KDM70 Controller Service Manual

Order Number: EK–KDM70–SV–001A

For Internal Use Only

February, 1990
December, 1991

This document was prepared using VAX DOCUMENT, Version 1.2

Contents

EXAMPLES

FIGURES

TABLES

Preface

This manual describes how to maintain and service the KDM70 controller.

Intended Audience

This document is for Digital Customer Services engineers and support engineers who service and maintain the KDM70 controller.

Scope

This guide does not cover KDM70 installation. Information on installation is in the *KDM70 User Guide*. The user guide also includes detailed equipment specifications.

Related Documentation

For more information related to the KDM70 controller, see the *KDM70 Controller User Guide* (EK–KDM70–UG)

Troubleshooting Reference Material

Refer to the following documents to run diagnostics and interpret error logs:

- *Getting Started with VAXsimPLUS* (AA–KN79A–TE)
- *VAXsimPLUS User Guide* (AA–KN80A–TE)
- *VAXsimPLUS Field Service Manual* (AA–KN82A–TE)
- *DSA Error Log Manual* (EK–DSAEL–MN)

1 General Information and Subsystem Overview

1.1 Minimum Revisions for Support on KDM70

OPERATING SYSTEM SUPPORT

VMS
The minimum release of VMS with full support for the KDM70 is V5.3-1.

ULTRIX
The minimum release of ULTRIX with full support for the KDM70 is V4.2.

TA90 minimum revision

Revision 2.3 is the minimum revision of TA90 formatter software that will support the KDM70. TA90's at a revision level less than 2.3 will not be accessible and attempts to use a down rev TA90 will result in an error log.

TS78 minimum revision

Revision 5 is the minimum revision of TS78 formatter software that will support the KDM70. When a TS78, below rev 5, is connected to a KDM70, the KDM70 will log false formatter requested errors.

SI cable external minimum length

The KDM70 requires a minimum external SI cable length of 12 feet. Cables less than 12 feet may result in SI pulse errors (Event Code = 10B). Six foot cable to tape drives is now an unsupported configuration. The six foot cable, period, is unsupported, if you read some of the SI documentation, but it will be supported on the RA90/RA70 drives because of the SA70 packaging in the VAX 6000.

MINIMUM KDM70 REVISION LEVEL

The minimum hardware version for the T2022 is rev E1.
The minimum hardware version for the T2023 is rev C1.

VAX-6000-M500 requires 2.4
VAX9000 requries 2.5
ESE-EP requires 2.4
ESE-EP HBS Requires 2.4 or 2.5 and PATCH 2
TA91 Requires V3.0

Vax 6000 Console ROM/EEPROM Minimum Revision

CPU Type = 62xx, ROM = 3.1, EEPROM = 2.0/3.6
CPU Type = 63xx, ROM = 4.1, EEPROM = 2.0/4.4
ROM = 6.0 does not require any patches
CPU Type = 64xx, ROM(0 & 1) = 1.0, EEPROM = 1.00/1.01

1.2 KDM70 Maintenance Philosophy

The KDM70 maintenance philosophy is to replace modules after analyzing module LEDs, system error logs, or software tools, such as VAXsimPLUS. Do not attempt to repair module components in the field.

1.2.1 Service Delivery Strategy

Real-time faults detected in the controller subsystem are recorded in the supporting system host error log.

Host error logs contain detailed information on intermittent and hard controller errors. This information can be used to isolate the field replaceable unit (FRU).

Controller-resident diagnostics are used to validate repairs to the KDM70 controller.

1.2.2 Maintenance Strategy

The following outlines the steps to successfully service the KDM70 controller:

1. Observe the status of the Go/No Go LEDs and module state LEDs.
2. Examine and analyze VAXsimPLUS.
3. Examine and analyze the system error logs.
4. Correlate the failure symptoms to the probable failing module through service documentation.
5. Replace the FRU only after a prime FRU is identified from the previous steps.
6. Verify repair through controller-resident diagnostics.
7. Verify that the controller is on line and operational through normal system-level commands that access the repaired unit.

Use host-based diagnostics only as a last resort to obtain information about symptoms and only if system error logs are unavailable.

1.3 Maintenance Features

The KDM70 controller provides a number of maintenance features, including:

- Go/No Go power-up self-tests (module internal self-tests)
- Go/No Go LEDs (amber) on each module
- Module (T2022) (red) LEDs
- Controller-based diagnostic and utility programs
- Host-based error logging

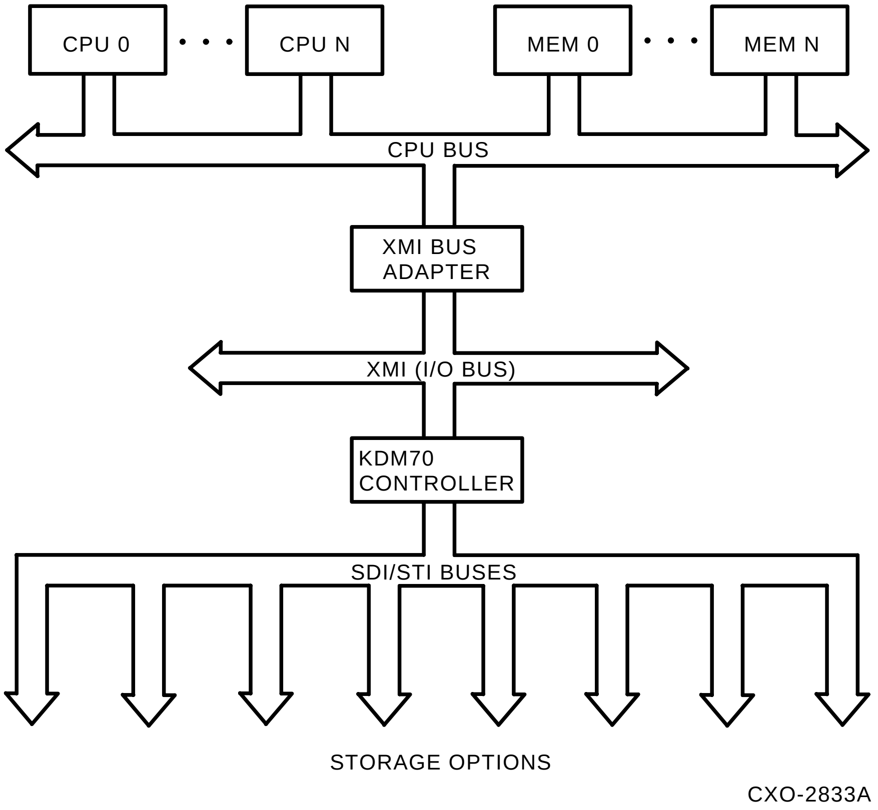
1.4 KDM70 I/O Subsystem Overview

The KDM70 I/O subsystem consists of eight parallel ports. These ports are capable of supporting any combination of:

- RA-series disk drives
- Electronic storage elements, such as the ESE20
- TA-series magnetic tape drives

Consult the *KDM70 Controller User Guide* for KDM70 controller configuration guidelines. Figure 1–1 shows how the KDM70 controller fits into the overall I/O subsystem.

Figure 1–1 KDM70 Controller Configured with XMI as a System Bus



Because the KDM70 controller is an intelligent Digital Storage Architecture (DSA) controller, it performs the following functions:

- Handles I/O management traditionally performed by the host
- Communicates with the host over the extended memory interconnect (XMI) bus using the following protocols:
  - Mass storage control protocol (MSCP)
  - Tape mass storage control protocol (TMSCP)
  - Diagnostic utility protocol (DUP)
- Communicates with mass storage media over the standard disk interface/standard tape interface (SDI/STI) bus

Communication in the KDM70 controller I/O subsystem is broken down into two sections:

- The KDM70 controller to the host
- The KDM70 controller to mass storage

1.4.1 The KDM70 Controller to Host Interface

The interface between the host and the KDM70 controller consists of:

- The extended memory interconnect (XMI) bus
- Mass storage control protocol (MSCP)
  Tape mass storage control protocol (TMSCP)
  Diagnostic utility protocol (DUP)

1.4.1.1 The Extended Memory Interconnect Bus

The XMI bus provides the communications link between the KDM70 controller and the host CPU. The bus includes the following features:

Feature	Explanation
Limited length	XMI clocks are distributed to each node over individual and equal length clock lines.
Pended	A node can request a read type transaction (Read, Interlock Read, or Ident) and release the bus after sending the command. The data source requests the bus to complete the read transaction when the data is available.
Synchronous	All transfers (discrete parts of a transaction) occur in fixed time intervals determined by XMI clocks.

The XMI identifies each device, including the KDM70 controller, as a node. Each node is identified by the backplane slot occupied by the module containing the XMI interface.

Because the largest XMI bus has 14 backplane slots, an XMI system can support 14 nodes. However, because the KDM70 controller requires two backplane slots and uses one node ID, no more than seven KDM70 controllers may be placed on a 14-node XMI bus.

Note: There may be other system-dependent restrictions. Consult system-specific user documentation.

Features of the XMI bus include:

- A maximum usable bandwidth of 100 Mbytes/second.
- Multiplexed address and data lines.
- Maximum transfer size of 16 words (hexaword) per transaction.
- Centralized bus arbitration.
- Simultaneous bus arbitration and data transfers.
- Bus error detection and reporting by all nodes.
- One gigabyte of address capability.
- Power-up and host-requested self-tests on all nodes.
- Two independent request queues: one for bus commanders and one for bus responders.

- Interlocked read and write operations permitted to memory and I/O space.
- Four consecutive data cycles allowed to complete a multicycle transfer.
- Parity protection of information transfer lines.
- A receiver/transmitter handshake required for all transfers.
- Data transfers that move naturally aligned data blocks.
- Octaword read transfers that are wraparound reads.

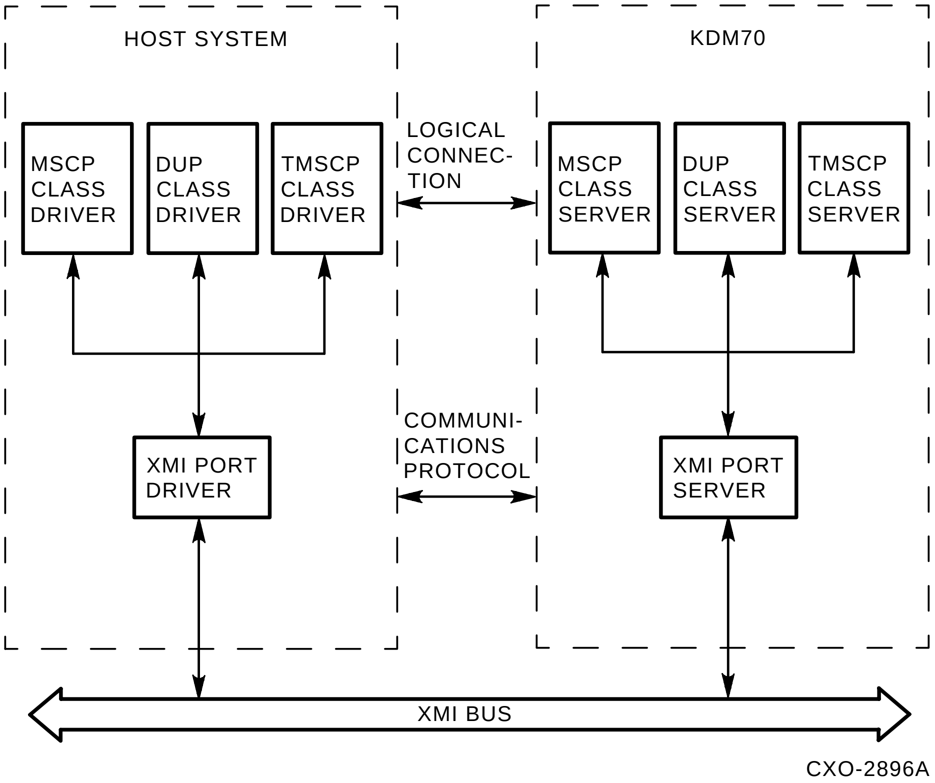
1.4.1.2 MSCP, TMSCP, and DUP Protocols

MSCP, TMSCP, and DUP protocols define command and response messages that pass between the host system and the KDM70 controller. The protocols make device-dependent characteristics, such as device geometry and error recovery strategies, invisible to the host system. Instead, the host system sees error-free disk and tape media with specific storage capacities.

The protocols are layered to simplify message transfers. Figure 1–2 illustrates the layered structure of the protocols.

The high-level processes, class driver and class server, provide I/O control of the storage device. Each high-level process has a logical connection made possible by the low-level processes called port drivers. The port drivers transfer information across the XMI bus.

Figure 1–2 Layered Structure of MSCP, TMSCP, and DUP Protocols



The following description illustrates how messages are passed between the host and the KDM70 controller:

1 The class driver assembles a command packet and stores it in the host memory command buffer. The command packet defines:

   - The unit number of the target disk or tape drive
   - The type of data transfer (read, write, or compare)
   - The amount of data to transfer
   - The location of the host buffer
   - The logical block number (LBN), if the transfer involves a disk drive

2 The host port driver attaches envelope information to define the command packet as MSCP, TMSCP, or DUP.

3 The host port driver accesses the KDM70 controller initialization and polling (IP) register.

4 The KDM70 controller port driver reads the command packet from host memory, removes the envelope information, and passes the command packet to the appropriate class server.

5 The class server reduces the command packet into discrete commands for the storage device. These commands include device management, such as positioning, data transfer, and, if necessary, error-recovery procedures.

6 The KDM70 controller completes the requested operations.

7 The class server builds a response packet that contains the completion status of the host request and passes the packet to the KDM70 controller port driver.

8 The KDM70 controller port driver attaches the envelope information to the response packet and writes it to the response buffer in host memory.

9 The KDM70 controller port driver interrupts the host system to signal that a response message is available in host memory.

10 The class driver reads the response message.

One class driver and one class server are present for each class device. One port driver supports multiple controllers connected to the system. One KDM70 controller port driver supports the MSCP, TMSCP, and DUP class servers.

1.4.2 The KDM70 Controller to Mass Storage Interface

A standard disk interface (SDI) connects the KDM70 controller to a disk drive; a standard tape interface (STI) connects the KDM70 controller to a tape formatter. Both buses are physically the same; however, protocols that define command and response message transfers across the buses differ.

The SDI and the STI buses consist of four directional coaxial lines. Two lines carry directional information from the KDM70 controller to a storage device. The remaining two lines carry directional information from a storage device to the KDM70 controller. The four lines are:

- Real-time controller state (RTCS) line
- Command/write data (WCD) line
- Response/read data (RD) line
- Real-time drive state (RTDS) line

Note: If the RTDS line connects to a tape formatter instead of a disk drive, the line is called real-time formatter state (RTFS).

The buses are radial configurations with a separate copy of the bus connecting to each storage device. Each bus receives service independent of the other buses and with equal priority.

The SDI and STI buses have the following common characteristics:

- Each detects extra or missing pulses on each of the four lines.
- Each defines errors in command and response messages exchanges.
- Each defines errors in data transfers across the SDI/STI.

- Each synchronizes transfers to the clock rate of the external storage device.

The protocols further define specific attributes of the SDI and the STI. For example, the SDI protocol defines commands and responses that support disk operations. Similarly, the STI protocol defines commands and responses that support tape operations.

1.5 Electrostatic Protection

Static protection eliminates static build-up or discharges static build-up quickly and safely.

If the charged object is a conductor, grounding eliminates discharge. The grounding cord connects the wrist strap and the conductive work surface to ground.

CAUTION: **To avoid product damage, use grounding straps when handling static-sensitive modules and components. Appendix E lists part numbers and descriptions for approved electrostatic discharge kits and materials.**

Use the following guidelines when handling static-sensitive components and modules:

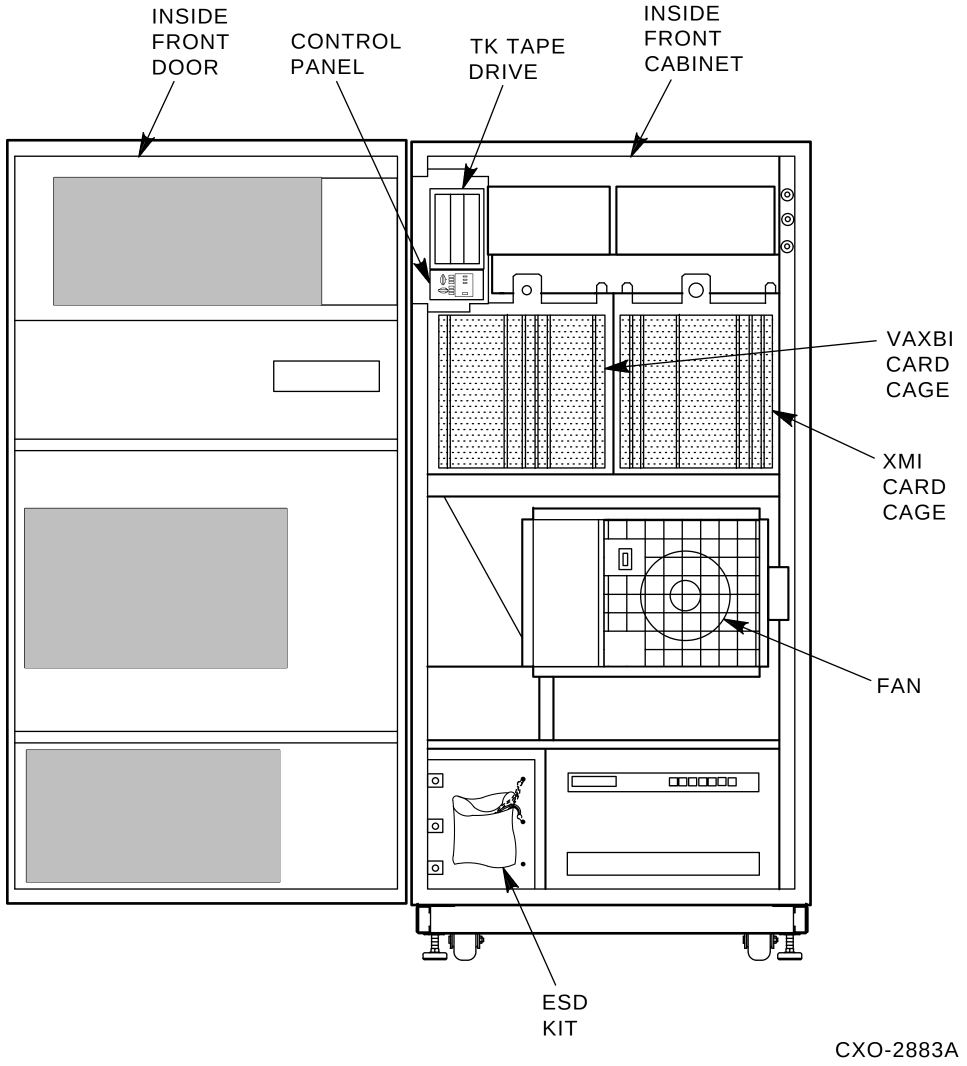
1. Be properly grounded when handling modules, components, or static-sensitive devices.
2. Use static-protective containers to transfer modules and components (including bags and tote boxes).

Figure 1–3 shows the location of the electrostatic discharge (ESD) kit.

When using an ESD wrist strap:

1. Ensure the wrist strap fits snugly for proper conductivity.
2. Do not overextend the grounding cord.
3. Attach the alligator clip securely to a clean, unpainted, grounded, metal surface, such as the cabinet chassis or module cage.

Figure 1–3 Ground Strap



2 KDM70 Controller Functional Description

2.1 Introduction

This chapter provides a brief hardware and software functional description of the KDM70 controller.

2.2 Failover

Failover between two KDM70's in different CPUs is automatic and supported providing the configuration is correct. The main concern is in device naming; if the disk is on controller PUA on one node, it must also be on controller PUA on the other. The disk can only be online to one controller at a time and other cluster members will access it via the MSCP server. The MSCP server must be loaded on both nodes and both nodes must have the same allocation class. The disk must have both ports enabled. If the serving node fails, the other node will bring it online and begin serving the disk.

KDM70 Supported Failover:

- Drive Port failover from KDM to KDM
- Drive Port failover from KDM to KDB, and KDB to KDM
- CPU failover of drive from KDM to KDM on different nodes
- CPU failover of drive from KDM to KDB, and KDB to KDM

Requirements for DISK failover on KDM70 Adapters:

- The disk(s) must be served at boot time
- KDM70 adapters must be located on different systems
- The KDM70 must have the same device mnemonic on both nodes; for example if the KDM70 is PUA0 on node 1, it must also be PUA0 on node 2

Note:

- **Failover between two KDMs on the same CPU is not supported**
- **Tape failover is not supported on Local Adapters**
- **Local controller failover to an HSC is not supported by VMS**

2.3 KDM70 Controller Hardware Functional Overview

The KDM70 controller is a specialized computer system. It uses a CVAX (3.8 VAX unit of performance (VUP) CPU) processor to execute policy software and state machines to manage data transfers. Figure 2–1 provides a functional block diagram consisting of the following components:

Component	Description
Buffer memory	A place that temporarily stores data transferred between host memory and the storage devices. Buffer memory also contains data structures to define state machine transfers.
CVAX kernel	A CVAX processor with dedicated memory and support logic.
XMI state machine (XISM)	A state machine that manages direct memory access DMA block data transfers between the host memory and the KDM70 controller buffer memory.
XMI interface	A dual-access communications path to the XMI bus. One path is controlled by the CVAX, the other path is controlled by the XISM.
Storage interconnect state machine (SISMs)	Two identical state machines that manage data transfers between the KDM70 controller buffer memory and the external disk drives or tape formatters.
Miscellaneous logic	Supporting logic, such as diagnostic and error registers.

Characteristics of the KDM70 controller include:

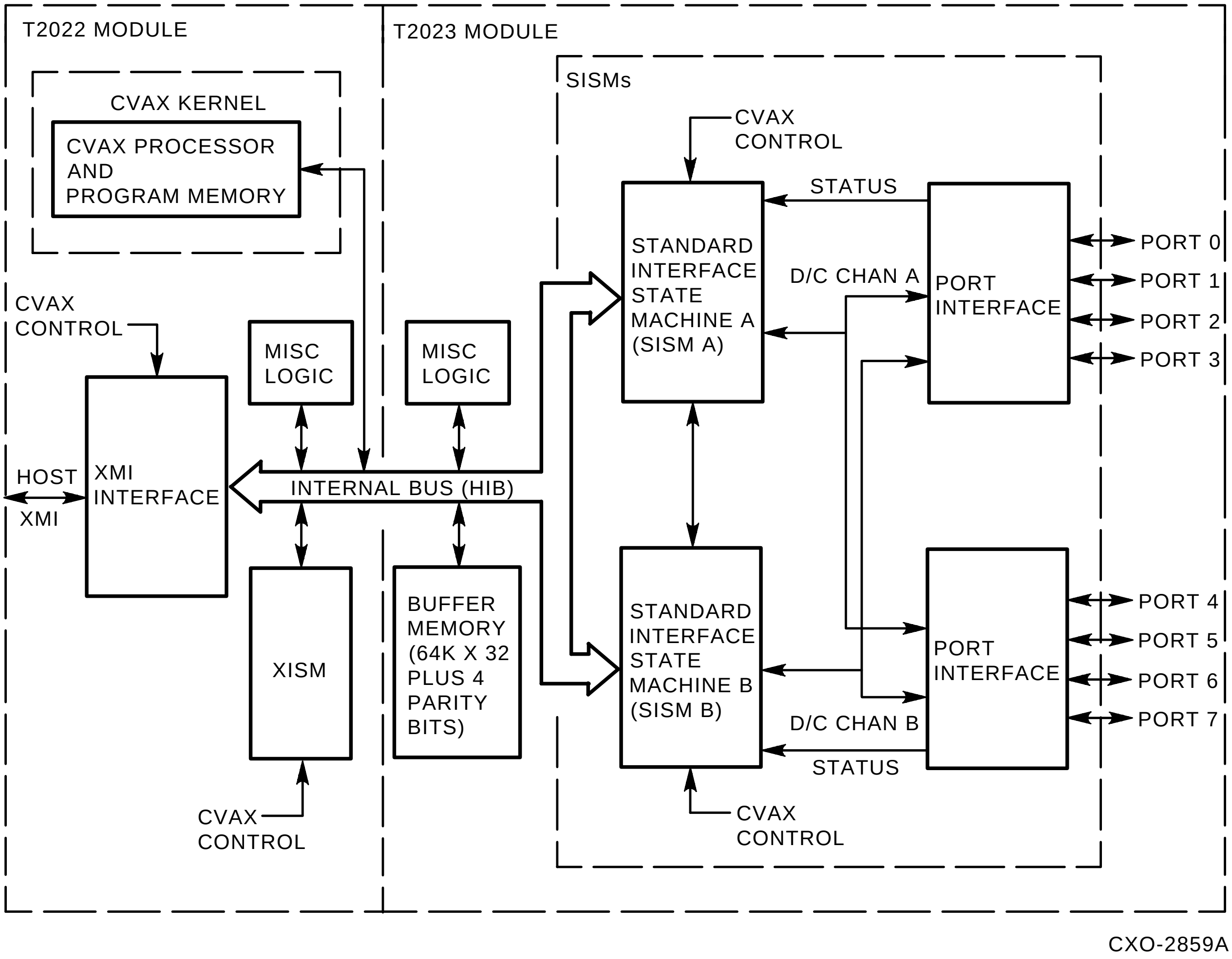
1. The policy processor (CVAX) and policy software are external to the data transfer path.
2. State machines can execute data transfers simultaneously with other state machines.
3. Data buffers have a defined format that contain 512 data bytes and a 2-byte error detection code (EDC).
4. CVAX and state machines execute extensive internal diagnostic tests on power-up or on demand.

The KDM70 controller’s buffer memory consists of three types of data structures:

1. Work rings provide a circular work schedule of order-independent work requests for state machines.
2. Work lists compose the work ring and schedule order-dependent work for state machines.
3. Work blocks compose the work lists and define work for state machines.

State machines traverse work rings, looking for and executing work blocks, without intervention from policy software.

Figure 2–1 KDM70 Controller Subsystem Functional Block Diagram



2.3.1 Buffer Memory and Data Structures

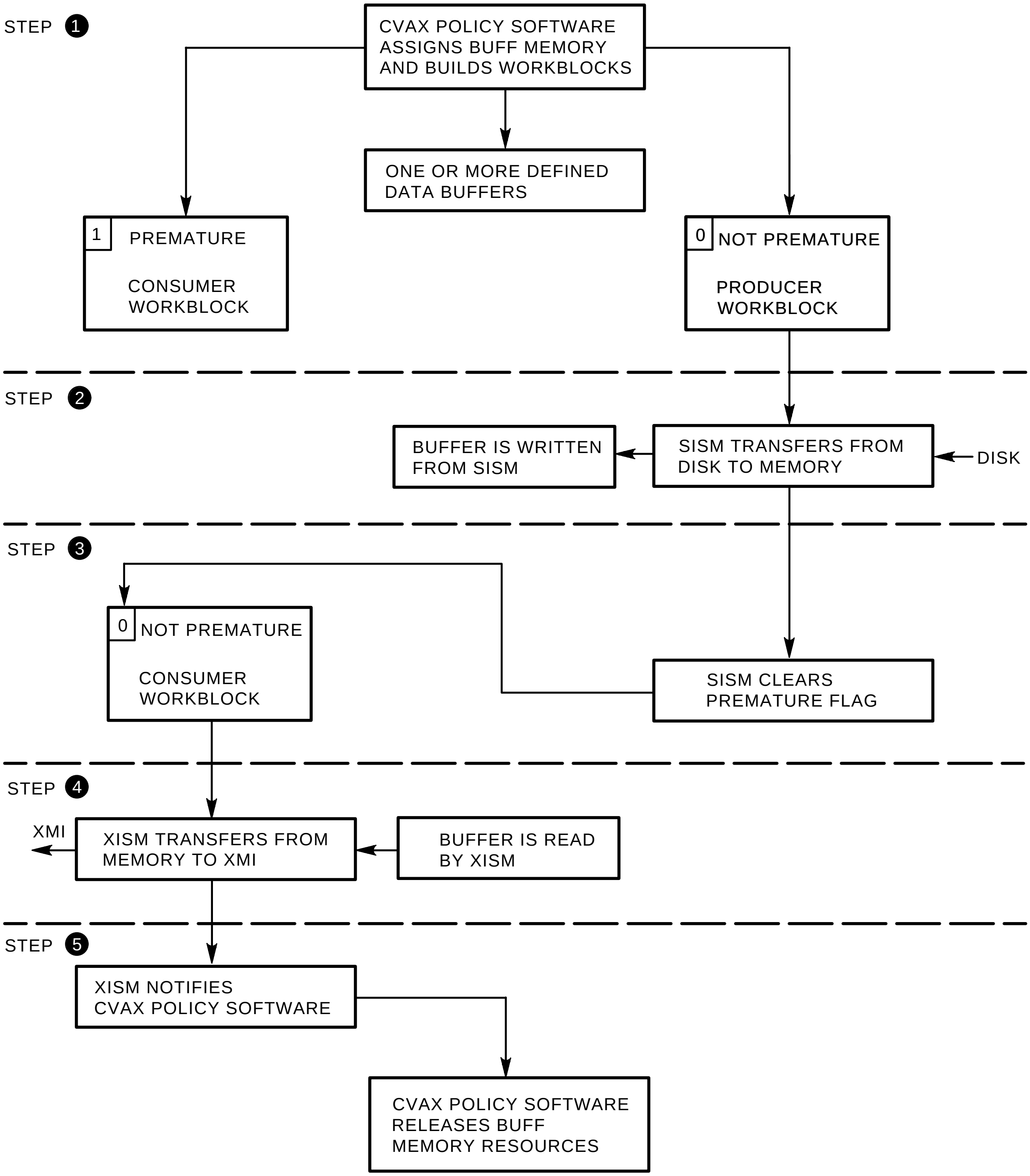
Producer and consumer work blocks transfer data between host memory and storage devices. A work block is flagged as a consumer or a producer work block when it is created by the CVAX. A bit position in the work block is defined as the premature flag.

A work block with a clear premature flag is a producer work block and can execute immediately. Producer work blocks cause a state machine to write KDM70 data buffers with data obtained from host memory or a storage device.

A work block that contains a set premature flag is a consumer work block. It cannot execute until a state machine completes the associated producer work block without error and clears the premature flag. Consumer work blocks cause a state machine to read KDM70 data buffers and transfer the data to host memory or to a storage device.

Figure 2–2 and the following example illustrate a transfer between a disk drive and the host memory system. The example assumes the host requested a read of one disk sector.

Figure 2–2 Functional Example of a Data Transfer


STEP 1
CVAX POLICY SOFTWARE ASSIGNS BUFF MEMORY AND BUILDS WORKBLOCKS
ONE OR MORE DEFINED DATA BUFFERS
1 PREMATURE
CONSUMER WORKBLOCK
0 NOT PREMATURE
PRODUCER WORKBLOCK
STEP 2
BUFFER IS WRITTEN FROM SISM
SISM TRANSFERS FROM DISK TO MEMORY
DISK
STEP 3
0 NOT PREMATURE
CONSUMER WORKBLOCK
SISM CLEARS PREMATURE FLAG
STEP 4
XMI
XISM TRANSFERS FROM MEMORY TO XMI
BUFFER IS READ BY XISM
STEP 5
XISM NOTIFIES CVAX POLICY SOFTWARE
CVAX POLICY SOFTWARE RELEASES BUFF MEMORY RESOURCES

CXO-2860A

❶ The policy software builds two work blocks defining the transfer as a two-step process. Each work block uses the same data buffer in KDM70 buffer memory.

