KA655 CPU System Maintenance
Order Number EK-306AA-MG-001
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:

<table>
<thead>
<tr>
<th>DEC</th>
<th>MicroVAX</th>
<th>UNIBUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECmate</td>
<td>MicroVMS</td>
<td>VAX</td>
</tr>
<tr>
<td>DECnet</td>
<td>PDP</td>
<td>VAXBI</td>
</tr>
<tr>
<td>DECUS</td>
<td>P/OS</td>
<td>VAXELN</td>
</tr>
<tr>
<td>DECwriter</td>
<td>Professional</td>
<td>VAXcluster</td>
</tr>
<tr>
<td>DELNI</td>
<td>Q-bus</td>
<td>VAXstation</td>
</tr>
<tr>
<td>DEQNA</td>
<td>Rainbow</td>
<td>VMS</td>
</tr>
<tr>
<td>DESTA</td>
<td>RSTS</td>
<td>VT</td>
</tr>
<tr>
<td>DIBOL</td>
<td>RSX</td>
<td>Work Processor</td>
</tr>
<tr>
<td>DSSI</td>
<td>RT</td>
<td></td>
</tr>
<tr>
<td>MASSBUS</td>
<td>ThinWire</td>
<td>digital</td>
</tr>
<tr>
<td>MicroPDP-11</td>
<td>U/LTRIX</td>
<td></td>
</tr>
</tbody>
</table>

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**

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

**Chapter 2  Configuration**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Introduction</td>
<td>2–1</td>
</tr>
<tr>
<td>2.2</td>
<td>General Module Order</td>
<td>2–1</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Module Order Rules for KA655 Systems</td>
<td>2–2</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Recommended Module Order for KA655 Systems</td>
<td>2–2</td>
</tr>
<tr>
<td>2.3</td>
<td>Module Configuration</td>
<td>2–3</td>
</tr>
<tr>
<td>2.4</td>
<td>DSSI Configuration</td>
<td>2–4</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Changing RF-Series ISE Parameters</td>
<td>2–6</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Changing the Unit Number</td>
<td>2–7</td>
</tr>
<tr>
<td>2.4.3</td>
<td>Changing the Allocation Class</td>
<td>2–9</td>
</tr>
</tbody>
</table>
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
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 DRVSTT 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.
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)

MEMORY CONNECTION  CONSOLE CONNECTION  CONSOLE SERIAL LINE CONNECTION

J3  LEDs  J2  J1

EPROM

SECOND LEVEL CACHE

Y1  Y2

SSC

Y3

CCLK

CMCTL  CVAX  CFPA  CQBIC

ML0–002367

1–2 KA655 CPU System Maintenance
Figure 1–2 shows a functional block diagram of the KA655 CPU.

Figure 1–2: KA655 CPU Functional Block Diagram
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.
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 translation 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

- **POWER UP MODE SWITCH**
  - LANGUAGE INQUIRY
  - NORMAL OPERATION (FACTORY SETTING)
  - LOOPBACK TEST

- **BREAK ENABLE SWITCH** (DOWN=DISABLED)

- **LED DISPLAY**

- **SLU CONNECTOR**

- **CABLE TO CPU**

- **BAUD RATE SELECT SWITCH** (9600 IS FACTORY SETTING)

- **BATTERY BACKUP UNIT (BBU)**

- **LIST OF SWITCH SETTINGS FOR BAUD RATES**
  - 0=300
  - 1=600
  - 2=1200
  - 3=2400
  - 4=4800
  - 5=9600
  - 6=19200
  - 7=38400

MLO–002372
Table 1–1: H3600–SA Controls and Connectors

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

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

<table>
<thead>
<tr>
<th>Switch</th>
<th>Position</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable/disable</td>
<td>Dot outside</td>
<td><strong>Breaks are disabled (factory setting).</strong> On power-up or restart, the</td>
</tr>
<tr>
<td>(two-position toggle)</td>
<td>circle</td>
<td>system tries to load software from one of the devices after completing</td>
</tr>
<tr>
<td></td>
<td>Dot inside</td>
<td><strong>Breaks are enabled.</strong> On power-up or restart, the system enters console</td>
</tr>
<tr>
<td></td>
<td>circle</td>
<td>I/O mode after completing the power-up diagnostics.</td>
</tr>
<tr>
<td>Power-up mode</td>
<td>Arrow</td>
<td><strong>Run (factory setting).</strong> If the console terminal supports the DEC</td>
</tr>
<tr>
<td>(three-position</td>
<td>Face</td>
<td>multinational character set (MCS), the system prompts the user for the</td>
</tr>
<tr>
<td>rotary)</td>
<td>T inside circle</td>
<td><strong>Language inquiry.</strong> If the console terminal supports the DEC MCS, the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>user is prompted for the console language on every power-up and restart.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>All power-up diagnostics run.</td>
</tr>
<tr>
<td>Baud rate select</td>
<td>300 to 38,400</td>
<td><strong>Test.</strong> ROM programs run the wraparound console serial line unit (SLU)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tests.</td>
</tr>
</tbody>
</table>

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             |

1–16 KA655 CPU System Maintenance
Figure 1-7: MS650-BA Memory Module (M7622-A)
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
KWV11–SA
DRV1J–SA
TSV05–SA
DMV11
LNV21
DEQNA/DELQA/DESQA–SA
DPV11–SA
KMV1A–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  KKJ11  DLV11J  DZQ11  DZV11  DFA01
   RLV12  TSV05  RXV21  DRV11W  DRV11B  DPV11
   DVM11  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
   KRW11C  ADV11D  AAV11D  VCB02  QDSS  DRV11J
   DRQ3B  VSV21  IBQ01  IDV11A  IDV11B  IDV11C
   IDV11D  IAV11C  IAV11B  MIRA  ADQ32  DTC04
   DESNA  IGG01

   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

<table>
<thead>
<tr>
<th>Device</th>
<th>Position</th>
<th>Node ID$^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>Right side</td>
<td>0</td>
</tr>
<tr>
<td>Second</td>
<td>Center</td>
<td>1</td>
</tr>
<tr>
<td>Third</td>
<td>Left side</td>
<td>2</td>
</tr>
</tbody>
</table>

$^1$KA655 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

<table>
<thead>
<tr>
<th>DSSI Node ID</th>
<th>1 (MSB)</th>
<th>2</th>
<th>3 (LSB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Down</td>
<td>Down</td>
<td>Down</td>
</tr>
<tr>
<td>1</td>
<td>Down</td>
<td>Down</td>
<td>Up</td>
</tr>
<tr>
<td>2</td>
<td>Down</td>
<td>Up</td>
<td>Down</td>
</tr>
<tr>
<td>3</td>
<td>Down</td>
<td>Up</td>
<td>Up</td>
</tr>
<tr>
<td>4</td>
<td>Up</td>
<td>Down</td>
<td>Down</td>
</tr>
<tr>
<td>5</td>
<td>Up</td>
<td>Down</td>
<td>Up</td>
</tr>
<tr>
<td>6</td>
<td>Up</td>
<td>Up</td>
<td>Down</td>
</tr>
<tr>
<td>7$^2$</td>
<td>Up</td>
<td>Up</td>
<td>Up</td>
</tr>
</tbody>
</table>

$^1$Up = toward the head disk assembly (HDA); Down = toward the drive module
$^2$Normally 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 DIan, 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

```bash
>>> 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.
```

```bash
>>> 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
HISTRY 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Current</th>
<th>Default</th>
<th>Type</th>
<th>Radix</th>
</tr>
</thead>
<tbody>
<tr>
<td>NODENAME</td>
<td>R3YBNE</td>
<td>RF71</td>
<td>String</td>
<td>Ascii</td>
</tr>
</tbody>
</table>

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

