed out by the CDAL Bus Timer (in the SSC) after 4 usec. Read as zero, NOP on write.
5. Access not allowed; accesses result in a reserved operand fault.
6. Accessible, but not fully implemented; accesses yield unpredictable results.

Refer to Table B-5 for a listing of each of the KA655 IPRs, along with its mnemonic, its access type (read or write), and its category number.

NOTE: An *I* following the category number in Table B-5 indicates that the register is initialized on power-up and by the negation of DCOK when the processor is halted.

Table B-5: KA655 Internal Processor Registers

Decimal	Hex	Register Name	Mnemonic	Type	Category
0	0	Kernel Stack Pointer	KSP	r/w	1
1	1	Executive Stack Pointer	ESP	r/w	1
2	2	Supervisor Stack Pointer	SSP	r/w	1
3	3	User Stack Pointer	USP	r/w	1
4	4	Interrupt Stack Pointer	ISP	r/w	1
7:5	7:5	Reserved			5
8	8	P0 Base	P0BR	r/w	1
9	9	P0 Length	P0LR	r/w	1
10	A	P1 Base	P1BR	r/w	1
11	B	P1 Length	P1LR	r/w	1
12	C	System Base	SBR	r/w	1
13	D	System Length	SLR	r/w	1
15:14	F:E	Reserved			5
16	10	Process Control Block Base	PCBB	r/w	1
17	11	System Control Block Base	SCBB	r/w	1
18	12	Interrupt Priority Level	IPL	r/w	1 I
19	13	AST Level	ASTLVL	r/w	1 I
20	14	Software Interrupt Request	SIRR	w	1
21	15	Software Interrupt Summary	SISR	r/w	1 I
23:22	17:16	Reserved			5
24	18	Interval Clock Control/Status	ICCS	r/w	3 I
25	19	Next Interval Count	NICR	w	5
26	1A	Interval Count	ICR	r	5
27	1B	Time-of-Year Clock	TODR	r/w	2
28	1C	Console Storage Receiver Status	CSRS	r/w	7 I
29	1D	Console Storage Receiver Data	CSRD	r	7 I
30	1E	Console Storage Transmit Status	CSTS	r/w	7 I
31	1F	Console Storage Transmit Data	CSTD	w	7 I
32	20	Console Receiver Control/Status	RXCS	r/w	4 I
33	21	Console Receiver Data Buffer	RXDB	r	4 I
34	22	Console Transmit Control/Status	TXCS	r/w	4 I
35	23	Console Transmit Data Buffer	TXDB	w	4 I
36	24	Translation Buffer Disable	TBDR	r/w	5
37	25	Cache Disable	CADR	r/w	3 I
38	26	Machine Check Error Summary	MCESR	r/w	5
39	27	Memory System Error	MSER	r/w	3 I
41:40	29:28	Reserved			5
42	2A	Console Saved PC	SAVPC	r	3
43	2B	Console Saved PSL	SAVPSL	r	3

Table B-5 (Cont.): KA655 Internal Processor Registers

Decimal	Hex	Register Name	Mnemonic	Type	Category
47:44	2F:2C	Reserved			5
48	30	SBI Fault/Status	SBIFS	r/w	5
49	31	SBI Silo	SBIS	r	5
50	32	SBI Silo Comparator	SBISC	r/w	5
51	33	SBI Maintenance	SBIMT	r/w	5
52	34	SBI Error	SBIER	r/w	5
53	35	SBI Timeout Address	SBITA	r	5
54	36	SBI Quadword Clear	SBIQC	w	5
55	37	I/O Bus Reset	IORESET	w	4
56	38	Memory Management Enable	MAPEN	r/w	1
57	39	TB Invalidate All	TBIA	w	1
58	3A	TB Invalidate Single	TBIS	w	1
59	3B	TB Data	TBDATA	r/w	5
60	3C	Microprogram Break	MBRK	r/w	5
61	3D	Performance Monitor Enable	PMR	r/w	5
62	3E	System Identification	SID	r	1
63	3F	Translation Buffer Check	TBCHK	w	1
64:127	40:7F	Reserved			6

B.3.1 KA655 VAX Standard IPRs

IPRs that are implemented as specified in the *VAX Architecture Reference Manual* are listed in Table B-6. See that manual for details on the operation and use of these registers.