- The first work block, a producer work block, defines a transfer from a disk drive to the KDM70 data buffer.
- The second work block, a consumer work block, defines a transfer from the KDM70 data buffer to a host memory buffer.

❷ The SISM transfers the requested disk sector to the prescribed KDM70 data buffer.

❸ If the transfer completes without error, the SISM clears the premature flag in the consumer work block. (Any detected error prevents modification of the premature flag in the consumer work block.)

❹ The XISM detects the premature flag is clear and executes the consumer work block to transfer the KDM70 data buffer to the host memory buffer.

❺ When the transfer completes, the consumer state machine (XISM, in this example) writes completion status into a status buffer and notifies the policy software that the transfer completed.

2.3.2 The CVAX Kernel

The CVAX kernel executes policy software characterizing the KDM70 controller as an intelligent DSA controller. The CVAX kernel performs the following functions:

- Maintains the logical communication path with host class drivers
- Determines storage device characteristics
- Reduces MSCP, TMSCP, and DUP command packets into state machine work blocks
- Optimizes transfer requests
- Implements error-recover strategies

To maintain the logical communication path, the CVAX transfers command and response packets with the host memory system, as explained in the following list:

1. When the CVAX requires a command packet, it requests the XMI interface to execute a series of quadword or octaword reads from the host command buffer.
2. As each read completes, the CVAX transfers the received data to the CVAX memory system.
3. When the CVAX returns a response packet to the host response buffer, the process reverses.
4. The CVAX transfers data from its memory to the XMI interface, then requests the XMI interface to perform a quadword or octaword write to the host response buffer.

5 The process continues until the response is completely transferred.

When a device is brought on line, the storage device characteristics must be defined. The CVAX policy software uses device characteristics to manage data transfers with the storage device.

High-level host requests in the form of command packets are reduced into state machine work blocks by the controller. A host request to read a single sector may require a number of work blocks to perform the following functions:

- Initiate a seek operation to the disk drive
- Test if the seek operation completed properly
- Transfer data from the disk sector to a KDM70 data buffer
- Complete the data transfer by moving the data from the KDM70 data buffer to host memory

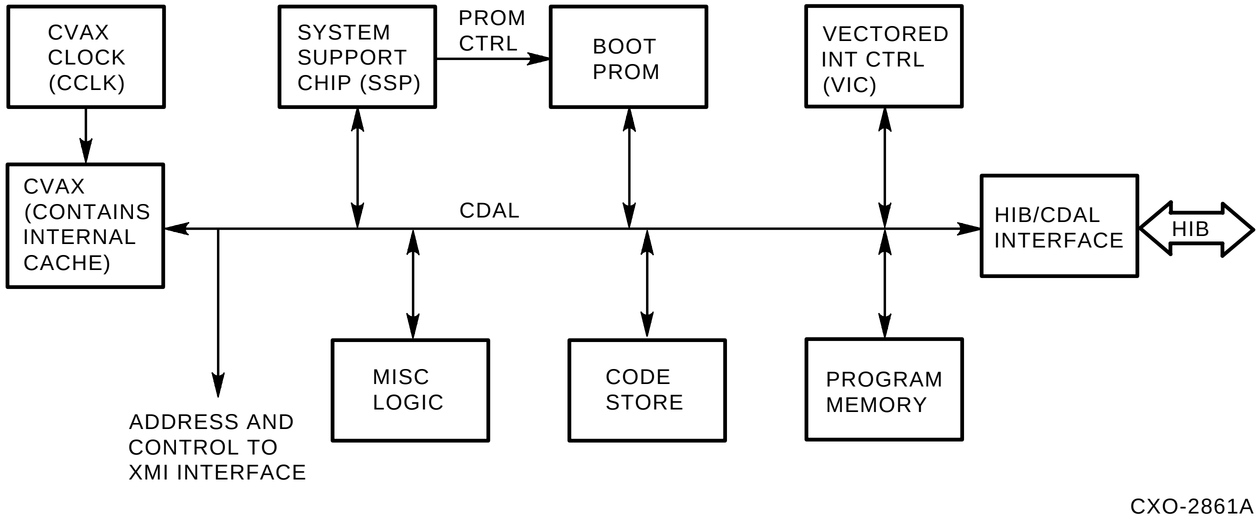
Optimization maximizes data throughput by ordering work blocks to use device geometry. Optimization is possible only with random access devices, such as disk drives.

Note: Because tape drives are sequential access devices, optimization of tape transfers is not possible.

Error-recovery procedures make storage devices appear error free to the host system. The policy software implements error-recovery strategies defined, in part, by the storage device. In cases where error recovery is not possible, the KDM70 controller notifies the host system of the error.

Figure 2–3 provides a block diagram of the CVAX Kernel.

Figure 2–3 CVAX Kernel Block Diagram



The components in Figure 2–3 are explained in the the following table:

Component	Description
CVAX	Provides a 32-bit virtual memory microprocessor with an internal cache system. The CVAX executes the policy software contained in program memory.
CVAX clock (CCLK)	Provides the KDM70 controller with system clocks. It also synchronizes VIC or SSC operations.
System support chip	Supplies the interval timer to the CVAX and interfaces the Boot PROM to the CVAX.
Vectored interrupt controller (VIC)	Provides a programmable interrupt controller. It prioritizes interrupt requests from state machines, the XMI interface, and the KDM70 internal bus (HIB) error logic and assigns them to CVAX interrupt request levels. The VIC supplies the interrupt vector to the CVAX when the CVAX acknowledges an interrupt asserted by the VIC.
Boot programmable read-only memory (PROM)	Supports 32 Kbytes of CVAX start-up code. Start-up code verifies the integrity of the CVAX Kernel before loading program memory and control store random access memory (CRAM) of the state machines.
Code store	Holds 384 Kbytes of code image. The code image includes policy software loaded into program memory and microcode loaded into state machine CRAM after system initialization.

Component	Description
Program memory	Contains executable policy software loaded from code store and provides working areas for CVAX policy software. Program memory is protected by byte parity and supports CVAX cache operations. Because program memory is on the CVAX side of the HIB/CDAL interface, it effectively removes the CVAX Kernel from the main data path of the KDM70 controller.
KDM70 internal bus to CVAX data and address lines (HIB/CDAL) interface	Stores address and data for CVAX-initiated references to devices on the HIB. Those devices include state machines, buffer memory, and various diagnostic registers.
Miscellaneous logic	Includes diagnostic registers, error registers, and programmable array logic (PAL) to support the CVAX Kernel.

2.3.3 The XMI State Machine

The XMI state machine (XISM) manages block data transfers between host memory and KDM70 buffer memory. Each transfer uses standard 512-byte data buffers. As each buffer transfers (partial data buffers are zero filled), the XISM generates an error detection code (EDC) across the data buffer. When the buffer transfer completes, the XISM performs the following functions:

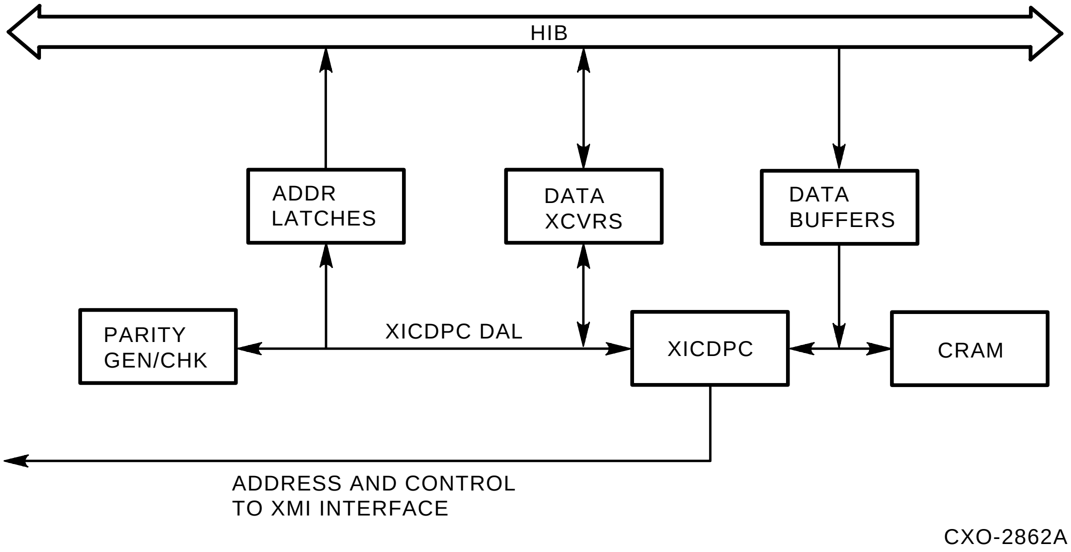
- Compares the generated EDC to the EDC read from the KDM70 buffer memory (buffer memory read operations)
- Stores the generated EDC in buffer memory immediately following byte 512 of the data buffer (buffer memory write operations)

The XISM performs all block data transfers with KDM70 buffer memory. Block transfers are controlled by work blocks defining the following operations:

- Compare operations between a host memory buffer and an KDM70 memory buffer
- Compare operations between two KDM70 memory buffers
- Read transfers from a host memory buffer to a KDM70 memory buffer
- Write transfers from a KDM70 memory buffer to a host memory buffer

Figure 2–4 shows a block diagram of the XISM.

Figure 2–4 XISM Block Diagram



The XISM in Figure 2–4 contains the following components:

Component	Description
Data transceivers	Permit the XISM to transfer data with the buffer memory system
CRAM	Contains microcode to control XICDPC operations
Data buffers	Permit CVAX loading of CRAM after system initialization
Address latches	Drive the HIB address bus with addresses of source or target KDM70 buffer memory locations
XICDPC	Functions as the XISM data path controller
Parity generator/checker	Maintains the integrity of XISM data transfers

2.3.4 The XMI Interface

The XMI interface manages communications with the host bus. It implements XMI protocol required to perform XMI transactions and provides command and data ports to the CVAX and the XISM.

The CVAX control and data port consists of registers that queue CVAX requests for XMI transactions. The registers hold:

- XMI commands (read, write, or interrupt)
- Addresses of host memory source or destination data
- Interrupt vectors
- Interrupt levels

The registers also provide an octaword buffer to temporarily store data. The octaword buffer permits the XMI interface to transfer up to an octaword per CVAX requested transaction. The CVAX transfers data to or from the octaword buffer and CVAX program memory.

The XISM control and data port is similar to the CVAX control and data port. It also contains registers to hold the XMI command (read or write) and addresses of host memory source or destination data. However, the XISM control and data port has two octaword data buffers. The octaword buffers wrap around to provide a circular address space. This allows the XMI interface to transfer one octaword buffer to host memory while the XISM transfers the other octaword buffer to KDM70 buffer memory.

Note: Transfers must complement each other. If the XMI interface is writing one octaword buffer, the XISM can only read the other octaword buffer. The reverse is also true.

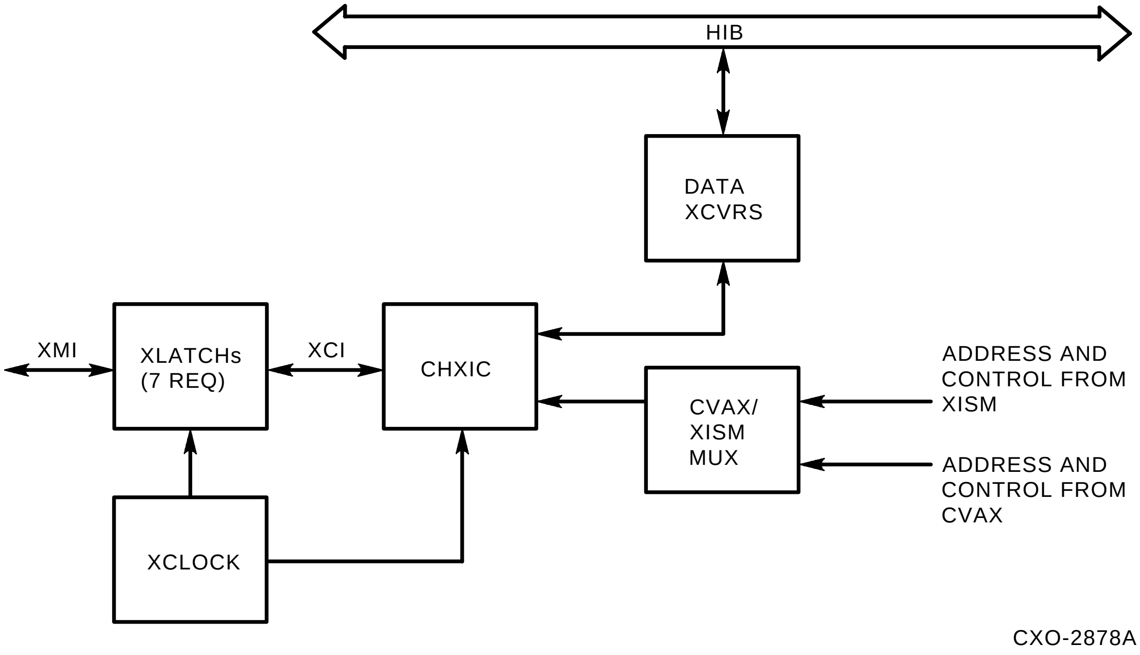
Because the CVAX and the XISM can request XMI transactions, the XMI interface may have two commands pending internally. But, only one transaction will be pending on the XMI bus.

Note: If commands are queued to both control and data ports, the XMI interface alternates between commands, unless policy software requests:

- **XISM transactions to be suspended**
- **A specific number of XISM transactions to complete**

Figure 2–5 provides a block diagram of the XMI interface.

Figure 2–5 XMI Interface Block Diagram



The components in Figure 2–5 are described in the following table:

Component	Description
XLATCH	Provides the primary interface to the XMI bus. Each XLATCH includes a read path and a transmit path. The read path transfers the contents of the XMI to the XCI (XMI chip interface). The transmit path transfers data from the XCI to the XMI. To facilitate pre-fetching of data for write-type operations, the transmit path contains two stages.
XCLOCK	Provides two sets of clocks synchronized to the XMI. One set drives the XLATCHs, the other set drives the CXHIC.
CXHIC	Provides the protocol required to interface to the XMI bus. Included in the CXHIC are the XMI required registers and the octaword buffers to support transfers with the host system.
Data transceivers	Interface the KDM70 HIB to the CXHIC.
CVAX/XISM multiplexer	Selects address and control information from the CVAX or the XISM. The address selects a CXHIC register. The control information specifies the type of register operation.

2.3.5 SI State Machine (SISM)

Two SISMs perform data transfers and diagnostic tests. Each SISM executes work blocks that define the following operations:

- Command and response transfers

- Diagnostic tests
- Disk read or write transfers
- Disk format transfers
- Tape read or write transfers

Command and response transfers, as well as diagnostic tests, can transfer less than standard length buffers. Disk or tape transfers use standard 512-byte data buffers. Each buffer is protected by an error detection code (EDC) and, in the case of disk buffers, an error correcting code (ECC).

Figure 2–6 shows a block diagram of the SISMs, including two each of the following:

- CSIC (CMOS storage interface controller)
- SIECL (storage interface emitter coupled logic)
- HIB interface
- CRAM (control-store random access memory)
- CRAM data transceivers
- Pulse transformers

Each CSIC converts data from the parallel format of the KDM70 HIB to the serial format required by the SIECL. The CSIC connects to each SIECL through data and control channels. The CSIC connects to one SIECL through a local port status path and to the other CSIC through a synchronization and remote test multiplexer path. As a result, the CSIC can access all eight ports of the KDM70 controller.

The CSIC performs the following functions:

- Generates the ECC for disk write data
- Checks the ECC for disk read data
- Generates EDC for every block transfer to KDM70 buffer memory
- Checks EDC on every block transfer from KDM70 buffer memory
- Detects header compare errors during disk read or write operations
- Implements some of the SDI and STI protocols

Each SIECL converts data from the CSIC serial format to a pulse-encoded format required on the SDI/STI bus. The SIECL connects to both CSICs through data and control channels and to one CSIC through a local port status path. The two data and control channels allow the SIECL ports to transfer data simultaneously over two ports. One transfer would be controlled by the CSIC connected to channel A, the other transfer would be controlled by the CSIC connected to channel B. The local port status path allows the SIECL to drive the status of its four ports to the local CSIC.

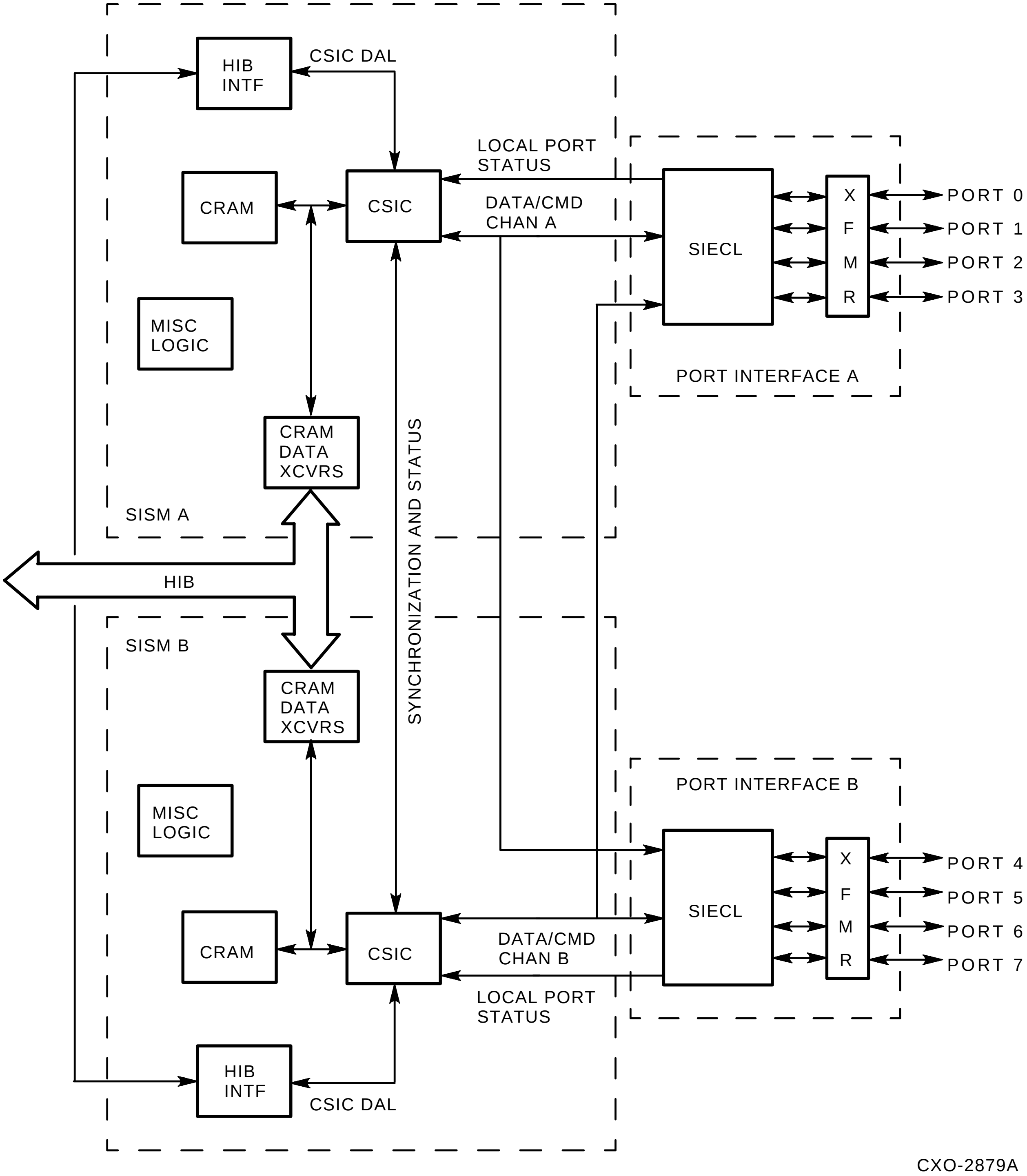
Each HIB interface transfers address, data, and control information between the HIB and the CSIC data and address lines (DALs).

Each CRAM contains microcode to control a CSIC. Microcode is parity protected and can be read or written by the CVAX kernel.

CRAM data transceivers interface CRAM to the HIB and permit the CVAX to load and verify SISM microcode during controller initialization.

Pulse transformers isolate the SDI/STI bus from the SIECL.

Figure 2–6 SISMs Block Diagram



2.3.5.1 Buffer Memory Temporarily Stores Data and Data Structures

Buffer memory provides 256 Kbytes of temporary storage for state machine data structures and data buffers. The data structures, which are work blocks linked as work lists and work rings, define and schedule work for the state machine. The data buffers hold data that is transferred between the host memory and the external storage devices.

2.3.6 Composite Block Diagram of the KDM70 Controller

Figure 2–7 and Figure 2–8 provide a complete block diagram of the KDM70 controller.

Figure 2–7 KDM70 T2022 Module Block Diagram

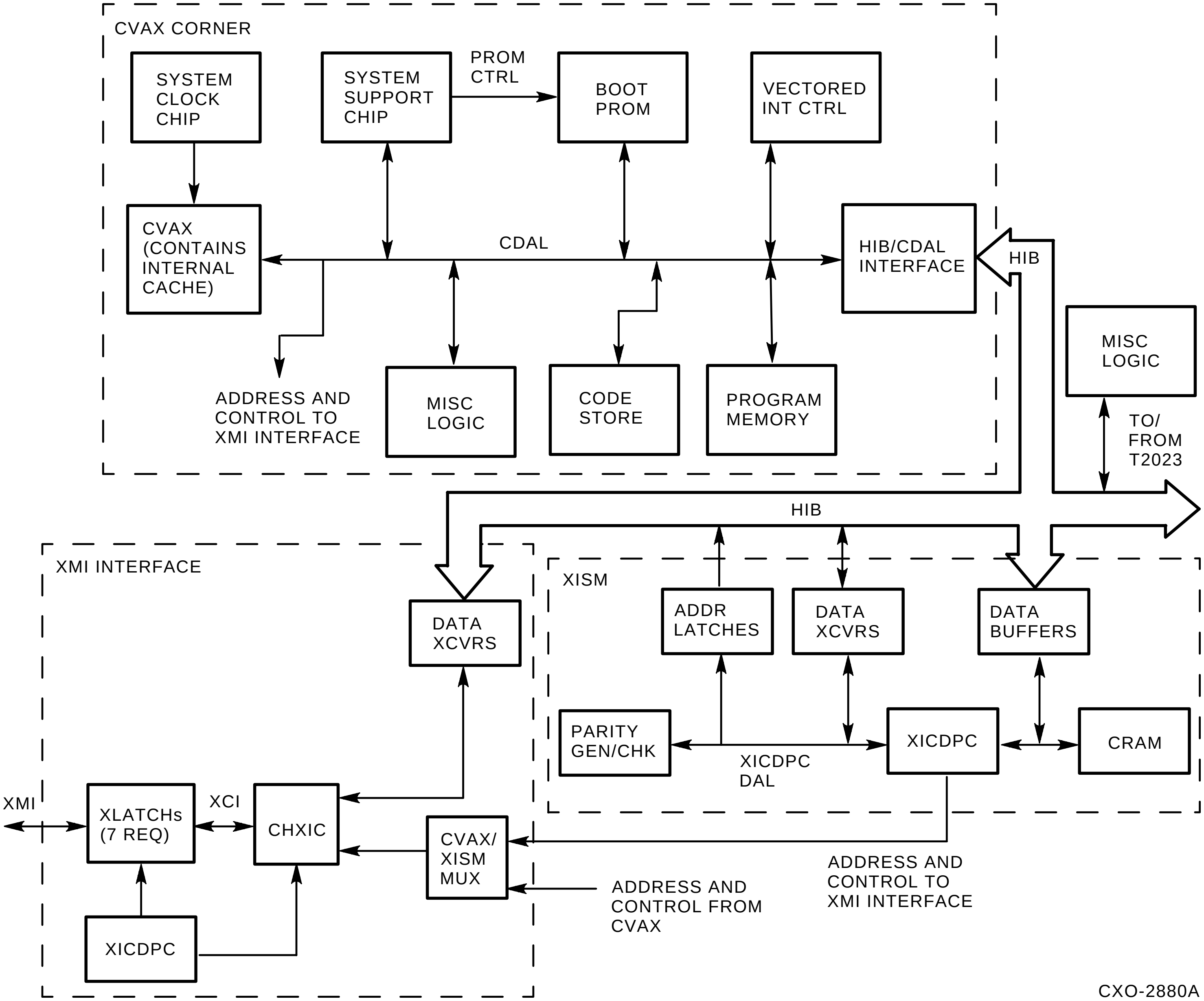
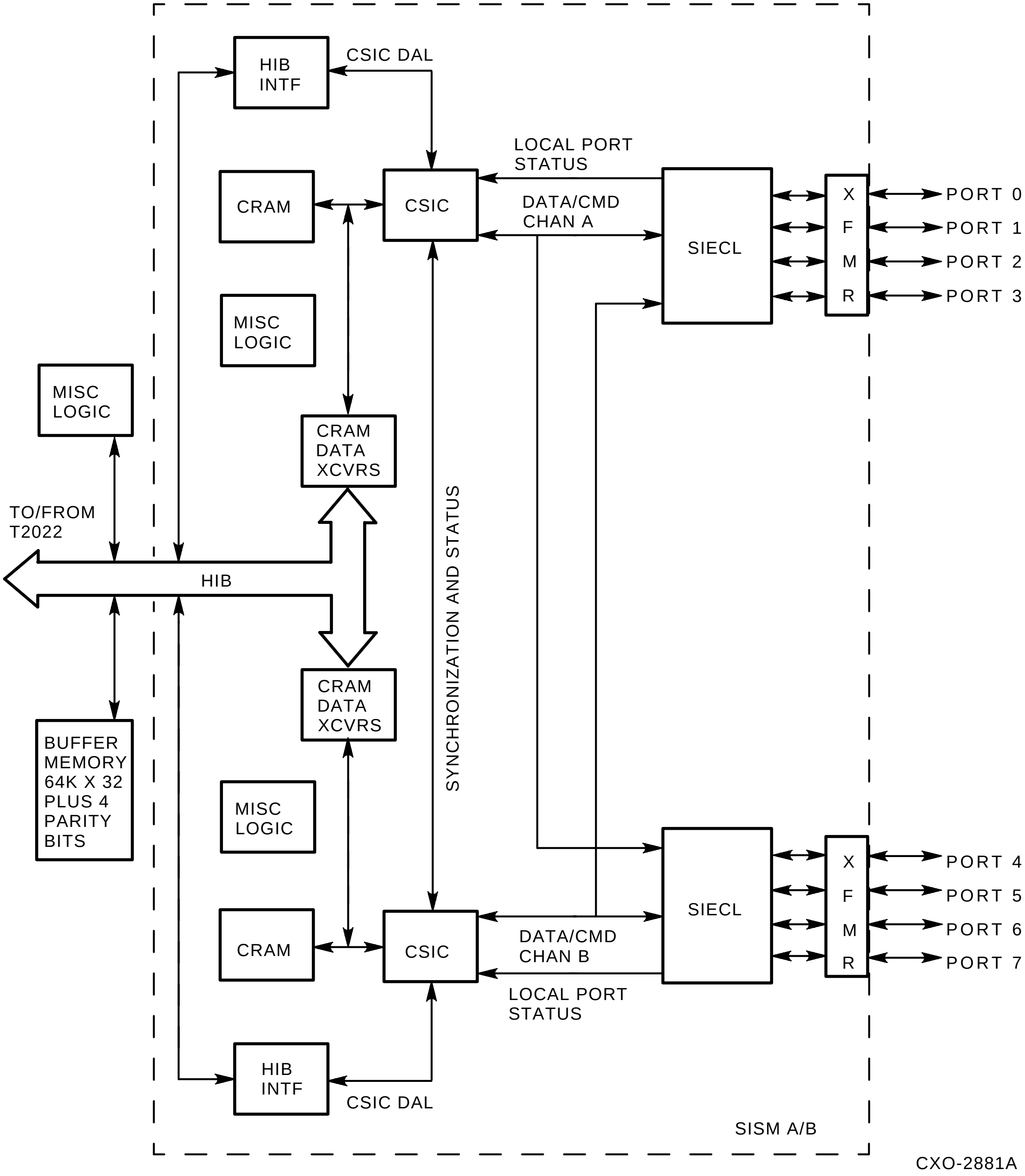


Figure 2–8 KDM70 T2023 Module Block Diagram



2.4 Software Architecture Overview

The KDM70 software is stored in nonvolatile ROM (EEPROM) memory on the T2022 processor module. The KDM70 software architecture is divided into two distinct layers (Figure 2–9):

1. Server layer
2. Device layer

The server layer consists of code that implements disk and tape MSCP and DUP protocol and also implements the high-level diagnostic interface into these servers.

The second layer, the device layer, consists of components unique to the hardware implementation. Disk communication management, SDI disk data error recovery or the XMI port layer are all examples of device layers that manipulate data structures and hardware in providing services to the server layer.