```plaintext
>>> 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
DRTST 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**

```plaintext
>>> set host /dup/ugssp/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
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>
<thead>
<tr>
<th>Option</th>
<th>Module</th>
<th>Current (Amps)</th>
<th>Power</th>
<th>Bus Loads</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>+5 V</td>
<td>+12 V</td>
<td>Watts</td>
</tr>
<tr>
<td>AAV11-SA</td>
<td>A1009-PA</td>
<td>1.8</td>
<td>0.0</td>
<td>9.0</td>
</tr>
<tr>
<td>ADV11-SA</td>
<td>A1008-PA</td>
<td>3.2</td>
<td>0.0</td>
<td>16.0</td>
</tr>
<tr>
<td>AXV11-SA</td>
<td>A026-PA</td>
<td>2.0</td>
<td>0.0</td>
<td>10.0</td>
</tr>
<tr>
<td>CXA16-AA/-AF</td>
<td>M3118-YA</td>
<td>1.6</td>
<td>0.20</td>
<td>10.4</td>
</tr>
<tr>
<td>CXB16-AA/-AF</td>
<td>M3118-YB</td>
<td>2.0</td>
<td>0.0</td>
<td>10.0</td>
</tr>
<tr>
<td>CXY08-AA/-AF</td>
<td>M3119-YA</td>
<td>1.64</td>
<td>0.395</td>
<td>12.94</td>
</tr>
<tr>
<td>DELQA-SA</td>
<td>M7516-PA</td>
<td>2.7</td>
<td>0.5</td>
<td>19.5</td>
</tr>
<tr>
<td>DFA01-AA/-AF</td>
<td>M3121-PA</td>
<td>1.97</td>
<td>0.40</td>
<td>14.7</td>
</tr>
<tr>
<td>DPV11-SA</td>
<td>M8020-PA</td>
<td>1.2</td>
<td>0.30</td>
<td>9.6</td>
</tr>
<tr>
<td>DRQ3B-SA</td>
<td>M7658-PA</td>
<td>4.5</td>
<td>0.0</td>
<td>22.5</td>
</tr>
<tr>
<td>DRV1J-SA</td>
<td>M8049-PA</td>
<td>1.8</td>
<td>0.0</td>
<td>9.0</td>
</tr>
<tr>
<td>DRV1W-SA</td>
<td>M7651-PA</td>
<td>1.8</td>
<td>0.0</td>
<td>9.0</td>
</tr>
<tr>
<td>DSV11-SA</td>
<td>M3108-PA</td>
<td>5.43</td>
<td>0.69</td>
<td>38.0</td>
</tr>
<tr>
<td>DZQ11-SA</td>
<td>M3106-PA</td>
<td>1.0</td>
<td>0.36</td>
<td>9.3</td>
</tr>
<tr>
<td>IBQ01-SA</td>
<td>M3125-PA</td>
<td>5.0</td>
<td>0.0</td>
<td>25.0</td>
</tr>
<tr>
<td>IEQ11-SA</td>
<td>M8634-PA</td>
<td>3.5</td>
<td>0.0</td>
<td>17.5</td>
</tr>
<tr>
<td>KA655-AA</td>
<td>M7625-AA/-BA</td>
<td>3.7</td>
<td>0.14</td>
<td>21.0</td>
</tr>
<tr>
<td>KDA50-Q</td>
<td>M7164</td>
<td>6.93</td>
<td>0</td>
<td>34.6</td>
</tr>
<tr>
<td>KDA50-Q</td>
<td>M6165</td>
<td>6.57</td>
<td>0.03</td>
<td>33.21</td>
</tr>
<tr>
<td>KFQSA-SA</td>
<td>M7769</td>
<td>5.5</td>
<td>0.0</td>
<td>27.0</td>
</tr>
<tr>
<td>KLESI-SA</td>
<td>M7740-PA</td>
<td>3.0</td>
<td>0.0</td>
<td>15.0</td>
</tr>
<tr>
<td>KMV1A-SA</td>
<td>M7500-PA</td>
<td>2.6</td>
<td>0.2</td>
<td>15.4</td>
</tr>
<tr>
<td>KVV11-SA</td>
<td>M4002-PA</td>
<td>2.2</td>
<td>0.13</td>
<td>11.15</td>
</tr>
<tr>
<td>LPV11-SA</td>
<td>M8096-PA</td>
<td>1.6</td>
<td>0.0</td>
<td>8.0</td>
</tr>
<tr>
<td>M9060-YA</td>
<td></td>
<td>5.3</td>
<td>0.0</td>
<td>26.5</td>
</tr>
<tr>
<td>MS650-AA</td>
<td>M7621-A</td>
<td>2.7</td>
<td>0.0</td>
<td>13.5</td>
</tr>
<tr>
<td>RF30-SA</td>
<td></td>
<td>1.10</td>
<td>0.80</td>
<td>15.1</td>
</tr>
<tr>
<td>RF71E-SA</td>
<td></td>
<td>1.25</td>
<td>4.54</td>
<td>26.5</td>
</tr>
<tr>
<td>TK50E-EA</td>
<td></td>
<td>1.35</td>
<td>2.4</td>
<td>36.6</td>
</tr>
<tr>
<td>TK70E-EA</td>
<td></td>
<td>1.5</td>
<td>2.4</td>
<td>36.3</td>
</tr>
<tr>
<td>TQK50</td>
<td>M7546</td>
<td>2.9</td>
<td>0.0</td>
<td>14.5</td>
</tr>
<tr>
<td>TQK70-SA</td>
<td>M7559</td>
<td>3.5</td>
<td>0.0</td>
<td>17.5</td>
</tr>
<tr>
<td>TSV05-SA</td>
<td>M7196</td>
<td>6.5</td>
<td>0.0</td>
<td>32.5</td>
</tr>
</tbody>
</table>
## Figure 2-3: BA213 Configuration Worksheet

### RIGHT POWER SUPPLY

<table>
<thead>
<tr>
<th>SLOT</th>
<th>MODULE</th>
<th>Current (Amps)</th>
<th>Power (Watts)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>+5 Vdc</td>
<td>+12 Vdc</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**MASS STORAGE:**

- TK Drive
- FIXED DISK

Total these columns:

- Must not exceed: 33.0 A 7.6 A 230.0 W

### LEFT POWER SUPPLY

<table>
<thead>
<tr>
<th>SLOT</th>
<th>MODULE</th>
<th>Current (Amps)</th>
<th>Power (Watts)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>+5 Vdc</td>
<td>+12 Vdc</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**MASS STORAGE:**

- FIXED DISK(S)
  - 1.
  - 2.

Total these columns:

- Must not exceed: 33.0 A 7.6 A 230.0 W

MLD-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
- **DLEDRI** 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
- **CBTICR** 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>
<thead>
<tr>
<th>Breaks Enabled on H3600-SA</th>
<th>Power-up Halt¹</th>
<th>Halt Action²</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>T²</td>
<td>T</td>
<td>X</td>
<td>Diagnostics, halt</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>0</td>
<td>Halt</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>X</td>
<td>Diagnostics, bootstrap, halt</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>0</td>
<td>Restart, bootstrap, halt</td>
</tr>
<tr>
<td>X</td>
<td>F</td>
<td>1</td>
<td>Restart, halt</td>
</tr>
<tr>
<td>X</td>
<td>F</td>
<td>2</td>
<td>Bootstrap, halt</td>
</tr>
<tr>
<td>X</td>
<td>F</td>
<td>3</td>
<td>Halt</td>
</tr>
</tbody>
</table>

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 μsec. To determine if the BHALT line is asserted, the firmware steps out into halt-unprotected space after 9 msec. If the processor halts again, the firmware concludes that the halt was caused by the BHALT and not by the deassertion of DCOK.

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

<table>
<thead>
<tr>
<th>Mode</th>
<th>Language Not Previously Set¹</th>
<th>Language Previously Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language Inquiry</td>
<td>Prompt²</td>
<td>Prompt</td>
</tr>
<tr>
<td>Normal</td>
<td>Prompt</td>
<td>No Prompt</td>
</tr>
</tbody>
</table>

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 PR$_TODR
   R5 Boot flags
   R10 Halt PC value
   R11 Halt PSL value (without halt code and map enable)
   AP Halt code
   SP Base of 128-Kbyte good memory block + 512
   PC Base of 128-Kbyte good memory block + 512
   R1, R6, R7, R8, 0
   R9, FP

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RPB$V_CONV</td>
<td>Conversational boot. At various points in the system boot procedure, the bootstrap code solicits parameters and other input from the console terminal.</td>
</tr>
<tr>
<td>2</td>
<td>RPB$V_INITBPT</td>
<td>Initial breakpoint. If RPB$V_DEBUG is set, the VMS operating system executes a BPT instruction in module INIT immediately after enabling mapping.</td>
</tr>
<tr>
<td>3</td>
<td>RPB$V_BBLOCK</td>
<td>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.</td>
</tr>
<tr>
<td>4</td>
<td>RPB$V_DIAG</td>
<td>Diagnostic bootstrap. When set, the load image requested is [SYS0.SYSMAINT]DIAGBOOT.EXE.</td>
</tr>
<tr>
<td>5</td>
<td>RPB$V_BOOBPT</td>
<td>Bootstrap breakpoint. When set, a breakpoint instruction is executed in VMB and control is transferred to XDELT4 before booting.</td>
</tr>
<tr>
<td>6</td>
<td>RPB$V_HEADER</td>
<td>Image header. When set, VMB transfers control to the address specified by the file's image header. When not set, VMB transfers control to the first location of the load image.</td>
</tr>
<tr>
<td>8</td>
<td>RPB$V_SOLICT</td>
<td>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.</td>
</tr>
<tr>
<td>9</td>
<td>RPB$V_HALT</td>
<td>Halt before transfer. When set, VMB halts before transferring control to the application image.</td>
</tr>
<tr>
<td>31:28</td>
<td>RPB$V_TOPSYS</td>
<td>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...].</td>
</tr>
</tbody>
</table>

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

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

### 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.

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 [CTRLS] 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.

- **CTRLD**: Ignores transmissions to the console terminal until the next [CTRLD] 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].

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>0</td>
<td>R1</td>
<td>1</td>
</tr>
<tr>
<td>R2</td>
<td>2</td>
<td>R3</td>
<td>3</td>
</tr>
<tr>
<td>R4</td>
<td>4</td>
<td>R5</td>
<td>5</td>
</tr>
<tr>
<td>R6</td>
<td>6</td>
<td>R7</td>
<td>7</td>
</tr>
<tr>
<td>R8</td>
<td>8</td>
<td>R9</td>
<td>9</td>
</tr>
<tr>
<td>R10</td>
<td>0A</td>
<td>R11</td>
<td>0B</td>
</tr>
<tr>
<td>R12</td>
<td>0C</td>
<td>R13</td>
<td>0D</td>
</tr>
<tr>
<td>R14</td>
<td>0E</td>
<td>R15</td>
<td>0F</td>
</tr>
<tr>
<td>AP</td>
<td>0C</td>
<td>FP</td>
<td>0D</td>
</tr>
<tr>
<td>SP</td>
<td>0E</td>
<td>PC</td>
<td>0F</td>
</tr>
<tr>
<td>PSL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

## IPR Address Space (/I)

<table>
<thead>
<tr>
<th>pr$_ksp</th>
<th>00</th>
<th>pr$_esp</th>
<th>01</th>
</tr>
</thead>
<tbody>
<tr>
<td>pr$_ssp</td>
<td>02</td>
<td>pr$_usps</td>
<td>03</td>
</tr>
<tr>
<td>pr$_isp</td>
<td>04</td>
<td>pr$_p0br</td>
<td>08</td>
</tr>
<tr>
<td>pr$_p0lr</td>
<td>09</td>
<td>pr$_p1br</td>
<td>0A</td>
</tr>
<tr>
<td>pr$_p1lr</td>
<td>0B</td>
<td>pr$_sbr</td>
<td>0C</td>
</tr>
<tr>
<td>pr$_slr</td>
<td>0D</td>
<td>pr$_pcbb</td>
<td>10</td>
</tr>
<tr>
<td>pr$_scbb</td>
<td>11</td>
<td>pr$_ipl</td>
<td>12</td>
</tr>
<tr>
<td>pr$_astlv</td>
<td>13</td>
<td>pr$_sirr</td>
<td>14</td>
</tr>
<tr>
<td>pr$_sisr</td>
<td>15</td>
<td>pr$_icrr</td>
<td>18</td>
</tr>
<tr>
<td>pr$_nicr</td>
<td>19</td>
<td>pr$_icr</td>
<td>1A</td>
</tr>
<tr>
<td>pr$_todr</td>
<td>1B</td>
<td>pr$_rxcs</td>
<td>20</td>
</tr>
<tr>
<td>pr$_rxdb</td>
<td>21</td>
<td>pr$_txcs</td>
<td>22</td>
</tr>
<tr>
<td>pr$_txdb</td>
<td>23</td>
<td>pr$_tbdr</td>
<td>24</td>
</tr>
<tr>
<td>pr$_cadr</td>
<td>25</td>
<td>pr$_mcesr</td>
<td>26</td>
</tr>
<tr>
<td>pr$_msdr</td>
<td>27</td>
<td>pr$_savpc</td>
<td>2A</td>
</tr>
<tr>
<td>pr$_savpsl</td>
<td>2B</td>
<td>pr$_ioreset</td>
<td>37</td>
</tr>
<tr>
<td>pr$_mapen</td>
<td>38</td>
<td>pr$_tbia</td>
<td>39</td>
</tr>
<tr>
<td>pr$_tbis</td>
<td>3A</td>
<td>pr$_sid</td>
<td>3E</td>
</tr>
<tr>
<td>pr$_tbck</td>
<td>3F</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Symbol</td>
<td>Address</td>
<td>Symbol</td>
<td>Address</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>------------</td>
<td>----------</td>
</tr>
<tr>
<td>qbio</td>
<td>20000000</td>
<td>qbmem</td>
<td>30000000</td>
</tr>
<tr>
<td>qbmbbr</td>
<td>20080010</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>rom</td>
<td>20040000</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>cacr</td>
<td>20084000</td>
<td>bdr</td>
<td>20084004</td>
</tr>
<tr>
<td>dsccr</td>
<td>20080000</td>
<td>dser</td>
<td>20080004</td>
</tr>
<tr>
<td>dmeacr</td>
<td>20080008</td>
<td>dsear</td>
<td>2008000C</td>
</tr>
<tr>
<td>ipcr0</td>
<td>20001F40</td>
<td>ipcr1</td>
<td>20001F42</td>
</tr>
<tr>
<td>ipcr2</td>
<td>20001F44</td>
<td>ipcr3</td>
<td>20001F46</td>
</tr>
<tr>
<td>ssc_ram</td>
<td>20140400</td>
<td>ssc_cr</td>
<td>20140010</td>
</tr>
<tr>
<td>ssc_cdal</td>
<td>20140020</td>
<td>ssc_dledr</td>
<td>20140030</td>
</tr>
<tr>
<td>ssc_ad0mat</td>
<td>20140130</td>
<td>ssc_ad0msk</td>
<td>20140134</td>
</tr>
<tr>
<td>ssc_ad1mat</td>
<td>20140140</td>
<td>ssc_ad1msk</td>
<td>20140144</td>
</tr>
<tr>
<td>ssc_tcr0</td>
<td>20140100</td>
<td>ssc_tir0</td>
<td>20140104</td>
</tr>
<tr>
<td>ssc_tmir0</td>
<td>20140108</td>
<td>ssc_tivr0</td>
<td>2014010C</td>
</tr>
<tr>
<td>ssc_tcr1</td>
<td>20140110</td>
<td>ssc_tir1</td>
<td>20140114</td>
</tr>
<tr>
<td>ssc_tmir1</td>
<td>20140118</td>
<td>ssc_tivr1</td>
<td>2014011C</td>
</tr>
<tr>
<td>memcsr0</td>
<td>20080100</td>
<td>memcsr1</td>
<td>20080104</td>
</tr>
<tr>
<td>memcsr2</td>
<td>20080108</td>
<td>memcsr3</td>
<td>2008010C</td>
</tr>
<tr>
<td>memcsr4</td>
<td>20080110</td>
<td>memcsr5</td>
<td>20080114</td>
</tr>
<tr>
<td>memcsr6</td>
<td>20080118</td>
<td>memcsr7</td>
<td>2008011C</td>
</tr>
<tr>
<td>memcsr8</td>
<td>20080120</td>
<td>memcsr9</td>
<td>20080124</td>
</tr>
<tr>
<td>memcsr10</td>
<td>20080128</td>
<td>memcsr11</td>
<td>2008012C</td>
</tr>
<tr>
<td>memcsr12</td>
<td>20080130</td>
<td>memcsr13</td>
<td>20080134</td>
</tr>
<tr>
<td>memcsr14</td>
<td>20080138</td>
<td>memcsr15</td>
<td>2008013C</td>
</tr>
<tr>
<td>memcsr16</td>
<td>20080140</td>
<td>memcsr17</td>
<td>20080144</td>
</tr>
</tbody>
</table>

3–16 KA655 CPU System Maintenance
Table 3–6 lists symbolic addresses that you can use in any address space.

Table 3–6: Symbolic Addresses Used in Any Address Space

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>The location last referenced in an EXAMINE or DEPOSIT command.</td>
</tr>
<tr>
<td>+</td>
<td>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.</td>
</tr>
<tr>
<td>−</td>
<td>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.</td>
</tr>
<tr>
<td>®</td>
<td>The location addressed by the last location referenced in an EXAMINE or DEPOSIT command.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Qualifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Control</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>/B</td>
<td>The data size is byte.</td>
</tr>
<tr>
<td>/W</td>
<td>The data size is word.</td>
</tr>
<tr>
<td>/L</td>
<td>The data size is longword.</td>
</tr>
<tr>
<td>/Q</td>
<td>The data size is quadword.</td>
</tr>
<tr>
<td>/N:(count)</td>
<td>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.</td>
</tr>
<tr>
<td>/STEP:(size)</td>
<td>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.</td>
</tr>
<tr>
<td>/WRONG</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Address Space Control</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>/G</td>
<td>General purpose register (GPR) address space, R0–R15. The data size is always longword.</td>
</tr>
<tr>
<td>/I</td>
<td>Internal processor register (IPR) address space. Accessible only by the MTPR and MPFR instructions. The data size is always longword.</td>
</tr>
<tr>
<td>/V</td>
<td>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&lt;M&gt; 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.</td>
</tr>
<tr>
<td>/P</td>
<td>Physical memory address space.</td>
</tr>
<tr>
<td>/M</td>
<td>Processor status longword (PSL) address space. The data size is always longword.</td>
</tr>
<tr>
<td>/U</td>
<td>Access to console private memory is allowed. This qualifier also disables virtual address protection checks. On virtual address writes, the PTE&lt;M&gt; bit is not set if the /U qualifier is present. This qualifier is not inherited; it must be respecified on each command.</td>
</tr>
</tbody>
</table>
### 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

<table>
<thead>
<tr>
<th>Processor Control</th>
<th>Data Transfer</th>
<th>Console Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOT</td>
<td>EXAMINE</td>
<td>CONFIGURE</td>
</tr>
<tr>
<td>CONTINUE</td>
<td>DEPOSIT</td>
<td>FIND</td>
</tr>
<tr>
<td>HALT</td>
<td>MOVE</td>
<td>REPEAT</td>
</tr>
<tr>
<td>INITIALIZE</td>
<td>SEARCH</td>
<td>SET</td>
</tr>
<tr>
<td>NEXT</td>
<td>X</td>
<td>SHOW</td>
</tr>
<tr>
<td>START</td>
<td></td>
<td>TEST</td>
</tr>
<tr>
<td>UNJAM</td>
<td></td>
<td>!</td>
</tr>
</tbody>
</table>

#### Table 3–9: Console Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Qualifiers</th>
<th>Argument</th>
<th>Other(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOT</td>
<td>/R5:[bitmap]/[bitmap]</td>
<td>[device_name]</td>
<td>—</td>
</tr>
<tr>
<td>CONFIGURE</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>CONTINUE</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>DEPOSIT</td>
<td>/B /N /L /Q /G /I /V /P /M /U /N:[count] /STEP:[size] /WRONG</td>
<td>[address]</td>
<td>[data] [data]</td>
</tr>
<tr>
<td>EXAMINE</td>
<td>/B /N /L /Q /G /I /V /P /M /U /N:[count] /STEP:[size] /WRONG/INSTRUCTION</td>
<td>[[address]]</td>
<td>—</td>
</tr>
<tr>
<td>FIND</td>
<td>/MEM /RPB</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>HALT</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>HELP</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>INITIALIZE</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Command</td>
<td>Qualifiers</td>
<td>Argument</td>
<td>Other(s)</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------</td>
<td>---------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>MOVE</td>
<td>/B /W /L /Q</td>
<td>(src_address)</td>
<td>(dest_address)</td>
</tr>
<tr>
<td></td>
<td>/N /P /U</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>/N:(count) /STEP:(size)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>/WRONG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEXT</td>
<td>–</td>
<td>([count])</td>
<td>–</td>
</tr>
<tr>
<td>REPEAT</td>
<td>–</td>
<td>{command}</td>
<td>–</td>
</tr>
<tr>
<td>SEARCH</td>
<td>/B /W /L /Q</td>
<td>(start_address)</td>
<td>(pattern) [(mask)]</td>
</tr>
<tr>
<td></td>
<td>/N /P /U</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>/N:(count) /STEP:(size)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>/WRONG/NOT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SET BFLAG</td>
<td>–</td>
<td>(bitmap)</td>
<td>–</td>
</tr>
<tr>
<td>SET BOOT</td>
<td>–</td>
<td>{device_string}</td>
<td>–</td>
</tr>
<tr>
<td>SET HOST</td>
<td>/DUP /UQSSP</td>
<td>(node) n</td>
<td>([task])</td>
</tr>
<tr>
<td></td>
<td>/DISK n</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>/TAPE n csr_address</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>/MAINTENANCE /UQSSP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>/SERVICE n csr_address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SET LANGUAGE</td>
<td>–</td>
<td>{language_type}</td>
<td>–</td>
</tr>
<tr>
<td>SHOW BFLAG</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>SHOW BOOT</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>SHOW DEVICE</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>SHOW ETHERNET</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>SHOW LANGUAGE</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>SHOW MEMORY</td>
<td>/FULL</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>SHOW QBUS</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>SHOW RLV12</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>SHOW UQSSP</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>SHOW VERSION</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>START</td>
<td>–</td>
<td>{address}</td>
<td>–</td>
</tr>
<tr>
<td>TEST</td>
<td>–</td>
<td>{test_number}</td>
<td>([parameters])</td>
</tr>
<tr>
<td>UNJAM</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>X</td>
<td>–</td>
<td>{address}</td>
<td>{count}</td>
</tr>
</tbody>
</table>
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:

```bash
>>> 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  DEQVA  DESQA  RQDX3  KDA50
RRD50  RQC25  KFQSA-DISK  TQK50  TQK70  TUB8E
RV20  KFQSA-TAPE  KMV11  IEQ11  DHQ11  HV11
CXA16  CXB16  CXY08  VCB01  QVSS  LNV11
LNV21  QPSS  DSV11  ADV11C  AAV11C  AXV11C
KWV11C  ADV11D  AAV11D  VCB02  QDSS  DRV11J
DRQ3B  VSV21  IBQ01  IDV11A  IDV11B  IDV11C
IDV11D  IAV11A  IAV11B  MIRA  ADQ32  DTC04
DESNA  IGQ11
Numbers:
   1 to 255, default is 1
Device,Number? rqdx3,2
Device,Number? dhv11
Device,Number? qdss
Device,Number? tqqk50
Device,Number? tqqk70
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, /N, /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, /N, /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
  G 00000000F FFFFFFFC
  ! Examine the PC.

>>> ex sp
  G 00000000E 00002000
  ! Examine the SP.

>>> ex psl
  M 00000000 041F0000
  ! Examine the PSL.

>>> e/m
  M 00000000 041F0000
  ! Examine PSL another way.

>>> e r4/n:5
  G 00000004 00000000
  G 00000005 00000000
  G 00000006 00000000
  G 00000007 00000000
  G 00000008 00000000
  G 00000009 801D9000
  ! Examine R4 through R9.
>>> ex pr$ scbb
  I 00000011 2004A000
     ! Examine the SCBB, IPR 17
     ! (decimal).

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

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

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

>>> e/ins
  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 00000000E 00000000
>>> find /mem                 ! Look for a valid 128 Kbytes.
>>> ex sp                     ! Note where it was found.
   G 00000000E 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 PATTERNN [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 RLIPv12
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:

<table>
<thead>
<tr>
<th>Register</th>
<th>State at Initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSL</td>
<td>041F0000</td>
</tr>
<tr>
<td>IPL</td>
<td>1F</td>
</tr>
<tr>
<td>ASTLVL</td>
<td>4</td>
</tr>
<tr>
<td>SISR</td>
<td>0</td>
</tr>
<tr>
<td>ICCS</td>
<td>Bits &lt;6&gt; and &lt;0&gt; clear; the rest are unpredictable</td>
</tr>
<tr>
<td>RXCS</td>
<td>0</td>
</tr>
<tr>
<td>TXCS</td>
<td>80</td>
</tr>
<tr>
<td>MAPEN</td>
<td>0</td>
</tr>
<tr>
<td>Caches</td>
<td>Flushed</td>
</tr>
<tr>
<td>Instruction buffer</td>
<td>Unaffected</td>
</tr>
<tr>
<td>Console previous reference</td>
<td>Longword, physical, address 0</td>
</tr>
<tr>
<td>TODR</td>
<td>Unaffected</td>
</tr>
<tr>
<td>Main memory</td>
<td>Unaffected</td>
</tr>
<tr>
<td>General registers</td>
<td>Unaffected</td>
</tr>
<tr>
<td>Halt code</td>
<td>Unaffected</td>
</tr>
<tr>
<td>Bootstrap-in-progress flag</td>
<td>Unaffected</td>
</tr>
<tr>
<td>Internal restart-in-progress flag</td>
<td>Unaffected</td>
</tr>
</tbody>
</table>

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:

```plaintext
>>> 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
P 00000000 00000000
P 00000004 00000000
P 00000008 00000000
P 0000000C 00000000
P 00000010 00000000

>>> ex/n:4 200
P 00000200 58DD0520
P 00000204 585E04C1
P 00000208 00FF8FB0
P 0000020C 5208A8D0
P 00000210 540CA8DE

>>> mov/n:4 200 0
P 00000200 58DD0520
P 00000204 585E04C1
P 00000208 00FF8FB0
P 0000020C 5208A8D0
P 00000210 540CA8DE

! Observe destination.

! Observe source data.

! Move the data.
>>> ex/n:4 0
P 00000000 58DD0520
P 00000004 585E04C1
P 00000008 00FF8FBB
P 0000000C 5208A9D0
P 00000010 540CA8DE

! Observe moved data.
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
    P 00001002  D6  INCL  R0

3-36 KA655 CPU System Maintenance
>>> n
P 00001004  D1 CMPL   S^05,R0
>>> n
P 00001007  12 BNEQ   00001002
>>> n
P 00001002  D6 INCL   R0
>>> n 5
P 00001004  D1 CMPL   S^05,R0
P 00001007  12 BNEQ   00001002
P 00001002  D6 INCL   R0
P 00001004  D1 CMPL   S^05,R0
P 00001007  12 BNEQ   00001002
>>> n 7
P 00001002  D6 INCL   R0
P 00001004  D1 CMPL   S^05,R0
P 00001007  12 BNEQ   00001002
P 00001002  D6 INCL   R0
P 00001004  D1 CMPL   S^05,R0
P 00001007  12 BNEQ   00001002
P 00001009  11 BRB    00001009
>>> n
P 00001009  11 BRB    00001009

! ...or multiple step
! the program.
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:

```plaintext
>>> 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
```

3–38  KA655 CPU System Maintenance
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:

<table>
<thead>
<tr>
<th>/NOT Qualifier</th>
<th>Match Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absent</td>
<td>True</td>
<td>Report address</td>
</tr>
<tr>
<td>Absent</td>
<td>False</td>
<td>No report</td>
</tr>
<tr>
<td>Present</td>
<td>True</td>
<td>No report</td>
</tr>
<tr>
<td>Present</td>
<td>False</td>
<td>Report address</td>
</tr>
</tbody>
</table>

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:

```plaintext
>>> dep /p/1/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
   ! of 12345678 on any byte
   ! boundary. Then try on
   ! longword boundaries.
   ! Search for all non-zero
   ! longwords.
>>> search /n:1000 /not 0 0
   ! Search for odd-numbered
   ! longwords on any boundary.
   !
   !
   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 all occurrences
   ! of the byte 12.
   !
   !
   P 00000502 87654321
   P 00000503 00876543
   P 00000504 00008765
   P 00000505 00000087
>>> search /n:1000 /b 0 12
   ! Search for all words that
   ! could be interpreted as
   ! a spin (10$: brb 10$).
   !
   !
   P 00000303 12
   P 00000404 12
>>> search /n:1000 /st:1 /w 0 FE11
   ! 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 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 Space
-000000DC (760334)=0000 (300) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-000000DE (760336)=0AA0
-000000E0 (760340)=0000 (304) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-000000E2 (760342)=0AA0
-000000E4 (760344)=0000 (310) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-000000E6 (760346)=0AA0
-000000E8 (760350)=0000 (314) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-000000EA (760352)=0AA0
-0001468 (772150)=0000 (154) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-000146A (772152)=0AA0
-0001920 (774440)=FF08 (120) DELQA/DEQNA/DESQA
-0001922 (774442)=FF00
-0001924 (774444)=FF2B
-0001926 (774446)=FF09
-0001928 (774450)=FFA3
-000192A (774452)=FF96
-000192C (774454)=0050
-000192E (774456)=1030
-0001940 (774500)=0000 (260) TQK50/TQK70/TU81E/RV20/KFQSA-TAPE
-0001942 (774502)=0BC0
-0001F40 (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 <NODE> <CSR_ADDRESS> <MODEL>
- CLEAR <NODE>
- SHOW
- HELP
- EXIT
- QUIT

Parameters:

- <NODE>
- <CSR_ADDRESS>
- <MODEL>

? set 6 /KFQSA
? show

<table>
<thead>
<tr>
<th>Node</th>
<th>CSR Address</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>772150</td>
<td>21</td>
</tr>
<tr>
<td>1</td>
<td>760334</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>760340</td>
<td>21</td>
</tr>
<tr>
<td>5</td>
<td>760344</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

? 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: Displays main memory configuration board by board.
  
  /FULL: Additionally, displays the normally inaccessible areas of memory, such as the PFN bitmap pages, the console scratch memory pages, the Q22-bus scatter-gather map pages. Also reports the addresses of bad pages, as defined by the bitmap.
- QBUS: Displays all Q22-bus I/O addresses that respond to an aligned word read, and vector and device name information. For each address, the console displays the address in the VAX I/O space in hex, the address as it would appear in the Q22-bus I/O space in octal, and the word data that was read in hex.

  This command may take several minutes to complete. Press [CTRLC] to terminate the command. During execution, the command disables the scatter-gather map.
- RLV12: Displays all RL01 and RL02 disks that appear on the Q22-bus.
- UQSSP: Displays the status of all disks and tapes that can be found on the Q22-bus that support the UQSSP protocol. For each such disk or tape on the Q22-bus, the firmware displays the controller number, the controller CSR address, and the boot name and type of each device connected to the controller. The command does not indicate whether the device contains a bootable image.

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

*Qualifiers:* Listed in the parameter descriptions above.

3–44 KA655 CPU System Maintenance
Examples:

>>> show bflag
  00000220
>>> show boot
  DUAA0
>>> 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)
  -MUAA0 (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
-2000000C (760334) = 0000 (300) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-2000000E (760336) = 0AA0
-2000000E0 (760340) = 0000 (304) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-2000000E2 (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 (CTRLC, CTRL/S, CTRL0, 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.
>>>
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.

---

4–2 KA655 CPU System Maintenance
• 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

<table>
<thead>
<tr>
<th>Test</th>
<th>Address1</th>
<th>Name</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004BC00</td>
<td>De_SCB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C6</td>
<td>2004D2F0</td>
<td>SSC_powerup</td>
<td>*********</td>
</tr>
<tr>
<td>C7</td>
<td>2004D3B2</td>
<td>CBTCR timeout</td>
<td>***</td>
</tr>
<tr>
<td>34</td>
<td>2004D4C5</td>
<td>ROM logic test</td>
<td>*</td>
</tr>
<tr>
<td>33</td>
<td>2004D544</td>
<td>CMCTL_powerup</td>
<td>*</td>
</tr>
<tr>
<td>32</td>
<td>2004D57C</td>
<td>CMCTLregs</td>
<td>MEMCSR0_addr *********</td>
</tr>
<tr>
<td>91</td>
<td>2004D6A0</td>
<td>CQBIC_powerup</td>
<td>**</td>
</tr>
<tr>
<td>90</td>
<td>2004D730</td>
<td>CQBIC_regs</td>
<td>*</td>
</tr>
<tr>
<td>80</td>
<td>2004D7B2</td>
<td>CQBIC-memory</td>
<td>*********</td>
</tr>
<tr>
<td>60</td>
<td>2004DDD9</td>
<td>Console serial</td>
<td>start_baud end_baud ********</td>
</tr>
<tr>
<td>62</td>
<td>2004E128</td>
<td>Console QDSS</td>
<td>mark_not_present selftest_r0 selftest_r1 *****</td>
</tr>
<tr>
<td>63</td>
<td>2004E2A8</td>
<td>QDSS self-test</td>
<td>input_csr selftest_r0 selftest_r1 *****</td>
</tr>
<tr>
<td>51</td>
<td>2004E40E</td>
<td>CFPA</td>
<td>*****</td>
</tr>
<tr>
<td>52</td>
<td>2004E5FA</td>
<td>Prog timer</td>
<td>which_timer wait_time_us ***</td>
</tr>
<tr>
<td>53</td>
<td>2004E8C0</td>
<td>TOY clock</td>
<td>repeat_count_250ms_ea *****</td>
</tr>
<tr>
<td>55</td>
<td>2004E6B5</td>
<td>Interval timer</td>
<td>*</td>
</tr>
<tr>
<td>5A</td>
<td>2004EAE0</td>
<td>VAX CMCTL CDAL</td>
<td>dont_report_memory_bad repeat_count *</td>
</tr>
<tr>
<td>45</td>
<td>2004EBF0</td>
<td>cache_mem_cqbiq</td>
<td>start_addr end_addr addr_incr ****</td>
</tr>
<tr>
<td>46</td>
<td>2004ED8</td>
<td>Cache1_diag_md</td>
<td>addr_incr *********</td>
</tr>
<tr>
<td>9E</td>
<td>2004F502</td>
<td>List diags</td>
<td>*</td>
</tr>
<tr>
<td>81</td>
<td>2004F528</td>
<td>MSCP-QBUS test</td>
<td>IP_csr ********</td>
</tr>
<tr>
<td>82</td>
<td>2004F6EA</td>
<td>DELQA</td>
<td>device_num_addr ****</td>
</tr>
<tr>
<td>C1</td>
<td>2004F8C5</td>
<td>SSC RAM</td>
<td>*</td>
</tr>
<tr>
<td>C2</td>
<td>2004FA8C</td>
<td>SSC RAM ALL</td>
<td>*</td>
</tr>
<tr>
<td>C5</td>
<td>2004FBFS</td>
<td>SSC regs</td>
<td>*</td>
</tr>
<tr>
<td>54</td>
<td>2004FCE9</td>
<td>Virtual mode</td>
<td>****</td>
</tr>
<tr>
<td>36</td>
<td>2004FF68</td>
<td>Cache2_memory</td>
<td>start_addr end_addr addr_incr ****</td>
</tr>
<tr>
<td>35</td>
<td>20050D4C</td>
<td>Cache2 integrity</td>
<td>start_addr end_addr addr_incr ****</td>
</tr>
<tr>
<td>44</td>
<td>20050CF4</td>
<td>Cache_memory</td>
<td>addr_incr *********</td>
</tr>
<tr>
<td>43</td>
<td>20050D4D</td>
<td>Cache1_cache2</td>
<td>addr_incr *********</td>
</tr>
<tr>
<td>41</td>
<td>2005107C</td>
<td>Board reset</td>
<td>***</td>
</tr>
<tr>
<td>42</td>
<td>20051269</td>
<td>Check_for_intrs</td>
<td>***</td>
</tr>
<tr>
<td>31</td>
<td>200512AC</td>
<td>MEM_setup_CSRs</td>
<td>*********</td>
</tr>
</tbody>
</table>

1V5.3 addresses; use as examples only.