Table B-6: IPRs Implemented According to Standard VAX Architecture

Number	Hex	Register Name	Mnemonic
12	C	System Base Register	SBR
13	D	System Length Register	SLR
16	10	Process Control Block Base	PCBB
17	11	System Control Block Base	SCBB
18	12	Interrupt Priority Level	IPL
20	14	Software Interrupt Request	SIRR
21	15	Software Interrupt Summary	SISR
27	1B	Time-of-Year Clock	TODR
56	38	Memory Management Enable	MAPEN
57	39	Translation Buffer Invalidate All	TBIA
58	3A	Translation Buffer Invalidate Single	TBIS
62	3E	System Identification	SID
63	3F	Translation Buffer Check	TBCHK

B.3.2 KA655 Unique IPRs

IPRs that are implemented on the KA655 but are not contained in, or do not fully conform to the standards in the *VAX Architecture Reference Manual*, are listed in Table B-7.

Table B-7: KA655 Unique IPRs

Number	Hex	Register Name	Mnemonic
24	18	Interval Clock Control/Status	ICCS
32	20	Console Receiver Control/Status	RXCS
33	21	Console Receiver Data Buffer	RXDB
34	22	Console Transmit Control/Status	TXCS
35	23	Console Transmit Data Buffer	TXDB
37	25	Cache Disable	CADR
39	27	Memory System Error	MSER
42	2A	Console Saved PC	SAVPC
43	2B	Console Saved PSL	SAVPSL
55	37	I/O Bus Reset	IORESET

B.4 Global Q22-Bus Address Space Map

The addresses and memory contents of the Q22-bus memory space are listed in Table B-8.

Table B-8: Q22-Bus Memory Space

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

The contents and addresses of the Q22-bus I/O space with BBS7 asserted are listed in Table B-9.

Table B-9: 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 (diagnostics)	1776 0000–1776 0007
Q22-bus floating address space	1776 0010–1776 3777
Q22-bus fixed and user address space	1776 4000–1776 7777
Q22-bus fixed and reserved address space	1777 0000–1777 7477
Interprocessor communication register (normal operation)	1777 7500
Interprocessor communication register (reserved)	1777 7502
Interprocessor communication register (reserved)	1777 7504
Interprocessor communication register (reserved)	1777 7506
Reserved Q22-bus I/O space	1777 7510–1777 7777

Appendix C

Related Documentation

The following documents contain information relating to the KA655 CPU.

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
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
IEU11-A/IEQ11-A User's Guide	EK-IEUQ1-UG
KDA50-Q CPU Module User's Guide	EK-KDA5Q-UG
KFQSA Installation Guide	EK-KFQSA-IN
KMV11 Programmable Communications Controller User's Guide	EK-KMV11-UG
KMV11 Programmable Communications Controller Technical Manual	EK-KMV11-TM
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	
RF30 Integrated Storage Element	EK-RF30D-UG
RF30 Integrated Storage Element Installation Manual	EK-RF30D-IN
RF71 Integrated Storage Element User's Guide	EK-RF71D-UG
TK50 Tape Drive Subsystem User's Guide	EK-LEP05-UG

Document Title	Order Number
Systems	
BA213 Enclosure Maintenance	EK-189AA-MG
BA215 Enclosure Maintenance	EK-191AA-MG
H9644 Cabinet Maintenance	EK-221AA-MG
Microsystems Options	EK-192AA-MG
Microsystems Site Preparation Guide	EK-067AB-PG
Diagnostics	
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

! (comment command), 3-52
9E utility, 4-10
9C utility, 4-28, 4-35

A

Acceptance testing, 4-26
Address assignments, B-1 to B-11
Autoboot, description of, 3-9