Figure 2–9 Server Layer/Device Layer Diagram

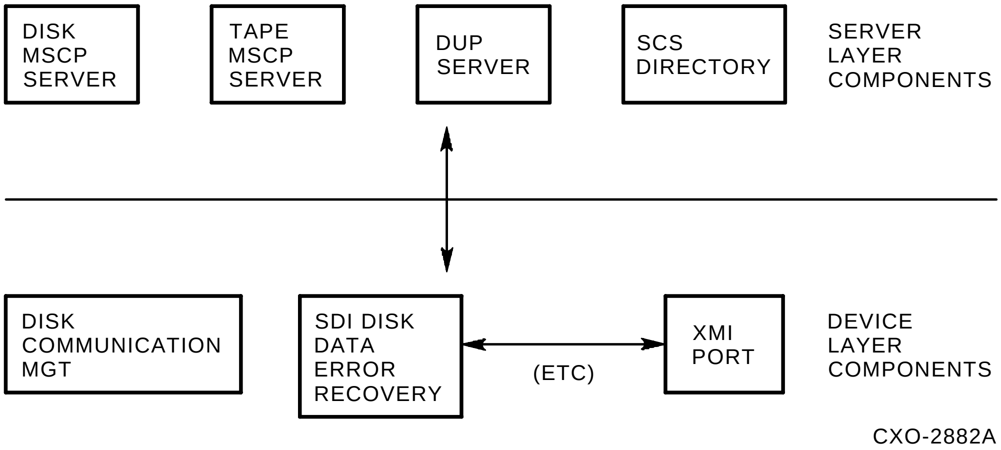
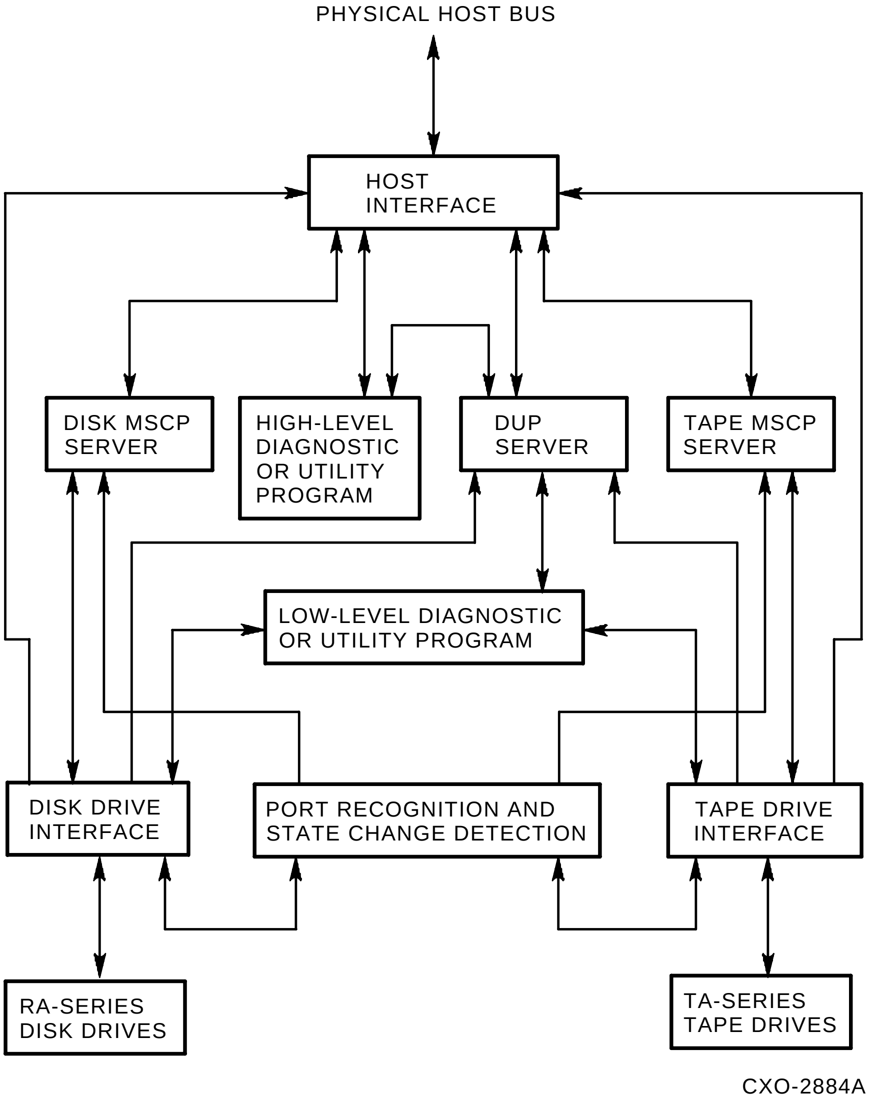


Figure 2–10 represents the overall KDM70 software architecture. The functional blocks in this figure are not related to the server/device layers in Figure 2–9. Functional blocks in Figure 2–10 may contain components from both layers. Each functional block represents a collection of smaller software compilable components called source modules. In turn, source modules are comprised of code written procedures.

Figure 2–10 KDM70 Software Block Diagram



2.5 Thread Structure

The functional blocks in Figure 2–10 operate independently and in parallel. Functions operate in a separate machine context (hardware context) from other functions. The interfaces between functions consist of queues, semaphores and data structures. Note that a machine's hardware context is defined by the state of general registers, stack pointer, program counter, and processor status register. To execute functional threads in a logical, orderly fashion, the exec provides synchronization services and scheduling for the multiple processes waiting to run. The exec uses queues, semaphores, and timers in synchronizing the various threads and services provided by the KDM70 controller.

2.5.1 Host Interface

The host interface controls the host bus hardware and implements port communications policy, the system service protocol (SSP) connection, and communications policies. The host interface provides message and block data services, as well as system communication services (SCS) and directory services to the servers.

2.5.2 Disk MSCP Server

The MSCP server interfaces to the host disk class driver through the host interface. Physical drive services required by MSCP protocol are invoked through the disk drive interface. The disk MSCP services are also provided to high-level diagnostic and DUP programs through the host disk class driver. Finally, drive state changes are monitored by the disk MSCP server through the port recognition and state change function.

2.5.3 Tape MSCP Server

The tape MSCP server is the analog function of the disk MSCP server used to control tape devices.

2.5.4 Diagnostic Utility Protocol Server

The diagnostic utility protocol server manages connection and communication services to host-based and on-board diagnostics. KDM70 DUP interfaces to the host-based DUP class driver through the host interface.

2.5.5 Disk Drive Interface

The disk drive interface controls policy for disk hardware, drive communications, and drive transfers. The disk function provides the following services:

- Transfers data services to the disk MSCP server
- Provides low-level diagnostic and utility programs
- Recognizes the port
- Detects state changes

The disk drive interface uses the host interface to map host buffer operations required for data transfers from the host to storage.

2.5.6 Tape Drive Interface

The tape drive interface is the tape equivalent of the disk drive interface, which uses TMSCP.

2.5.7 Port Recognition and State Change Detection

The port recognition and state change detection function monitors SI port state changes and device state changes. State changes are reported to the disk or tape MSCP server. These functions also help the disk or tape drive interfaces determine the type of device attached to a particular SI port.

2.5.8 High-Level Diagnostic and Utility Program

The high-level diagnostic/utility function works with the DUP server to control program execution and operates on targeted devices. Normal disk and tape services are provided by the disk/tape MSCP servers through the host interface. Programs run in parallel with normal controller functions.

2.5.9 Low-Level Diagnostic and Utility Program

The low-level diagnostic and utility function uses the DUP server to control program execution and operate on targeted devices. Programs run in parallel with normal controller functions.

2.6 Software Transfer Command Flow

The software transfer command flow is a high-level overview of how the KDM70 software processes transfer commands. This example assumes an error-free transfer. Relevant data structures are included in this overview.

2.6.1 KDM70 Software Concepts and Constructs

The KDM70 software is divided into software modules that perform specific tasks. Each module is connected to other modules through a series of linked data structures, queues, and semaphores.

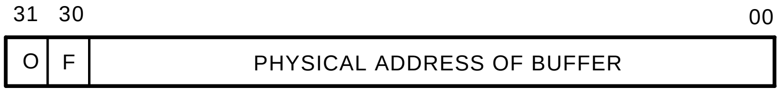
Executing processes or tasks in the KDM70 controller cannot be interrupted by a waiting process. The executing process runs to completion and then clears its own scheduling bit, indicating completion of the its task.

Software data structures are packets of information used to pass device characteristics and status information. Data structures are used to pass MSCP commands and associated flags and command modifiers. Data structures include command and response packets, which allow the host, controller, and target devices to communicate their status and the status of MSCP commands.

2.6.2 Host Data Structures: Command Ring and Response Ring

Host memory uses two data structures called the command ring and the response ring to pass and receive MSCP commands and MSCP-related information (Figure 2–11). Both rings are circular buffers. The host uses these data structures to store pointers to areas of host memory that contain MSCP command packets. These packets also contain system services protocol (SSP) information used in performing system overhead functions.

Figure 2–11 Command and Response Rings



O BIT = OWNERSHIP BIT
0 = HOST OWNS DESCRIPTOR
1 = PORT OWNS DESCRIPTOR

F BIT = FLAG BIT
PORT RETURNS DESCRIPTOR TO HOST:
0 = DESCRIPTOR PROCESSING NOT COMPLETED
1 = DESCRIPTOR PROCESSING COMPLETED
HOST RETURNS DESCRIPTOR TO PORT:
0 = PORT NOT TO INTERRUPT ON RING TRANSITION *
1 = PORT IS TO INTERRUPT ON RING TRANSITION *

* RING TRANSITION = COMMAND RING GOING FROM FULL TO NOT - FULL
RESPONSE RING GOING FROM EMPTY TO NOT - EMPTY

CXO-2885A

The KDM70 controller checks the state of the high-order bit in the command ring to determine if there is an outstanding MSCP command (packet). A set high-order bit indicates an outstanding command. The KDM70 notifies the host that it has accessed either the command ring or response ring by writing a non-zero value to a register called the command interrupt reason or response interrupt reason. This, in turn, generates an interrupt to the host.

2.6.3 Device Data Structures: Ringleader, SI Work Blocks, and XI Work Blocks

The device side of the controller includes the ringleader. One ringleader exists for each port. The ringleader is a circular structure with the tail pointing to the head of the structure (Figure 2–12). Processed work blocks are placed on the ringleader in the order received. Work blocks cause the controller state machines (SI and XMI) to do work by sending or receiving data to and from the device attached to the KDM70 ports.

Figure 2–12 Ringleader

ADDRESS		
0	SMPTR *SI_PORT_RING_LEADER	SISM RING LEADER POINTER
4	SMWB *SI_NORMAL_WL_PTR	IF "NON_EMPTY", DIRECT POINTER
8	SMWB *SI_ERROR_WL_PTR	IF "NON_EMPTY", DIRECT POINTER
12	SMPTR *XI_PORT_RING_LEADER	XI RING LEADER POINTER
16	SMWB *XI_NORMAL_WL_PTR	IF "NON_EMPTY", DIRECT POINTER
20	SMWB *XI_ERROR_WL_PTR	IF "NON_EMPTY", DIRECT POINTER
24	BITFIELD: FLAGS	
28	LINKS *FIRST	QUEUE OF CURRENT TRANSACTIONS
32	LINKS *LAST	QUEUE OF CURRENT TRANSACTIONS
36	_RLEADER *SI_NEXT_RL_PTR	POINTER TO SI RING LEADER FOR NEXT PORT
40	_RLEADER *XI_NEXT_RL_PTR	POINTER TO XMI RING LEADER FOR NEXT PORT

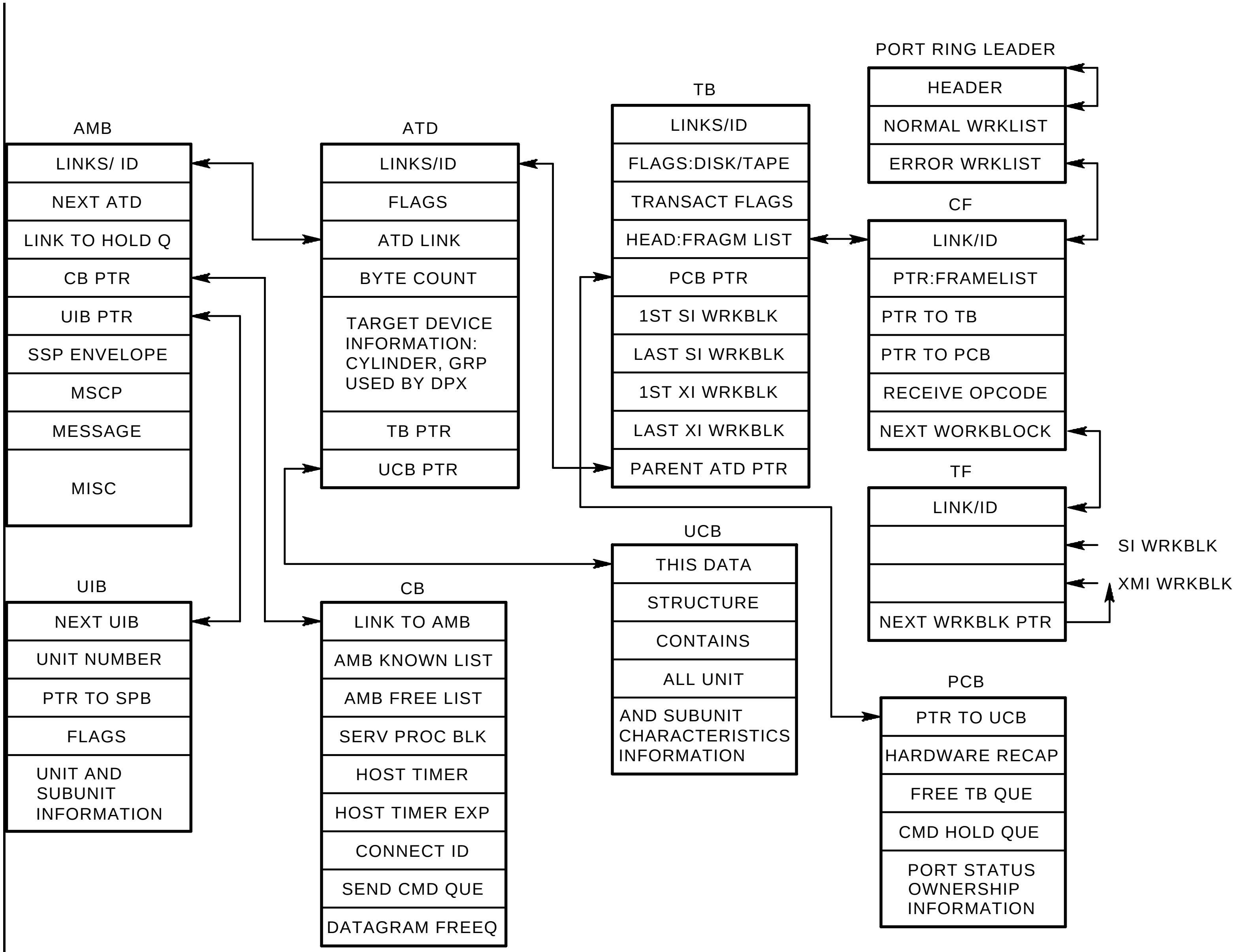
CXO-2886A

2.7 Transfer Command Flow Through the KDM70 Controller

This section describes an MSCP disk transfer command through the KDM70 controller. The terms data structure, structure, and buffers are used interchangeably. Threads, tasks, and processes are also used interchangeably. Refer to Figure 2–13 during this discussion.

When the host has work for the controller to perform, the host places an address of an MSCP command packet in the host command ring. This action sets the high-order bit in the command ring, which generates an interrupt to the controller. The controller, through its executive function, schedules the host port thread to run.

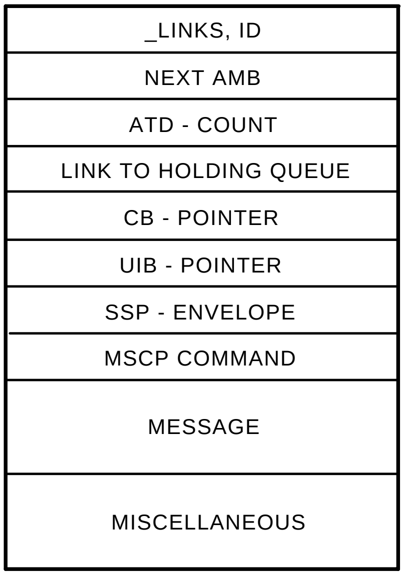
Figure 2–13 Buffers Associated with a Transfer Command



CXO-2887A

The host port thread reads in the MSCP packet from the referenced host address. The MSCP packet is stored in a data structure called the application message block (AMB) in controller program memory (Figure 2–14). The AMB also contains SSP-related information, state context information, and pointers to other data structures used to process the MSCP command.

Figure 2–14 Application Message Block



CXO-2888A

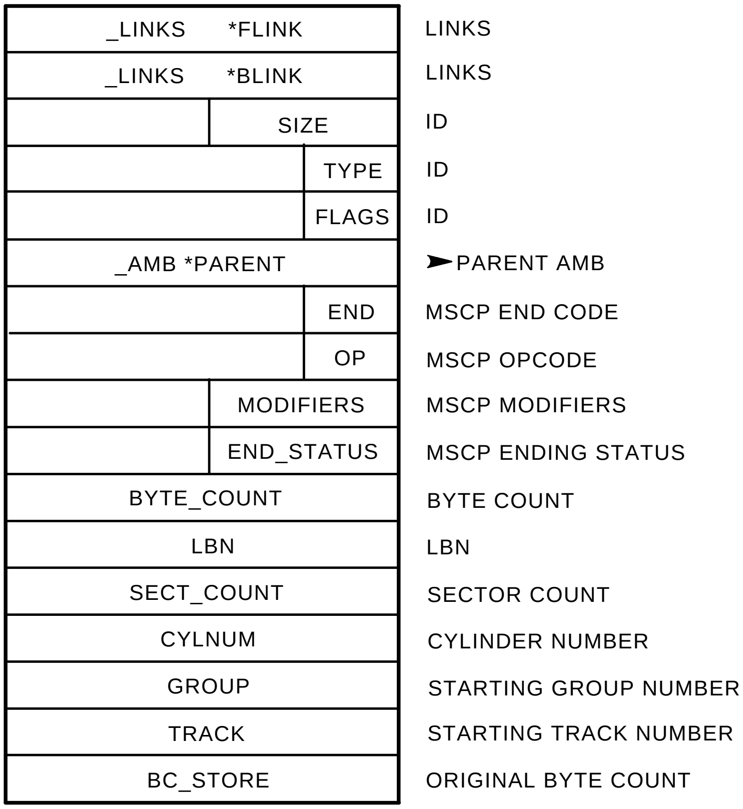
The host port thread then passes the AMB to the disk server thread, clears its own scheduling bit, and returns to a wait state.

The arrival of the AMB on the disk server threads input queue causes the disk server's scheduling bit to be set. The now executing disk server thread validates the AMB by comparing the MSCP command, including any flags or modifiers, to a table of legal MSCP opcodes.

Once the AMB has been validated, the disk server thread compares the target device's unit number, referenced in the MSCP command, to the units associated unit control block (UCB). The UCB is a data structure containing the selected device's unit and subunit characteristics and unit context.

Once the UCB is validated and the device is ready for a transfer, the disk server thread locates a data structure called the atomic transfer descriptor (ATD) (Figure 2–15). An ATD represents a transfer of contiguous logical block numbers (LBNs), all within the same cylinder. (If a transfer crosses cylinder boundaries, multiple ATDs must be built to accommodate the transfer.) Relevant MSCP command packet information from the AMB is copied into the ATD, such as byte count, MSCP opcode, end flags, modifiers, and the LBN to be used.

Figure 2–15 Atomic Transfer Descriptor (ATD)



_LINKS *FLINK	LINKS
_LINKS *BLINK	LINKS
SIZE	ID
TYPE	ID
FLAGS	ID
_AMB *PARENT	➢PARENT AMB
END	MSCP END CODE
OP	MSCP OPCODE
MODIFIERS	MSCP MODIFIERS
END_STATUS	MSCP ENDING STATUS
BYTE_COUNT	BYTE COUNT
LBN	LBN
SECT_COUNT	SECTOR COUNT
CYLNUM	CYLINDER NUMBER
GROUP	STARTING GROUP NUMBER
TRACK	STARTING TRACK NUMBER
BC_STORE	ORIGINAL BYTE COUNT

CXO-2889A

The disk server sends the ATD through a queued interface to the disk transfer manager (DPX) (TPX for tape). The disk server clears its scheduling bit and waits for more work.

The DPX receives the ATD from a structure to which the PCB is pointing, called the input queue. The input queue receives new work for each port. The input queue is a standard receive queue with semaphore; that is, it is a blocking queue.

DPX checks if the work pipeline is currently full. If it is full, the work is left on the port input queue. If work can be done, DPX calls a transfer routine.

DPX then allocates a transaction block (TB) (2 per port) for the operation. DPX computes the number of transfer fragments (TFs) and communication fragments (CFs) required for the transfer to occur, and then allocates them.

DPX allocates the needed buffers for the data and calls the disk interface manager to initiate the transfer.

The TB is a holder for the TF and CF. Both the TF and CF contain two work blocks. The TF has an SI side and an XI side. The CF has a send and receive side. The send and receive opcodes show the start of the work blocks.

After DPX builds its TB, TFs, and CFs and allocates the necessary buffers from data memory, DPX sends the TB to the state machine management code. The state machine processes the work block portion of the TFs and CFs and queues the TFs and CFs to the ringleader of the selected port.

After the state machines execute the work blocks, the TB is marked “success normal” and is sent back to the DPX on the PCB hardware recapture queue. If the transaction completes error free, the DPX removes the completed TB from the queue and retires it along with other reserved resources used during the transfer.

DPX then returns the associated ATD back to the disk server thread. The disk server thread retires the ATD and AMB, freeing those resources for other commands. The disk server thread calls a response routine and notifies the host, through the host response ring, that the transfer completed successfully.

3 KDM70 Controller Initialization and Operation

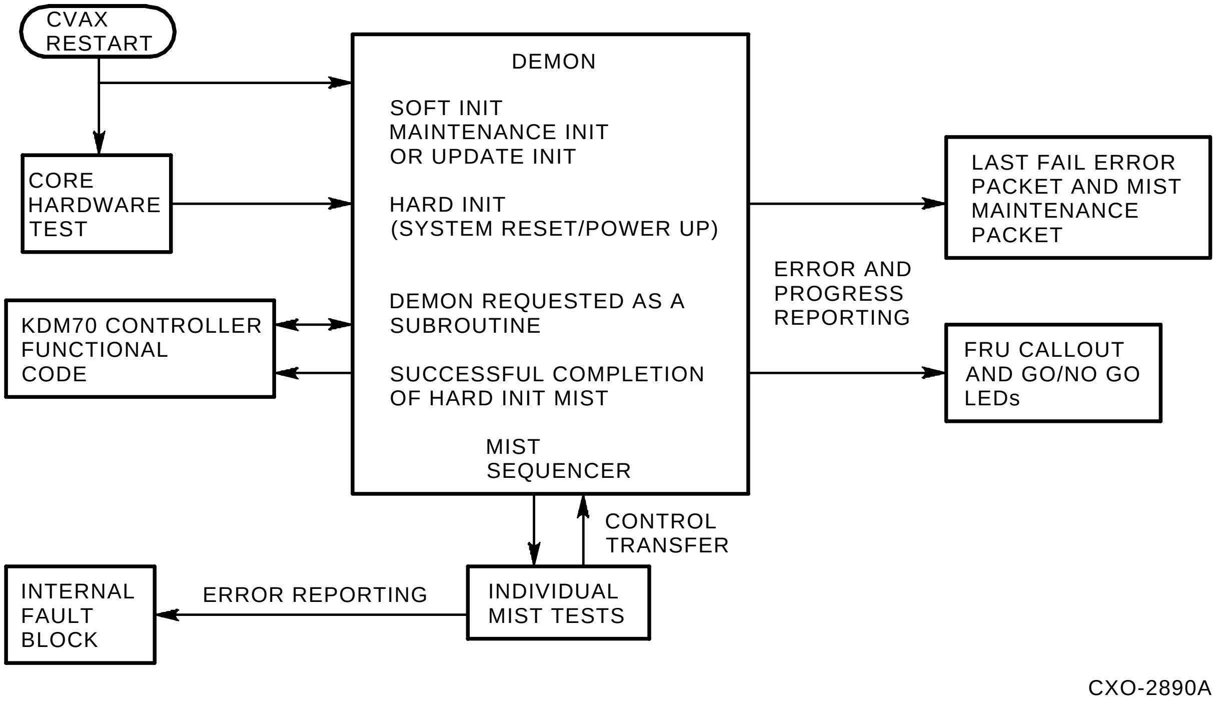
3.1 Introduction

This chapter describes KDM70 controller boot sequence and initialization. This chapter is intended as a high-level overview.

3.2 DEMON Flow

Figure 3–1 shows the logical connections between the diagnostic execution monitor (DEMON), module internal self-test (MIST) and KDM70 controller functional code. Logically represented in the flow is error handling by MIST/DEMON.

Figure 3–1 DEMON Flow


CVAX RESTART
CORE HARDWARE TEST
KDM70 CONTROLLER FUNCTIONAL CODE
DEMON
SOFT INIT
MAINTENANCE INIT
OR UPDATE INIT
HARD INIT
(SYSTEM RESET/POWER UP)
DEMON REQUESTED AS A SUBROUTINE
SUCCESSFUL COMPLETION OF HARD INIT MIST
MIST SEQUENCER
LAST FAIL ERROR PACKET AND MIST MAINTENANCE PACKET
ERROR AND PROGRESS REPORTING
FRU CALLOUT AND GO/NO GO LEDs
CONTROL TRANSFER
INTERNAL FAULT BLOCK
ERROR REPORTING
INDIVIDUAL MIST TESTS
CXO-2890A

3.2.1 KDM70 Core Hardware Initialization Test

The hardcore initialization tests check the sanity of the CVAX microprocessor and supporting hardware. Hardcore tests execute from boot programmable read-only memory (PROM). Refer to Appendix D for a description of hardcore tests.

3.2.2 DEMON

DEMON is invoked after the occurence of one of the following:

- Completion of core hardware tests during a port hard initialization (hard init)
- Port soft initialization (soft init) (requested by the host)
- Maintenance initialization (requested by the host)
- Port update initialization (requested by the host)
- A functional code thread call

A hard init is caused by:

- KDM70 controller power-up
- Host initiation of a node reset
- Manual reset of the KDM70 controller

After the succesful completion of core hardware tests, CVAX control is transferred to the DEMON, which controls further execution of MIST tests.

3.2.3 MIST

MIST performs the following functions:

- Executes all MIST tests
- Tests SI corner
- Verifies RAM memory
- Tests the state machine

Refer to Appendix D for a description of MIST tests.

After MIST has successfully completed, DEMON turns on the Go/No Go LEDs, writes a successful completion code to the FRU callout LEDs, and writes the last crash error log packet.

3.2.4 MIST Errors

If an error occurs during MIST testing, the following steps are performed:

1. MIST FAIL is asserted in the diagnostic write register.
2. Error information is logged to an on-board internal error log (both modules).

3 A host reset of the KDM70 controller is enabled.

4 MIST attempts to clear any errors or write error information into the SA register.

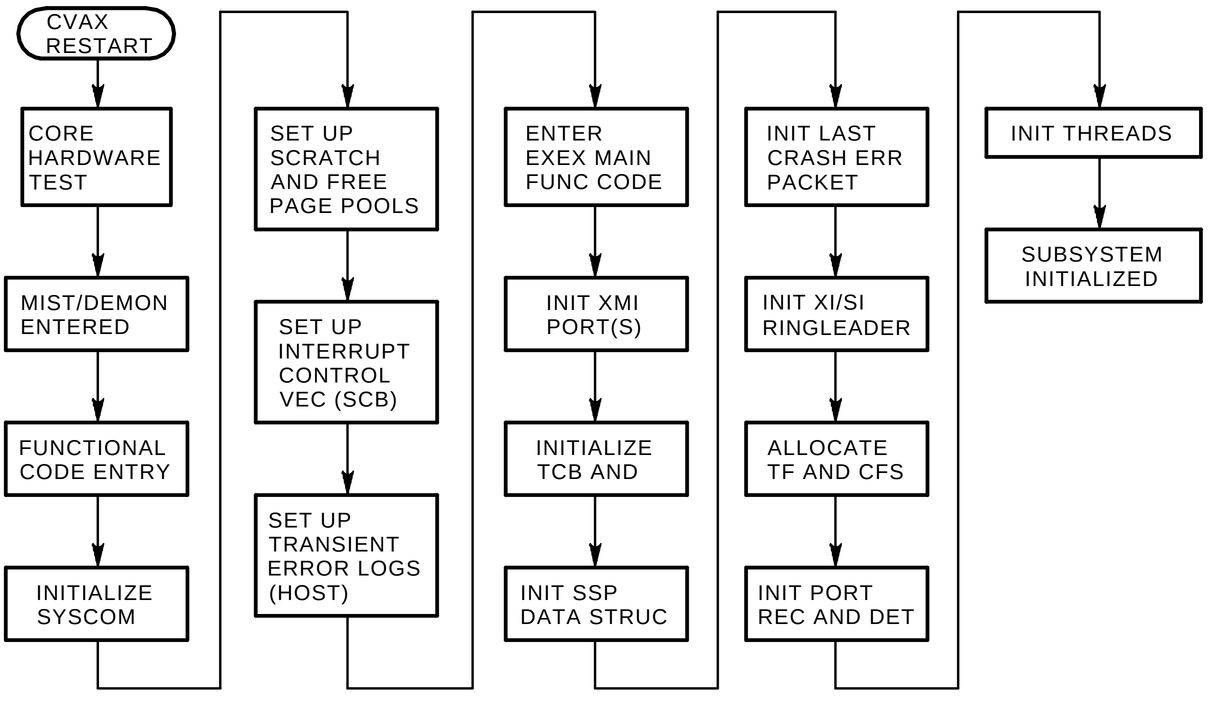
Nonfatal faults are cleared and recovery is attempted. If recovery succeeds, testing continues. If recovery fails, the fault is considered fatal and handled as such.

Soft and maintenance initializations are handled similiarly; however, no hardcore tests are executed.

3.3 Functional Intialization Entry

Once MIST testing has completed, the KDM70 controller functional code is invoked to complete the intialization process. Refer to Figure 3–2.

Figure 3–2 KDM70 Controller Functional Initialization



CXO-2891A

The KDM70 controller functional code takes control from DEMON once MIST has completed. The following components are initialized and set up to complete KDM70 initialization:

SYSCOM
Interrupt control vector tables
Main functional code
Thread control blocks (TCBs)
The last crash fatal error packet
Resources for transaction fragments (TFs) and connection fragments (CFs)
KDM70 controller threads
Scratch and free page pools
Transient error logs
XMI port
Systems services protcol (SSP) data structures
The XI and SI (work blocks) ringleaders
Port recognition and detection (PRD) queues

4 KDM70 Controller Error Analysis

4.1 Troubleshooting Reference Material

Refer to the following documentation when troubleshooting:

- *Getting Started with VAXsimPLUS* (AA–KN79A–TE)
- *VAXsimPLUS User Guide* (AA–KN80A–TE)
- *VAXsimPLUS Field Service Manual* (AA–KN82A–TE)
- *DSA Error Log Manual* (EK–DSAEL–MN)

Note: The *DSA Error Log Manual* provides a general description of DSA error logs. Refer to device-specific error log manuals for detailed error description.

4.2 Introduction

This chapter describes errors that may occur during the KDM70 controller module internal self-test (MIST) and during normal KDM70 controller operation. In addition, this chapter explains how to evaluate errors and determine their cause. For information on errors that occur during a KDM70 in-line diagnostic or utility, refer to the specific diagnostic or utility description.

4.3 Host Error Log

The host error log is used to record error information for eventual use by Digital Customer Services.

The error information in the host error log is in the form of error log packets produced by the host or KDM70 controller in response to error conditions. These error log packets may include specific information on the operation of the KDM70 controller, its attached drives, or other elements of the system (host processor, memory, software, and so on) that may be important in diagnosing problem sources.

The host software records error information in the host error log for the following events related to the KDM70 controller:

1. Controller initialization
2. Controller-detected errors
3. Device errors

Some error log reports may reflect changes in the configuration or operation of the system that are informational and do not represent an error condition. The following are examples of error log reports that are not error conditions:

- Completion of the initialization sequence between the port driver and the KDM70 controller
- Attention messages pertaining to availability of a disk or tape drive

Error log reports provide error/status information that occurred during the error. Error logs have the following benefits:

1. Error information can be obtained without recreating the error with diagnostics.
2. Error information can be obtained on intermittent errors, even if the error cannot be duplicated with diagnostic or exerciser programs.
3. Error information can be obtained without interrupting work.
4. Error reports may indicate possible problem areas, including failing unit serial number, revision levels, error codes, status indicators, and other applicable information.