4-4 KA655 CPU System Maintenance
Table 4–1 (Cont.):  Test and Utility Numbers

<table>
<thead>
<tr>
<th>Test</th>
<th>Address</th>
<th>Name</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>200518CF</td>
<td>MEM_bitmap</td>
<td>*** mark_Hard_SBEs ******</td>
</tr>
<tr>
<td>4F</td>
<td>20051D0A</td>
<td>MEM_data</td>
<td>start_add end_add add_incr cont_on_err ******</td>
</tr>
<tr>
<td>4E</td>
<td>20051EC8</td>
<td>MEM_byte</td>
<td>start_add end_add add_incr cont_on_err ******</td>
</tr>
<tr>
<td>4D</td>
<td>20051FDD</td>
<td>MEM_address</td>
<td>start_add end_add add_incr cont_on_err ******</td>
</tr>
<tr>
<td>4C</td>
<td>2005216F</td>
<td>MEM_ECC_error</td>
<td>start_add end_add add_incr cont_on_err ******</td>
</tr>
<tr>
<td>4B</td>
<td>20052630</td>
<td>MEM_masked_errs</td>
<td>start_add end_add add_incr cont_on_err ******</td>
</tr>
<tr>
<td>4A</td>
<td>2005280A</td>
<td>MEM_correction</td>
<td>start_add end_add add_incr cont_on_err ******</td>
</tr>
<tr>
<td>49</td>
<td>20052A1D</td>
<td>MEM_FDM_logic</td>
<td>*** cont_on_err ******</td>
</tr>
<tr>
<td>48</td>
<td>20052FEC</td>
<td>MEM_addr_shrts</td>
<td>start_add end_add * cont_on_err pat1 pat2 ******</td>
</tr>
<tr>
<td>47</td>
<td>20053643</td>
<td>MEM_refresh</td>
<td>start_end incr cont_on_err time_seconds ******</td>
</tr>
<tr>
<td>40</td>
<td>200537CE</td>
<td>MEM_count_errs</td>
<td>First_board Last_board ****** Soft_errs_allowed</td>
</tr>
<tr>
<td>9D</td>
<td>20053B14</td>
<td>Utilities</td>
<td>Expnd_err_msg get_mode init_LEDs clr_ps_cnt</td>
</tr>
<tr>
<td>9C</td>
<td>20053C18</td>
<td>List CPU regs</td>
<td>*</td>
</tr>
<tr>
<td>9F</td>
<td>200541D4</td>
<td>Create script</td>
<td>*</td>
</tr>
</tbody>
</table>

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 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**

<table>
<thead>
<tr>
<th>Script&lt;sup&gt;1&lt;/sup&gt;</th>
<th>Enter with TEST Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>A0</td>
<td>Soft script created by de_test9f. Also referred to as user script. Enter T 9F to create.</td>
</tr>
<tr>
<td>A1</td>
<td>A1, AA, AB, AC, 0, 3</td>
<td>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.</td>
</tr>
<tr>
<td>A7</td>
<td>A7, A8</td>
<td>Memory test portion invoked by script A8. Reruns the memory tests without rebuilding and reinitializing the bitmap. Runs script A8 once before running script A7 separately to allow mapping out of both single-bit and double-bit main memory ECC errors.</td>
</tr>
<tr>
<td>A8</td>
<td>A8</td>
<td>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.</td>
</tr>
<tr>
<td>A9</td>
<td>A9</td>
<td>Memory tests. Halts and reports the first error. Does not reset the bitmap or busmap.</td>
</tr>
<tr>
<td>AA</td>
<td>AA, 0</td>
<td>Console SLU. Invokes scripts BA, BC, and A1. Does not invoke any tests directly.</td>
</tr>
<tr>
<td>AD</td>
<td>AD</td>
<td>Console program. Runs memory tests, marks bitmap, resets busmap, and resets caches. Calls script AE.</td>
</tr>
<tr>
<td>AE</td>
<td>AE, AD</td>
<td>Console program. Resets memory CSRs and resets caches. Also called by the INIT command.</td>
</tr>
<tr>
<td>AF</td>
<td>AF</td>
<td>Console program. Resets busmap and resets caches.</td>
</tr>
<tr>
<td>BA</td>
<td>BA, 2, AA</td>
<td>Initial power-up script for console SLU before first console announcement. Invoked at power-up.</td>
</tr>
<tr>
<td>BC</td>
<td>BC, AA, AC, 0, 3</td>
<td>Called by scripts AA and AC. Provides console announcements. Invoked at power-up.</td>
</tr>
<tr>
<td>BD</td>
<td>BD, A1, AA, AB, AC, 0, 3</td>
<td>Common section of power-up script. Script A1 invokes this script at power-up.</td>
</tr>
</tbody>
</table>

<sup>1</sup>Scripts A2–A6, B0–B3, and B5 are for manufacturing use. They should not be used by Field Service. Scripts AB and BB are used to test the QDSS, which is not supported. 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**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Automatically invokes the proper scripts; runs the same tests as during power-up.</td>
</tr>
<tr>
<td>A9</td>
<td>Primarily runs the memory tests; halts upon first hard or soft error.</td>
</tr>
<tr>
<td>A8</td>
<td>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.</td>
</tr>
<tr>
<td>A7</td>
<td>Can be run by itself; useful when you want to bypass the bitmap test.</td>
</tr>
<tr>
<td>A1</td>
<td>Power-up script that can be run by itself. Bypasses the bitmap test.</td>
</tr>
</tbody>
</table>

### 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.

   a. DE executes Script AC.
      
      Script AC directs DE to execute scripts BC and A1.
      
      – Script BC directs DE to execute tests (console announce-
        ments 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 announce-
        ments are off).