B

Battery backup unit (BBU), 1-15
Baud rate, 1-12
 changing, 1-14
Binary load and unload (X
 command), 3-50
Boot, naming device for, 3-10
BOOT command, 3-21
Boot devices, supported, 3-9
Boot flags, 3-8
Bootstrap
 conditions, 3-6
 device names, 3-8
 initialization, 3-7
Break enable/disable switch
 disable setting, description of,
 3-9
Bus length (DSSI), 2-12

C

Cables, CPU to H3600-SA, 1-15
Cabling
 DSSI, 2-10
 RF-series ISEs, 2-10
Cache memory, 1-9
Central processing unit (CPU), 1-8

CFPA60 chip, 1-9
Changing the baud rate, 1-14
Clock chip (CCLK), 1-9
CMCTL chip, 1-10
Comment command (!), 3-52
Configuration, 2-1 to 2-16
 and module order, 2-1
 DSSI, 2-4
 limitations on dual-host, 2-14
 rules, 2-2
 worksheet, 2-14
CONFIGURE command, 2-3, 3-23
Connectors on CPU module, 1-12
Console commands
 address space control qualifiers,
 3-18
 address specifiers, 3-14
 binary load and unload (X), 3-50
 BOOT, 3-21
 ! (comment), 3-52
 CONFIGURE, 3-23
 CONTINUE, 3-25
 data control qualifiers, 3-17
 DEPOSIT, 3-26
 EXAMINE, 3-27
 FIND, 3-29
 HALT, 3-30
 HELP, 3-31
 INITIALIZE, 3-33
 keywords, 3-19
 list of, 3-19
 MOVE, 3-34
 NEXT, 3-36
 qualifier and argument
 conventions, 3-14
 qualifiers, 3-17
 REPEAT, 3-38

Console commands (cont'd.)

- SEARCH, 3-39
- SET, 3-41
- SHOW, 3-44
- START, 3-47
- symbolic addresses, 3-14
- syntax, 3-13
- TEST, 3-48
- UNJAM, 3-49
- X (binary load and unload), 3-50
- Console displays, 4-14
 - and FRUs, 4-18
- Console error messages, 4-24
 - list of, 4-25
 - sample of, 4-15
- Console I/O mode
 - restart caution, 3-3
 - special characters, 3-13
- Console port, testing, 4-38
- CONTINUE command, 3-25
- CQBIC chip, 1-11
- Current and power values, 2-14

D

- DEPOSIT command, 3-26
- Diagnostic executive, 4-3
 - error field, 4-16
- Diagnostics, RF-series, 4-40
- Diagnostic tests
 - list of, 4-4
 - parameters for, 4-4
- DRVEXR local program, 4-43
- DRVTST local program, 4-42
- DSSI
 - bus length, 2-12
 - bus termination, 2-12
 - cabling, 2-10
 - configuration, 2-4
 - drive order, 2-4
 - dual-host, 2-13
 - expansion configurations, 2-14
 - node ID, 2-4
 - node name, changing, 2-6
 - unit number, changing, 2-7

Dual-host

- capability, 2-13
- limitations, 2-14

E

- Entry and dispatch code, 3-2
- ERASE local program, 4-45
- Error messages
 - console, list of, 4-25
 - console, sample of, 4-15
 - halt, 4-24
 - VMB, 4-26
- EXAMINE command, 3-27

F

- FE utility, 4-31
- FIND command, 3-29
- Firmware, 1-11, 3-1 to 3-52
 - power-up sequence, 3-4
- Floating point accelerator (CFPA), 1-9
- FRUs
 - and console display, 4-18
- Fuse, on KA655 module, 4-38

G

- General purpose registers (GPRs)
 - in error display, 4-17
 - initialization of, 3-7
 - symbolic addresses for, 3-14

H

- H3103 loopback connector, 4-38
 - testing serial line with, 3-4
- H3600-SA, 1-12
 - changing the baud rate, 1-14
 - enable/disable switch differences, 1-14
 - hex LED display, 1-14
 - with KA630 CPU, 1-14
 - with KA650 CPU, 1-14
- H3600-SA I/O panel, 4-38