4.4 KDM70 Controller Errors

There are three major types of KDM70 controller errors. Refer to the appropriate section for fault isolation.

- Module internal self-test (MIST) errors, Section 4.5
- Fault management events detected during self test, Section 4.17
- Fatal controller errors detected by functional code, Section 4.6

4.5 Errors During KDM70 Controller Module Internal Self-Test (MIST)

The KDM70 controller self-test is executed during powerup and also during a node reset. A node reset will occur during system initialization or as requested by the operating system port driver.

Both amber LEDs are turned off at the beginning of MIST. These LEDs will be turned on only if the self-test is successful. A failure during MIST will result in both amber LEDs off, and an error code displayed in the red LEDs on board 1 (T2022). See Table 4–1.

As part of the error reporting for self-test failures, MIST will write a KDM70 conttroller-specific error code in the SA register. This code will provide module isolation and further detail on the test that failed. The SA code is available either in the error log, or by examining the SA register from the local console terminal. This register will contain valid error information only if the error bit (15) is set. Refer to Table B–1 for a complete listing of SA codes.

```
   15                 10                                  0
+------+--+--+--+--+-----------------------------------+
|      |  |  |  |  |                                   |
| ERR  |  |  |  |  |             SA ERROR CODE         |
|      |  |  |  |  |                                   |
+------+--+--+--+--+-----------------------------------+
```

Note: The SA register may be examined from the console at node base address + 44. See Table C–1 to determine the correct node base address.

```
>>> Examine 21900044            ;node 2
>>> 21900044  0000835F          ;Board 2 SRAM Test
```

MIST failures are also written to an internal error log. This information is primarily used by manufacturing for module repair. It will contain an error identifier and other specific failure information. In some cases it may be useful to Digital Customer Services for module isolation.

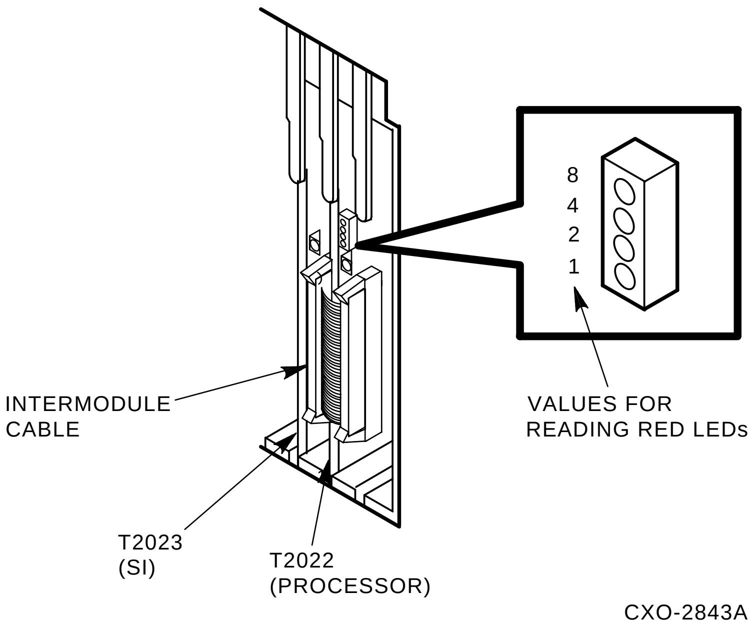
The MIST internal error log is a ring buffer that contains information on the last six errors. This internal error log may be used in cases where the failure is intermittent, or the LED code is not available. See Chapter 12 for details.

Table 4–1 LED Code Interpretation

LED Code (Hex)	Description	FRU
0 (0000)	Successful completion of MIST	N/A
2 (0010)	Reserved	N/A
3 (0011)	Reserved	N/A
4 (0100)	Host Memory Test failure	T2022
5 (0101)	Board 2 failure	T2023
6 (0110)	Unexpected Restart Occurred	N/A
7 (0111)	Waiting for Update Init	N/A
8 (1000)	Code update in progress	N/A
9 (1001)	Board 1 - Board 2 interface error	T2022, T2023 or bus cable
A (1010)	Board 1 failure	T2022
B (1011)	Forced update mode	T2022
C (1100)	Core SRAM test failure	T2022
D (1101)	Core CVAX Parity test failure	T2022
E (1110)	Core hardware test failure	T2022
F (1111)	Failure at boot time	T2022

Codes are in Hex and should be read from the top (most significant bit) to the bottom (least significant bit). See Figure 4–1. (Refer to Appendix D for more information on MIST testing.)

Figure 4–1 Status Code Decoding (Hex)



4.6 KDM70 Controller Bug Checks

KDM70 controller bug checks occur when an error is detected by functional code during normal controller operation. Bug checks result in a last crash error log packet (LCELP) being written to an internal buffer. This buffer is sent to the host error log during controller initialization. The LCELP is also written to an internal real-time error log, which may be displayed by running EVRLM from the VAX diagnostic supervisor, Chapter 12.

There are two types of KDM70 controller bug checks: software and hardware. A software bug check may be distinguished from a hardware bug check by bit 31 of the bug check code. If bit 31 is clear, a software bug check exists; otherwise, the bug check was detected by KDM70 hardware.

Note: Appendix A contains a complete list of all KDM70 bug check codes and the recommended actions.

4.7 Software-Detected Bug Checks

Software bug checks are inconsistencies detected by KDM70 functional software. Software bug checks do not necessarily mean the KDM70 software is at fault. It simply means that software detected the failure.

To troubleshoot this type of failure, look for other error symptoms. Avoid replacing hardware on the first failure. If many types of bug checks occur, the KDM70 software is probably not the problem.

If only one software bug check occurs (for example, the same bug check code), this failure is probably caused by the KDM70 functional code. Report these failures to KDM70 engineering by submitting a PRISM report.

4.8 Hardware-Detected Bug Checks

Hardware bug checks result from an error detected by the KDM70 hardware (a control RAM parity error). These errors are reported in the same way as software bug checks, by sending a last crash error log packet to the host error log.

Hardware bug checks do not necessarily require you to replace a KDM70 module. For example, a memory parity error results in a hardware bug check. The port driver resets the controller, and MIST adds the bad page to the bad page list. The KDM70 functions normally until it exceeds the maximum number of bad pages. See Section 4.17 for further details on the KDM70 bad page list.

4.9 Last Crash Error Log Packets

A last crash error log packet (LCELP) is the result of a bug check in the controller. It does not indicate a problem with the new connection. During controller initialization, functional code determines if there is a valid LCELP. If so, it is sent to the host error log as part of the initialization process.

The following example shows a last crash error log:

Example 4–1 Last Crash Error Log

```
V A X / V M S               SYSTEM ERROR REPORT     COMPILED 27-NOV-1956 18:54
                                                                     PAGE   1.

******************************* ENTRY       1. *******************************
ERROR SEQUENCE 63.                             LOGGED ON:        SID 0A000005
DATE/TIME 26-NOV-1956 13:50:20.68                            SYS_TYPE 02410201
SCS NODE: BARNUN                                              VAX/VMS V5.3

ERL$LOGMESSAGE ENTRY  KA62B  CPU REV# 6.  FW REV# 4.1
                      XMI NODE # 1.

"DSA" PORT SUB-SYSTEM, UNIT _BARNUN$PUA0:

      MESSAGE TYPE        0004
                                      UDA PORT MESSAGE
      MSLG$L_CMD_REF  00000000
      MSLG$W_SEQ_NUM      0000
                                      SEQUENCE #0.
      MSLG$B_FORMAT         00 ❶
                                      CONTROLLER ERROR
      MSLG$B_FLAGS          01
                                      SEQUENCE NUMBER RESET
                                      UNRECOVERABLE ERROR
      MSLG$W_EVENT        000A ❷
                                      CONTROLLER ERROR
                                      HOST CMD TIMEOUT/LASTFAIL ERR
      MSLG$Q_CNT_ID   87002489
                      021B0001
                                      UNIQUE IDENTIFIER, 000187002489(X)
                                      DISK CLASS DEVICE (166)
                                      KDM70 CONTROLLER
      MSLG$B_CNT_SVR        22
                                      CONTROLLER SOFTWARE VERSION #34.
      MSLG$B_CNT_HVR        00
                                      CONTROLLER HARDWARE REVISION #0.
      "LASTFAIL" CODE     837E ❸
                                      "LASTFAIL" CODE
                                      INTERNAL BUS HW ERROR

CONTROLLER DEPENDENT INFORMATION

      BUGCK CODE      C068202C ❹
      XMI TRANS ERR       0000
                                      XMI TRANSIENT ERROR CNT = 0.
      MEM TRANS ERR       0000
                                      MEMORY TRANSIENT ERROR CNT = 0.
      CDAL ADR        401C0004
      BD1 READ DIAG   664FFF8F
      B1 HIB BUS      05F4FFC1
      BD2 READ DIAG   030560AC
      BD2 HIB BUS     FCFFD70B
      PC              8020FC91

      ERROR PSL       00000000
                                      INTERRUPT PRIORITY LEVEL = 00.
                                      PREVIOUS MODE = KERNEL
                                      CURRENT MODE  = KERNEL
```

The following list explains the callouts in the example:

❶ Controller error format packet.

❷ Event code 0A indicates an LCELP.

❸ SA code determines the format of the KDM70 controller-specific information contained in the LCELP.

❹ KDM70 controller error-specific bug check code.

The codes listed in Table 4–2 are KDM70-specific SA codes seen in LCELP. These codes are used to describe the format of the LCELP and provide further detail on the failure.

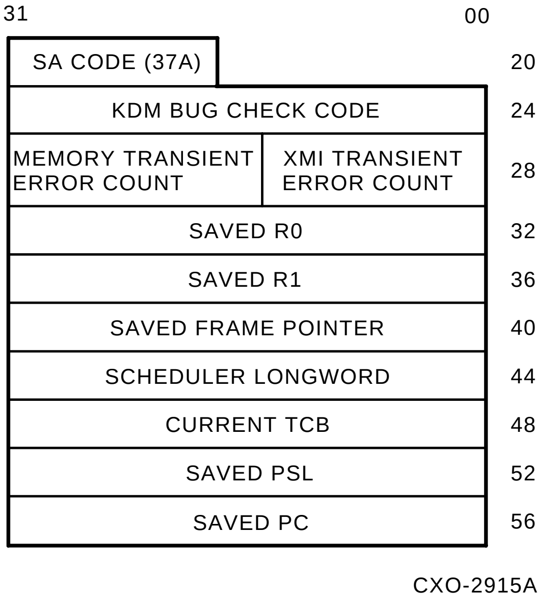
Table 4–2 KDM70 Controller-Specific SA Codes

SA Code	Description	Figure
37A	Software logic error	(Figure 4–2)
37B	Host interface hardware error	(Figure 4–3)
37C	Drive interface hardware error	(Figure 4–4)
37D	Controller memory error	(Figure 4–5)
37E	Internal bus hardware error	(Figure 4–6)
37F	Policy processor error	(Figure 4–7)

4.10 Software Logic Error Format

A software logic error format packet is used to report all software-detected bug checks (Figure 4–2). Software bug checks are inconsistencies detected by the KDM70 software. The information supplied in this error log packet is primarily for use by KDM70 engineering.

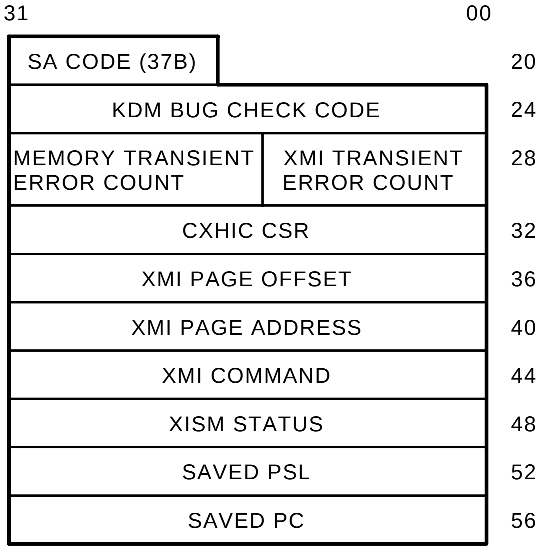
Figure 4–2 Software Logic Error Format



4.11 Host Interface Hardware Error Format

The host interface hardware error format packet is used to report errors associated with the XMI state machine (Figure 4–3). An error detected during a CVAX or DMA (XMI state machine) transfer on the XMI is reported using this error log format. Error-specific registers are supplied in this packet. They may be needed for fault isolation. Refer to Section 4.18 and Section 4.19 for the register description.

Figure 4–3 Host Interface Hardware Error Format



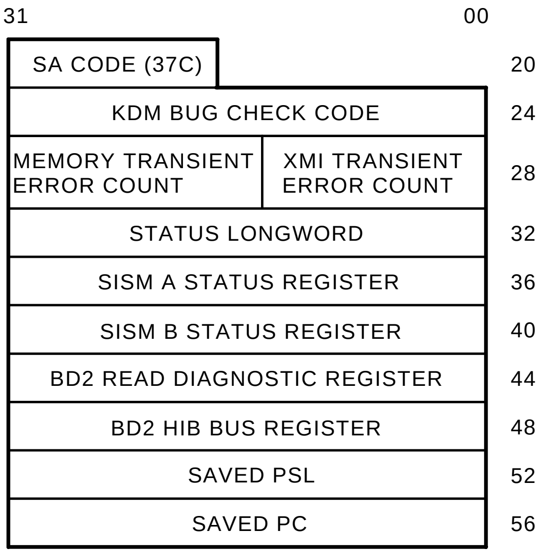
31	00	
SA CODE (37B)		20
KDM BUG CHECK CODE		24
MEMORY TRANSIENT ERROR COUNT	XMI TRANSIENT ERROR COUNT	28
CXHIC CSR		32
XMI PAGE OFFSET		36
XMI PAGE ADDRESS		40
XMI COMMAND		44
XISM STATUS		48
SAVED PSL		52
SAVED PC		56

CXO-2928A

4.12 Drive Interface Hardware Error Format

The drive interface hardware error format packet is used to report errors associated with the SI state machine (Figure 4–4). Error-specific registers are supplied in this packet. They may be needed for fault isolation. Refer to Section 4.18 and Section 4.19 for the register description.

Figure 4–4 Drive Interface Hardware Error Format



31	00	
SA CODE (37C)		20
KDM BUG CHECK CODE		24
MEMORY TRANSIENT ERROR COUNT	XMI TRANSIENT ERROR COUNT	28
STATUS LONGWORD		32
SISM A STATUS REGISTER		36
SISM B STATUS REGISTER		40
BD2 READ DIAGNOSTIC REGISTER		44
BD2 HIB BUS REGISTER		48
SAVED PSL		52
SAVED PC		56

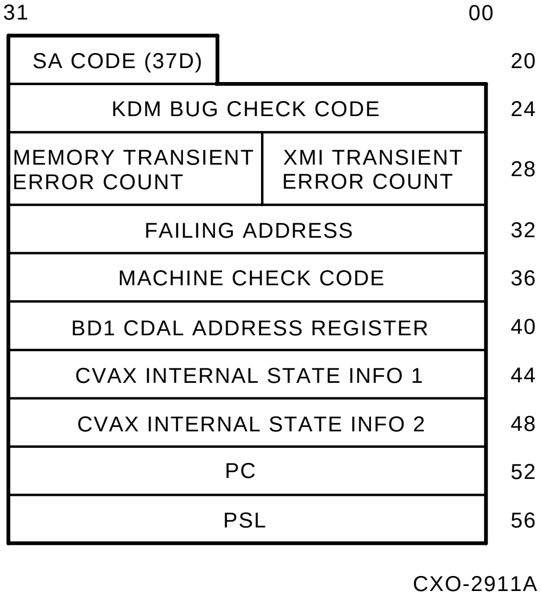
CXO-2913A

4.13 Controller Memory Error Format

The controller memory error format packet is used to report KDM70 memory parity errors (Figure 4–5). The failing address will determine in which module the error occurred.

In most cases, module replacement is not required. The KDM70 will add the failing location to the bad page list. This prevents its use by functional code. Once the error threshold is reached, a fault management event error log will be sent (see Section 4.17). The KDM70 controller will continue to function normally until it reaches the maximum number of bad pages.

Figure 4–5 Controller Memory Error Format

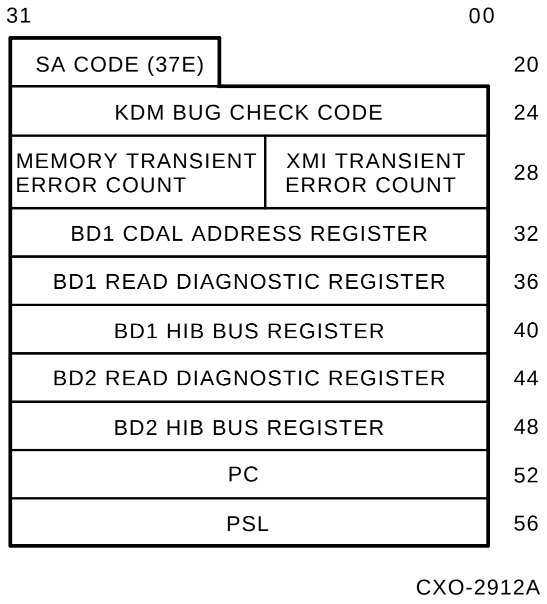

31
00
SA CODE (37D)
20
KDM BUG CHECK CODE
24
MEMORY TRANSIENT ERROR COUNT
XMI TRANSIENT ERROR COUNT
28
FAILING ADDRESS
32
MACHINE CHECK CODE
36
BD1 CDAL ADDRESS REGISTER
40
CVAX INTERNAL STATE INFO 1
44
CVAX INTERNAL STATE INFO 2
48
PC
52
PSL
56
CXO-2911A

4.14 Internal Bus Hardware Error Format

The internal bus hardware error format packet is used to report KDM70 parity errors detected on the internal bus (HIB) (Figure 4–6). The internal bus provides the data path between board 1 and board 2.

Error-specific registers are supplied in this packet. They may be needed for fault isolation. Refer to Section 4.18 and Section 4.19 for a description of these registers.

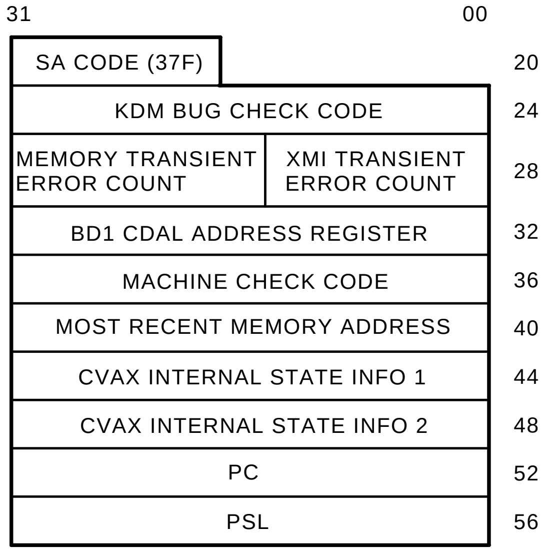
Figure 4–6 Internal Bus Hardware Error Format


31
00
SA CODE (37E)
20
KDM BUG CHECK CODE
24
MEMORY TRANSIENT ERROR COUNT
XMI TRANSIENT ERROR COUNT
28
BD1 CDAL ADDRESS REGISTER
32
BD1 READ DIAGNOSTIC REGISTER
36
BD1 HIB BUS REGISTER
40
BD2 READ DIAGNOSTIC REGISTER
44
BD2 HIB BUS REGISTER
48
PC
52
PSL
56
CXO-2912A

4.15 Policy Processor Error Format

A policy processor error format packet is used to log KDM70 machine checks Figure 4–7. A machine check occurs as a result of serious microcode or hardware error conditions.

Figure 4–7 Policy Processor Error Format



31	00	
SA CODE (37F)		20
KDM BUG CHECK CODE		24
MEMORY TRANSIENT ERROR COUNT	XMI TRANSIENT ERROR COUNT	28
BD1 CDAL ADDRESS REGISTER		32
MACHINE CHECK CODE		36
MOST RECENT MEMORY ADDRESS		40
CVAX INTERNAL STATE INFO 1		44
CVAX INTERNAL STATE INFO 2		48
PC		52
PSL		56

CXO-2914A

The following table explains the machine check codes:

Table 4–3 Machine Check Codes

Machine Check Code (Hex)	Description
01	CVAX Floating Point Accelerator — protocol error
02	CVAX Floating Point Accelerator — reserved instruction
03	CVAX Floating Point Accelerator — unknown error
04	CVAX Floating Point Accelerator — unknown error
05	Process PTE in P0 space (TB miss)
06	Process PTE in P1 space (TB miss)
07	Process PTE in P0 space (M = 0)
08	Process PTE in P1 space (M = 0)
09	Undefined interrupt ID code
0A	Impossible microcode state (MOVCx)
80	Read bus error, normal read
81	Read bus error, SPTE, PCB, or SCB read
82	Write bus error, normal write
83	Write bus error, SPTE or PCB write

The following table describes the bit fields in internal state information 1:

Table 4–4 Internal State Information 1

Bit Field	Field Description
<31:24>	Current contents of OPCODE <7:0>
<23:20>	1111
<19:16>	Current contents of HSIR <3:0>
<15:8>	Current contents of Cache Disable Register <7:0>
<7:0>	Current contents of Memory System Error Register <7:0>, Table 4–6

The following table describes the bit fields in internal state information 2:

Table 4–5 Internal State Information 2

Bit Field	Field Description
<31:24>	Current contents of SC <7:0>
<23:22>	11
<21:16>	Current contents of STATE <5:0>
<15>	Current contents of VAX CANT RESTART bit
<14:12>	111
<11:8>	Current ALU condition codes
<7:0>	Delta PC at time of exception

The following table describes the bit fields in the memory system error register:

Table 4–6 Memory System Error Register

Bit Field	Field Description
<31:8>	Always read as 0’s
<7>	1 = Cache Hit, 0 = Cache Miss
<6>	DAL parity error
<5>	Machine check abort - DAL parity error
<4>	Machine check abort - Cache parity error
<3:2>	Always read as 0’s
<1>	Cache data error
<0>	Cache tag error

4.16 Real-Time Internal Error Log

The real-time internal error log contains a copy of the last six KDM70 controller bug checks. This internal error log may be accessed by running EVRLM. Refer to Chapter 12 for assistance in displaying the real-time error log.

4.16.1 Real-Time Internal Error Log to Host Error Log Relationship

The real-time internal error log contains the same bug check information as the host error log, but is displayed in a different format. This is valuable for two reasons:

1 If the failure prevents the controller from sending an error log packet to the host.

2 If the host error log is not enabled or because the system does not support error logging.

If host error logs are available, use them to determine what type of error has occurred. The host error log formatter is much easier to read and is available on line. The utility used to display this error log is EVRLM, and must be run standalone.

4.16.2 Interpreting Real-Time Internal Error Log Fields

The real-time internal error log is a ring buffer located in board 1 (T2022) EEPROM. There is also a duplicate copy located in board 2 (T2023) EEPROM. This error log contains the last six KDM70 controller bug checks. Note that multiple occurrences of the same bug check will only be logged once if all of the extended status longwords (ESLs) are the same. The bug check will, however, be logged to the system error log. The following is an example of a KDM70 controller real-time internal error log:

```
Bug Check Code C068202C          00000000           00000000
ESL 1          00000000          00000000           00000000
ESL 2          401C0004          00000000           00000000
ESL 3          664FFF8F          00000000           00000000
ESL 4          05F4FFC1          00000000           00000000
ESL 5          030560AC          00000000           00000000
ESL 6          FCFFD70B          00000000           00000000
ESL 7          8020FC91          00000000           00000000
ESL 8          00000000          00000000           00000000
Time Stamp     00000050          00000000           00000000
    .
    .
    .
```

In the example, the bug check code is called out. For a detailed description on each KDM70 controller bug check code, refer to Appendix A.

The format of the ESLs is determined by the type of bug check. In the example, the bug check code is C068202C. This code is described as an internal bus error, which follows the format described in Figure 4–6. The ESLs are also decoded by the VMS error formatter, as shown in Example 4–1.

4.17 Event Code 01EA: Fault Management Analysis Event

Fault management analysis events are recoverable and indicate that hardware needs to be replaced under scheduled maintenance. Event Code 01EA is an informational message that the KDM70 controller has reached an error threshold. The controller will continue to operate normally until the maximum error threshold is reached. This maximum threshold depends on the type of error and is shown in the error log packet.

Fault management analysis event error logs have an MSCP event code of 1EA. Information in the maintenance error log packet (MELP) is formatted by the KDM70 self-test code (MIST). These packets are returned to the host error log during KDM70 initialization.

Port error codes are used to describe specific fault management events. There are four possible port error codes as listed in Table 4–7. Figure 4–8 through Figure 4–11 show the four possible error log formats.

Table 4–7 Fault Management Events

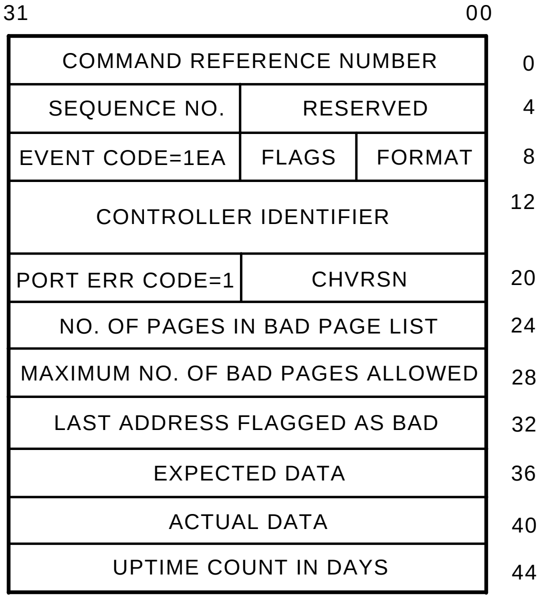
Port Error Code	Description
1	Board 1 bad page list overflow
2	Board 2 bad page list overflow
3	Code store EEPROM entry overflow
4	CVAX cache failure

4.17.1 Board 1 Bad Page List Overflow

This error log indicates that the number of entries in the board 1 (T2022) bad page list has reached or surpassed its threshold. The error log packet contains the number of entries in the bad page list, the maximum number allowed, and the address of the most recent error.

The error log packet format of a board 1 bad page list overflow is shown in Figure 4–8.

Figure 4–8 Board 1 BPL Overflow



31		00	
COMMAND REFERENCE NUMBER			0
SEQUENCE NO.	RESERVED		4
EVENT CODE=1EA	FLAGS	FORMAT	8
CONTROLLER IDENTIFIER			12
PORT ERR CODE=1	CHVRSN		20
NO. OF PAGES IN BAD PAGE LIST			24
MAXIMUM NO. OF BAD PAGES ALLOWED			28
LAST ADDRESS FLAGGED AS BAD			32
EXPECTED DATA			36
ACTUAL DATA			40
UPTIME COUNT IN DAYS			44

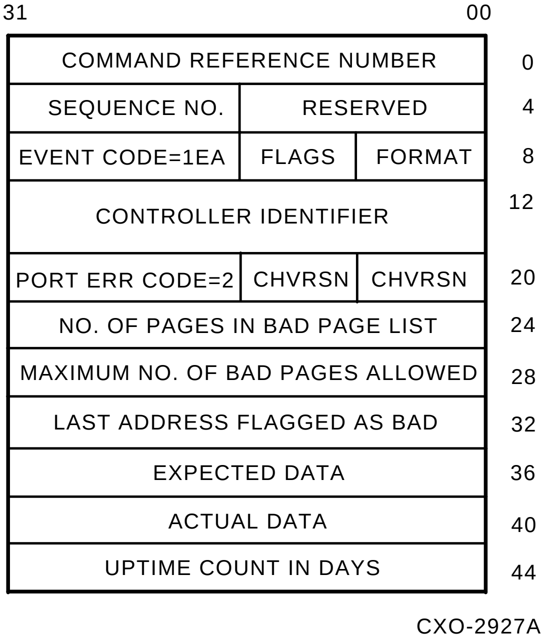
CXO-2926A

4.17.2 Board 2 Bad Page List Overflow

This error log indicates that the number of entries in the board 2 (T2023) bad page list has reached or surpassed its threshold. The error log packet contains the number entries in the bad page list, the maximum number allowed, and the address of the most recent error.

The error log packet format of a board 2 bad page list overflow is shown in Figure 4–9.

Figure 4–9 Board 2 BPL Overflow



31		00	
COMMAND REFERENCE NUMBER			0
SEQUENCE NO.	RESERVED		4
EVENT CODE=1EA	FLAGS	FORMAT	8
CONTROLLER IDENTIFIER			12
PORT ERR CODE=2	CHVRSN	CHVRSN	20
NO. OF PAGES IN BAD PAGE LIST			24
MAXIMUM NO. OF BAD PAGES ALLOWED			28
LAST ADDRESS FLAGGED AS BAD			32
EXPECTED DATA			36
ACTUAL DATA			40
UPTIME COUNT IN DAYS			44

CXO-2927A

4.17.3 Code Store EEPROM Entry Overflow

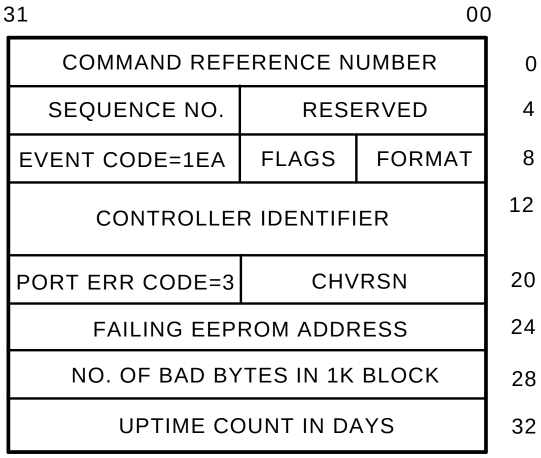
The KDM70 software is stored in EEPROM on board 1 (T2022). The software is copied from EEPROM to SRAM during self-test. The EEPROM is protected by ECC and EDC. Errors during the copy process will be corrected by ECC. Up to 8 bytes per 1K block can be corrected. The correct data will be written to SRAM and written back to the EEPROM.

EEPROM errors will not be logged until an error threshold has been reached. Exceeding the error threshold will cause an error log packet to be sent.

If the number of ECC corrections becomes excessive, the controller may be reset by the port driver before all corrections can be completed. If this occurs, an SA code of 375 (hex) will be written by MIST. As shown in Table B–1, this code is described as “Extended ECC Recovery: EEPROM.”

The error log packet format of a code store EEPROM overflow is shown in Figure 4–10.