      – Script BC directs DE to execute tests (console announce-
        ments 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 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

```plaintext
>>>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
CPFA  >>Run from ?[0=ROM,1=DIAG_RAM,2=RAM,3=fastest possible] (0):
CPFA  >>Repeat? [0=no,1=on error,2=forever,>2=count<FF] (0):
CPFA  >>Error severity ? [0,1,2,3] (2):
CPFA  >>Console error report? [0=none,1=full] (1):
CPFA  >>Stop script on error? [0=NO,1=YES] (1):
CPFA  >>LED on entry (05):
CPFA  >>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 [CTRLC] 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
    2004D2F0  SSC_powerup          *******
    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       2004DD99  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       2004ED8  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       2004FC99  Virtual mode ******
    36       2004FF68  cache2_memory start_addr end_addr addr_incr *******
    35       200504DC  Cach2_integrity 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 20052FDD 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 20052FE0 MEM_Addr_shrts start_add end_add * cont_on_err pat2 pat3 ****
47 20003643 MEM_Refresh start end incr cont_on_err time_seconds *****
40 2000537CE MEM_Count_Errs First_board Last_board Soft_errs_allowed *****
3D 20005B14 Utilities Expnd_err_msg get_mode init_LEDs clr_ps_cnt
3C 20005B18 List MPU regs *
3F 2000541D4 Create script *****

24 bytes left
Next test number: 40

MEM_Count_Errs>>Run from ?[0=ROM,1=DIAG_RAM,2=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 :

>> TA0
^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).

4–14  KA655 CPU System Maintenance
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=AAAAAAAA
r5=00000080  r6=01EF0000  r7=20080144  r8=00010000  ERF=20140770

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

<table>
<thead>
<tr>
<th>Test</th>
<th>Severity</th>
<th>Error</th>
<th>De_error</th>
<th>Vector</th>
<th>Count</th>
</tr>
</thead>
</table>

- **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 autoboote. An error of severity level 1 causes a display of the first line of the error printout, but does not interrupt an autoboote. 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

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

Table 4–5: Values Saved, Exception During Executive

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

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**

![LED Diagram](image-url)
### Table 4-6: KA655 Console Displays and FRUs

<table>
<thead>
<tr>
<th>Hex LED</th>
<th>Normal Console Display</th>
<th>Error Console Display</th>
<th>Default on Error</th>
<th>Description</th>
<th>FRU(^1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Power-Up Tests</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>None</td>
<td>None</td>
<td>Loop on self</td>
<td>Power-up</td>
<td>6, 1, 4, 5(^2,) 3</td>
</tr>
<tr>
<td>D</td>
<td>None</td>
<td>None</td>
<td>Loop on self</td>
<td>WAIT_POK</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>None</td>
<td>None</td>
<td>Loop on self</td>
<td>Entering IPT</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>None</td>
<td>None</td>
<td>Loop on self</td>
<td>SLU_EXT_LOOPBACK(^4)</td>
<td>7, 1</td>
</tr>
</tbody>
</table>

**Script AA**

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

**Script BA**

<table>
<thead>
<tr>
<th>C</th>
<th>None</th>
<th>?9D</th>
<th>Continue</th>
<th>Utilities</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>None</td>
<td>?42</td>
<td>Continue</td>
<td>Check_for_intrs</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>None</td>
<td>?C6</td>
<td>Continue</td>
<td>SSC_powerup</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>None</td>
<td>?60</td>
<td>Continue</td>
<td>CONSOLE_SERIAL</td>
<td>1</td>
</tr>
</tbody>
</table>

End of script

**Script AC**

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

1\(^{\text{FRU key:}}\)

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

\(^{2}\text{In the case of multiple FRUs, refer to Section 4.5.2 for further information.}\)

\(^{3}\text{If a problem recurs with the same FRU, check that the tolerance for system power supply +5 Vdc, +12 Vdc, and AC ripple are within specification.}\)

\(^{4}\text{This test only runs if the power-up mode switch on the H3600–SA is set to TEST mode. See Section 4.6.1.}\)
# Table 4–6 (Cont.): KA655 Console Displays and FRUs

<table>
<thead>
<tr>
<th>Hex LED</th>
<th>Normal Console Display</th>
<th>Error Console Display</th>
<th>Default on Error</th>
<th>Description</th>
<th>FRU$^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Script BC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>40</td>
<td>?91</td>
<td>Continue</td>
<td>CQBIC_power-up</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>39</td>
<td>?90</td>
<td>Continue</td>
<td>CQBIC_registers</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>38</td>
<td>?33</td>
<td>Continue</td>
<td>CMCTL_power-up</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>37</td>
<td>?32</td>
<td>Continue</td>
<td>CMCTL_registers</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>36</td>
<td>?31</td>
<td>Continue</td>
<td>CMCTL_setup_CSRs</td>
<td>1, 2, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>35</td>
<td>?49</td>
<td>Continue</td>
<td>MEMORY_FDM_logic</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>34</td>
<td>?30</td>
<td>Halt</td>
<td>MEMORY_bitmap</td>
<td>1, 2, 3, 5</td>
</tr>
</tbody>
</table>

End of script.

---

Script A1

Invoke script BD.

---

# Script BD

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Script BD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>33</td>
<td>?52</td>
<td>Continue</td>
<td>PROG_TIMER_0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>32</td>
<td>?52</td>
<td>Continue</td>
<td>PROG_TIMER_1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>31</td>
<td>?53</td>
<td>Continue</td>
<td>TOY_CLOCK</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>30</td>
<td>?C1</td>
<td>Continue</td>
<td>SSC_RAM</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>29</td>
<td>?34</td>
<td>Continue</td>
<td>ROM_logic</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>28</td>
<td>?C5</td>
<td>Continue</td>
<td>SSC_registers</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>27</td>
<td>?55</td>
<td>Continue</td>
<td>INTERVAL_TIMER</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>26</td>
<td>?51</td>
<td>Continue</td>
<td>CFPA</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>25</td>
<td>?C7</td>
<td>Continue</td>
<td>CBTCR_timeout</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>24</td>
<td>?46</td>
<td>Continue</td>
<td>CACHE_DIAG_MODE</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>23</td>
<td>?35</td>
<td>Continue</td>
<td>Cache2_integrity</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>22</td>
<td>?C2</td>
<td>Continue</td>
<td>SSC_RAM_ALL</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>21</td>
<td>?43</td>
<td>Continue</td>
<td>Cache1_cache2</td>
<td>1</td>
</tr>
</tbody>
</table>

$^1$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

<table>
<thead>
<tr>
<th>Hex LED</th>
<th>Normal Error Console Display</th>
<th>Error Console Display</th>
<th>Default on Error</th>
<th>Description</th>
<th>FRU¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>Script BD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>?4F</td>
<td>Continue</td>
<td>MEMORY_data</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>19</td>
<td>?4E</td>
<td>Continue</td>
<td>MEMORY_byte</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>18</td>
<td>?4D</td>
<td>Continue</td>
<td>MEMORY_addr</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>17</td>
<td>?4C</td>
<td>Continue</td>
<td>MEMORY_ECC_error</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>?4B</td>
<td>Continue</td>
<td>MEMORY_masked_errors</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>?4A</td>
<td>Continue</td>
<td>MEMORY_correction</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>14</td>
<td>?48</td>
<td>Continue</td>
<td>MEMORY_address_shorts</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td>?47</td>
<td>Continue</td>
<td>MEMORY_refresh</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>?40</td>
<td>Continue</td>
<td>MEMORY_count_errors</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>B</td>
<td>11</td>
<td>?44</td>
<td>Continue</td>
<td>CACHE1_MEMORY</td>
<td>1, 2, 3, 5</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>?36</td>
<td>Continue</td>
<td>Cache2_memory</td>
<td>1, 2, 3, 5</td>
</tr>
<tr>
<td>7</td>
<td>09</td>
<td>?80</td>
<td>Continue</td>
<td>CQBIC_MEMORY</td>
<td>1, 2, 3, 5</td>
</tr>
<tr>
<td>B</td>
<td>08</td>
<td>?54</td>
<td>Continue</td>
<td>VIRTUAL_MODE</td>
<td>1, 2, 3, 5</td>
</tr>
<tr>
<td>B</td>
<td>07</td>
<td>?55</td>
<td>Continue</td>
<td>SSC_registers</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>06</td>
<td>?34</td>
<td>Continue</td>
<td>ROM logic test</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>05</td>
<td>?45</td>
<td>Continue</td>
<td>CACHE_MEM_CQBIC</td>
<td>1, 2, 4, 3, 5</td>
</tr>
<tr>
<td>9</td>
<td>04</td>
<td>?5A</td>
<td>Continue</td>
<td>CVAX CMCTL CDAL</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>03</td>
<td>?41</td>
<td>Continue</td>
<td>Board reset</td>
<td>1, 4</td>
</tr>
</tbody>
</table>

End of script.

Script A8

<table>
<thead>
<tr>
<th>Hex LED</th>
<th>Normal Error Console Display</th>
<th>Error Console Display</th>
<th>Default on Error</th>
<th>Description</th>
<th>FRU¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>31</td>
<td>?31</td>
<td>Halt</td>
<td>CMCTRL_setup_CSRS</td>
<td>1, 2, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>49</td>
<td>?49</td>
<td>Halt</td>
<td>MEMORY_FDM_logic</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>30</td>
<td>?30</td>
<td>Halt</td>
<td>MEMORY_bitmap</td>
<td>1, 2, 3, 5</td>
</tr>
</tbody>
</table>

Invoke script A7.
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

Troubleshooting and Diagnostics 4-21
Table 4–6 (Cont.): KA655 Console Displays and FRUs

<table>
<thead>
<tr>
<th>Hex LED</th>
<th>Normal Console Display</th>
<th>Error Console Display</th>
<th>Default on Error</th>
<th>Description</th>
<th>FRU¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>Script A7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4F</td>
<td>?4F</td>
<td>Halt</td>
<td>MEMORY_data</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>4E</td>
<td>?4E</td>
<td>Halt</td>
<td>MEMORY_byte</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>4D</td>
<td>?4D</td>
<td>Halt</td>
<td>MEMORY_addr</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>4C</td>
<td>?4C</td>
<td>Halt</td>
<td>MEMORY_ECC_error</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>4B</td>
<td>?4B</td>
<td>Halt</td>
<td>MEMORY_masked_errors</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>4A</td>
<td>?4A</td>
<td>Halt</td>
<td>MEMORY_correction</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>48</td>
<td>?48</td>
<td>Halt</td>
<td>MEMORY_address_shorts</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>47</td>
<td>?47</td>
<td>Halt</td>
<td>MEMORY_refresh</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>40</td>
<td>?40</td>
<td>Cont</td>
<td>MEMORY_count_errors</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>7</td>
<td>80</td>
<td>?80</td>
<td>Cont</td>
<td>CQBIC_memory</td>
<td>1, 2, 4, 3, 5</td>
</tr>
<tr>
<td>C</td>
<td>41</td>
<td>?41</td>
<td>Halt</td>
<td>Board reset</td>
<td>1, 4</td>
</tr>
</tbody>
</table>

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

4–22 KA655 CPU System Maintenance
<table>
<thead>
<tr>
<th>Hex LED</th>
<th>Normal Console Display</th>
<th>Error Console Display</th>
<th>Default on Error</th>
<th>Description</th>
<th>FRU¹</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Script AD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>None</td>
<td>?30</td>
<td>Continue</td>
<td>MEM_bitmap</td>
<td>1, 2, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>None</td>
<td>?4F</td>
<td>Continue</td>
<td>MEM_data</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>None</td>
<td>?4E</td>
<td>Continue</td>
<td>MEM_byte</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>None</td>
<td>?4D</td>
<td>Continue</td>
<td>MEMORY_addr</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>None</td>
<td>?4C</td>
<td>Continue</td>
<td>MEMORY_ECC_error</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>None</td>
<td>?4B</td>
<td>Continue</td>
<td>MEMORY_masked_errors</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>None</td>
<td>?4A</td>
<td>Continue</td>
<td>MEMORY_correction</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>None</td>
<td>?48</td>
<td>Continue</td>
<td>MEMORY_address_shorts</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>8</td>
<td>None</td>
<td>?40</td>
<td>Continue</td>
<td>MEMORY_count_errors</td>
<td>2, 1, 3, 5</td>
</tr>
<tr>
<td>7</td>
<td>None</td>
<td>?80</td>
<td>Continue</td>
<td>CQBIC_MEMORY</td>
<td>1, 2, 4, 3, 5</td>
</tr>
<tr>
<td></td>
<td>None</td>
<td>?41</td>
<td>Continue</td>
<td>Board reset</td>
<td>1, 4</td>
</tr>
</tbody>
</table>

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.

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

Troubleshooting and Diagnostics 4–23
### 4.3.6 System Halt Messages

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>?02</td>
<td>EXT HLT</td>
<td>External halt, caused either by console BREAK condition, or because Q22-bus BHALT_L or DBR&lt;AUX_HLT&gt; bit was set while enabled.</td>
</tr>
<tr>
<td>?04</td>
<td>ISP ERR</td>
<td>Caused by attempt to push interrupt or exception state onto the interrupt stack when the interrupt stack was mapped NO ACCESS or NOT VALID.</td>
</tr>
<tr>
<td>?05</td>
<td>DBL ERR</td>
<td>A second machine check occurred while the processor was attempting to service a normal exception.</td>
</tr>
<tr>
<td>?06</td>
<td>HLT INST</td>
<td>The processor executed a HALT instruction in kernel mode.</td>
</tr>
<tr>
<td>?07</td>
<td>SCB ERR3</td>
<td>The vector had bits &lt;1:0&gt; = 3.</td>
</tr>
<tr>
<td>?08</td>
<td>SCB ERR2</td>
<td>The vector had bits &lt;1:0&gt; = 2.</td>
</tr>
<tr>
<td>?0A</td>
<td>CHM FR ISTK</td>
<td>A change mode instruction was executed when PSL&lt;IS&gt; was set.</td>
</tr>
<tr>
<td>?0B</td>
<td>CHM TO ISTK</td>
<td>The SCB vector for a change mode had bit &lt;0&gt; set.</td>
</tr>
<tr>
<td>?0C</td>
<td>SCB RD ERR</td>
<td>A hard memory error occurred during a processor read of an exception or interrupt vector.</td>
</tr>
<tr>
<td>?10</td>
<td>MCHK AV</td>
<td>An access violation or an invalid translation occurred during machine check exception processing.</td>
</tr>
<tr>
<td>?11</td>
<td>KSP AV</td>
<td>An access violation or an invalid translation occurred during invalid kernel stack pointer exception processing.</td>
</tr>
<tr>
<td>?12</td>
<td>DBL ERR2</td>
<td>Double machine check error. A machine check occurred during an attempt to service a machine check.</td>
</tr>
<tr>
<td>?13</td>
<td>DBL ERR3</td>
<td>Double machine check error. A machine check occurred during an attempt to service a kernel stack not valid exception.</td>
</tr>
<tr>
<td>?19</td>
<td>PSL EXC5</td>
<td>PSL &lt;26:24&gt; = 5 on interrupt or exception.</td>
</tr>
<tr>
<td>?1A</td>
<td>PSL EXC6</td>
<td>PSL &lt;26:24&gt; = 6 on interrupt or exception.</td>
</tr>
<tr>
<td>?1B</td>
<td>PSL EXC7</td>
<td>PSL &lt;26:24&gt; = 7 on interrupt or exception.</td>
</tr>
<tr>
<td>?1D</td>
<td>PSL RE15</td>
<td>PSL &lt;26:24&gt; = 5 on an rei instruction.</td>
</tr>
<tr>
<td>?1E</td>
<td>PSL RE16</td>
<td>PSL &lt;26:24&gt; = 6 on an rei instruction.</td>
</tr>
<tr>
<td>?1F</td>
<td>PSL RE17</td>
<td>PSL &lt;26:24&gt; = 7 on an rei instruction.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>?20</td>
<td>CORRPTN</td>
<td>The console data base was corrupted. The console simulates a power-up sequence and rebuilds its data base.</td>
</tr>
<tr>
<td>?21</td>
<td>ILL REF</td>
<td>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.</td>
</tr>
<tr>
<td>?22</td>
<td>ILL CMD</td>
<td>The command string cannot be parsed.</td>
</tr>
<tr>
<td>?23</td>
<td>INV DGT</td>
<td>A number has an invalid digit.</td>
</tr>
<tr>
<td>?24</td>
<td>LITL</td>
<td>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.</td>
</tr>
<tr>
<td>?25</td>
<td>ILL ADR</td>
<td>The specified address is not in the address space.</td>
</tr>
<tr>
<td>?26</td>
<td>VAL TOO LRG</td>
<td>The specified value does not fit in the destination.</td>
</tr>
<tr>
<td>?27</td>
<td>SW CONF</td>
<td>Switch conflict. For example, an EXAMINE command specifies two different data sizes.</td>
</tr>
<tr>
<td>?28</td>
<td>UNK SW</td>
<td>The switch is not recognized.</td>
</tr>
<tr>
<td>?29</td>
<td>UNK SYM</td>
<td>The EXAMINE or DEPOSIT symbolic address is not recognized.</td>
</tr>
<tr>
<td>?2A</td>
<td>CHKSM</td>
<td>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 &quot;Illegal command.&quot;</td>
</tr>
<tr>
<td>?2B</td>
<td>HLTED</td>
<td>The operator entered a HALT command.</td>
</tr>
<tr>
<td>?2C</td>
<td>FND ERR</td>
<td>A FIND command failed either to find the RPB or 64 Kbytes of good memory.</td>
</tr>
<tr>
<td>?2D</td>
<td>TMOUT</td>
<td>Data failed to arrive in the expected time during an X command.</td>
</tr>
<tr>
<td>?2E</td>
<td>MEM ERR</td>
<td>Memory error or machine check occurred.</td>
</tr>
<tr>
<td>?2F</td>
<td>UNXINT</td>
<td>An unexpected interrupt or exception occurred.</td>
</tr>
<tr>
<td>?30</td>
<td>UNIMPLEMENTED</td>
<td>Unimplemented function.</td>
</tr>
<tr>
<td>?31</td>
<td>QUAL NOVAL</td>
<td>Qualifier does not take a value.</td>
</tr>
<tr>
<td>?32</td>
<td>QUAL AMBG</td>
<td>Ambiguous qualifier.</td>
</tr>
<tr>
<td>?33</td>
<td>QUAL REQ VAL</td>
<td>Qualifier requires a value.</td>
</tr>
<tr>
<td>?34</td>
<td>QUAL OVERF</td>
<td>Too many qualifiers.</td>
</tr>
<tr>
<td>?35</td>
<td>ARG OVERF</td>
<td>Too many arguments.</td>
</tr>
<tr>
<td>?36</td>
<td>AMBG CMD</td>
<td>Ambiguous command.</td>
</tr>
<tr>
<td>?37</td>
<td>INSUF ARG</td>
<td>Too few arguments.</td>
</tr>
</tbody>
</table>
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

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

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:

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

Press `CTRLC` to terminate the scripts.

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