H3600-SA mode switch
 set to language inquiry, 3-5
 set to normal, 3-6
 set to test, 3-4
H8572 loopback connector, 4-38
HALT command, 3-30
Halts
 conditions for external, 3-3
 entry and dispatch code, 3-2
 messages, list of, 4-24
 registers saved, 3-2
 registers set to fixed values, 3-2
Hardware error summary register,
 4-32
HELP command, 3-31
Hex LED display, 1-14
HISTORY local program, 4-44

I

INITIALIZE command, 3-33
Initial power-up test
 See IPT
Internal processor registers (IPR)
 symbolic addresses for, 3-15
IPT, 3-4, 4-19, B-6

K

KA655
 connectors, 1-12
 features, 1-7
 functional block diagram, 1-3
 fuse, 4-38
 LEDs, 4-18
 system block diagram, part I,
 1-5
 system block diagram, part II,
 1-6
 variants, 1-4
KFQSA storage adapter
 programming instructions for,
 A-1

L

Language selection menu
 conditions for display of, 3-5
 example of, 3-5
 messages, list of, 3-5
Load module, M9060-YA, 2-14
Loopback connectors
 H3103, 3-4, 4-38
 H8572, 4-38
 list of, 4-39
 tests, 4-38

M

M9060-YA load module, 2-14
MEMCSR 0-15, 4-28
Memory
 acceptance testing of, 4-28
 cache, 1-9
 controller chip (CMCTL), 1-10
 isolating FRU, 4-28, 4-34
 MS650-BA, 1-16
 on KA655, 1-9
 testing, 4-34
Messages
 console error, 4-25
 halt, 4-24
 VMB error, 4-26
Module
 configuration, 2-3
 for KA655 systems, recommended
 order, 2-2
 order, in backplane, 2-1
 self-tests, 4-38
MOVE command, 3-34
MS650-BA memory module, 1-16
MS650-BF option kit, contents of,
 1-16

N

NEXT command, 3-36

O

OCP cabling, 2-10
Operator console panel
 See OCP

P

Parameters
 for diagnostic tests, 4-5
 in error display, 4-16
PARAMS local program, 4-46
 commands, 4-46
Physical memory
 symbolic addresses for, 3-15
POST
 errors handled by, 4-40
Power-on self test
 See POST
Power-up mode switch, 1-15
Power-up sequence, 3-4
Power values, 2-14

Q

Q22-bus interface chip (CQBIC),
 1-11

R

REPEAT command, 3-38
Restart caution, 3-3
Restart sequence, 3-11
RF-series ISE
 cabling, 2-10
 diagnostic error codes, 4-49
 diagnostics, 4-40
 errors, 4-40
 node ID switches, 2-6
 node name, 2-6
 unit number, 2-7
RF-series ISE local programs
 DRVEXR, 4-43
 DRVST, 4-42
 ERASE, 4-45
 HISTORY, 4-44
 list of, 4-41
 PARAMS, 4-46

ROM-based diagnostics, 4-2 to
 4-49
 and memory testing, 4-34
 console displays during, 4-14,
 4-15
 list of, 4-3
 parameters, 4-4
 utilities, 4-3
RPB, locating, 3-12

S

Scripts, 4-3, 4-6 to 4-14
 calling sequence for, 4-8
 creation of, using 9F utility, 4-10
 field service, 4-8
 list of, 4-7
SEARCH command, 3-39
Self-test, for modules, 4-38
Serial line test using H3103, 3-4
SET command, 3-41
SET HOST/DUP command, 3-41
SHOW command, 3-44
SLU cable, 1-15
SLU connector, 1-14
SSC (system support chip), 1-10
START command, 3-47
Symbolic addresses, 3-14
 for any address space, 3-17
 for GPRs, 3-14
 for IPRs, 3-15
 for physical memory, 3-15
System support chip (SSC), 1-10

T

TEST command, 3-48
Tests, diagnostic
 list of, 4-3
 parameters for, 4-5
Troubleshooting, 4-31 to 4-49
 procedures, general, 4-1
 suggestions, additional, 4-37

U

UNJAM command, 3-49

Utilities, diagnostic, 4-3

V

Virtual memory bootstrap

See VMB

VMB, 3-7

boot flags, 3-8

error messages, 4-26

X

X command (binary load and
unload), 3-50