Figure 4–10 Code Store EEPROM Entry Overflow



31		00	
COMMAND REFERENCE NUMBER			0
SEQUENCE NO.	RESERVED		4
EVENT CODE=1EA	FLAGS	FORMAT	8
CONTROLLER IDENTIFIER			12
PORT ERR CODE=3	CHVRSN		20
FAILING EEPROM ADDRESS			24
NO. OF BAD BYTES IN 1K BLOCK			28
UPTIME COUNT IN DAYS			32

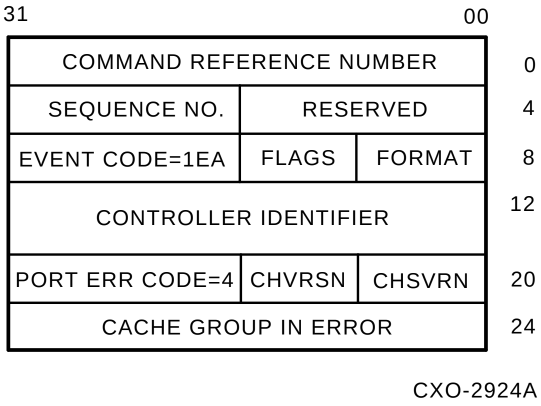
CXO-2925A

4.17.4 CVAX Cache Set Failure

If a CVAX (T2022) cache error is detected during self-test, the failing group will be disabled. The controller will continue the initialization process. Although a bad cache will impact performance, the KDM70 controller will be available for use.

The error log packet format of a CVAX cache failure is shown in Figure 4–11.

Figure 4–11 CVAX Cache Set Failure



31		00	
COMMAND REFERENCE NUMBER			0
SEQUENCE NO.	RESERVED		4
EVENT CODE=1EA	FLAGS	FORMAT	8
CONTROLLER IDENTIFIER			12
PORT ERR CODE=4	CHVRSN	CHSVRN	20
CACHE GROUP IN ERROR			24

CXO-2924A

4.18 KDM70 Board 1 Register Descriptions

The following section describes the KDM70 board 1 (T2022) registers. These registers are not accessible from the local console. The register contents are supplied as part of the last crash error log packet.

4.18.1 Read Diagnostic Register

The board 1 read diagnostic register is updated once every CVAX cycle. When an error is detected, the register's contents are frozen until they are read by the software error handling code, thus freeing them for further data capture.

This register shows the data source selected at the time of error and additional data that may be needed for fault isolation.

Table 4–8 Read Diagnostic Register

Bit Field	Field Description
00	JUMPER 1 H - Selects MIST code to execute after a Node Reset
01	JUMPER 2 H - Selects MIST code to execute after a Node Reset
02	DPC IDLE L - The board 1 Data Path Controller uCode is in the IDLE state
03	REGISTERED WRITE L - Write in progress
04	NXM ACCESS H - CVAX attempted to access a location outside the decode range
05	Reserved
06	Reserved
07	BOARD 2 CHIP SELECT L - Board 2 selected at time of error
08	DBUF - Board 2 Memory selected at time of error
09	SICA - SI State Machine A selected at time of error
10	SICB - SI State Machine B selected at time of error
11	RAM1 - Board 1 (Bank 1) memory selected at time of error
12	RAM2 - Board 1 (Bank 2) memory selected at time of error
13	RAM3 - Board 1 (Bank 2) memory selected at time of error
14	DPCS - Board 1 Data Path Controller selected at time of error
15	CXHIC - Board 1 XMI Controller selected at time of error
19:16	CVAX CYCLE STATUS/DATA PARITY at time of error
20	XMI Backplane Identifier L (XMI 1 or XMI 2)
21	Board 2 EEPROM DATA L
22	Board 2 Serial EEPROM Write L
23	Board 2 EEPROM CLK L
24	Reserved
25	Vectored Interrupt Controller Intr Ack Error L(Board 1)
26	Board 1 Data Path Controller Parity Error L
27	CVAX Parity Error L - Parity error detected on CDAL
31:28	Reserved

4.18.2 CVAX Data and Address Lines Address Register

The board 1 CVAX data and address lines (CDAL) address register is updated with the current address on the CVAX data and address lines every processor cycle until an error is detected. Then, the register latches the address on the bus at the time of the error.

This register remains frozen with the error address so that error recovery code can read this register and obtain the location of the faulty device. If this register is read without an error condition, then the register's own address is received on the DAL bus.

4.18.3 Write Diagnostic Register

The board 1 write diagnostic register provides software control of the KDM70 hardware. Resets to the VIC, DPC, CXHIC, and board 2 are removed by writing ones to the appropriate bits of this register.

The write diagnostic register is used by MIST to force parity errors on any of the KDM70 internal buses and to receive or disable error interrupts.

Table 4–9 Write Diagnostic Register

Bit Field	Field Description
00	CVAX DATA FORCED ERROR CHECKING H
01	HIB ADDRESS FORCED ERROR CHECKING H
02	HIB CONTROL FORCED ERROR CHECKING H
03	XIC DATA FORCED ERROR CHECKING H
04	CXHIC RESET L
05	BOARD 2 RESET L
06	VIC RESET L
07	DPC RESET L
08	WRITE ENABLE H - Enables writes to the EEPROM
09	MIST FAIL H - Failure during self-test
10	MIST DONE H
11	DIAGNOSTIC PRIORITY INTERRUPT REQUEST L
12	XIC ERROR RESET L - Clears the Board 1 HIB Bus Register
13	CVAX DPE DISABLE L - Disables CVAX parity checking
14	ERROR DISABLE L - Disables Memory Error Interrupts
15	CVAX PARITY FORCED ERROR CHECKING H
16	Reserved
17	SER EEPROM WR L
18	SER EEPROM DATA H
19	BDR2 EEPROM CLK H
20	LED ENA H - Enable Board 1 module LED (Amber)
21	XMI REQ DIS H
22	PREEMPT H - Allow preemption of work block execution by DPC
23	DPC TEST L - Enable DPC test mode
31:24	Reserved

4.18.4 HIB Bus Register

The HIB bus register contents consist of the address bus, control bus, and chip selects. If HIB errors are detected, the clock to the HIB bus register is inhibited until the error condition is cleared. By freezing the contents of this register, error recovery code can determine the type of operation and location.

This register allows module isolation for FRU callouts. MIST verifies the integrity of the board 1 HIB address and control buses by writing to locations on board 2 while this module is held reset. Then, diagnostics read back the contents of the HIB bus register and compare the captured data with the address and control data generated during the write operation. If the two quantities match, board 1 is functioning properly. If the two quantities do not match, board 1 is the suspect module and is called out as the failing module.

Table 4–10 Board 1 HIB Bus Register

Bit Field	Field Description
8:0	HIB/CVAX ADDRESS
17:9	HIB ADDRESS LINES H
18	HIB ADDRESS LOW PARITY H - HIB Address Parity (8:0)
19	HIB ADDRESS HIGH PARITY H - HIB Address Parity (17:09)
20	BRD2 Chip Select
21	SICA Chip Select
22	SICB Chip Select
23	DPC Chip Select
24	CXHIC Chip Select
25	HIB MEMORY SELECT L - Board 2 Data Buffer Memory select
26	HIB WRITE L - Board 1 Write in Progress
30:27	HIB BYTE MASK
31	HIB CONTROL BUS PARITY H

4.18.5 CXHIC Control and Status Register

The KDM70 controller communicates to the host through a custom chip (CXHIC). The CXHIC is used to control XMI transfers. The CXHIC control and status register contains 32 bits of control and status information. It is read/write accessible only by policy software. It is not accessible to the host system or the XMI state machine. It is used to report errors in the XMI interface located on board 1. Table 4–11 describes the bit fields of the CXHIC CSR.

Table 4–11 CXHIC Control and Status Register

Bit Field	Field Description
3:0	DMA Transfer Count
4	DMA Transaction Count Enable
5	CVAX Transaction Commander ID Enable
9:6	Reserved for future use
10	CXHIC Transaction Timeout Error
11	CXHIC Command or Write Data NOACK Error
12	CXHIC Read Response Error
13	CXHIC Read Sequence Error
14	CXHIC No Read Response Error
15	Internal Bus (HIB) Parity Error
16	Assert Lockout
17	DMA Transaction OK
18	DMA Transaction Not OK
19	CVAX Transaction OK
20	CVAX Transaction Not OK
21	Ident Response Sent
22	LOCK Response Received - Invalid response
23	CXHIC Masked Error Summary
24	Register Collision Error
25	Internal Byte Parity Error
26	Diagnostic Loopback Mode
27	Diagnostic Forced Error - Diagnostic mode
31:28	KDM70 controller node number

4.18.6 XISM Status Register

The XISM status register contains the state of the XMI state machine. Table 4–12 describes the bit fields of the XISM status register.

Table 4–12 XISM Status Register

Bit Field	Field Description
21:0	Reserved
22	XMI State Machine Parity Error
24:23	Reserved
29:26	Error Code - Table 4–13
30	Reserved
31	Error Summary

The following table describes the error codes in the XSIM status register (bits <29:26>.)

Table 4–13 XISM Status Error Codes

Error Code	Description
0001	Internal EDC error
0010	XMI compare error
0011	Buffer compare error
0100	DPC timeout error
0101	CXHIC error
0110	XMIC error
0111	Programming error

4.19 KDM70 Board 2 Register Descriptions

The following section describes the KDM70 board 2 (T2023) registers. These registers are not accessible from the local console. The register contents are supplied as part of the last crash error log packet.

4.19.1 HIB Bus Register

The HIB bus error register on board 2 continually captures address, control, and parity information about transactions on the HIB as seen on board 2. When an error occurs, the clock enable for the register is removed and the information is preserved.

Table 4–14 HIB Bus Register

Bit Field	Field Description
00	SIC A BUS GRANT L - CSIC A granted control of HIB
01	SIC B BUS GRANT L - CSIC B granted control of HIB
02	XI GRANT L - XMI State Machine granted control of HIB
03	READ H
04	MEM BM0 L - HIB Date byte mask
05	MEM BM1 L - HIB Date byte mask
06	MEM BM2 L - HIB Date byte mask
07	MEM BM3 L - HIB Date byte mask
25:08	B2 HIB ADDR H - Board 2 HIB Address lines
26	CTRL PERR L - Parity error on the control lines received from board 1
27	ADDR PERR L - Parity error on the address lines received from board 1
28	HIB DATA0 PERR L - HIB parity error on byte 0
29	HIB DATA1 PERR L - HIB parity error on byte 1
30	HIB DATA2 PERR L - HIB parity error on byte 2
31	HIB DATA3 PERR L - HIB parity error on byte 3

4.19.2 Read Diagnostic Register

The board 2 read diagnostic register is latched at the time of error. It contains error and status information related to the SI state machine located on board 2 (T2023).

Table 4–15 Read Diagnostic Register

Bit Field	Field Description
10:00	SISM A ADDRESS
21:11	SISM B ADDRESS
22	SIECL A SHIFT REG OUT H
23	SIECL B SHIFT REG OUT H
24	SISM A PERR FDBCK1 L - SI State Machine A parity error
25	SISM B PERR FDBCK1 L - SI State Machine B parity error
26	BD2 EE DATA H
27	BUS ERR REG EN L
28	CRAM A OVERFLOW H
29	CRAM B OVERFLOW H
30	CHAN A PORT ENA H
31	XMI UPDATE EN H - Read from XMI Backplane

4.19.3 Write Diagnostic Register

The diagnostic write register on board 2 is used by MIST to force error conditions and test the error detection logic on board 2.

Table 4–16 Write Diagnostic Register

Bit Field	Field Description
00	SIECL A DIAG MODE A H
01	SIECL A DIAG MODE B H
02	FORCE PULSE ERROR A H
03	SIECL A SH REG CLK EN H
04	SISM A EN H
05	SISM A CRAM TEST H
06	SIC A RESET L
07	Reserved
08	SIECL SHIFT REG IN H
09	SIECL B DIAG MODE A H
10	SIECL B DIAG MODE B H
11	FORCE PULSE ERROR B H
12	SIECL B SH REG CLK EN H
13	SISM B EN H
14	SISM B CRAM TEST H
15	SIC B RESET L
16	DIAG2 WR BIT11 H
17	DIAG CLK EN H
18	HIB FEC H
19	HIB CTRL FEC H
20	CLR PERR L
21	CAL CLK TEST A H
22	CAL CLK TEST A H
23	MANUAL DIAG CLK L
24	EEWR DIAG24 H
25	EEDATA DIAG25 H
26	EECLK DIAG26 H
27	SISM A RD EN L
28	SISM B RD EN L
29	SISMCODE WR RD EN H
30	YELLOW LED ON H
31	Reserved

The following table describes the fields in the SISM status register:

Table 4–17 SISM Status Register

Bit Field	Field Description
7:0	Major Event Counter - Transfers remaining in current work block
15:8	Transfer Word Counter - Serial transfers remaining for current operation
23:16	Error Code - Table 4–18
27:24	0
28	State Machine Number - 0 = SISM A, 1 = SISM B
29	Halt - Halted after detecting a fatal error
30	1
31	Error Summary

The following table describes the error codes of the SISM status register (<23:16>):

Table 4–18 SISM Status Error Codes

Error Code	Description
00-0F	Reserved
10	SISM Running
11	Running Tape - SISM processing tape work blocks
20	Fatal Error Halt - Work block Error
30	HALT: Halted due to invalid next work block pointer
40	Unresolved Error Halt
50	Internal SISM error
60	Program Error
70	SERDES Overrun
80-F0	Diagnostic Codes

4.20 Physical Memory Address Map

Memory on board 1 is divided between SRAM and EEPROM banks. The EEPROM is used for nonvolatile storage of KDM70 software. The code image is copied from EEPROM to SRAM during self-test.

Memory on board 2 consists of SRAM and serial EEPROM. The SRAM is used for data memory. It is allocated by functional code as needed for data transfers. Serial EEPROM contains non-volatile module history (serial numbers, revision, and so on).

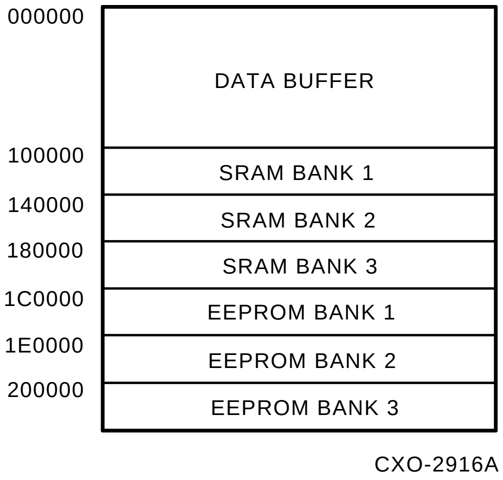
Table 4–19 and Figure 4–12 show the physical address ranges of KDM70 controller memory.

Table 4–19 KDM70 Physical Memory Map

Address Range	Physical Location	Purpose	Type
100000 - 1BFFFF	T2022	Program execution	SRAM
1C0000 - 1FFFFF	T2022	Code store, error logs, bad page list	EEPROM
000000 - 0FFFFF	T2023	Data buffers	SRAM

Data buffer memory is contained on board 2. The address range for board 2 memory is 000000 through 0FFFFF. This information enables you to translate a physical address location to board 1 or board 2.

Figure 4–12 Physical Memory Address Map



4.21 Physical I/O Space

Figure 4–13 shows the physical addresses for KDM70 I/O space.

Figure 4–13 Physical I/O Space Map

20009000	SISM uCODE (LOWER)
20009400	SISM uCODE (UPPER)
20008000	CSIC A REGISTERS
20008400	CSIC B REGISTERS
20008800	BOARD 2 DIAGNOSTIC READ REG
20008804	BOARD 2 DIAGNOSTIC WRITE REG
20008C00	BD2 HIB BUS REGISTER
20009800	CXHIC REGISTER
20009C00	DPC REGISTER
2000A000	XISM uCODE
2000C000	BOARD 1 DIAGNOSTIC READ REG
2000C400	BOARD 1 DIAGNOSTIC WRITE REG
2000C800	BD1 CDAL ADDR REGISTER
2000D000	VIC REGISTERS
2000E000	BD1 HIB BUS REGISTER
20040000	BOOT ROM
20140000	SSC REGISTERS
201407FC	MONITOR FLAGS

CXO-2917A

5 Invoking Diagnostics, Exercisers, and Utilities

5.1 Introduction

This chapter describes the on-line and standalone implementation of diagnostics, exercisers, and utilities. Refer to individual chapters for program-specific information.

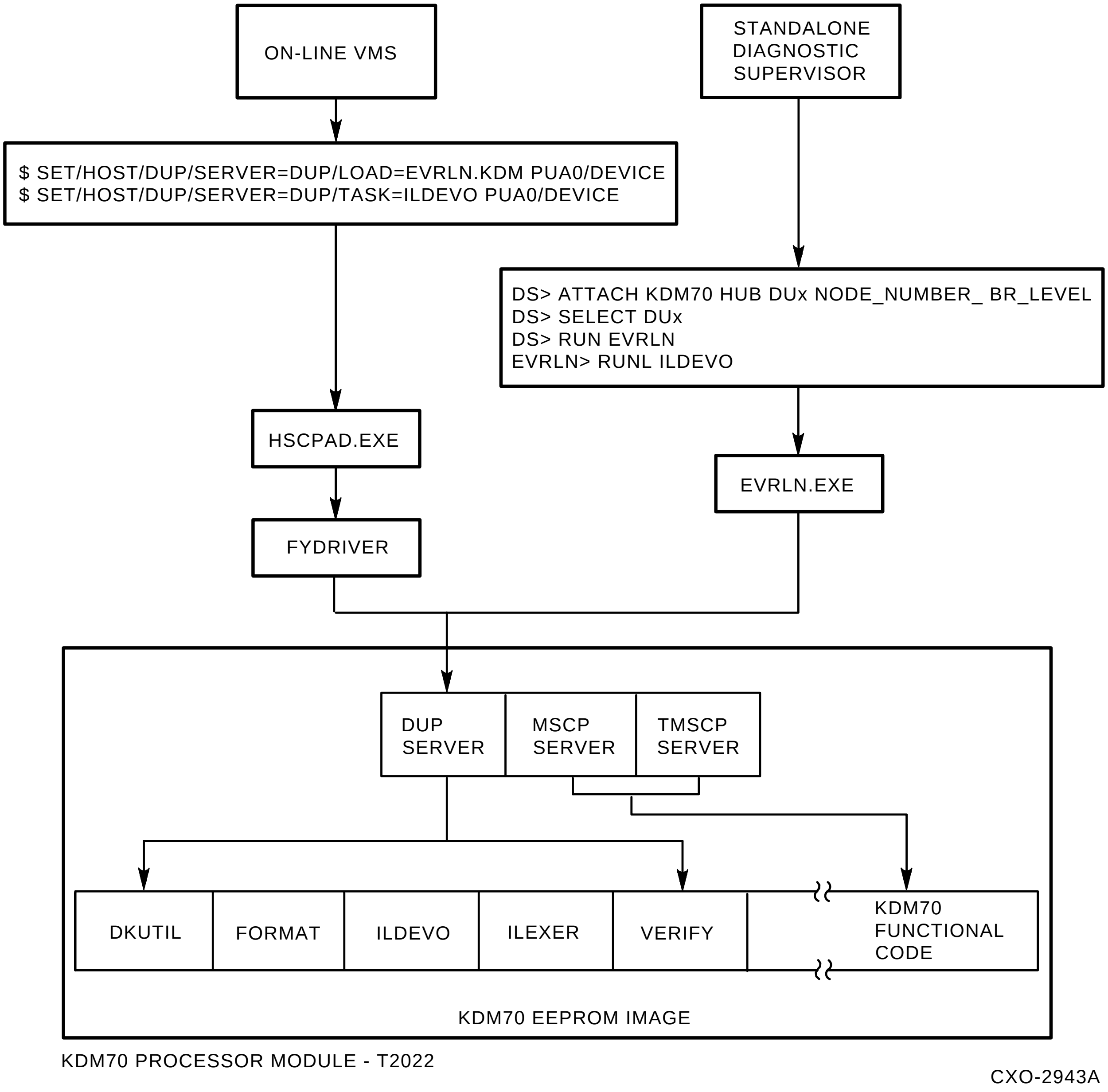
5.2 How to Run Diagnostics, Exercisers, and Utilities

Sometimes it is necessary to run diagnostics, exercisers, and utilities when troubleshooting disks and tapes attached to the KDM70 controller. For the purpose of this discussion, diagnostics, exercisers, and utilities are referred to as programs.

You must first decide if you will run the program on line or in standalone mode. It is always preferred that you run the programs on line. Taking a system off line should be done only in extreme cases and only after obtaining permission from the customer.

Figure 5–1 shows how KDM70 controller-resident programs (such as ILDEVO) are accessed. The resident diagnostics are accessed through the diagnostic utility protocol (DUP). DUP supplies the connection management and communications services to diagnostics running in the KDM70 controller. This connection is provided by HSCPAD/FYDRIVER when running on line under VMS, or by “EVRLN.EXE” when running standalone from the VAX diagnostic supervisor. Any of the resident programs listed in Table 5–1 are accessed in the same manner.

Figure 5–1 KDM70 Controller-Resident Programs


ON-LINE VMS
STANDALONE DIAGNOSTIC SUPERVISOR
$ SET/HOST/DUP/SERVER=DUP/LOAD=EVRLN.KDM PUA0/DEVICE
$ SET/HOST/DUP/SERVER=DUP/TASK=ILDEVO PUA0/DEVICE
DS> ATTACH KDM70 HUB DUx NODE_NUMBER_ BR_LEVEL
DS> SELECT DUx
DS> RUN EVRLN
EVRLN> RUNL ILDEVO
HSCPAD.EXE
EVRLN.EXE
FYDRIVER
DUP SERVER
MSCP SERVER
TMSCP SERVER
DKUTIL
FORMAT
ILDEVO
ILEXER
VERIFY
KDM70 FUNCTIONAL CODE
KDM70 EEPROM IMAGE
KDM70 PROCESSOR MODULE - T2022
CXO-2943A

5.3 KDM70-Resident Programs

The programs shown in Table 5–1 are resident on the KDM70 controller processor module (T2022). They may be accessed through a DUP connection either from VMS or standalone from the VAX diagnostic supervisor.

VMS uses the FYDRIVER for the required DUP connection. This connection is provided by "EVRLN.EXE" when running standalone from the VAX diagnostic supervisor.

Table 5–1 KDM70 Controller-Resident Programs

Task	Description
DKUTIL	Disk Utility
FORMAT	Disk Formatter
ILEXER	In-line Exerciser - Disk/Tape
ILDEVO	In-line Device Operations Test - Disk/Tape
VERIFY	Disk Verify

5.4 Invoking KDM70 Controller-Resident Programs

The following procedures describe how to run the KDM70 controller-resident programs, either on line or standalone.

5.5 HSCPAD = Diagnostic Utility Protocal (DUP) virtual terminal program for VMS

The DUP virtual terminal program dupterm provides access to the diagnostic and utility features of DSA (DIGITAL Storage Architecture) intelligent disk and tape controllers which are running under the control of the VMS Operating system.

Depending upon the particular controller, operations may include the execution of controller resident diagnostics and utilities or those downloaded from the host system.

Further information regarding the operation of the utility and diagnostic capabilities associated with a particular controller may be obtained from the appropriate documentation set for that controller.

5.5.1 COMMANDS to use DUP on VMS

SET HOST/DUP

PARAMETER

node-name

node-name/device

Specifies the node name of the storage controller. If, HSC then enter SCS node name (ie, rock). If, KDM then enter pudriver name/device (ie, pua0/device).

QUALIFIERS

/LOG

/LOG[=filespec]

/NOLOG (default)

Controls whether a log file for the session is kept. If you use the /LOG qualifier without the file specification, the log information is stored in the file HSCPAD.LOG.

/LOAD

/LOAD=supplied-program-name

Specifies the supplied utility or diagnostic program name. It is first loaded to the target storage controller. Then executes under the direction of the server.

Either this qualifier or the TASK qualifier is required.

/SERVER

/SERVER=DUP

The only server applicable to the KDM70 is DUP.

/TASK

/TASK=local-program-name

Specifies the local utility or diagnostic program name to be executed on the target storage controller under direction of the server.

Either this qualifier or the LOAD qualifier is required.

EXAMPLES

```
$ SET HOST/DUP/SERVER=DUP/TASK=DIRECT
PUA0/DEVICE %HSCPAD-I-LOCPROGEXE, Local
program executing - type ^\ to exit
```

The SET HOST/DUP command in this example connects the user terminal to the utility program called DIRECT executing on a KDM storage controller named PUA0 under direction of the DUP server.

```
$ SET HOST/DUP/SERVER=DUP/LOAD=PATCH.KDM
PUA0/DEVICE %HSCPAD-I-SUPPROGEXE, Supplied
program executing - type ^\ to exit... Done!
```

The SET HOST/DUP command in this example down-line loads a supplied utility program called PATCH.KDM and then connects the user terminal to that program which is excuting on PUA0.

5.5.2 Running Programs On Line from VMS

Use the following procedure to access and run on-line programs. Go to Section 5.5.3 for instructions on accessing and running programs in standalone mode.

Note: You cannot run on-line diagnostics, exercisers, and utilities without first running EVRLN.KDM. It is important that you follow this procedure.

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CONNECT FYA0/NOADAPTER
SYSGEN> EXIT
$ SET DEFAULT SYS$MAINTENANCE
$ SET HOST/DUP/SERVER=DUP/LOAD=EVRLN.KDM PUA0/DEVICE
$ SET HOST/DUP/SERVER=DUP/TASK=ILDEVO PUA0/DEVICE
```

5.5.3 Running Programs Standalone from the VAX Diagnostic Supervisor

Use the following procedure to invoke programs from VAX DS:

```
DS> ATTACH KDM70 HUB DUx N BR
                         |  |
                         |  |__________ BUS REQUEST
                         |
                         |____________ NODE NUMBER
DS> SELECT DUx
DS> RUN EVRLN
EVRLN> RUNL ILDEVO
```

5.6 EVRLN Overview

EVRLN is a standalone DUP control program that runs under the VAX diagnostic supervisor (VAX/DS). EVRLN enables the local and supplied DUP diagnostics and utilities to run when on-line testing is not possible. EVRLN must be invoked prior to running in-line diagnostics on the KDM70 controller.

EVRLN is located in the SYS$MAINTENANCE area of the system disk (in a VMS-based system) or on the local load media, which also contains the VAX diagnostic supervisor (VAX/DS). EVRLN supports only one controller at a time. Multiple controllers must be tested individually.

5.7 Invoking EVRLN

To invoke EVRLN, boot the VAX/DS in standalone mode. At the VAX/DS prompt (DS), attach and select the KDM70 controller. The following is a example of this procedure:

```
DS> ATTACH KDM70 HUB DUx N BR
DS> SELECT DUx
DS> RUN EVRLN
```

The variables are described in the following table:

Variable	Explanation
n	The XMI node number, in hex, for the KDM70 controller. The node number is determined by the slot location in which T2022 is placed. If T2022 were placed in slot two, the XMI node number would be 2.
br	The bus request level (4–7). The normal bus request level is 5.

EVRLN performs the following steps:

1. Initialize the KDM70 controller and check for errors.
2. Establish a DUP connection.

5.8 EVRLN Commands

The commands listed in Table 5–2 are legal EVRLN commands.

Table 5–2 EVRLN Commands

Command	Description
HELP	Prints information on EVRLN commands
CAT	Prints a catalog of SUPPLIED and LOCAL DUP programs
INFO	Prints information about the selected controller
RUNS	Executes a SUPPLIED DUP program
RUNL	Executes a LOCAL DUP PROGRAM
EXIT	Terminates EVRLN and returns control to VAX/DS

5.8.1 EVRLN HELP Command

There are two levels of HELP available. The first is the VAX/DS help facility. It is invoked at the DS> prompt and provides information on the ATTACH sequence for supported controllers and drive types. It is invoked in the following way:

```
 DS> HELP EVRLN ATTACH
```

EVRLN also provides the second level of help. It is invoked in the following way:

```
EVRLN> HELP
```

The following message is printed when HELP is invoked:

```
EVRLN will allow the field engineer to run both SUPPLIED and/or
LOCAL DUP programs. To get information about the controller
(microcode revision, hardware revision, DUP extension, etc.)
type INFO. To find out what programs are available for the
device selected, type CAT.

To run a DUP program, type RUNL to run a LOCAL program (or
optionally type RUNS to run a SUPPLIED program) followed by
the program name (no extension) to be run.

Additional information is available on the following topics --
type HELP <topic> after the EVRLN> prompt.

      CAT      EXIT      INFO      RUNL      RUNS
```

5.8.2 EVRLN CAT Command

CAT is a directory lookup command that provides a listing of available DUP programs. Use the CAT command in the following manner:

```
EVRLN>CAT
```

EVRLN prints the contents of a text file called EVRLNCAT.TXT. Since this is not a true directory look-up, the information may not match the contents of SYS$MAINTENANCE directory. For example, if a copy of a new DUP SUPPLIED program is added to SYS$MAINTENANCE, it will not be reflected in EVRLNCAT.TXT unless the information is added. The following disclaimer is printed:

```
THIS IS NOT A TRUE DIRECTORY LOOKUP.  IT IS  JUST  A  PRINTOUT  OF  AN
ASCII FILE, WHICH MAY NOT MATCH THE ACTUAL CONTENTS OF YOUR DIRECTORY.
```

If a requested program is not in the directory, a PROGRAM NOT FOUND error message is printed by EVRLN:

```
        There are no LOCAL programs available for this controller.
```

If any other message comes back, a message indicating the problem is printed. For example:

```
Unable to run DIRECT program on controller.  Reason for failure:

PROGRAM NOT FOUND
```

5.8.3 EVRLN EXIT Command

EXIT returns control to the VAX/DS. A CTRL/C aborts a running program.

A running DUP program must first be aborted with CTRL/C, which returns control to EVRLN. Use the EXIT command to exit to the VAX/DS prompt.

5.8.4 EVRLN INFO Command

INFO provides the following information about a selected controller:

- Software revision
- Hardware revision
- DUP extension
- Controller class
- Model and unique identifier

5.8.5 EVRLN RUNL Command

A RUN LOCAL (RUNL) command causes EVRLN to issue an EXECUTE LOCAL PROGRAM command with the given program name. If no local program is available by the selected name, the following error message is printed:

```
DUP LOCAL program not found -- "Name_of_program"
```

The *name_of_program* string is the program requested in the command.

Control then returns to the EVRLN> prompt.

```
The format of the RUNL command is:
```

```
RUNL <program_name>
```

```
In order to abort a running program, type CTRL/Y or CTRL/C.
```

5.8.6 EVRLN RUNS Command

When a RUN SUPPLIED PROGRAM (RUNS) command is received, EVRLN appends the DUP extension to the end of the supplied program name and attempts to open the latest revision of the file on the load device. If the file does not exist, the following error message is printed:

```
DUP SUPPLIED program not found -- program_name.DUP_extension
```

The string *program_name* is the program requested with the the RUNS command (without the DUP extension). Control returns to the EVRLN prompt.

The request is executed and control returns to EVRLN> prompt when the program completes, hangs, detects a controller error, or aborts from a CTRL/Y or CTRL/C.