```bash
>>> 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:

```bash
>>> 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 `CTRLC` 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 00FFFFFE, 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 MEMCSRR 0–15 with the memory installed in the system:

>>> 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 =00000000  TXCS =00000000  TXDB =00000030
MSE0 =00000000  CADR =00000000C  CACR =F5B40040
BDR =FFFFFF50  DLEDRI=00000000C  SSCTR=00D45577  CBTRC=00000004
SCR =00000000  DSER =00000000  QBEAR=00000000A  DEAR =00000000
QMBR=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=00000003C

One memory bank is enabled for each 4 Mbytes of memory. The MEMCSRMs 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 increase 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 increase 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 columnn = 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
dev_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=00000000 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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Machine check code</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Machine check code</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Machine check code</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Machine check code</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Machine check code</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Machine check code</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Machine check code</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Machine check code</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>MSER &lt;6&gt;</td>
<td>CDAL data parity error.</td>
</tr>
<tr>
<td>22</td>
<td>MSER &lt;5&gt;</td>
<td>Mchn chk CDAL parity error.</td>
</tr>
<tr>
<td>21</td>
<td>MSER &lt;4&gt;</td>
<td>Machine check cache parity.</td>
</tr>
<tr>
<td>20</td>
<td>MSER &lt;1&gt;</td>
<td>Cache data parity error.</td>
</tr>
<tr>
<td>19</td>
<td>MSER &lt;0&gt;</td>
<td>Cache tag parity error.</td>
</tr>
<tr>
<td>18</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>MEMCSR16 &lt;31&gt;</td>
<td>Uncorrectable ECC error.</td>
</tr>
<tr>
<td>16</td>
<td>MEMCSR16 &lt;30&gt;</td>
<td>Two or more uncorrectable errors.</td>
</tr>
<tr>
<td>15</td>
<td>MEMCSR16 &lt;29&gt;</td>
<td>Correctable single-bit error.</td>
</tr>
<tr>
<td>14</td>
<td>MEMCSR16 &lt;25&gt;</td>
<td>Page address bits 25:22 of</td>
</tr>
<tr>
<td>13</td>
<td>MEMCSR16 &lt;24&gt;</td>
<td>Location that caused error.</td>
</tr>
<tr>
<td>12</td>
<td>MEMCSR16 &lt;23&gt;</td>
<td>These four bits point to the</td>
</tr>
<tr>
<td>11</td>
<td>MEMCSR16 &lt;22&gt;</td>
<td>failing 4-Mbyte bank of memory.</td>
</tr>
<tr>
<td>10</td>
<td>MEMCSR16 &lt;8&gt;</td>
<td>DMA read/write error.</td>
</tr>
<tr>
<td>9</td>
<td>MEMCSR16 &lt;7&gt;</td>
<td>CDAL parity error on write.</td>
</tr>
<tr>
<td>8</td>
<td>CBCTR &lt;31&gt;</td>
<td>CDAL bus timeout.</td>
</tr>
<tr>
<td>7</td>
<td>CBCTR &lt;30&gt;</td>
<td>CPU read/write bus timeout.</td>
</tr>
<tr>
<td>6</td>
<td>DSER &lt;7&gt;</td>
<td>Q22-bus NXM.</td>
</tr>
<tr>
<td>4</td>
<td>DSER &lt;5&gt;</td>
<td>Q22-bus parity error.</td>
</tr>
<tr>
<td>3</td>
<td>DSER &lt;4&gt;</td>
<td>Read main memory error.</td>
</tr>
<tr>
<td>2</td>
<td>DSER &lt;3&gt;</td>
<td>Lost error.</td>
</tr>
<tr>
<td>1</td>
<td>DSER &lt;2&gt;</td>
<td>No grant timeout.</td>
</tr>
<tr>
<td>0</td>
<td>IPCRn &lt;15&gt;</td>
<td>DMA Q22-bus memory error.</td>
</tr>
</tbody>
</table>
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. TA9

>>> 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
TCSR =00000000 TIR0 =000207E3 TNIR0=00000000 TIVR0=00000078
TCSR =00000000 TIR1 =00000000 TNIR1=00000000 TIVR1=0000007C
RXCS =00000000 RXDB =00000000 TXCS =00000000 TXDB =00000030
MSER =00000000 CADR =00000000 CACR =F5B40040
BDR =FFFFFF50 DLED =00000000 SSCR =00D45577 CBTCR =00000004
SCR =0000C000 DSR =00000000 QBEAR =0000000A DEAR =00000000
QBMMR =00FF8000 IPCRn =0020
MEM_FRU 1 MEMCSR _0=80000000 1=80400000 2=80800017 3=80C00017
MEM_FRU 2 MEMCSR _4=81000000 5=81400000 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 =81540000 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

\[ \begin{array}{ccc}
3 & 2 & 2 \\
1 & 5 & 2 \\
\end{array} \]

MEMCSR16 = 8154000F Hex = 1000 0001 0101 0100 0000 0000 0000 1111
\[ \begin{array}{ccc}
| & | & | \\
\end{array} \]
MEMCSR_5 = 81400017 Hex = 1000 0001 0100 0000 0000 0000 0001 0111
\[ \begin{array}{ccc}
^ & ^ & \\
\end{array} \]

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

<table>
<thead>
<tr>
<th>Device</th>
<th>Module Loopback</th>
<th>Cable Loopback</th>
</tr>
</thead>
<tbody>
<tr>
<td>CXA16/CXB16</td>
<td>H3103 + H8572(^1)</td>
<td></td>
</tr>
<tr>
<td>CXY08</td>
<td>H3046 (50-pin)</td>
<td>H3197 (25-pin)</td>
</tr>
<tr>
<td>DELQA</td>
<td>12–22196–02</td>
<td></td>
</tr>
<tr>
<td>DPV11</td>
<td>H3259</td>
<td>H3260</td>
</tr>
<tr>
<td>DSSI(^2)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>DZQ11</td>
<td>12–15336–00 or H325</td>
<td>H329 (12–27351–01)</td>
</tr>
<tr>
<td>Ethernet(^3)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>LPV11</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>KA655/H3600–SA</td>
<td>H3103</td>
<td>H3103 + H8572</td>
</tr>
<tr>
<td>KMV1A</td>
<td>H3255</td>
<td>H3251</td>
</tr>
</tbody>
</table>

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

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

\(^3\)For ThinWire, use H8223–00 plus two H8225–00 terminators. For standard Ethernet, use 12–22196–02.
4.8 RF-Series ISE Troubleshooting and Diagnostics

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

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

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

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

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

Three configuration errors also commonly occur:

• More than one node with the same node number

• Identical node names

• Identical unit numbers

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

If the ISE is connected to the OCP, you must install a unit ID plug in the corresponding socket on the OCP. If the ISE is not connected to the OCP, it reads the unit ID from the three-switch DIP switch on the side of the drive.
The RF-series ISE contains the following local programs (described in the following sections):

DIRECT       A directory, in DUP specified format, of available local programs
DRVTSST      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

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Copyright © 1988 Digital Equipment Corporation</td>
</tr>
<tr>
<td>Q</td>
<td>Write/read anywhere on the medium? [1=yes/(0=no)]</td>
</tr>
<tr>
<td>Q</td>
<td>User data will be corrupted. Proceed? [1=yes/(0=no)]</td>
</tr>
<tr>
<td>I</td>
<td>5 minutes to complete.</td>
</tr>
<tr>
<td>T</td>
<td>Test passed.</td>
</tr>
</tbody>
</table>

Or:

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE</td>
<td>Unit is currently in use.(^1)</td>
</tr>
<tr>
<td>FE</td>
<td>Operation aborted by user.</td>
</tr>
<tr>
<td>FE</td>
<td>xxxx—Unit diagnostics failed.(^2)</td>
</tr>
<tr>
<td>FE</td>
<td>xxxx—Unit read/write test failed.(^2)</td>
</tr>
</tbody>
</table>

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

\(^2\) Refer to the diagnostic error code list at the end of this chapter.

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

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

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

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

Table 4-13: DRVEXR Messages

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Copyright © 1988 Digital Equipment Corporation</td>
</tr>
<tr>
<td>Q</td>
<td>Write/read anywhere on the medium? [1=yes/(0=no)]</td>
</tr>
<tr>
<td>Q</td>
<td>User data will be corrupted. Proceed? [1=yes/(0=no)]</td>
</tr>
<tr>
<td>Q</td>
<td>Test time in minutes? [(10)-100]</td>
</tr>
<tr>
<td>I</td>
<td>dddd minutes to complete.</td>
</tr>
<tr>
<td>I</td>
<td>dddddddd blocks (512 bytes) transferred.</td>
</tr>
<tr>
<td>I</td>
<td>dddddddd bytes in error (soft).</td>
</tr>
<tr>
<td>I</td>
<td>dddddddd uncorrectable ECC errors (recoverable).</td>
</tr>
<tr>
<td>T</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

Or:

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE</td>
<td>Unit is currently in use.(^1)</td>
</tr>
<tr>
<td>FE</td>
<td>Operation aborted by user.</td>
</tr>
<tr>
<td>FE</td>
<td>xxxx—Unit diagnostics failed.(^2)</td>
</tr>
<tr>
<td>FE</td>
<td>xxxx—Unit read/write test failed.(^2)</td>
</tr>
</tbody>
</table>

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

\(^2\)Refer to the diagnostic error code list at the end of this chapter.

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

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

NOTE: If you press the write-protect switch on the OCP (LED on) and you answer Yes to the second question, the drive does not allow the test to run. DRVEXR displays the error message 2006—Unit read/write test failed. In this case, the test has not failed, but has been prevented from running.
DRVEXR saves the error counters, then calls the timed I/O routine. After the timed I/O routine ends, DRVEXR saves the counters again. It then reports the total number of blocks transferred, bits in error, bytes in error, and uncorrectable errors.

DRVEXR uses the same timed I/O routine as DRVTST, with two exceptions. First, DRVTST always uses a fixed time of 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>
<thead>
<tr>
<th>Message Type</th>
<th>Field Length</th>
<th>Field Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>47 ASCII characters</td>
<td>Copyright notice</td>
</tr>
<tr>
<td>I</td>
<td>4 ASCII characters</td>
<td>Product name</td>
</tr>
<tr>
<td>I</td>
<td>12 ASCII characters</td>
<td>Drive serial number</td>
</tr>
<tr>
<td>I</td>
<td>6 ASCII characters</td>
<td>Node name</td>
</tr>
<tr>
<td>I</td>
<td>1 ASCII character</td>
<td>Allocation class</td>
</tr>
<tr>
<td>I</td>
<td>8 ASCII characters</td>
<td>Firmware revision level</td>
</tr>
<tr>
<td>I</td>
<td>17 ASCII characters</td>
<td>Hardware revision level</td>
</tr>
<tr>
<td>I</td>
<td>6 ASCII characters</td>
<td>Power-on hours</td>
</tr>
<tr>
<td>I</td>
<td>5 ASCII characters</td>
<td>Power cycles</td>
</tr>
<tr>
<td>I</td>
<td>4 ASCII characters</td>
<td>Hexadecimal fault code</td>
</tr>
<tr>
<td>T</td>
<td>4 ASCII characters</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

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 HISTORY:

Copyright © 1988 Digital Equipment Corporation

RF71
EN01082
SUSAN
0
RFX V101
RF71 PCB-5/ECO-00
617
21
A04F
A04F
A103
A04F
A404
A04F
A404
A04F
A404
A04F
A404
Complete.

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

4.8.4 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**

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Copyright © 1989 Digital Equipment Corporation</td>
</tr>
<tr>
<td>Q</td>
<td>Write/read anywhere on the medium? [1=yes/(0=no)]</td>
</tr>
<tr>
<td>Q</td>
<td>User data will be corrupted. Proceed? [1=yes/(0=no)]</td>
</tr>
<tr>
<td>I</td>
<td>6 minutes to complete.</td>
</tr>
<tr>
<td>T</td>
<td>Complete.</td>
</tr>
</tbody>
</table>

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

---

4–46 KA655 CPU System Maintenance
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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLCLASS</td>
<td>The controller allocation class. The allocation class should be set to match that of the host.</td>
</tr>
<tr>
<td>FIVEDIME</td>
<td>True (1) if MSCP should support five connections with ten credits each. False (0) if MSCP should support seven connections with seven credits each.</td>
</tr>
<tr>
<td>UNITNUM</td>
<td>The MSCP unit number.</td>
</tr>
<tr>
<td>FORCEUNI</td>
<td>True (1) if the unit number should be taken from the DSSI ID. False (0) if the UNITNUM value should be used instead.</td>
</tr>
<tr>
<td>NODENAME</td>
<td>The controller’s SCS node name.</td>
</tr>
<tr>
<td>FORCENAM</td>
<td>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.</td>
</tr>
</tbody>
</table>
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.
Example A-1: KFQSA (M7769) Service Mode Switch Settings

KFQSA Four-Position Switchpack

<table>
<thead>
<tr>
<th>S/N</th>
<th>Mode</th>
<th>Fx/F1</th>
<th>MSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

S/N = Service mode/Normal operating mode
Fx/F1 = fixed/浮动ing 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) = 0AAA
-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
-20001944 (774504) = 0200 (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
  KUV11C  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)

<table>
<thead>
<tr>
<th>NODE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>772150</td>
</tr>
<tr>
<td>1</td>
<td>760334</td>
</tr>
<tr>
<td>2</td>
<td>760340</td>
</tr>
<tr>
<td>3</td>
<td>760344</td>
</tr>
<tr>
<td>4</td>
<td>760350</td>
</tr>
<tr>
<td>5</td>
<td>760354</td>
</tr>
<tr>
<td>6</td>
<td>KFQSA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NODE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>772150</td>
</tr>
<tr>
<td>1</td>
<td>760334</td>
</tr>
<tr>
<td>2</td>
<td>760340</td>
</tr>
<tr>
<td>3</td>
<td>760344</td>
</tr>
<tr>
<td>4</td>
<td>760350</td>
</tr>
<tr>
<td>5</td>
<td>760354</td>
</tr>
<tr>
<td>7</td>
<td>KFQSA</td>
</tr>
</tbody>
</table>

Configuring the KFQSA A-9
Example A–5: Display for Programming the First KFQSA

```plaintext
>>> 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

SET <NODE> <CSR_ADDRESS> <MODEL>

CLEAR <NODE>

SHOW

HELP

EXIT

QUIT

Parameters:

<NODE>

<CSR_ADDRESS>

<MODEL>

? 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 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
6 -------- KFQSA --------

? exit

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

Configuring the KFQSA A–11
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 OBUS` 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/DEQSA
-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

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

<table>
<thead>
<tr>
<th>Node</th>
<th>CSR Address</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>772150</td>
<td>21</td>
</tr>
<tr>
<td>1</td>
<td>760334</td>
<td>21</td>
</tr>
<tr>
<td>2</td>
<td>760340</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>760344</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>760350</td>
<td>21</td>
</tr>
<tr>
<td>5</td>
<td>760354</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>----- KFQSA ------</td>
<td></td>
</tr>
</tbody>
</table>

? clear 5
? show

<table>
<thead>
<tr>
<th>Node</th>
<th>CSR Address</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>772150</td>
<td>21</td>
</tr>
<tr>
<td>1</td>
<td>760334</td>
<td>21</td>
</tr>
<tr>
<td>2</td>
<td>760340</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>760344</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>760350</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>----- KFQSA ------</td>
<td></td>
</tr>
</tbody>
</table>

? set 5 760354 21
? show

<table>
<thead>
<tr>
<th>Node</th>
<th>CSR Address</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>772150</td>
<td>21</td>
</tr>
<tr>
<td>1</td>
<td>760334</td>
<td>21</td>
</tr>
<tr>
<td>2</td>
<td>760340</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>760344</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>760350</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>----- KFQSA ------</td>
<td></td>
</tr>
</tbody>
</table>

? exit

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

A–16 KA655 CPU System Maintenance
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 \texttt{SHOW FORCEUNI} to show the new forceunit flag values.