```
The format of the RUNS command is:
```

```
RUNS <program_name>
```

6 FORMAT and VERIFY Utilities

6.1 Introduction

This chapter contains the information required to run the VERIFY and FORMAT utilities.

Topics in this chapter include initiating the utility, using commands, and interpreting error messages. The utilities are interactive and are prompt-oriented.

Note: **A prompt displayed in square brackets is the default.**

6.2 Disk Formatter Utility (FORMAT)

FORMAT is used to format 512-byte disks. It can format the read-only DBN space or the LBN area and the read-only DBN space.

CAUTION: **Before using the FORMAT utility, make sure you are familiar with the DSA. Otherwise, you may destroy user data.**

The DBN area is always formatted. If the user requests it, the LBN area also is formatted.

CAUTION: **Using CTRL/C or CTRL/Y to abort the FORMAT utility may destroy the contents of the FCT and/or the RCT. The FORMAT utility should only be aborted under fatal or unrecoverable disk failure conditions.**

6.2.1 Invoking FORMAT

FORMAT is a utility that can be invoked either on line or in standalone mode.

6.2.1.1 Invoking FORMAT On Line from VMS

Use the following procedure to access and run on-line programs. Go to Section 6.2.1.2 for instructions on accessing and running programs in standalone mode.

Note: **You cannot run on-line diagnostics, exercisers, and utilities without first running EVRLN.KDM. It is important that you follow this procedure.**

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CONNECT FYA0/NOADAPTER
SYSGEN> EXIT
$ SET DEFAULT SYS$MAINTENANCE
$ SET HOST/DUP/SERVER=DUP/LOAD=EVRLN.KDM PUA0/DEVICE
$ SET HOST/DUP/SERVER=DUP/TASK=FORMAT PUA0/DEVICE
```

6.2.1.2 Invoking FORMAT Standalone from the VAX Diagnostic Supervisor

Use the following procedure to invoke FORMAT from VAX DS:

```
DS> ATTACH KDM70 HUB DUx N BR
                         |  |
                         |  |__________ BUS REQUEST
                         |
                         |____________ NODE NUMBER
DS> SELECT DUx
DS> RUN EVRLN
EVRLN> RUNL FORMAT
```

You can enter responses to more than one prompt by separating the responses with commas. For example, if unit D133 is formatted with no special options, you can type D133,Y,N at the first prompt.

The following prompt asks for the unit number of the disk to format:

```
Select drive to FORMAT (Dnnnn) [] ?
```

The next prompt determines whether the LBN (user data) area should be formatted or whether only the DBN (diagnostic) area should be formatted.

```
Format user data area (Y/N) [N] ?
```

Enter N to execute and format only DBN space. Enter Y to execute and format LBN space. The program prompts for special options for debugging purposes and to to increase data reliability.

```
Select special options (Y/N) [N] ?
```

Enter N to allow FORMAT to start processing. Enter Y to enable the special options. The following three special option prompts appear.

```
    Revector blocks with 1 symbol ECC errors (Y/N) [N] ?
```

Normally, blocks discovered during the check pass of formatting with one-symbol ECC errors are not retired. The program assumes this level of error is tolerable. Enter Y to retire , all blocks with solid (nontransient) ECC errors. However, in all cases, blocks with two-symbol (or more) ECC errors are always retired, regardless of the drive's ECC symbol threshold.

The second special option prompt is:

```
    Revector blocks with transient errors (Y/N) [N] ?
```

After a track is formatted, it is read once. If an error is detected, blocks with errors are read twice more. If those blocks show errors again, they are retired and the track is reformatted. If those blocks show no errors again, they are not retired. Such errors are considered tolerable transient errors. Enter Y to retire any block with an error.

The third special option prompt is:

```
Report position of bad blocks (Y/N) [N] ?
```

Blocks retired during the format process are reported with a single line printout. The type, block number, and cause are printed. Enter Y to print the PBN number, cylinder, track, group, and position as well.

6.2.2 Running FORMAT

The following is a sample session using FORMAT, under VAX/DS and EVRLN.

After booting the diagnostic supervisor, issue the following commands:

Example 6–1 Sample FORMAT Session

```
DS> ATTACH KDM70 HUB DUB 4 5
DS> SEL DUB
DS> RUN EVRLN
EVRLN> RUNL FORMAT

***
*** FORMAT (Disk Formatter) V 001  *** 10-June-1989 18:04:47 ***
***

Select drive to FORMAT (Dnnnn) [] ? D1
Format user data area (Y/N) [N] ? Y
Select special options (Y/N) [N] ? Y
Revector blocks with 1 symbol ECC errors (Y/N) [N] ?
Revector blocks with transient errors (Y/N) [N] ?
Report position of bad blocks (Y/N) [N] ?

*** Format begun at 18:05:06.
***     2 cylinders left in DBN space at 18:05:08.
*** LBN 534149 is bad (FCT), a primary revector to RBN 16186.
*** LBN 532332 is bad (FCT), a primary revector to RBN 16131.
*** LBN 531210 is bad (FCT), a primary revector to RBN 16097.
*** LBN 522770 is bad (FCT), a primary revector to RBN 15841.
***  1413 cylinders left in LBN space at 18:06:01.
*** LBN 498954 is bad (FCT), a primary revector to RBN 15119.
***   413 cylinders left in LBN space at 18:14:56.
.......(deleted some LBN data here...)
*** LBN 133982 is bad (FCT), a primary revector to RBN 4060.
*** LBN 114144 is bad (FCT), a primary revector to RBN 3458.
***   313 cylinders left in LBN space at 18:15:49.
***   213 cylinders left in LBN space at 18:16:42.
***   113 cylinders left in LBN space at 18:17:35.
*** LBN 9075 is bad (FCT), a primary revector to RBN 275.
*** LBN 9020 is bad (FCT), a primary revector to RBN 270.
*** LBN 8453 is bad (FCT), a primary revector to RBN 256.
***    13 cylinders left in LBN space at 18:18:28.

*** Format completed at 18:19:07.
```

Example 6–1 Cont’d on next page

Example 6–1 (Cont.) Sample FORMAT Session

```
    Statistics:
                             0 Bad RBNs,
                            10 Revectored LBNs,
                            10 Primary Revectored LBNs,
                             0 Non-Primary Revectored LBNs,
                             0 Bad Blocks in the RCT Area,
                             0 Bad Blocks in DBN Area,
                             0 Blocks Retried in Check Pass.

   *************************************************************
   *                                                           *
   *    VERIFY must be RUN to complete FORMAT verification!    *
   *                                                           *
   *************************************************************

***
*** FORMAT is exiting.
***
```

CAUTION: The message in the box indicates VERIFY must be run to complete verification. This is an essential step and should not be skipped.

The example output is a session with an RA70 disk drive that has 10 bad PBNs in the FCT. The PBNs were retired because they were in the FCT. The informational message is printed every 100 cylinders to show the pace at which progress is being made.

Note: The final statistics indicate 10 LBNs were revectored. These blocks were not retried during the check pass.

6.2.3 FORMAT Errors and Information Messages

This section describes the error and information messages printed by FORMAT.

6.2.3.1 Error Message Variables

Variable output in the error and information messages is shown in **bold print**. These fields are formed as follows:

n = A decimal number
x = The way a block was found bad: FCT or check
xBN = A space: DBN, XBN, or LBN
hh = Hours
mm = Minutes
ss = Seconds

6.2.3.2 Fatal Error Messages

This section describes the fatal error messages printed by FORMAT.

***** Cannot Position to DBN Area.** — FORMAT attempts to verify it has positioned the heads to the DBN area before it formats the disk. FORMAT does this by reading the first sector of every track in the DBN area until a sector is read without a header error. This fatal error message is printed if no such sector can be found.

***** DBN Format Error (Drive FORMAT Command Failed).** — A FORMAT command fails for five retries when formatting the DBN area.

***** Drive Is Write Protected.** — The requested drive is hardware write-protected and therefore cannot be formatted.

***** FCT Does Not Have Enough Good Copies of Each Block.** — Any block in the FCT does not have two good copies.

***** Invalid FCT.** — One or more PBNs remain to be processed. When the program finishes formatting the LBN area, it checks to see if all PBNs in the FCT have been processed. It usually indicates an FCT where some PBNs are out of order.

***** FCT Read Error.** — All copies of some given block of the FCT cannot be successfully read.

***** FCT Write Error.** — All copies of some given block of the FCT cannot be successfully written.

***** FORMAT Initialization Failure: Storage Could Not Be Acquired.** — FORMAT cannot acquire enough data buffers or control blocks to start formatting.

***** LBN Format Error (Drive FORMAT Command Failed).** — A FORMAT command fails for five retries when formatting the LBN area.

***** RCT Does Not Have Enough Good Copies of Each Block.** — Any block in the RCT does not have two good copies.

***** RCT Is Full.** — So many bad blocks are encountered that the RCT overflows.

***** RCT Read Error.** — All copies of some given block of the RCT cannot be successfully read.

***** RCT Write Error.** — All copies of some given block of the RCT cannot be successfully written.

***** Too Many Bad RBNs Found Before RCT Was Formatted.** — More RBNs than can be recorded in memory are encountered before the RCT area has been formatted.

***** Unsuccessful SDI Command.** — The drive fails to respond to an SDI command. FORMAT issues SEEK, RECALIBRATE, and DRIVE CLEAR SDI commands.

6.2.3.3 Warning Message

The FORMAT utility prints only one warning message.

***** WARNING: Possible Head Addressing Problem.** — No sector was successfully read from one or more tracks in the DBN area. Note that all cylinders are checked. This is a simple check for a bad head.

6.2.3.4 Information Messages

Following are the informational messages printed by FORMAT.

***** LBN n Is Bad (x), A Non-Primary Revector.** — Prints for LBNs retired by being revectored to some RBN other than the primary RBN; they are marked in the RCT as nonprimaries. They are formatted with a header code of non-primary or with a header code of bad if their header area is bad.

***** LBN n Is Bad (x), A Primary Revector to RBN n.** — Prints for LBNs retired by being revectored to the first RBN on the same track; they are marked in the RCT as primaries. They are formatted with a header code of primary.

***** LBN n Is Bad (x), in the RCT Area.** — Retired LBNs in the RCT area are formatted with a header code of bad.

***** RBN n Is Bad (x).** — Retired RBNs are marked bad in the RCT and are formatted with a header code of bad.

***** Cylinder n, Group n, Track n, Position n, PBN n.** — The information in this message identifies the position of the bad block. This message prints if the user requested the special option to print bad block position.

***** FORMAT: Aborted by User!** — FORMAT was aborted. The disk may be in an unusable state if the format has begun.

***** n Cylinders Left in xBN Space at hh:mm:ss** — Prints after every 100 cylinders are formatted to record the progress of the FORMAT program.

***** Only DBN Area Formatted (n Bad DBNs).** — FORMAT is formatting the DBN area only. It prints after the format of the DBN area is completed. After this message prints, the program terminates.

***** Drive Went Offline.** — The unit selected went off line or was declared inoperative while FORMAT was running.

6.2.3.5 Error Messages

Following are the error messages printed by FORMAT.

***** Drive Does Not Exist.** — The unit requested does not exist. FORMAT reprompts for the correct input.

***** Input Value Must Be "Y" or "N".** — FORMAT expected a Y or N but received a different value. FORMAT reprompts with the same question.

***** No Default Is Allowed.** — FORMAT expected a value but received null input. FORMAT reprompts with the same question.

***** Input Value Is an Invalid Drive Specification.** —- FORMAT expected input in the form Dnnn but received a different response. FORMAT reprompts with the same question.

***** Tape Drives Are Not Allowed.** — FORMAT expected the name of a disk drive but instead received the name of a tape drive. FORMAT reprompts with the same question.

***** Drive Could Not Be Acquired.** — The unit requested is unavailable. It may be in use by a host of another diagnostic, or it may be inoperative. The program reprompts for another unit.

***** Drive Could Not Be Brought Online, MSCP Status: m.** — the unit requested is available but the ONLINE command failed. The unit is released and FORMAT reprompts for another unit.

6.2.3.6 Success Messages

Following is the FORMAT success message.

***** Format Begun at hh:mm:ss.** — FORMAT is formatting the disk.

***** Format Completed at 18:19:07.** — successfully completed.

6.3 Disk Verifier Utility (VERIFY)

VERIFY checks the integrity of the disk architectural structure. This utility helps Digital Customer Services personnel ensure disks conform to the Digital standard disk format.

VERIFY may print various messages during the test. These messages have significance only when VERIFY reports the drive is bad.

Note: The VERIFY utility only reads the disk. It does not destroy user data and does not perform bad block replacement.

The following steps describe the process by which this utility verifies a disk:

1. The first block of the factory control table (FCT) is read to determine how the disk is formatted. The following items are printed:

   - Mode
   - Date last formatted
   - State of the FCT
   - Scratch area parameters (offset, size of not last, and size of last)
   - Format version
   - Date first formatted
   - Format instance
   - Number of bad PBNs
   - Flags

2. The first block of the revector control table (RCT) is then read. The information in the RCT is printed, including the serial number, flags, and bad block replacement variables (LBN being replaced, replacement RBN, and bad RBN).

3. All copies of the first two blocks in the RCT (used by bad block replacement) are read and compared. Discrepancies or bad blocks are reported.

4. All copies of the rest of the RCT are read and compared. Any discrepancies or bad blocks are reported. The information about revectors and bad RBNs is dumped. A summary of the number of bad blocks and revectors by type is printed.

5. All copies of FCT block 0 are read and compared, and bad blocks or discrepancies are reported.

6. All copies of the appropriate FCT subtable are read (if not null) and bad blocks or discrepancies are reported.

7. The list of bad PBNs is printed. Each entry is printed with the header bits, PBN number, and xBN number (in parentheses) as separate fields. If a bad PBN that should be in the RCT is found, the xBN field is printed in brackets instead of parentheses. An error message indicating the total number of PBNs is printed at the end of the bad PBN list.

8. After reading and dumping the FCT, a quick scan of DBN space is done. Every block is accessed only once. Counts of various errors are recorded for a summary printed at the end of the scan. If more than nine positioner errors are detected, a message is printed suggesting DBN space be reformatted.

9 All LBN space up to the RCT and all RBNs are scanned. Any block with an error is reread five more times to determine the type of error. Information about bad blocks and revectors collected in this phase is compared with information collected from reading the RCT. During the scan, four error classes can be found:

Error	Description
Structure errors	Considered inconsistencies; always reported.
Permanent recoverable errors	Usually ECC errors; reported if requested.
Permanent unrecoverable errors	Considered inconsistencies; always reported.
Transient errors	Occurs if one of the five reads of a block with an error does not report an error. Transient errors are reported if requested.

10 At the end of the scan, certain other errors are reported. Some errors can only be determined by examining information collected during the scan.

11 Finally, a summary, by type, of the errors detected and certain other information is printed. If no inconsistencies were discovered, a message prints saying the drive is OK. Otherwise, the message indicates the number of inconsistencies.

Note: VERIFY under the KDM70 controller only runs on drives formatted in 512-byte sector mode. Otherwise, a fatal message is printed and VERIFY exits.

VERIFY does not run on drives without RBNs, such as an ESE. A fatal error message is printed and VERIFY exits.

6.3.1 Invoking VERIFY

VERIFY is a utility and can be invoked either online or in standalone mode.

6.3.1.1 Invoking VERIFY On Line from VMS

Use the following procedure to access and run on-line programs. Go to Section 6.3.1.2 for instructions on running programs in standalone mode.

Note: You cannot run on-line diagnostics, exercisers, and utilities without first running EVRLN.KDM. It is important that you follow this procedure.

```
$ RUN SYS$SYSTEM:SYSGEN

SYSGEN> CONNECT FYA0/NOADAPTER

SYSGEN> EXIT

$ SET DEFAULT SYS$MAINTENANCE

$ SET HOST/DUP/SERVER=DUP/LOAD=EVRLN.KDM PUA0/DEVICE

$ SET HOST/DUP/SERVER=DUP/TASK=VERIFY PUA0/DEVICE
```

6.3.1.2 Invoking VERIFY Standalone from the VAX Diagnostic Supervisor

Use the following procedure the invoke VERIFY from standalone mode:

```
DS> ATTACH KDM70 HUB DUx N BR
                         |  |
                         |  |__________ BUS REQUEST
                         |
                         |____________ NODE NUMBER

DS> SELECT DUx

DS> RUN EVRLN

EVRLN> RUNL VERIFY
```

6.3.2 Interrupting VERIFY

To interrupt VERIFY, press CTRL/Z and Return. Or, you can use CTRL/Y, CTRL/C, or CTRL/ \ (backslash). However, CTRL/Z is not recognized by VERIFY when running under VAX/DS.

6.3.3 VERIFY Errors and Information Messages

This section describes error and information messages that may be printed by VERIFY.

Error message fields with variable output print are in **bold print**. Definitions for these fields are:

xCT = FCT or RCT
n = A decimal number
xBN = LBN, RBN, or XBN
m = MSCP status
e = Error: ECC, EDC, and so on

6.3.3.1 Fatal Error Messages

Following is a list of the error messages fatal to the VERIFY utility. The program terminates after printing one of these messages.

***** All Copies of xCT Block n Are Bad:** —- All copies of some block in either the RCT or the FCT are bad. The program cannot continue to run because vital information is missing. In any case, it has verified that the unit is bad.

***** Drives Formatted in 576 Byte Mode Are not Supported.** —- The mode field in FCT block zero indicates the unit is formatted in 576-byte mode. In this case, VERIFY cannot run because it cannot read sectors 576 bytes long.

***** Drive Went Offline.** —- The unit selected goes offline while VERIFY is running.

***** VERIFY Initialization Failure: Storage Could Not Be Acquired.** —- VERIFY cannot acquire the necessary resources to run or the disk functional code is not loaded.

***** Mode Is Bad or Format Is in Progress on this Unit:** —- The mode field in FCT block 0 of the selected unit is not valid.

***** Drives with No RBNs Are Not Supported:** —- The drive has no RBNs. One such drive is the ESE20.

6.3.3.2 Warning Messages

The following messages are warning messages. In many cases, they are true warnings; in other cases, they simply precede a reprompt.

***** n Bad PBNs (in [...] above) Not in the RCT.** —- The count is not 0. After the RCT has been collected, the appropriate subtable of the FCT is read. The list of PBNs is printed. The collected RCT is searched for RBNs and non-RCT LBNs corresponding to PBNs; they should be there. If they are not found, the LBN or RBN corresponding to the PBN is printed in brackets and counted.

***** Drive Could Not Be Brought Online, MSCP status: m.** —- The unit requested is available but the ONLINE command failed. The unit is released. You are prompted for another unit.

***** Copy n of xCT Block n (xBN n) Does Not Compare.** —- A block does not compare to the first good one. All copies of every RCT or FCT block are read and compared to the first good copy read.

***** LBN n, a Non-Primary Revector, Is Improper.** —- A nonprimary LBN is not a revector but is recorded in the RCT as such. When VERIFY reads an LBN with a header indicating it is a nonprimary revector, it looks it up in the collected RCT information and flags the fact.

***** LBN n, a Primary Revector, Is Improper.** —- An LBN is not a primary revector but is recorded in the RCT as such. When VERIFY reads an LBN with a header indicating it is primarily revectored, it looks it up in the collected RCT information and flags the fact. This message prints after the scan for unflagged blocks in the RCT.

***** LBN n Revectors to RBN n Which Is Bad.** —- VERIFY finds an RBN is good (can be read with error recovery) or only has a forced error (after error recovery), it looks it up in the collected RCT table. If the RBN is found, VERIFY re-marks it as good. This message prints after the scan for unflagged blocks in the RCT.

***** Drive Does Not Exist.** —- The unit number entered does not correspond to any known unit. The program reprompts for the unit number.

***** Drive Could Not Be Acquired.** —- The unit requested is unavailable. It may be in use by a host or another diagnostic, or it may be inoperative. The program reprompts for another unit.

***** xBN n Has a Hard EDC Error.** —- LBNs and RBNs have a bad EDC (neither correct nor forced error). This error is classed as an inconsistency.

***** xBN n Is Bad but Not in the RCT.** —- VERIFY accesses a particular track for LBNs or RBNs only once. LBNs or RBNs with errors are recorded. They are then read five more times, one LBN or RBN at a time. If errors are detected each time the LBN or RBN is accessed, and all of the errors are header errors but the LBN or RBN is not recorded in the RCT, this error message is printed.

***** xBN n has An Error, MSCP Status: m** —- The return from the I/O operation is not a SUCCESS or a forced error, EDC error, or uncorrectable ECC error.

***** Input Value Must Be "Y" or "N".** —- Verify expected a Y or N but did not receive one. The program reprompts with the same question.

***** No Default Allowed.** —- VERIFY expected a value but received a null input. The program reprompts with the same question.

***** Input Value is an Invalid Drive Specification.** —- VERIFY expected a string in the form Dnnn but received a different string. The program reprompts with the same question.

***** Tape Drives Are Not Allowed.**— You replied to the prompt with a drive specification for a tape drive. The program reprompts with the same question.

***** LBN n, A Primary, Is Not in the RCT.** —- VERIFY accesses a particular track for LBNs or RBNS only once. LBNs where errors are detected in this initial pass are recorded. They are then reread five more times, one LBN at a time. If the header for the LBN indicates that it is a primary but the LBN is not recorded in the RCT, this error message is printed.

This section contains a list of Warning and Informational messages. Warning messages are always printed. Informational messages are only printed if requested by the operator.

***** LBN n Has Corrupted Data (Forced Error).** — Prints if you answer Y to the FORMAT prompt. However, if the unit underwent bad block replacement, this message is printed only if information messages were requested.

Normally, all LBNs have a correct EDC indicating their data is good. However, a bad block replacement that occurs when the data could not be recovered produces a revectored LBN with a forced error flag. This indicates the data probably is bad. No such LBNs should exist just after FORMAT has run.

***** RBN n Is Good But Not Used for a Revector.** — A good RBN with a valid EDC is found in the verification pass but not recorded in the RCT as used. Unused RBNs on a disk are written with a forced error indication. (The EDC is the complement of the proper EDC.) No such records should exist just after FORMAT has been run. This message prints only if you answered Y to the FORMAT prompt. However, if the unit underwent bad block replacement, this message is printed only if informational messages are requested.

***** RBN n Marked Bad in the RCT Was Not Bad.** — Prints if the question was answered with a "Y" to the prompt about FORMAT. However, if the unit has been subject to bad block replacement, this message is printed only if informational messages are requested. When VERIFY reads a bad RBN (bad header or header code of bad), it looks it up in the collected RCT information and flags the fact it was indeed found to be bad. If bad RBNs recorded in the RCT are in fact all right, this flag is not set. No such RBNs should exist just after FORMAT has been run.

***** xBN n Has an Uncorrectable ECC Error.** — Prints conditionally. For example, no LBN should have an uncorrectable ECC error; it should be revectored either by FORMAT or by bad block replacement. Thus, for an LBN, this error is considered an inconsistency. Also, FORMAT should have discovered all RBNs with uncorrectable ECC errors and marked them as bad in the RCT. If an RBN is found with an uncorrectable ECC error, but that RBN is not in the RCT, it is also considered an inconsistency. In both of these cases, this message is printed. If an RBN is discovered with an uncorrectable ECC error marked bad in the RCT, this message is printed only if information messages are requested.

6.3.3.3 Informational Messages

Following are descriptions of the informational messages printed by VERIFY. Note that this type of message may or may not need informational messages enabled in order to print.

***** VERIFY: Aborted by User!** — VERIFY was aborted.

***** Drive is OK.** — No inconsistencies were discovered.

***** There Were n Inconsistencies Found for this Drive.** — Inconsistencies were discovered.

***** Copy n of xCT Block n (xBN n) Is Bad.** — RCT or FCT blocks cannot be read correctly with error recovery.

***** NOTE: Table Is Null or Empty (no bad PBNs).** — FCTs are null or empty FCTs. This message prints whether or not informational messages are enabled.

***** DBN Area Should Probably Be Reformatted.** — More than nine DBNs were detected with position errors. This message prints whether or not informational messages are enabled.

***** LBN n, a Primary Has a Bad Header (is Non-Primary).** — LBNs are recorded in the RCT as primary revectors but have garbled headers. Such a condition is abnormal but not erroneous. This message prints if informational messages are enabled.

***** xBN n Has a Transient (n out of 6) Error.** — An LBN or RBN has been read six times with a least one error-free. This message prints when informational and transient error messages are enabled. The number of times that errors were detected is indicated in the message.

***** xBN n Has a n Symbol Correctable ECC Error.** — LBNs or RBNs have solid ECC errors (errors on all six accesses) that are correctable. This message prints when informational messages are enabled. The highest number of symbols corrected on any given access is indicated in the message.

***** xBN n Has a Solid Unrecoverable e Error.** — LBNs or RBNs appear errors on all six accesses. This message appears when informational messages are enabled. The errors include those other than ECC or EDC. The record is read a seventh time with error recovery to determine if the error is correctable. If it is not, a warning message is printed.

7 ILEXER: In-line Exerciser

7.1 ILEXER Overview

ILEXER is a KDM70 controller-resident high-level exerciser designed to verify the data transfer capabilities of the KDM70 subsystem. The subsystem includes the disk and/or tape drives attached to the KDM70 controller, as well as the controller-resident software.

Note: ILEXER writes to the customer area of the disk (LBN). It does not use the diagnostic write area of the disk (DBN). Ensure customer data has been protected through back-up procedures.

7.2 Invoking ILEXER

ILEXER is initiated by a host command through a DUP connection to the KDM70 controller. The KDM70 controller can use either of the following host DUP control programs:

- EVRLN: VAX/DS standalone environment
- HSCPAD: VMS on-line environment

7.2.1 Invoking ILEXER On Line from VMS

Use the following procedure to access and run on-line programs. Go to Section 7.2.2 for instructions on accessing and running programs in standalone mode.

Note: You cannot run on-line diagnostics, exercisers, and utilities without first running EVRLN.KDM. It is important that you follow this procedure.

```
$ RUN SYS$SYSTEM:SYSGEN

SYSGEN> CONNECT FYA0/NOADAPTER

SYSGEN> EXIT

$ SET DEFAULT SYS$MAINTENANCE

$ SET HOST/DUP/SERVER=DUP/LOAD=EVRLN.KDM PUA0/DEVICE

$ SET HOST/DUP/SERVER=DUP/TASK=ILEXER PUA0/DEVICE
```

7.2.2 Invoking ILEXER Standalone from the VAX Diagnostic Supervisor

Use the following procedure to invoke ILEXER from VAX DS:

```
DS> ATTACH KDM70 HUB DUx N BR
                         |  |
                         |  |___________ BUS REQUEST
                         |
                         |_____________ NODE NUMBER

DS> SELECT DUx

DS> RUN EVRLN

EVRLN> RUNL ILEXER
```

7.3 Running ILEXER

The following is an example of an ILEXER run:

```
EVRLN> RUNL ILEXER

*** ILEXER (InLine Exerciser) V 001  *** 29-AUG-1989 17:46:37 ***

Enable Bad Block Replacement (Y/N) [N] ?
Available Disk Drives: D0001 D0002 D0003 D0004 D0010 D0012 D0607
Available Tape Drives: NONE
Select next drive to test (Tnnnn/Dnnnn) [] ? D1
Write enable drive (Y/N) [N] ?

              *** Available tests are ***

1. Random I/O
2. Seek Intensive I/O
3. Data Intensive I/O
4. Oscillatory Seek

Select test number (1:4) [1] ? 1
Select start block number (0:547040) [0] ?
Select end block number (0:547040) [547040] ?
Select data pattern number 0=ALL (0:15) [0] ?
Select another drive (Y/N) [] ? N
Select execution time limit, 0=Infinite, minutes (0:65535) [0] ? 2
Select report interval, minutes (0:65535)[1]? 1
Select hard error limit (0:32) [0] ?
Report soft errors (Y/N) [N] ?

            Execution Performance Summary at 29-AUG-1989 17:53:34

Unit    Serial Number Requests   KB Read     KB Written   Hard   Soft   ECC
D0001       182507185     2037      4579            0      0      0      0

            Execution Performance Summary at 29-AUG-1989 17:54:34

Unit    Serial Number Requests   KB Read     KB Written   Hard   Soft   ECC
D0001       182507185     4093      9250            0      0      0      0

            Execution Performance Summary at 29-AUG-1989 17:54:34

Unit    Serial Number Requests   KB Read     KB Written   Hard   Soft   ECC
D0001 *     182507185     4098      9259            0      0      0      0

                      *** ILEXER is exiting. ***
```

7.4 ILEXER Runtime Options

ILEXER prompts you for various options and parameters, which are explained in the following sections.

7.4.1 Bad Block Replacement

ILEXER prompts you to enable or disable bad block replacement (BBR) during testing:

```
> Enable Bad Block Replacement (Y/N) [N]?
```

Enter Y to enable BBR or N to disable BBR for all drives under test.

7.4.2 Disk and Tape Selection

ILEXER displays the disk and tape drives available for testing.

```
> Available Disk Drives: D0006 D0018 D0076 D0149 D0152
> Available Tape Drives: T0000 T0001
```

This is followed by a selection prompt:

```
> Select next drive to test (Tnnnn/Dnnnn) [] ? D6
```

Leading zeros may be omitted. There is no default answer for this prompt. ILEXER has separate categories of prompts, depending on whether you choose to test a disk drive or tape drive. Those prompts are listed in the following sections.

7.4.3 Disk Drive Runtime Parameters

If you chose to test a disk drive, the following prompts are displayed:

7.4.3.1 Disk Write Enable

ILEXER does not write to the user area of the disk unless enabled through this parameter. ILEXER displays the following question:

```
> Write enable drive (Y/N) [N]?
```

Enter N to write protect the selected disk drive.

CAUTION: **A Y destroys the user data area of the disk. Protect user data by first backing up the data on the disk drive to be tested.**

ILEXER verifies a Y response with the following prompt:

```
> Should user data really be overwritten (Y/N) [N]?
```

Enter Y to write enable the disk drive. Enter N to cause ILEXER to issue read-only commands.

7.4.3.2 Disk Initial Write

This prompt allows specific data patterns to be protein to a disk drive. (A data pattern is written only to the selected area of the disk to be tested. This parameter is prompted for later in the program.)

```
> Perform initial write (Y/N) [N] ?
```

Enter N to prevent an initial write. Enter Y to enable an initial write to the selected drive.

7.4.3.3 Disk Test Selection

The disk portion of ILEXER consists of four tests. The tests are displayed in a message menu followed by a prompt requiring a number, as in the following example:

```
> *** Available tests are:
>     1. Random I/O
>     2. Seek Intensive I/O
>     3. Data Intensive I/O
>     4. Oscillatory Seek

> Select test number (1:4) [1]?
```

Illegal responses cause ILEXER to reprompt.

7.4.3.4 Disk LBN Range

The range of logical block numbers (LBNs) can be specified when exercising disk drives. Two prompts allow you to select the starting and ending range of block numbers to be exercised.

```
> Select start block number (0:1216664) [0] ?

> Select end block number (0:1216664) [1216664] ?
```

The end block number is adjusted to reflect the response to the start block prompt. The response must be a number in the range specified. The default is the entire LBN range.

7.4.4 Tape Runtime Parameters

If you chose to test a tape drive, the following prompts are displayed.

7.4.4.1 Tape Test Selection

The tape portion of ILEXER consists of three tests. The tests are displayed in a message menu followed by a prompt requiring a number, as in the following example:

```
> *** Available tests are:
>     1. Random I/O
>     2. Position Intensive I/O
>     3. Data Intensive I/O


> Select test number (1:3) [3]?
```

Illegal responses cause ILEXER to reprompt.