12. Type \texttt{WRITE}, then type \texttt{Y} to save the new values into the EEROM, or \texttt{N} to cancel the reprogramming.

13. Type \texttt{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.

\textbf{Example A–10: Display for Changing Allocation Class and Unit Number}

\begin{verbatim}
>>> set host/dup/eqssp/disk 0 params
Starting DUP server...
UQSSP Disk Controller 0 (772150)
Copyright (c) 1988 Digital Equipment Corporation

PARAMS> show allclass

\begin{tabular}{llll}
\hline
Parameter & Current & Default & Type & Radix \\
\hline
ALLCLASS & 1 & 0 & Byte & Dec B \\
\hline
\end{tabular}

PARAMS> set allclass 2

PARAMS> show unitnum

\begin{tabular}{llll}
\hline
Parameter & Current & Default & Type & Radix \\
\hline
UNITNUM & 0 & 0 & Word & Dec B \\
\hline
\end{tabular}

PARAMS> set unitnum 20

PARAMS> set forceuni 0

PARAMS> show allclass

\begin{tabular}{llll}
\hline
Parameter & Current & Default & Type & Radix \\
\hline
ALLCLASS & 2 & 0 & Byte & Dec B \\
\hline
\end{tabular}

PARAMS> show unitnum

\begin{tabular}{llll}
\hline
Parameter & Current & Default & Type & Radix \\
\hline
UNITNUM & 20 & 0 & Word & Dec U \\
\hline
\end{tabular}

PARAMS> show forceuni

\begin{tabular}{llll}
\hline
Parameter & Current & Default & Type & Radix \\
\hline
FORCEUNI & 0 & 1 & Boolean & 0/1 U \\
\hline
\end{tabular}

\end{verbatim}

\textit{Example A–10 Cont’d. on next page}
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

<table>
<thead>
<tr>
<th>Contents</th>
<th>Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local memory space (64 Mbytes)</td>
<td>0000 0000–03FF FFFF</td>
</tr>
<tr>
<td>Reserved memory space (64 Mbytes)</td>
<td>0400 0000–07FF FFFF</td>
</tr>
<tr>
<td>Reserved memory space (64 Mbytes)</td>
<td>0800 0000–0BFF FFFF</td>
</tr>
<tr>
<td>Reserved memory space (64 Mbytes)</td>
<td>0C00 0000–0FFF FFFF</td>
</tr>
<tr>
<td>Cache diagnostic space (64 Mbytes)</td>
<td>1000 0000–13FF FFFF</td>
</tr>
<tr>
<td>Reserved cache diagnostic space (64 Mbytes)</td>
<td>1400 0000–17FF FFFF</td>
</tr>
<tr>
<td>Reserved cache diagnostic space (64 Mbytes)</td>
<td>1800 0000–1BFF FFFF</td>
</tr>
<tr>
<td>Reserved cache diagnostic space (64 Mbytes)</td>
<td>1C00 0000–1FFF FFFF</td>
</tr>
</tbody>
</table>
### Table B–2: VAX Input/Output Space

<table>
<thead>
<tr>
<th>Address Range</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000 0000–2000 1FFF</td>
<td>Local Q22-bus I/O space (8 Kbytes)</td>
</tr>
<tr>
<td>2000 2000–2003 FFFF</td>
<td>Reserved local I/O space (248 Kbytes)</td>
</tr>
<tr>
<td>2004 0000–2005 FFFF</td>
<td>Local ROM space—halt protected space (128 Kbytes)</td>
</tr>
<tr>
<td>2006 0000–2007 FFFF</td>
<td>Local ROM space—halt unprotected space (128 Kbytes)</td>
</tr>
<tr>
<td>2008 0000–201F FFFF</td>
<td>Local register I/O space (1.5 Mbytes)</td>
</tr>
<tr>
<td>2200 0000–23FF FFFF</td>
<td>Reserved local I/O space (62.5 Mbytes)</td>
</tr>
<tr>
<td>2400 0000–27FF FFFF</td>
<td>Reserved local I/O space (64 Mbytes)</td>
</tr>
<tr>
<td>2800 0000–2BFF FFFF</td>
<td>Reserved local I/O space (64 Mbytes)</td>
</tr>
<tr>
<td>2C08 0000–2FFF FFFF</td>
<td>Reserved local I/O space (64 Mbytes)</td>
</tr>
<tr>
<td>3000 0000–303F FFFF</td>
<td>Local Q22-bus memory space (4 Mbytes)</td>
</tr>
<tr>
<td>3040 0000–33FF FFFF</td>
<td>Reserved local I/O space (60 Mbytes)</td>
</tr>
<tr>
<td>3400 0000–37FF FFFF</td>
<td>Reserved local I/O space (64 Mbytes)</td>
</tr>
<tr>
<td>3800 0000–3BFF FFFF</td>
<td>Cache tag diagnostic space (64 Mbytes)*</td>
</tr>
<tr>
<td>3C00 0000–3FFF FFFF</td>
<td>Reserved cache tag diagnostic space (64 Mbytes)</td>
</tr>
</tbody>
</table>

*Not visible during normal operation.

### B.2 Detailed Local Address Space Map

### Table B–3: Detailed VAX Memory Space

<table>
<thead>
<tr>
<th>Contents</th>
<th>Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local memory space (up to 64 Mbytes)</td>
<td>0000 0000–03FF FFFF</td>
</tr>
<tr>
<td>Q22-bus map—top 32 Kbytes of main memory</td>
<td></td>
</tr>
<tr>
<td>Reserved memory space</td>
<td>0400 0000–0FFF FFFF</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Cache diagnostic space</td>
<td>1000 0000–13FF FFFF</td>
</tr>
<tr>
<td>Reserved cache diagnostic space</td>
<td>1800 0000–1FFF FFFF</td>
</tr>
<tr>
<td>Contents</td>
<td>Address Range</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Local Q22-bus I/O Space</td>
<td>2000 0000–2000 1FFF</td>
</tr>
<tr>
<td>Reserved Q22-bus I/O space (diagnostics)</td>
<td>2000 0000–2000 0007</td>
</tr>
<tr>
<td>Q22-bus floating address space</td>
<td>2000 0008–2000 07FF</td>
</tr>
<tr>
<td>Q22-bus fixed and user I/O space</td>
<td>2000 0800–2000 0FFF</td>
</tr>
<tr>
<td>Q22-bus fixed and reserved I/O space</td>
<td>2000 1000–2000 1FFF</td>
</tr>
<tr>
<td>Interprocessor communication register (normal operation)</td>
<td>2000 1F40</td>
</tr>
<tr>
<td>Interprocessor communication register (reserved)</td>
<td>2000 1F42</td>
</tr>
<tr>
<td>Interprocessor communication register (reserved)</td>
<td>2000 1F44</td>
</tr>
<tr>
<td>Interprocessor communication register (reserved)</td>
<td>2000 1F46</td>
</tr>
<tr>
<td>Reserved Q22-bus I/O space</td>
<td>2000 1F48–2000 1FFF</td>
</tr>
<tr>
<td><strong>Reserved Local I/O Space</strong></td>
<td><strong>2000 2000–2003 FFFF</strong></td>
</tr>
<tr>
<td>Local ROM Space</td>
<td>2004 0000–2007 FFFF</td>
</tr>
<tr>
<td>Local ROM protected space</td>
<td>2004 0000–2005 FFFF</td>
</tr>
<tr>
<td>MicroVAX system type register (in ROM)</td>
<td>2004 0004</td>
</tr>
<tr>
<td>Local ROM unprotected space</td>
<td>2006 0000–2007 FFFF</td>
</tr>
<tr>
<td><strong>Local Register I/O Space</strong></td>
<td><strong>2008 0000–201F FFFF</strong></td>
</tr>
<tr>
<td>DMA system configuration register</td>
<td>2008 0000</td>
</tr>
<tr>
<td>DMA system error register</td>
<td>2008 0004</td>
</tr>
<tr>
<td>DMA master error address register</td>
<td>2008 0008</td>
</tr>
<tr>
<td>DMA slave error address register</td>
<td>2008 000C</td>
</tr>
<tr>
<td>Q22-bus map base register</td>
<td>2008 0010</td>
</tr>
<tr>
<td>Reserved local register I/O space</td>
<td>2008 0014–2008 00FF</td>
</tr>
<tr>
<td>Main memory configuration register 00</td>
<td>2008 0100</td>
</tr>
<tr>
<td>Main memory configuration register 01</td>
<td>2008 0104</td>
</tr>
<tr>
<td>Main memory configuration register 02</td>
<td>2008 0108</td>
</tr>
<tr>
<td>Main memory configuration register 03</td>
<td>2008 010C</td>
</tr>
<tr>
<td>Main memory configuration register 04</td>
<td>2008 0110</td>
</tr>
<tr>
<td>Main memory configuration register 05</td>
<td>2008 0114</td>
</tr>
<tr>
<td>Main memory configuration register 06</td>
<td>2008 0118</td>
</tr>
<tr>
<td>Main memory configuration register 07</td>
<td>2008 011C</td>
</tr>
<tr>
<td>Main memory configuration register 08</td>
<td>2008 0120</td>
</tr>
<tr>
<td>Main memory configuration register 09</td>
<td>2008 0124</td>
</tr>
<tr>
<td>Main memory configuration register 10</td>
<td>2008 0128</td>
</tr>
<tr>
<td>Main memory configuration register 11</td>
<td>2008 012C</td>
</tr>
<tr>
<td>Main memory configuration register 12</td>
<td>2008 0130</td>
</tr>
<tr>
<td>Main memory configuration register 13</td>
<td>2008 0134</td>
</tr>
<tr>
<td>Main memory configuration register 14</td>
<td>2008 0138</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Contents</th>
<th>Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main memory configuration register 15</td>
<td>2008 013C</td>
</tr>
<tr>
<td>Main memory error status register</td>
<td>2008 0140</td>
</tr>
<tr>
<td>Main memory control/diagnostic status register</td>
<td>2008 0144</td>
</tr>
<tr>
<td>Reserved local register I/O space</td>
<td>2008 0148–2008 3FFF</td>
</tr>
<tr>
<td>Cache control register</td>
<td>2008 4000</td>
</tr>
<tr>
<td>Boot and diagnostic register</td>
<td>2008 4004</td>
</tr>
<tr>
<td>Reserved local register I/O space</td>
<td>2008 4008–2008 7FFF</td>
</tr>
<tr>
<td>Q22-bus map registers</td>
<td>2008 8000–2008 FFFF</td>
</tr>
<tr>
<td>Reserved local register I/O space</td>
<td>2009 0000–2013 FFFF</td>
</tr>
<tr>
<td>SSC base address register</td>
<td>2014 0000</td>
</tr>
<tr>
<td>SSC configuration register</td>
<td>2014 0010</td>
</tr>
<tr>
<td>CDAL bus timeout control register</td>
<td>2014 0020</td>
</tr>
<tr>
<td>Diagnostic LED register</td>
<td>2014 0030</td>
</tr>
<tr>
<td>Reserved local register I/O space</td>
<td>2014 0034–2014 0068</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Contents</th>
<th>Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-of-year register</td>
<td>2014 006C</td>
</tr>
<tr>
<td>Console storage receiver status</td>
<td>2014 0070*</td>
</tr>
<tr>
<td>Console storage receiver data</td>
<td>2014 0074*</td>
</tr>
<tr>
<td>Console storage transmitter status</td>
<td>2014 0078*</td>
</tr>
<tr>
<td>Console storage transmitter data</td>
<td>2014 007C*</td>
</tr>
<tr>
<td>Console receiver control/status</td>
<td>2014 0080</td>
</tr>
<tr>
<td>Console receiver data buffer</td>
<td>2014 0084</td>
</tr>
<tr>
<td>Console transmitter control/status</td>
<td>2014 0088</td>
</tr>
<tr>
<td>Console transmitter data buffer</td>
<td>2014 008C</td>
</tr>
<tr>
<td>Reserved local register I/O space</td>
<td>2014 0090–2014 00DB</td>
</tr>
<tr>
<td>* These registers are not fully implemented; accesses yield unpredictable results.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Contents</th>
<th>Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O bus reset register</td>
<td>2014 00DC</td>
</tr>
<tr>
<td>Reserved local register I/O space</td>
<td>2014 00E0–2014 00EF</td>
</tr>
<tr>
<td>ROM data register</td>
<td>2014 00F0**</td>
</tr>
<tr>
<td>Bus timeout counter</td>
<td>2014 00F4**</td>
</tr>
<tr>
<td>Contents</td>
<td>Address Range</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Interval timer</td>
<td>2014 00F8**</td>
</tr>
<tr>
<td>Reserved local register I/O space</td>
<td>2014 00FC–2014 00FF</td>
</tr>
</tbody>
</table>

** 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)**

<table>
<thead>
<tr>
<th>Contents</th>
<th>Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timer 0 control register</td>
<td>2014 0100</td>
</tr>
<tr>
<td>Timer 0 interval register</td>
<td>2014 0104</td>
</tr>
<tr>
<td>Timer 0 next interval register</td>
<td>2014 0108</td>
</tr>
<tr>
<td>Timer 0 interrupt vector</td>
<td>2014 010C</td>
</tr>
<tr>
<td>Timer 1 control register</td>
<td>2014 0110</td>
</tr>
<tr>
<td>Timer 1 interval register</td>
<td>2014 0114</td>
</tr>
<tr>
<td>Timer 1 next interval register</td>
<td>2014 0118</td>
</tr>
<tr>
<td>Timer 1 interrupt vector</td>
<td>2014 011C</td>
</tr>
<tr>
<td>Reserved local register I/O space</td>
<td>2014 0120–2014 012F</td>
</tr>
<tr>
<td>CACR address decode match register</td>
<td>2014 0130</td>
</tr>
<tr>
<td>CACR decode mask register</td>
<td>2014 0134</td>
</tr>
<tr>
<td>Reserved local register I/O space</td>
<td>2014 0138–2014 013F</td>
</tr>
<tr>
<td>BDR address decode match register</td>
<td>2014 0140</td>
</tr>
<tr>
<td>BDR decode mask register</td>
<td>2014 0144</td>
</tr>
<tr>
<td>Reserved local register I/O space</td>
<td>2014 0148–2014 03FF</td>
</tr>
<tr>
<td>Battery backed-up RAM</td>
<td>2014 0400–2014 07FF</td>
</tr>
<tr>
<td>Reserved local register I/O space</td>
<td>2014 0800–201F FFFF</td>
</tr>
</tbody>
</table>

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.