7.4.4.2 Recording Density

ILEXER allows you to specify the recording density to be used. Only the densities supported by the type of tape drive under test are shown.

```
> *** Available Tape Density Values:
>     1 PE 1600 BPI
>     2 GCR 6250 BPI

> Select density (1, 2) [2]?
```

The response must be a number in the range specified. Any other response is illegal and causes ILEXER to reprompt for a legal value.

7.4.4.3 Tape Speed

ILEXER allows automatic speed management of tape speed on tape drives with speed selection. The following prompt is presented if the tape drive supports speed selection:

```
> Automatic speed management (Y/N) [Y]?
```

Enter Y to direct the drive to automatically select tape speed based upon recording density. Enter N to use the drive's default speed.

7.4.4.4 Record Size

ILEXER permits you to specify the size of the record to be written. The following prompt is displayed:

```
> Select tape record size, bytes (4:65536) [8192] ?
```

The response must in the range specified. Any other response is illegal and causes ILEXER to reprompt for a legal value. The default record size is 8 Kbytes.

7.4.4.5 Record Count

ILEXER allows you to specify the number of records to read/write. The following prompt is displayed:

```
> Select tape record count (0:65536) [0] ?
```

Note: Enter a zero to specify a read or write to the end of tape (EOT).

The response must be a number in the specified range. Any other response is illegal and causes ILEXER to reprompt for a legal value.

7.4.5 ILEXER Common Configuration Parameters for Disk or Tape

The following parameters are common to disk or tape.

7.4.5.1 Data Pattern

One of 16 data patterns can selected for testing disk or tape. The following prompt is displayed:

```
> Select data pattern number 0=ALL (0:15) [0] ?
```

The response must be a number in the range specified. Any other response is illegal and causes ILEXER to reprompt for a legal value. The default causes ILEXER to use all 16 patterns, one for each 512-byte buffer.

7.4.5.2 Data Compare

ILEXER permits comparisons of data read with the expected patterns when INITIAL WRITE has been selected. The following prompt is displayed:

```
> Data compare (Y/N) [N] ?
```

Enter N to prevent the data check. Enter Y to perform a data check. This generates a second prompt. (The media must have been previously written with one of the disk patterns.)

```
> Data compare on all reads (Y/N) [N] ?
```

Enter Y to implement data checking for every read. Enter N to implement data checking on only 15 percent of disk reads.

7.4.5.3 Additional Drives

Additional drives may be configured for testing. The following prompt is displayed:

```
> Select another drive (Y/N) [] ? Y
```

Enter Y to repeat the drive configuration sequence, starting in Section 7.4.2. Enter N to stop the drive configuration sequence and start the execution control sequence.

7.4.5.4 Report Interval

ILEXER displays a periodic activity report for all drives. The report interval may specified with the following prompt:

```
> Select report interval, minutes (0:65536)[1] ?
```

The response must be in the range specified. Any other response is illegal and causes ILEXER to reprompt for a legal value. A 0 disables the periodic report; the default interval is 1 minute.

7.4.5.5 Hard Errors

You can specify the number of hard errors allowed for each drive during testing. If the limit is exceeded for a drive, ILEXER drops the drive from further testing. The following prompt is used for this parameter:

```
  > Select hard error limit (0:32) [0] ?
```

The response must be in the range specified. Any other response is illegal causes ILEXER to reprompt for a legal value. Zero is the default response and allows unlimited hard errors for all drives.

7.4.5.6 Soft Errors

You can enable or disable the display of soft errors with this parameter. The following prompt is displayed:

```
  > Report soft errors (Y/N) [Y] ?
```

Enter Y to enable the display of soft errors. Enter N to disable the display of soft errors.

7.4.5.7 Data Compare Errors

Data compare errors are called out for each drive, along with the LBN, byte offset into the bad sector, the expected longword, and actual longword. The following is an example of a data compare error display:

```
> *** Compare Error (Pattern)

>     Disk Unit   6
>     LBN         818
>     Offset      0
>     Good Data   0001CCCC
>     Real Data   000070AC

> *** 17-NOV-1858 00:05:19
```

7.4.6 ILEXER Runtime Performance Summary

Once each report interval, ILEXER displays the execution performance summary. The summary lists drives configured for test, along with number of requests issued, KBytes read and written, and error counts for the drive. The following example shows a summary:

```
>                                                      -------ERRORS-------
> Unit    Serial Number Requests   KB Read  KB Written Hard   Soft   ECC
> -----  ------------- --------   --------  ---------- ------ ------ ------
> D0006     70203686      142        247         92      0      0      0
> D0018 *   166610        115         57          0      8      0      0
```

Note: Drives that have been dropped due to errors are displayed with an asterisk (*).

7.5 Terminating ILEXER

ILEXER terminates under any one of the following conditions:

- The execution time limit is reached.
- The operator sends a DUP ABORT through a CTRL/C.
- All drives under test have been dropped due to error(s).

When completed, ILEXER displays the final execution performance summary and the following message:

```
> ***
> *** ILEXER is exiting.
> ***
```

8 ILDEVO: In-Line Device Operations Test

8.1 ILDEVO Overview

The in-line device operations test (ILDEVO) is a KDM70 controller-resident low-level diagnostic program that exercises individual disk or tape drives. ILDEVO is a disk/tape data integrity test.

ILDEVO requires exclusive access to the device under test. Therefore, the device under test must be off line to system users.

Note: ILDEVO uses only the diagnostic write area of the disk (DBN).

8.2 Invoking ILDEVO

ILDEVO can be invoked on line from VMS or in standalone mode.

8.2.1 Invoking ILDEVO On Line from VMS

Use the following procedure to access and run on-line programs. Go to Section 8.2.2 for instructions on accessing and running programs in standalone mode.

Note: You cannot run on-line diagnostics, exercisers, and utilities without first running EVRLN.KDM. It is important that you follow this procedure.

```
$ RUN SYS$SYSTEM:SYSGEN

SYSGEN> CONNECT FYA0/NOADAPTER

SYSGEN> EXIT

$ SET DEFAULT SYS$MAINTENANCE

$ SET HOST/DUP/SERVER=DUP/LOAD=EVRLN.KDM PUA0/DEVICE

$ SET HOST/DUP/SERVER=DUP/TASK=ILDEVO PUA0/DEVICE
```

8.2.2 Invoking ILDEVO Standalone from the VAX Diagnostic Supervisor

Use the following procedure to run ILDEVO from standalone mode.

```
DS> ATTACH KDM70 HUB DUx N BR
                         |  |
                         |  |__________ BUS REQUEST
                         |
                         |____________ NODE NUMBER

DS> SELECT DUx

DS> RUN EVRLN

EVRLN> RUNL ILDEVO
```

ILDEVO is initiated by a host command through a DUP connection to the KDM70 controller. The KDM70 controller can use either of the following host DUP control programs:

- EVRLN: VAX/DS standalone environment
- HSCPADD: VMS on-line environment

8.3 Running ILDEVO

When ILDEVO has successfully started, a message similar to the following is printed:

```
ILDEVO version 0.00

 Port                           Drive
Number   State   Device      Type      Number(s)
  0    Available Tape        TA79      10
  1    Available Disk        RA60      79
  2    Available Tape        TA90      4, 6
  3    Host      Unknown
  4    Available Disk        RA82      11
  5    Host      Unknown
  6    Host      Unknown
  7    Available Tape        TA81      0
```

Next, the following prompt is displayed:

```
Enter (D)isk, (T)ape or (P)ort number to test

[(Show all disks and tapes), D0 - D4095, T0 - T4095, P0 - P7] ?D79
```

Enter the type of device to be tested: Select one of the following:

D for disk
T for tape
P for port

The selected device must be followed by a unit or port number.

Refer to Section 8.4 for descriptions of disk drive tests. Refer to Section 8.5 for descriptions of tape drive tests.

After you select a device type and unit to test, ILDEVO prompts for one of the following modes of operation:

- Default mode runs through each diagnostic test once. (Section 8.3.1)
- Tailor mode allows you to loop on a test or perform repetitive iterations of a test. (Section 8.3.2)

- Internal drive diagnostic mode executes drive-resident diagnostics available on the device selected. Not all devices have self-tests. Consult device-specific documentation. (Section 8.3.1)

The following prompt is displayed:

```
Do you wish to run ILDEVO in:
     Default mode (run through all tests once)
     Tailor mode (loop on test or multiple passes)
     Internal drive diagnostic mode (execute drive resident diagnostics)
[(Default), Tailor, Internal] ?
```

Enter D for the Default mode, T for the Tailor mode, or I the Internal mode.

8.3.1 Default Mode for Disks

The default mode performs the following tests on the selected disks:

- Basic Communication test
- Disk Interrogation and Setup test
- On-line and Setup test
- Disk Internal Diagnostic test
- Read test
- Seek (Positioning) test
- Write/Read test
- Format test
- Error Detection test
- Exercise test
- Disconnect test

8.3.2 Tailor Mode for Disks

Tailor mode is really a loop-on-selected-test mode or loop-on-entire-test mode. The following message is printed in Tailor mode:

```
 1) Basic communication test
 2) Disk interrogation and setup test
 3) On-line and setup test
 4) Disk internal diagnostic test
 5) Read test
 6) Seek (positioning) test
 7) Write/read test
 8) Format test
 9) Error detection test
10) Exercise test
11) Disconnect test

Enter test number to loop on [(Run all tests), 1 - 11] ?
```

Select one of the diagnostics and ILDEVO will loop on the selected test until interrupted by a CTRL/ \ (control backslash), or CTRL/C if in standalone mode.

If you enter RETURN, ILDEVO prompts for the number of passes requested for the entire chain of tests. The following prompt is displayed:

```
Number of passes [(0),0 - 65535] ?
```

Enter the desired number of passes and carriage return. Zero allows an infinite number of passes.

8.3.3 Internal Mode for Disks

If internal mode is selected, tests 1 through 3 are executed. When the tests are completed, ILDEVO prompts for further testing with the following prompt:

```
Test 4: Device internal diagnostic test
Drive test number or exit [0 - FFFF, EXit] ?
```

Type EXIT to execute the diagnose region zero text, which is followed by device internal diagnostics tests, which are device dependent. See the device-specific documentation for listings of device internal diagnostics. Not all devices can access resident diagnostics.

The following example contains an example of ILDEVO test run:

```
ILDEVO version 1.00
 Port                            Drive

 Number   State   Device      Type      Number(s)
   0    Available Disk        RA82      3
   1    Host      Unknown
   2    Available Disk        RA90      607
   3    Available Disk        RA70      1
   4    Host      Unknown
   5    Available Disk        RA70      2
   6    Available Disk        RA70      12
   7    Available Disk        RA70      4

Enter (D)isk, (T)ape or (P)ort number to test
[(Show all disks and tapes),D0 - D4095, T0 - T4095, P0 - P7] ? D1
Do you wish to run ILDEVO in:
Default mode (run through all tests once)
Tailor mode (loop on test or multiple passes)
Internal drive diagnostic mode (execute drive resident diagnostics)
[(Default), Tailor, Internal] ?

Test 1: Basic communication test
Test 2: Disk interrogation and setup test
Test 3: On-line and setup test

RA70 (drive type 18), SDI Version 4
Microcode revision 76, Hardware revision 7

Test 4: Device internal diagnostic test

         ***** Do not abort ILDEVO until further notice. *****
***** Aborting ILDEVO at this point may cause the controller to crash *****

                 *** ILDEVO may now be aborted ***
```

```
Results from SDI DIAGNOSE command:
Drive type: 18
Test 5: Read test
Test 6: Seek (positioning) test
Test 7: Write/read test
Test 8: Format test
Test 9: Error detection test
     Subtest  1 Level 1 GROUP SELECT with illegal group number
     Subtest  2 Level 2 SEEK with illegal cylinder number
     Subtest  3 Write on write protected drive
Test 10: Exercise test
Test 11: Disconnect test
ILDEVO diagnostic complete
No errors occurred
```

8.4 ILDEVO Disk Drive Test Descriptions

The following ILDEVO subtests test SDI disk drives only. Refer to Section 8.5 for information on tests for tape drives.

8.4.1 Basic Communications Test

The device is initialized and the diagnostic checks that clocks are deasserted, then reasserted.

8.4.2 Drive Interrogation Test

This portion of ILDEVO is centered around the results of a level two GET STATUS command. Errors are reported during testing.

If drive interrogation fails, the test loops on drive interrogation indefinitely.

8.4.3 Drive On-line and Setup Test

ILDEVO brings the target device on line and sets up internal data structures. The following commands are issued to the target device:

- GET COMMON CHARACTERISTICS

  The following errors can be generated by this command:

  - No Device on Port — No clocks are present on selected port.
  - A Device Failed to Deassert Clocks After Init — No clock pulse interrupts are present.
  - Drive Failed to Reassert Clocks After Init — Deasserted clocks exceed the time-out period.
  - Receiver Ready — Receiver ready is not asserted after the reassertion of clocks.

- ONLINE
- GET STATUS
- CLEAR ERRORS

- CHANGE MODE
- RUN
- GET SUBUNIT CHARACTERISTICS
- CHANGE CONTROLLER FLAGS (Set to zero)
- RECALIBRATE

The target device is now initialized. The device name, number, SDI version, microcode, and hardware revision are reported.

If the Drive On-line and Setup test fails, the test loops indefinitely.

8.4.4 Read Test

If the Drive On-line and Setup test discovers that the drive cannot be spun up, the Read test is skipped. Otherwise, the Read test performs the following steps:

1. ILDEVO reads FCT block zero to ensure the drive is in 512-byte/sector mode. If so, ILDEVO seeks to the first read-only DBN cylinder, group, and track and ensures at least one sector can be correctly read.
2. Data is compared to a known pre-written data pattern.
3. This test is repeated on all read-only cylinders, groups and tracks. Each track must have at least one readable sector or an error is reported. If errors are detected, the drive is marked inoperative when ILDEVO exits.

Two errors can cause the Read test to terminate:

1. ILDEVO attempts to read at least one copy of the first block of the FCT. An error is reported if all copies of the FCT (block zero) are unreadable. The drive is marked inoperative and ILDEVO exits.
2. ILDEVO checks the media mode. If the drive is other than 512 bytes/sector, an error is reported, the device is marked inoperative, and ILDEVO exits.

8.4.5 Seek (Positioning) Test

The Seek test consists of performing sawtooth seeks across most of the drive while verifying the head's position at each endpoint. A formula used for computing endpoints ensures thorough testing of the entire disk. LBN, XBN, and DBN spaces are all checked.

If the Seek test fails, the test loops indefinitely.

8.4.6 Write/Read Test

This test is skipped if either the Drive On-line and Setup test discovers that the drive could not be spun up or if the drive is write protected.

ILDEVO continues to execute if write or read errors are detected; however, the drive is marked inoperative when ILDEVO exits.

The test performs the following steps:

1. A seek is issued to the first cylinder, group, and track of the DBN space. This entire track is written with known data.
2. ILDEVO attempts to read at least one sector, performing data compares on the data read. If an error occurs during the write/read, the next sector is used. At least one sector must pass the write/read test on each track.

This test is performed on every track in writable DBN space.

8.4.7 Format Test

The Format test is skipped if the Drive On-line and Setup test discovers that the drive cannot be spun up, if the drive is write protected, or if data corruption or positioning errors are detected during the Write/Read test.

Otherwise, the Format test issues a seek and read (for positioning) to the last writable track in DBN space. A FORMAT command is issued for the first half of the track. At least one sector must be read successfully from the first half of the formatted track.

8.4.8 Error Detection Test

The Error Detection test is skipped if the Drive On-line and Setup test discovers that the drive cannot be spun up.

The Error Detection test tests the drives ability to detect and report error conditions. ILDEVO forces the following errors:

- A GROUP SELECT to a nonexistent group
- A level 2 SEEK to a nonexistent cylinder
- A write operation on a write-protected drive — skipped if the drive is write protected.

Error conditions are detected through the protocol error (PE) bit in response to an ILDEVO-issued GET STATUS command.

8.4.9 Exercise Test

The Exercise test is skipped if the Drive On-line and Setup test discovers that the drive cannot be spun up or if the drive is write protected.

This test is performed only in the writable DBN area.

ILDEVO continues testing if data errors are detected during the Exercise test. If errors are detected, the drive is marked inoperative when ILDEVO completes.

The Exercise test is similar to the Write/Read test, except that ILDEVO picks a random DBN and performs a write, then a read on the selected DBN space. All data reads have ECC and EDC checked, as well as having the data compared against known patterns. The disk's operation during this test is similar to its normal on-line operation, except the data transferred is in the DBN space and not the LBN space.

Two hundred operations (reads/writes) are performed before this test completes.

8.4.10 Disconnect Test

Once the Exercise test has completed, a DISCONNECT command is sent to the drive. (If errors occur during this operation, they are reported.) The drive is released to the KDM70 subsystem.

If errors occur during the test, the drive is released in a inoperative state.

If ILDEVO executed without an error, the drive is enabled. All resources requested for ILDEVO execution return to the subsystem, and a completion message is displayed. The following example shows a sample test of the disk drive:

Example 8–1 Disk Drive Test Example

```
ILDEVO version 0.00
 Port                            Drive
Number   State   Device      Type      Number(s)
  0    Host      Unknown
  1    Available Disk        RA70      238
  2    Available Disk        RA80      139
  3    Available Disk        RA81      43
  4    Available Disk        RA82      39
  5    Available Disk        RA90      18
  6    Available Disk       ESE20      122
  7    Available Disk       ESE20      119
Enter (D)isk, (T)ape or (P)ort number to test
[(Show all disks and tapes),D0 - D4095, T0 - T4095, P0 - P7] ?D18

Do you wish to run ILDEVO in:
     Default mode (run through all tests once)
     Tailor mode (loop on test or multiple passes)
     Internal drive diagnostic mode (execute drive resident diagnostics)
[(Default), Tailor, Internal] ?

Test 1: Basic communication test
Test 2: Disk interrogation and setup test
Test 3: On-line and setup test

     RA90 (drive type 19), SDI Version 4
     Microcode revision 10, Hardware revision 16

Test 4: Device internal diagnostic test
Results from SDI DIAGNOSE command:
Drive type: 19
Device reported the following binary information after a DIAGNOSE
(displayed in hex, right to left):
0F 0E 0D 0C 0B 0A 09 08 07 06 05 04 03 02 01 00 Offset
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  000
Test 5: Read test
Test 6: Seek (positioning) test
Test 7: Write/read test
Test 8: Format test
Test 9: Error detection test
Test 10: Exercise test
Test 11: Disconnect test
ILDEVO diagnostic complete
No errors occurred

ILDEVO complete
```

8.5 ILDEVO Tape Test Descriptions

The Tape test exercises the formatter and the selected tape drive. A scratch tape must be mounted on the selected unit. The tape test of ILDEVO consists the following subtests:

- Basic Communication test
- Formatter Interrogation and Setup test
- On-line and Setup test
- Device Internal Diagnostic test
- Drive Interrogation and Setup test
- Tape Movement Test

- Disconnect test

Tests are preceded by a warning that user data will be destroyed. The following format is used:

```
All customer data will be destroyed, continue [(No), YES] ?
```

Any answer other than Y causes this test to abort.

If a port with a tape formatter is selected for testing, ILDEVO prompts for which transport to test.

8.5.1 User Setup Test

After selecting Tape for the device to be tested, ILDEVO prints the following message:

```
Do you wish to run ILDEVO in:
     Default mode (run through all tests once)
     Tailor mode (loop on test or multiple passes)
     Internal drive diagnostic mode (execute drive resident diagnostics)
[(Default), Tailor, Internal] ?
```

If you select Tailor mode, ILDEVO prints the following message:

```
The following tests are available for this device:

1) Basic Communication

2) Formatter Interrogation and Setup Test

3) On-line and Setup Test

4) Device Internal Diagnostic Test

5) Drive Interrogation and Setup Test

6) Tape Movement test

7) Disconnect Test

Enter test number to loop on [(Run all tests),1 - 11] ? Return
```

ILDEVO either requests you to input a test number or begins testing. Test descriptions are given in the following sections.

8.5.2 Basic Communication Test

The device is initialized and the diagnostic checks that clocks are deasserted then reasserted.

8.5.3 Formatter Interrogation and Setup Test

This portion of ILDEVO is centered around the results of a Level 2 GET SUMMARY STATUS command. If a successful response is received, testing continues to the On-line and Setup Test.

If the Formatter Interrogation and Setup test fails, the test loops indefinitely.

8.5.4 On-line and Setup Test

ILDEVO brings the formatter on line and sets up its internal data structures. Errors encountered during formatter setup (such as Level 2 communication errors) are reported in ASCII format.

ILDEVO uses the response from the GET FORMATTER CHARACTERISTICS command to convert short and long SDI timeouts to facilitate timing of outstanding commands.

Next, ILDEVO executes a normal error recovery using the following sequence of commands:

- ONLINE GET UNIT CHARACTERISTICS
- GET SUMMARY STATUS
- CLEAR FORMATTER ERRORS

Errors detected in the GET SUMMARY STATUS response are not reported because these errors existed before the test was run, and therefore would confuse the results of the test.

Finally, ILDEVO issues a CHANGE CONTROLLER FLAGS command and sets all flags to zero and then proceeds to the next test.

If the On-line and Setup test fails, the test loops indefinitely.

8.5.5 Device Internal Diagnostic Test

ILDEVO sends a DIAGNOSE command requesting the formatter to diagnose region 0. This region is supported by all STI tape drives. Tests consist of the drive's power-up self-tests.

A failure in device internal testing causes the test to loop on the formatter resident diagnostic indefinitely.

8.5.6 Drive Interrogation and Setup Test

The following parameters are tape specific:

- Tape density — The default tests all tape densities supported by the drive.
- Tape speed — Automatic speed manager (ASM) may be selected or testing defaults to include all tape speeds, including the automatic tape speed function.
- Testing data pattern — The default is a standard data pattern. Otherwise, the words and data that make up the data pattern (0000—FFFF) are can be selected through ILDEVO prompting.

Once the drive is set up, ILDEVO proceeds to the Tape Movement tests.

If the Drive Interrogation and Setup test fails, the test loops indefinitely. If the unit to be tested is not found, a setup error message is printed and the test finishes.

8.5.7 Tape Movement Test

The Tape Movement test consists of a number of subtests. Each test is repeated for each combination of speed and density of the unit under test.

Both state machines must be on line to execute tape movement tests.

The Tape Movement tests are executed in the order the appear.

8.5.7.1 Tape Initial Write

An error during the Tape Initial Write test causes ILDEVO to retry the failing operation until it succeeds or is aborted. However, if an end of tape (EOT) is detected during the test, a SETUP error message is printed indicating a longer tape should be mounted. ILDEVO aborts further testing.

A number of write algorithms are employed to ensure good data is being written to tape. After each operation (write or write tape mark), a GET EXTENDED STATUS command is sent to the drive. The gap count is then compared against a known good gap count.

ILDEVO then issues a SET UNIT EXECUTION MODE command, which sets the diagnose (DI) bit. This ensures that good data is written to the tape.

8.5.7.2 Read Forward Test

ILDEVO skips two gaps, reads a 1-byte record, skips another gap and reads a 2-byte record. After each read and skip combination, ILDEVO issues a GET EXTENDED DRIVE STATUS. Gap counts are compared against a known good gap count.

An error during the Read Forward test causes ILDEVO to retry the failing operation until it succeeds or is aborted.

8.5.7.3 Read Reverse Test

If the tape drive selected for testing does not support read reverse, this test is skipped.

This test executes the same as Read Forward test, except in reverse. The tape is left at the beginning of tape (BOT).

An error in the Read Reverse test causes ILDEVO to retry the operation until it succeeds or is aborted.

8.5.7.4 Tape Positioning Test

Each response to the POSITION TAPE and READ commands is checked for correct tape position. ILDEVO issues a GET EXTENDED DRIVE STATUS and checks to ensure that the intended tape position matches the actual tape position.

A number of tape positioning algorithms are used to ensure the unit is functioning properly.

An error in the Tape Positioning test causes ILDEVO to retry the failing operation until it succeeds or is aborted.

8.5.7.5 Write/Read Test

This test is started when ILDEVO issues an INITIATE REWIND command. Records are written in varying lengths and then read backward and forward.

An error in the Write/Read test causes ILDEVO to retry the failing operation until it succeeds or is aborted.

8.5.7.6 Exercise Test

This test is started when ILDEVO issues an INITIATE REWIND command. The test performs the following operations:

1. The test gets a random number ranging from the number of records to the start of the tape to the number of records to the end of the tape.
2. If the random number is positive, the test moves forward that number of records. If the number is negative, it moves backward that number of records (plus 1 if read reverse is not supported).
3. If the random number is zero, the test does not move. However, if read reverse is not supported, the test moves backward one record.
4. If the number is negative or zero, the test reads in reverse. If the number is positive or read reverse is not supported, the test reads forward.
5. The test checks the record number of the record read to make sure it is the correct record.
6. The test sends a GET EXTENDED DRIVE STATUS command and checks the gap count to ensure it is correct.

At the end of Read/Positioning test, an INITIATE REWIND command is sent to the drive.

An error in the Write/Read test causes ILDEVO to retry the failing operation until it succeeds or is aborted.

8.5.7.7 Write/Read Summary

At the end of all the Write/Read tests, a summary of the results is printed. Errors are divided into eight categories. The following is an example of the format used for this report:

```
Tape transfer error summary:
            Double Track      Single Track
             Correction        Correction      Other     Other     Other
 Media      Read    Read      Read    Read     Error     Error     Error
 Error      Fwd     Rev       Fwd     Rev         A         B         C
     0         0       0         0       0         0         0         0
ILDEVO diagnostic complete
No errors occurred

ILDEVO complete
```

8.5.8 Error Detection Test

ILDEVO uses previously written records to position the tape at the start of a 512-byte record. The following two exchanges are attempted:

1. Read a 512-byte record with a byte count of 511
2. Read a 512-byte record with a byte count of 513

If the formatter/controller does not detect each of the incorrect record lengths, an error message is printed.

8.5.9 Disconnect Test

After the drive tests have completed, a DISCONNECT command is sent to the device under test. (If errors occur during this operation, they are reported.) The device is released back to the KDM70 subsystem.

If errors occurred during the test, the drive is released in an inoperative state. If ILDEVO executes without errors, the drive is enabled.

Once released back to the system, all resources requested for ILDEVO execution are returned to the subsystem, and a completion message is printed.

8.6 Error Messages

Messages printed by ILDEVO have the following format:

```
**** KDM70 In-line Device Operations Test V0.0 **** 13-JUL-1989 10:05:35 ****
Hard error number 156, Port 7, Test 4, Subtest 1, Pass 1
```

where:

error number is the number assigned to this error.
port number is included on ILDEVO errors and messages.
test is the number of the executing test that failed.
subtest is the number of the executing subtest that failed.
pass is the current pass count.

9 DKUTIL

9.1 DKUTIL Overview

DKUTIL is a general utility for displaying disk structures and disk data. Unlike some other utilities, DKUTIL is a command language interpreter. It is intended for debugging utilities, diagnostics, error recovery, and bad block replacement. DKUTIL has become a general utility for displaying disk structure and data.

Initially, the program goes into command mode. You issue a GET command to obtain the unit to which other commands are to be applied. DKUTIL then returns to the command mode, prompting for a command, executing it, and prompting for another command.

9.2 Invoking DKUTIL

DKUTIL can be invoked on line or in standalone mode.

9.2.1 Invoking DKUTIL On Line from VMS

Use the following procedure to access and run on-line programs. Go to Section 9.2.2 for instructions on accessing and running programs in standalone mode.

Note: You cannot run on-line diagnostics, exercisers, and utilities without first running EVRLN.KDM. It is important that you follow this procedure.

```
$ RUN SYS$SYSTEM:SYSGEN

SYSGEN> CONNECT FYA0/NOADAPTER

SYSGEN> EXIT

$ SET DEFAULT SYS$MAINTENANCE

$ SET HOST/DUP/SERVER=DUP/LOAD=EVRLN.KDM PUA0/DEVICE

$ SET HOST/DUP/SERVER=DUP/TASK=DKUTIL PUA0/DEVICE
```

9.2.2 Invoking DKUTIL Standalone from the VAX Diagnostic Supervisor

Use the following procedure to invoke DKUTIL from standalone mode:

```
DS> ATTACH KDM70 HUB DUx N BR
                         |  |
                         |  |__________ BUS REQUEST
                         |
                         |____________ NODE NUMBER

DS> SELECT DUx

DS> RUN EVRLN

EVRLN> RUNL DKUTIL
```

9.3 Running DKUTIL

DKUTIL uses the following commands:

- DEFAULT
- DISPLAY
- DUMP
- EXIT
- GET
- POP
- PUSH
- REVECTOR

You can shorten commands, command options, and modifiers. For example, DUMP can be entered as DUM, DU, or D. To prevent shortened commands from becoming confusing, use enough letters to make the command unique. The shortened command depends on an order based on history and expected frequency of usage. Thus, D specifies DUMP, DI specifies DISPLAY, and DE specifies DEFAULT.

In the following descriptions, only the command (or part of the command) in **bold print** must be specified.

Some command options take optional parameters which, if omitted, default.

9.4 Command Modifiers

You can add modifiers to some commands. Modifiers are preceded by a slash (one slash for each modifier). The following commands are equivalent:

DUMP /NOEDC RBN 0

DUMP RBN/NOEDC 0

DUMP RBN 0/NOEDC

DUMP RBN 0 /NOEDC

Modifiers are processed left to right and applied to the current default modifiers. The DUMP command is the exception. The default modifiers for DUMP can be changed via the DEFAULT command. The initial default modifiers for DUMP are /DATA, /EDC, and /IFERROR.