KA655 CPU Address Assignments  B–5
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.


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>
<thead>
<tr>
<th>Decimal Hex</th>
<th>Register Name</th>
<th>Mnemonic</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Kernel Stack Pointer</td>
<td>KSP</td>
<td>r/w</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Executive Stack Pointer</td>
<td>ESP</td>
<td>r/w</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Supervisor Stack Pointer</td>
<td>SSP</td>
<td>r/w</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>User Stack Pointer</td>
<td>USP</td>
<td>r/w</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Interrupt Stack Pointer</td>
<td>ISP</td>
<td>r/w</td>
<td>1</td>
</tr>
<tr>
<td>7:5</td>
<td>Reserved</td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>P0 Base</td>
<td>P0BR</td>
<td>r/w</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>P0 Length</td>
<td>P0LR</td>
<td>r/w</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>P1 Base</td>
<td>P1BR</td>
<td>r/w</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>P1 Length</td>
<td>P1LR</td>
<td>r/w</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>System Base</td>
<td>SBR</td>
<td>r/w</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>System Length</td>
<td>SLR</td>
<td>r/w</td>
<td>1</td>
</tr>
<tr>
<td>15:14</td>
<td>Reserved</td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>16</td>
<td>Process Control Block Base</td>
<td>PCBB</td>
<td>r/w</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>System Control Block Base</td>
<td>SCBB</td>
<td>r/w</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>Interrupt Priority Level</td>
<td>IPL</td>
<td>r/w</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>AST Level</td>
<td>ASTLVL</td>
<td>r/w</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>Software Interrupt Request</td>
<td>SIRR</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>Software Interrupt Summary</td>
<td>SISR</td>
<td>r/w</td>
<td>1</td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved</td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>24</td>
<td>Interval Clock Control/Status</td>
<td>ICCS</td>
<td>r/w</td>
<td>3</td>
</tr>
<tr>
<td>25</td>
<td>Next Interval Count</td>
<td>NICR</td>
<td>w</td>
<td>5</td>
</tr>
<tr>
<td>26</td>
<td>Interval Count</td>
<td>ICR</td>
<td>r</td>
<td>5</td>
</tr>
<tr>
<td>27</td>
<td>Time-of-Year Clock</td>
<td>TODR</td>
<td>r/w</td>
<td>2</td>
</tr>
<tr>
<td>28</td>
<td>Console Storage Receiver Status</td>
<td>CSRS</td>
<td>r/w</td>
<td>7</td>
</tr>
<tr>
<td>29</td>
<td>Console Storage Receiver Data</td>
<td>CSRD</td>
<td>r</td>
<td>7</td>
</tr>
<tr>
<td>30</td>
<td>Console Storage Transmit Status</td>
<td>CSTS</td>
<td>r/w</td>
<td>7</td>
</tr>
<tr>
<td>31</td>
<td>Console Storage Transmit Data</td>
<td>CSTD</td>
<td>w</td>
<td>7</td>
</tr>
<tr>
<td>32</td>
<td>Console Receiver Control/Status</td>
<td>RXCS</td>
<td>r/w</td>
<td>4</td>
</tr>
<tr>
<td>33</td>
<td>Console Receiver Data Buffer</td>
<td>RXDB</td>
<td>r</td>
<td>4</td>
</tr>
<tr>
<td>34</td>
<td>Console Transmit Control/Status</td>
<td>TXCS</td>
<td>r/w</td>
<td>4</td>
</tr>
<tr>
<td>35</td>
<td>Console Transmit Data Buffer</td>
<td>TXDB</td>
<td>w</td>
<td>4</td>
</tr>
<tr>
<td>36</td>
<td>Translation Buffer Disable</td>
<td>TBDR</td>
<td>r/w</td>
<td>5</td>
</tr>
<tr>
<td>37</td>
<td>Cache Disable</td>
<td>CADR</td>
<td>r/w</td>
<td>3</td>
</tr>
<tr>
<td>38</td>
<td>Machine Check Error Summary</td>
<td>MCESR</td>
<td>r/w</td>
<td>5</td>
</tr>
<tr>
<td>39</td>
<td>Memory System Error</td>
<td>MSER</td>
<td>r/w</td>
<td>3</td>
</tr>
<tr>
<td>41:40</td>
<td>Reserved</td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>42</td>
<td>Console Saved PC</td>
<td>SAVPC</td>
<td>r</td>
<td>3</td>
</tr>
<tr>
<td>43</td>
<td>Console Saved PSL</td>
<td>SAVPSL</td>
<td>r</td>
<td>3</td>
</tr>
<tr>
<td>Decimal</td>
<td>Hex</td>
<td>Register Name</td>
<td>Mnemonic</td>
<td>Type</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>-----------------------------</td>
<td>----------</td>
<td>------</td>
</tr>
<tr>
<td>47:44</td>
<td>2F:2C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>30</td>
<td>SBI Fault/Status</td>
<td>SBIFS</td>
<td>r/w</td>
</tr>
<tr>
<td>49</td>
<td>31</td>
<td>SBI Silo</td>
<td>SBIS</td>
<td>r</td>
</tr>
<tr>
<td>50</td>
<td>32</td>
<td>SBI Silo Comparator</td>
<td>SBISC</td>
<td>r/w</td>
</tr>
<tr>
<td>51</td>
<td>33</td>
<td>SBI Maintenance</td>
<td>SBIMT</td>
<td>r/w</td>
</tr>
<tr>
<td>52</td>
<td>34</td>
<td>SBI Error</td>
<td>SBIER</td>
<td>r/w</td>
</tr>
<tr>
<td>53</td>
<td>35</td>
<td>SBI Timeout Address</td>
<td>SBITA</td>
<td>r</td>
</tr>
<tr>
<td>54</td>
<td>36</td>
<td>SBI Quadword Clear</td>
<td>SBIQC</td>
<td>w</td>
</tr>
<tr>
<td>55</td>
<td>37</td>
<td>I/O Bus Reset</td>
<td>IORESET</td>
<td>w</td>
</tr>
<tr>
<td>56</td>
<td>38</td>
<td>Memory Management Enable</td>
<td>MAPEN</td>
<td>r/w</td>
</tr>
<tr>
<td>57</td>
<td>39</td>
<td>TB Invalidate All</td>
<td>TBIA</td>
<td>w</td>
</tr>
<tr>
<td>58</td>
<td>3A</td>
<td>TB Invalidate Single</td>
<td>TBIS</td>
<td>w</td>
</tr>
<tr>
<td>59</td>
<td>3B</td>
<td>TB Data</td>
<td>TBDATA</td>
<td>r/w</td>
</tr>
<tr>
<td>60</td>
<td>3C</td>
<td>Microprogram Break</td>
<td>MBRK</td>
<td>r/w</td>
</tr>
<tr>
<td>61</td>
<td>3D</td>
<td>Performance Monitor Enable</td>
<td>PMR</td>
<td>r/w</td>
</tr>
<tr>
<td>62</td>
<td>3E</td>
<td>System Identification</td>
<td>SID</td>
<td>r</td>
</tr>
<tr>
<td>63</td>
<td>3F</td>
<td>Translation Buffer Check</td>
<td>TBCHK</td>
<td>w</td>
</tr>
<tr>
<td>64:127</td>
<td>40:7F</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 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**

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

<table>
<thead>
<tr>
<th>Number</th>
<th>Hex</th>
<th>Register Name</th>
<th>Mnemonic</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>18</td>
<td>Interval Clock Control/Status</td>
<td>ICCS</td>
</tr>
<tr>
<td>32</td>
<td>20</td>
<td>Console Receiver Control/Status</td>
<td>RXCS</td>
</tr>
<tr>
<td>33</td>
<td>21</td>
<td>Console Receiver Data Buffer</td>
<td>RXDB</td>
</tr>
<tr>
<td>34</td>
<td>22</td>
<td>Console Transmit Control/Status</td>
<td>TXCS</td>
</tr>
<tr>
<td>35</td>
<td>23</td>
<td>Console Transmit Data Buffer</td>
<td>TXDB</td>
</tr>
<tr>
<td>37</td>
<td>25</td>
<td>Cache Disable</td>
<td>CADR</td>
</tr>
<tr>
<td>39</td>
<td>27</td>
<td>Memory System Error</td>
<td>MSER</td>
</tr>
<tr>
<td>42</td>
<td>2A</td>
<td>Console Saved PC</td>
<td>SAVPC</td>
</tr>
<tr>
<td>43</td>
<td>2B</td>
<td>Console Saved PSL</td>
<td>SAVPSL</td>
</tr>
<tr>
<td>55</td>
<td>37</td>
<td>I/O Bus Reset</td>
<td>IORESET</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Contents</th>
<th>Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q22-bus memory space (octal)</td>
<td>0000 0000–1777 7777</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Contents</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q22-bus I/O space (Octal)</td>
<td>1776 0000–1777 7777</td>
</tr>
<tr>
<td>Reserved Q22-bus I/O space (diagnostics)</td>
<td>1776 0000–1776 0007</td>
</tr>
<tr>
<td>Q22-bus floating address space</td>
<td>1776 0010–1776 3777</td>
</tr>
<tr>
<td>Q22-bus fixed and user address space</td>
<td>1776 4000–1776 7777</td>
</tr>
<tr>
<td>Q22-bus fixed and reserved address space</td>
<td>1777 0000–1777 7477</td>
</tr>
<tr>
<td>Interprocessor communication register (normal operation)</td>
<td>1777 7500</td>
</tr>
<tr>
<td>Interprocessor communication register (reserved)</td>
<td>1777 7502</td>
</tr>
<tr>
<td>Interprocessor communication register (reserved)</td>
<td>1777 7504</td>
</tr>
<tr>
<td>Interprocessor communication register (reserved)</td>
<td>1777 7506</td>
</tr>
<tr>
<td>Reserved Q22-bus I/O space</td>
<td>1777 7510–1777 7777</td>
</tr>
</tbody>
</table>
Appendix C

Related Documentation

The following documents contain information relating to the KA655 CPU.

<table>
<thead>
<tr>
<th>Document Title</th>
<th>Order Number</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Modules</strong></td>
<td></td>
</tr>
<tr>
<td>CXA16 Technical Manual</td>
<td>EK–CAB16–TM</td>
</tr>
<tr>
<td>CXY08 Technical Manual</td>
<td>EK–CXY08–TM</td>
</tr>
<tr>
<td>DPV11 Synchronous Controller Technical Manual</td>
<td>EK–DPV11–TM</td>
</tr>
<tr>
<td>DZQ11 Asynchronous Multiplexer Technical Manual</td>
<td>EK–DZQ11–TM</td>
</tr>
<tr>
<td>IEU11–A/IEQ11–A User's Guide</td>
<td>EK–IEUQ1–UG</td>
</tr>
<tr>
<td>KDA50–Q CPU Module User's Guide</td>
<td>EK–KDA5Q–UG</td>
</tr>
<tr>
<td>KFQSA Installation Guide</td>
<td>EK–KFQSA–IN</td>
</tr>
<tr>
<td>KMV11 Programmable Communications Controller User's Guide</td>
<td>EK–KMV11–UG</td>
</tr>
<tr>
<td>KMV11 Programmable Communications Controller Technical Manual</td>
<td>EK–KMV11–TM</td>
</tr>
<tr>
<td>RQDX2 Controller Module User's Guide</td>
<td>EK–RQDX2–UG</td>
</tr>
<tr>
<td>RQDX3 Controller Module User's Guide</td>
<td>EK–RQDX3–UG</td>
</tr>
<tr>
<td><strong>Disk and Tape Drives</strong></td>
<td></td>
</tr>
<tr>
<td>RF30 Integrated Storage Element</td>
<td>EK–RF30D–UG</td>
</tr>
<tr>
<td>RF30 Integrated Storage Element Installation Manual</td>
<td>EK–RF30D–IN</td>
</tr>
<tr>
<td>TK50 Tape Drive Subsystem User's Guide</td>
<td>EK–LEEP05–UG</td>
</tr>
<tr>
<td>Document Title</td>
<td>Order Number</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td><strong>Systems</strong></td>
<td></td>
</tr>
<tr>
<td>BA213 Enclosure Maintenance</td>
<td>EK–189AA–MG</td>
</tr>
<tr>
<td>BA215 Enclosure Maintenance</td>
<td>EK–191AA–MG</td>
</tr>
<tr>
<td>H9644 Cabinet Maintenance</td>
<td>EK–221AA–MG</td>
</tr>
<tr>
<td>Microsystems Options</td>
<td>EK–192AA–MG</td>
</tr>
<tr>
<td>Microsystems Site Preparation Guide</td>
<td>EK–067AB–PG</td>
</tr>
<tr>
<td><strong>Diagnostics</strong></td>
<td></td>
</tr>
<tr>
<td>MicroVAX Diagnostic Monitor Reference Card</td>
<td>AV–FMXAA–DN</td>
</tr>
<tr>
<td><strong>Networks</strong></td>
<td></td>
</tr>
<tr>
<td>Ethernet Transceiver Tester User’s Manual</td>
<td>EK–ETHTT–UG</td>
</tr>
<tr>
<td>VAX/VMS Networking Manual</td>
<td>AA–Y512C–TE</td>
</tr>
<tr>
<td>VAX NI Exerciser User’s Guide</td>
<td>AA–HI06A–TE</td>
</tr>
</tbody>
</table>
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
CQBIChip, 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
DRVEXXR local program, 4-43
DRVVTST 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

Index–3
O

OCP cabling, 2–10
Operator console parallel
   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 (CQ8IC),
   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
   DRVTST, 4–42
   ERASE, 4–45
   HISTRY, 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