The following is a sample session using DKUTIL. This example is standalone and runs under EVRLN.

```
EVRLN> RUNL DKUTIL

***
*** DKUTIL (Disk Utility) V 001 *** 15-SEP-1989 09:00:00 ***
***

DKUTIL> GET D95

 Serial Number:       0172200370
 Mode:                512
 First Formatted:     02-AUG-1987 00:35:47
 Date Formatted:      28-JUL-1988 00:05:09
 Format Instance:     4
 FCT:                 VALID
 Bad PBNs in FCT:     26 (512), 0 (576)

DKUTIL> DIS/F FCT

         Factory Control Table for D0095

 Serial Number:       0172200370
 Mode:                512
 First Formatted:     02-AUG-1987 00:35:47
 Date Formatted:      28-JUL-1988 00:05:09
 Format Instance:     4
 FCT:                 VALID
 Bad PBNs in FCT:     26 (512), 0 (576)


 Scratch Area Offset: 63
 Size (Not Last):     417
 Size (Last):         289

 Flags:               000000
 Format Version:      0


C 244865 (LBN  237213), C 556854 (LBN  540498), C 526287 (LBN  510826),
C 480242 (LBN  466118), C 470981 (LBN  457139), C 449212 (LBN  435999),
A   2043 (LBN    2011), A   1699 (LBN    1648),

DKUTIL> REV 1000

*** BBR attempted for LBN 1000, MSCP Status: BBR (Success).

DKUTIL> DIS/F RCT

    ***  Revector Control Table for D0095

 Serial Number:       0172200370
 Flags:               000000

 LBN Being Replaced:  1000
 Replacement RBN:     30
 Bad RBN:             0

1000 *->    33,  25512 -->   822,   139512 -->  4500

 RCT Statistics:         0 Bad RBNs.
                         3 Bad LBNs.
                         3 Primary Revectors.
                         0 Probationary RBNs.
                         0 Bad RCT Blocks.
    0 Bad First Copy RCT Blocks.

DKUTIL> DEF/NODATA
DKUTIL> DUMP LBN 1000

 ****** Buffer for LBN 1000, MSCP Status: Success

 Error Code 52:                    Block Number: 1000 (000003E8)

 Recovery Flags:  000             ECC Symbols Corrected: 0
 Retry Counts: 0                  Recovery Command: 0
 Header: 6000001E    6000001E    6000001E    6000001E
 EDC:    0045      EWC: 0000
 ECC:    4C00 5800 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000
DKUTIL> DIS CHAR LBN 1000
      Characteristics for LBN 1000 (000003E8)
 Cylinder 2, Group 8, Track 0, Position 6
 PBN 1026 (000402)
 Primary RBN 30 (6000001E) in RCT Block 3 at Offset 120
DKUTIL> DIS CHAR DISK
      Drive Characteristics for D0095
 Type:          RA70
 Media:         FIXED
 Cylinders:     1511 LBN. 4 XBN. 2 DBN
 Geometry:      1 tracks/group, 11 groups/cylinder, 11 tracks/cylinder
                33 LBNs/track, 1 RBNs/track, 34 sectors/track, 34 XBNs/track
                374 XBNs/cylinder, 363 LBNs/cylinder, 11 RBNs/cylinder
 Group Offset:  8 (LBN),  8 (XBN)
 LBNs:          547041 (host), 548493 (total)
 RBNs:          16621
 XBNs:          1496
 DBNs:          374  (read/write), 374 (read only)
 PBNs:          567041
 RCT:           198 (size), 132 (non-pad), 7 (copies)
 FCT:           204 (size), 131 (non-pad), 7 (copies)
 SDI Version:   4
 Transfer Rate: 116
 Timeouts:      3 (short), 7 (long)
 Retry Limit:   5
 Error Recover: 9 command levels
 ECC Threshold: 4 symbols
 Revision:      60 (microcode), 6 (hardware)
 Drive ID:      180E00000000
 Drive Type ID: 18
 DBN RO Groups: 11
 Preamble Size: 14 (data), 4 (header)
DKUTIL> DUMP RCT BLOCK 3/DATA
        ****** RCT Block 3, Copy 1 ******
 ****** Buffer for LBN 547043, MSCP Status: Success
```

```
  Data = 00000000 00000000 00000000 00000000
    +16  2000008D 00000000 00000000 00000000
    +32  00000000 00000000 00000000 00000000
    +48  00000000 00000000 00000000 00000000
    +64  00000000 00000000 00000000 00000000
    +80  00000000 00000000 00000000 00000000
    +96  00000000 00000000 00000000 00000000
   +112  00000000 00000000 200003E8 00000000
   +128  00000000 00000000 00000000 00000000
   +144  00000000 00000000 00000000 00000000
   +160  00000000 00000000 00000000 00000000
   +176  00000000 00000000 00000000 00000000
   +192  00000000 20000670 00000000 00000000
   +208  00000000 00000000 00000000 00000000
   +224  00000000 00000000 00000000 00000000
   +240  200007DB 00000000 00000000 00000000
   +256  00000000 00000000 00000000 00000000
   +272  00000000 00000000 00000000 00000000
   +288  00000000 00000000 00000000 00000000
   +304  00000000 00000000 00000000 00000000
   +320  00000000 00000000 00000000 00000000
   +336  00000000 00000000 00000000 00000000
   +352  00000000 00000000 00000000 00000000
   +368  00000000 00000000 00000000 00000000
   +384  00000000 00000000 00000000 00000000
   +400  00000000 00000000 00000000 00000000
   +416  00000000 00000000 00000000 00000000
   +432  00000000 00000000 00000000 00000000
   +448  00000000 00000000 00000000 00000000
   +464  00000000 00000000 00000000 00000000
   +480  00000000 00000000 00000000 00000000
   +496  00000000 00000000 00000000 00000000

   EDC:  6D5F   EWC: 0000

  DKUTIL> EXIT

***
***  DKUTIL is exiting.
***
```

9.5 Command Descriptions

The DKUTIL commands are described in the following paragraphs. Command options are separated by lines in the syntax specification. Parameters are enclosed in braces ({}). Options in brackets ([ ]) can be omitted.

9.5.1 DEFAULT Command

The DEFAULT command is outlined as follows:

- Purpose: To change the default modifiers for the DUMP command.
- Syntax: DEFAULT.
- Parameters: None.
- Modifiers: Shown in the following table.

Modifier	Related Modifier	Default	Description
/IFERROR	**NOI**FERROR	ON	Dumps the error, header, and ECC fields in the buffer if an error occurs when reading the block. When this modifier is used with the /RAW modifier, the error must occur on the reread of the block with the header code extracted from the first read.
/ERRORS	**NOE**RRORS	OFF	Dumps the error fields in the buffer.
/EDC	**NOED**C	ON	Dumps the EDC and calculated EDC fields in the buffer.
/ECC	**NOEC**C	OFF	Dumps the ECC fields in the buffer.
/DATA	**NOD**ATA	ON	Displays the data in the buffer unless the /NZ modifier is also specified.
/HEADERS	**NOH**EADERS	OFF	Displays the header fields in the buffer.
/ALL	**NON**E		The same as /ERRORS/EDC/ECC/DATA/HEADERS. Requests all fields be displayed. Its opposite, /NONE, requests no fields be displayed. When using the /NONE qualifier, only the MSCP status line prints.
/RAW	**NOR**AW		Allows reading the original LBN that was revectored rather than the RBN that would be read without the /RAW qualifier. /RAW only affects revectored (primary or non-primary) LBNs. If /IFERROR is in effect, this modifier applies only to dumping a revectored LBN.
/NZ	**NONZ**	OFF	Prevents the data from being displayed if it is all zeros. Instead, a single line indicating the data is zero is printed. It has no effect if the /DATA modifier is not specified or if it is defaulted OFF.
/BBR	**NOB**BR	OFF	Usually inhibited when a block is accessed. If this modifier is specified, bad block replacement can occur. It only occurs, however, if the error recovery code detects the block being accessed as bad and the block is an LBN in the host area.
/ORIGINAL	**NOOR**IGINAL	OFF	Saves the first data seen for display. When a block is accessed for dumping, the data is seen twice by the program if an error occurs. It is seen first just after the K detects the error and sends it to error recovery. It is seen again after error recovery takes place and the data has been corrected or reread. Usually, the data is saved for displaying when it is last seen.

- Usage: The modifiers specified are applied to the current default modifiers for the DUMP command. The result becomes the new default. Examples are:

```
DEFAULT/NONE
DEF/RAW/NODATA
DE/A/OR/NZ
```

9.5.2 DISPLAY Command

The DISPLAY command is outlined as follows:

- Purpose: To display the disk characteristics, the characteristics of a given block, the error history in the drive, the FCT, and/or the RCT.
- Syntax:
  - **DI**SPLAY **A**LL
  - **DI**SPLAY **C**HARACTERISTICS **DB**N {block}
  - **DI**SPLAY **C**HARACTERISTICS **D**ISK
  - **DI**SPLAY **C**HARACTERISTICS **L**BN {block}
  - **DI**SPLAY **C**HARACTERISTICS **P**BN {block}
  - **DI**SPLAY **C**HARACTERISTICS **R**BN {block}
  - **DI**SPLAY **C**HARACTERISTICS **X**BN {block}
  - **DI**SPLAY **E**RRORS **/PROMPT**
  - **DI**SPLAY **F**CT
  - **DI**SPLAY **R**CT
- Parameters: Block is a number specifying the DBN, LBN, PBN, RBN, or XBN whose characteristics are displayed. The default radix is decimal, and can be changed to hex by prefixing the number with the letter X.
- Modifiers:
  - **/F**ULL — Displays all defined fields in xCT block 0. /FULL applies only to the RCT and FCT command options. For the RCT option, the bad block replacement fields in RCT block 0 are only displayed if the appropriate flags in the flags field are set. These flags indicate they are currently in use (BBR). This modifier forces all fields to be displayed, regardless of the flags' settings. For the FCT option, the scratch area parameters, format version, and format flags are normally not displayed. This modifier forces all fields in FCT block 0 to be displayed.
  - **/PROMPT**— Prompts for further input after displaying a full page of error information.
  - **/N**OITEMS — Does not display the individual items in the FCT or RCT. It applies only to the FCT and RCT command options. If given, only the block 0 information is displayed.
- Usage:
  - **D**ISPLAY **A**LL — Displays FCT, RCT, disk characteristics, and error history.
  - **D**ISPLAY **C**HARACTERISTICS **D**ISK — Displays the following characteristics:

Drive type	Media	Cylinders
Geometry	Group offsets	Number of LBNs
Number of RBNs	Number of XBNs	Numbers of DBNs
Number of PBNs	RCT parameters	FCT parameters
SDI version	Transfer rate	SDI timeouts
SDI retry limit	Error recovery command levels	ECC threshold
Revision levels	Drive ID	Drive type ID
DBN Read/Only groups	Preamble sizes	

— **D**ISPLAY **C**HARACTERISTICS **x**BN {block} — Displays the characteristics of the given block. For DBNs and XBNs, these are the block numbers in decimal and hex. Cylinder, group, track, position, and PBN are also in decimal and hex. For RBNs, the RCT block numbers and offset also are displayed. For LBNs, the primary RBN number and its RCT block number and offset also are displayed. For PBNs, the display depends on the type of block: DBN, LBN, RBN, or XBN.

— **D**ISPLAY **E**RRORS — Reads the error history in the drive. The error history in the drive is read from region 2, offset 0, and dumped in hexadecimal. For RA60 drives an end message is printed indicating there is no error region. The RA70, RA80, RA81, and RA82 drives display only 16 bytes of error log data. Succeeding drives display the error log header and all selected error log entries.

— **D**ISPLAY **F**CT — Displays the information in FCT block 0. Certain fields are not displayed unless the /FULL modifier is given. The list of bad PBNs is displayed unless the /NOITEMS modifier is given. For each item in the list, the header bits, PBN number, type (DBN, LBN, RBN, or XBN), and xBN number are displayed.

— **D**ISPLAY **R**CT — Displays the information in RCT block 0. Certain fields are not displayed unless the /FULL modifier is given. The list of revectors, bad RBNs, and probationary RBNs are displayed unless the /NOITEMS modifier is given. For bad and probationary RBNs, just the RBN number is displayed (in decimal). For revectors, the LBN number and RBN number to which it is revectored are displayed (in decimal). A primary revector is distinguished by the character sequence "–>". A non-primary revector is distinguished by the character sequence "*->".

— Examples are:

```
DISPLAY/FULL ALL
DI/F A
```

```
DI C D
DIS CHAR LBN 1000
DI/NOI RCT
```

9.5.3 DUMP Command

The DUMP command is outlined as follows:

- Purpose: To dump the given block or table of blocks.
- Syntax:
  - **D**UMP **B**UFFER
  - **D**UMP **D**BN {block}
  - **D**UMP **F**CT [**B**LOCK {number}] [**C**OPY {copy}]
  - **D**UMP **L**BN {block}
  - **D**UMP **RB**N {block}
  - **D**UMP **R**CT [**B**LOCK {number}] [**C**OPY {copy}]
  - **D**UMP **X**BN {block}
- Parameters:
  - Block is a number specifying the DBN, LBN, RBN, or XBN to be dumped. The default radix is decimal. It can be changed to hex by prefixing the number with the letter X.
  - Number is the relative block number in the FCT or RCT to be dumped. The default radix is decimal and can be changed to hex by prefixing the number with the letter X. The value must be in the range 1 through the FCT or RCT block size. That is, the first block is number 1 (not 0) and the block must lie in the range.
  - Copy specifies which copy of the given block in the FCT or RCT is to be dumped. The first copy is number 1. The value must not exceed the number of copies.
  - **DUMP xBN {block}** — The specified DBN, LBN, RBN, or XBN is read in and dumped subject to the given modifiers. The block number must be specified.
  - **DUMP xCT [BLOCK {number}] [COPY {copy}]** — If a BLOCK number is given, that block in the FCT or RCT is read in and dumped. If none is specified, every block in the FCT or RCT is read in and dumped. If COPY is not specified, it defaults to copy 1.
  - Examples of DUMP command parameters are:

```
DUMP RCT BLOCK 3 COPY 4
DU/NZ RCT C 2
DU LBN 1000
D F B 2
D X 0
```

- Modifiers:

Modifier	Related Modifier	Default	Description
/IFERROR	**NOI**FERROR	ON	Dumps the error, header, and ECC fields in the buffer when an error occurs while reading the block. When used with the /RAW modifier, the error must occur on the read of the LBN (reread) with the header code extracted from the RBN (first read). Refer to Section 9.5.1.
/ERRORS	**NOE**RRORS	OFF	Dumps the error fields in the buffer.
/EDC	**NOED**C	ON	Dumps the EDC and calculated EDC fields in the buffer.
/ECC	**NOEC**C	OFF	Dumps the ECC fields in the buffer.
/DATA	**NOD**ATA	ON	Displays the data in the buffer unless the /NZ modifier is also specified.
/HEADERS	**NOH**EADERS	OFF	Displays the header fields in the buffer.
/ALL	**NON**E		The same as /ERRORS/EDC/ECC/DATA/HEADERS. It requests display of all fields. Its opposite, /NONE, requests display of no fields. When using the /NONE qualifier, only the MSCP status line prints.
/RAW	**NOR**AW		Allows a read of the original revectored LBN (rather than the RBN that would be read without the /RAW qualifier). /RAW only affects revectored (primary or non-primary) LBNs. If in effect, the /IFERROR modifier applies only to dumping a revectored LBN.
/NZ	**NONZ**		Prevents data from being displayed when it is all 0s. Instead, a single line prints indicating the data is 0s. /NZ has no effect unless the /DATA modifier is specified. It also has no effect if /DATA is not specified (or is defaulted OFF).
/BBR	**NOB**BR	OFF	Permits bad block replacement. Normally, bad block replacement is inhibited when a block is accessed. BBR occurs if the block being accessed is detected as bad by the error recovery code and is an LBN in the host area.
/ORIGINAL	**NOOR**IGINAL		Saves the first data seen for display. When a block is accessed for dumping, the data is seen twice by the program when an error occurs. It is seen first just after the program detects the error and sends it to error recovery. It is seen again after error recovery takes place and the data has been corrected or reread. Normally, the data is saved for displaying when it is last seen.

9.5.4 EXIT Command

The EXIT command is outlined as follows:

- Purpose: To terminate execution of the program.
- Syntax: **E**XIT.
- Parameters: None.
- Modifiers: None.

- Usage: The current drive is released, all resources are returned, and the program exits. Examples are:

  ```
  EXIT
  E
  ```

9.5.5 GET Command

The GET command is outlined as follows:

- Purpose: To obtain a drive or change the current drive.
- Syntax: **G**ET {drive}.
- Parameters: Drive is a valid drive unit specification of the form Dnnn.
- Modifiers:
  - — **/NOI**MF — Allows the reading of FCT block 0 to determine the mode and the reading and writing of RCT block 0 to verify the RCT is sane. If this modifier is specified, the IMF MSCP modifier is not used in the on-line mode and these actions take place. By default, a new drive is brought on line with the IMF (MD.IMF) MSCP modifier.
  - — **/W**P — Brings the drive on line with the MSCP SET WRITE PROTECT modifier (MD.SWP) and WRITE PROTECT unit flag (UF.WPS). The drive is then software or volume write-protected.
  - — **/NOW**P — Brings the drive on line with the MSCP SET WRITE PROTECT modifier. The drive is *not* software or volume write-protected.
  - — **NOON**LINE — The drive is acquired but not brought on line with the MSCP ONLINE command. Only the display characteristics and display errors commands can be executed on a drive in this state.
- Usage: The current drive is released. The new drive is acquired and then brought on line with the requested modifiers and unit flags. If the drive is nonexistent, in use, or inoperative, the user is put back in command mode. The modifiers cannot be changed for this other unit. Examples are:

  ```
  GET D133
  G/WP D64
  ```

9.5.6 POP Command

The POP command is outlined as follows:

- Purpose: To restore the data in the current buffer from the save buffer.
- Syntax: **P**OP.
- Parameters: None.
- Modifiers: None.

- Usage: The data in the save buffer is restored to the current buffer. The data in the current buffer is lost. Examples are:

  ```
  POP
  P
  ```

9.5.7 PUSH Command

The PUSH command is outlined as follows:

- Purpose: To save the data in the current buffer in the save buffer.
- Syntax: **PU**SH.
- Parameters: None.
- Modifiers: None.
- Usage: The data previously in the current buffer is saved in the save buffer. The data in the save buffer is lost. Examples are:

  ```
  PUSH
  PU
  ```

9.5.8 REVECTOR Command

The REVECTOR command is outlined as follows:

- Purpose: To force bad block replacement for one or more given LBNs.
- Syntax: **R**EVECTOR {block}.
- Parameters: Block is a number specifying the LBN to be replaced. The default radix is decimal. It can be changed to hex by prefixing the number with the letter X.
- Modifiers: None.
- Usage: The specified LBNs are sent to the bad block replacement module to be revectored. If the LBNs are not valid or not in the RCT, the revector fails and an error message prints. Otherwise, the result of the replace attempt is printed in a message. The data in the replacement RBN is read from the specified LBN. Examples are:

  ```
  REVECTOR 1000
  R 100
  ```

9.6 Error Messages

DKUTIL error messages conform to the KDM70 controller utility error message format.

9.6.1 Error Message Variables

Certain portions of the error messages are variable and have the following meanings:

n = A decimal number
par = BLOCK, COPY, DBN, LBN, PBN, RBN, REVECTOR OR XBN
parm = The part of the command in error
status = MSCP status
text = The actual text in error
xBN = DBN, LBN, and so on
xCT = FCT or RCT

9.6.2 Information and Error Messages

The following is a list of the DKUTIL information and error messages:

- ***** Drive went offline.** — The selected unit went off line when DKUTIL attempted I/O to the selected drive.
- ***** Nonexistent unit number.** — The unit number entered does not correspond to any known unit. DKUTIL goes into command mode and you must issue a GET or EXIT command at the DKUTIL> prompt.
- ***** Drive could not be acquired.** — The unit requested is unavailable. The unit may be in use by a host or another diagnostic or it may be inoperative. DKUTIL goes into command mode and you must issue a GET or EXIT command at the DKUTIL> prompt.
- ***** Drive could not be brought on line, MSCP status: status.** — The requested unit is available, but the ONLINE command failed. The unit is released, and DKUTIL then goes into command mode. Issue a GET or EXIT command at the DKUTIL> prompt.
- ***** Invalid input value specification.** — An invalid number is entered in a command line.
- ***** Input value is an invalid drive specification.** — The command line contains an invalid drive specification.
- ***** Missing parameter.** — A command line is entered with a required parameter missing.
- ***** No buffer to dump.** — The DUMP BUFFER command is entered, and there is no current buffer. This can only happen if a drive has just been selected.
- ***** Missing modifier.** — A command line is entered with a slash (/) followed by a blank or is entered at the end of the line. A modifier is expected, but is missing.
- ***** n is an invalid par number; range is M-M.** — An out-of-range number is entered for a value for the DUMP, DISPLAY CHARACTERISTICS, or REVECTOR command.
- ***** xxx is an invalid param.** — An invalid command, command option, modifier, block type, or SET option is specified in a command line.

- ***** Copy n of xCT Block n (xBN n) is bad.** — FCT or RCT blocks cannot be read correctly with error recovery when the FCT or RCT is being read just after a drive has been selected. It also occurs when the DISPLAY FCT or DISPLAY RCT command is being used.
- ***** All copies of xCT Block n are bad.** — All copies of FCT or RCT blocks are bad. It occurs when the FCT or RCT is being read just after a drive has been selected or when the DISPLAY FCT or DISPLAY RCT command is being used.
- ***** Unable to read error log.** — The DISPLAY ERRORS command is unable to execute the read memory command.
- ***** Error log not implemented in drive.** — The DISPLAY ERRORS command is executed on an RA60 drive.
- ***** No drive is acquired.** — The requested command requires that a drive be acquired before the command can be executed. A drive can be acquired and not brought on line by using the /NOONLINE modifier with the GET command.
- ***** No drive is online.** — The requested command requires that a drive be acquired and brought on line before the command can be executed.
- ***** DKUTIL: Aborted by user!** — DKUTIL is aborted by typing CTRL/Y or CTRL/C.
- ***** Tape drives are not allowed.** — A tape drive was specified for the GET command. DKUTIL then goes into command mode and you must issue a GET or EXIT command at the DKUTIL> prompt.
- ***** Extraneous parameters.** — Extra parameters are added to a command. The command should be reentered without the extra parameters.
- ***** Missing parm.** — A required command, command option, or SET option is left out in a command line. The command should be reentered with the correct syntax.
- ***** Bad entry in Copy n of xCT Block n (xBN n) at Offset nnn: xxxxxxxx.** — During a display of the items in the FCT or RCT (due to a DISPLAY FCT or DISPLAY RCT command), an invalid entry is found.
- ***** BBR is disabled.** — A REVECTOR command is issued to a drive for which BBR is disabled because of a previous BBR failure or the drive is write-protected.
- ***** BBR attempted for LBN n, MSCP Status: status.** — A BBR is attempted. BBR is attempted in response to a REVECTOR command or when GET/NOIMF is issued for a drive which has a pending BBR.

10 EVRAE

10.1 EVRAE Overview

EVRAE is a generic level 2R on-line disk exerciser that runs under the VAX diagnostic supervisor (VAX/DS). EVRAE can exercise up to 16 disk drives. Although EVRAE is primarily a disk exerciser, it also tests controller functionality.

EVRAE is intended to test the functional level of DSA disk drives and only provides PASS/FAIL information. The error messages are informational in nature. If EVRAE fails, run other on-line exercisers for more specific fault isolation.

EVRAE can be used for the following:

- To verify ECO installations, new installations, and repairs
- To implement preventive maintenance

10.2 Using EVRAE

Note: EVRAE requires that all drives under test be mounted for exclusive access. Use the following VMS command before running EVRAE.

```
$ MOUNT/FOREIGN DUA5  ; Mount the device to test
```

Use the following procedure to invoke EVRAE:

1. Invoke the VAX diagnostic supervisor (VAX/DS). Refer to the appropriate VAX/DS user documentation for more information.
2. Use the ATTACH command at the VAX/DS prompt to attach the KDM70 controller:

   ```
   DS> ATTACH
   Device type? KDM70
   Device link? HUB
   Device name? DUA
   XMI node # (1,2,3,4,B,C,D,E) ? 2
   Bus Request Level (4 - 7) ? 5
   ```

   This sequence attaches the KDM70 controller to the HUB. The HUB is the XMI main I/O bus for the KDM70. The prompts also ask for the KDM70 controller designation, the XMI node number, and the bus request level.
3. Use the same ATTACH command format to attach disk drives for testing as in the following example:

```
DS> ATTACH
Device type? RA90
Device link? DUA
Device name? DUA3
```

This example attaches the RA90 fixed disk to the KDM70 controller. The controller designation and unit number are also specified.

4 Select the device for testing. Issue the SELECT command at the VAX/DS prompt, as in the following example:

```
DS> SELECT DUA3
```

5 Invoke EVRAE, which can be run in the autostart mode or in the manual mode.

To start EVRAE in the autostart mode, type the following command:

```
DS> RUN EVRAE
```

EVRAE displays the following prompts:

```
FOR UNIT _DUAn, VOLUME "name" ALLOW WRITES TO CUSTOMER DATA AREA
ON THIS PLATTER [(No), Yes])
```

6 Enter NO to prevent EVRAE from performing writes to the LBN area of the selected disk drive. If a volume is labeled SCRATCH, writes/reads are performed only on the DBN area of the disk.

Enter YES to begin testing on the selected devices. The following warning and confirmation message appears:

```
** WARNING - CUSTOMER DATA WILL BE OVERWRITTEN! ... CONFIRM [(No), Yes]
```

If you issue the /SECTION:MANUAL qualifier, various runtime parameter options are displayed, which are explained in the next section.

Note that after EVRAE is started, it can be interrupted with a CTRL/C to examine the summary report. Typing CONTINUE after EVRAE has been interrupted restarts EVRAE.

10.3 EVRAE Options

The following section documents EVRAE runtime options. Included in this section are parameter descriptions and parameter prompts.

Set the QUICK (verify) flag to reduce runtime from 10 minutes to 1 minute. Use either of the following commands:

```
DS> SET FLAGS QUICK
```

```
DS> SET QUICK
```

Number of minutes per pass (0 for no limit) [(10), 0-65535(D)] : Allows you to select the time interval between passes in minutes. Selecting zero puts EVRAE in an infinite pass loop.

Hard error limit for dropping a unit (0 for no limit) [(32), 0-65535(D)]: Allows you to select the upper error threshold limit for each unit under test.

Specify non-default starting and ending LBNs [(No), Yes]: Answer NO to enables testing over the entire LBN area of the disk. Answer YES to limit testing to within selected starting and ending LBNs. The next two prompts selects the starting and ending LBNs.

_Dxan starting LBN [(0), 0-max(D)]: Allows you to select starting LBN for each unit selected for testing.

_Dxan ending LBN [(max), n-max(D)]: Allows you to select the ending LBN for each unit selected for testing.

Random seek mode [(Yes), No]: Enables random block I/O within the selected LBN range. Answer NO to enable starting block numbers to be sequentially selected.

Enable read data checks if not already enabled [(Yes), No]: Answer NO to disable read data compares. Answer YES to enable read data compares.

The remaining questions only refer to write-enabled disks.

Write only [(No), Yes]: Answer NO to enable read/write operations. Answer YES to enable writes only.

Enable write data checks if not already enabled [(Yes), No]: Answer YES to enable host data compares after each write operation. Answer NO to disable host data compares.

User-defined data pattern [(No), Yes]: Answer YES to define a data pattern. Answer NO to select between 21 data available data patterns.

Select predefined data pattern (0 for sequential selection) [(0), 0-21(D)]: Select zero to causes data patterns 1–21 to be sequentially used for each write operation. Pattern number one (1) is a random pattern. (Refer to <REFERENCE>(evraepat) for data patterns.)

Number of words in data pattern [(16), 1-16(D)]: Select the number of words for the user-defined data pattern.

Pattern value [00000000-0000FFFF(X)]: Select the pattern value for the user-defined data pattern.

10.4 EVRAE Error Information

EVRAE provides three levels of informational error messages:

1. System-fatal error
2. Device-fatal error
3. Hard error

Level 3 error messages include the list of possible messages associated with a particular error.

Units are dropped from testing if one of the following conditions occur:

1. A unit exceeds its error threshold limit.
2. A device-fatal error is detected during testing.

Interrupting EVRAE with a CTRL/C and then restarting with a RUN command or a START command clears the inactive status bit of all dropped units. Testing is restarted with the devices hard or fatal error count set to zero.

A system fatal error causes EVRAE to abort. This error indicates not enough system resources were allocated to EVRAE for testing.

```
Error Number 1 Level 1: Failed to get memory space for I/O buffers
```

11 EVRLJ Subsystem Exerciser

11.1 EVRLJ Overview

The EVRLJ subsystem exerciser is a level 3 diagnostic that runs under the control of the VAX diagnostic supervisor (VAX/DS). This diagnostic exercises UDA50–A, KDB50, or KDM70 disk subsystems to verify that the subsystems are properly functioning.

Note: This diagnostic can only access the customer data area on the drives under test. Any operations requiring a write to the drive should only be used if the customer data has been adequately backed up or, on drives with removable media, a scratch pack is put in the drive.

Note: EVRLJ version 2.2 is required for KDM70 controller support.

The EVRLJ subsystem exerciser uses the MSCP interface to the selected disk controller to perform extensive input/output operations on selected standard disk interface-compatible disk drives and selected controllers. The following is a listing of tests within EVRLJ:

1. Controller Verification test
2. Subsystem Functional test
3. Deterministic subsystem exerciser
4. Modifiable subsystem exerciser
5. Memory Access Verification test

11.1.1 Controller Verification Test

The verification test can test two controllers at a time to make sure they properly initialize and pass module internal self-tests (MISTs). Controller memory and data path memory are verified through read/write operations. This test consists of the following subtests:

1. Bus Address subtest
2. Diagnostic Wrap Mode subtest
3. Controller Interrupt subtest
4. Controller RAM Verification subtest

11.1.2 Subsystem Functional Test

The Subsystem Functional test is a multidrive test. Controllers and drives are initialized and set on line and available. EVRLJ performs reads, seeks, and data compare operations on the selected devices.

11.1.3 Subsystem Exerciser (Deterministic)

The Subsystem Exerciser test is a multidrive exerciser. This test sequentially writes a test data pattern to all blocks in the customer data area (LBN space) of the disk. The first 512 blocks of data are read and compared while remaining blocks are read.

11.1.4 Modifiable Subsystem Exerciser

The Modifiable Subsystem Exerciser test is a multidrive exerciser. You can modify this exerciser by selecting the test parameters. This test performs extensive simultaneous I/O operations to selected devices.

11.1.5 Memory Access Verification Test

The Memory Access Verification test exercises multiple controllers. This test makes sure the controller can perform data transfers to and from host memory locations.

11.2 Invoking EVRLJ

Refer to the *VAX Diagnostic Supervisor User Guide* (EK–VXDSU–UG–00n) for instructions on how to load and start the VAX diagnostic supervisor.

11.3 EVRLJ Parameters

EVRLJ allows you to modify runtime parameters, which are described in this section.

```
DO YOU WISH TO CHANGE THE GLOBAL TEST PARAMETERS [(No), Yes]
```

Enter YES to modify global parameters through a series of diagnostic queries. Enter NO to use the default parameters and to begin testing. Defaults parameters are in parenthesis. No further intervention is required of the operator.

```
DRIVE HARD ERROR LIMIT [(1), 0-65535 (D)]
```

The value selected becomes the upper hard-error threshold. If a device exceeds this limit, it is dropped from further testing. Enter zero to set the the hard error threshold to allow unlimited errors. This prevents the device from being dropped, regardless of the error count.

```
EXERCISER TIME LIMIT IN MINUTES [(60), 0-65535 (D)]
```

You can select how long to run EVRLJ. Enter zero to run the test until either the error threshold has been exceeded or the program is aborted.

```
MINUTES BETWEEN STATISTICAL REPORTS [(15), 0-65535 (D)]
```

You can choose the elapsed time between reports. Enter zero to disable reports.

```
PRINT SOFT ERROR MESSAGES [(Yes), No]
```

Enter YES to print soft errors. Enter NO to print hard errors, preparation errors, system fatal errors, and device fatal errors.

```
DO DATA PATTERN VERIFICATION ON READS [(No), Yes]
```

Select YES only if all disk drives have been initially written with the deterministic subsystem exerciser. The deterministic subsystem exerciser and operator subsystem exerciser can be run by using the following VAX/DS command:

```
DS>Run EVRLJ/section:exercise
```

Enter YES to perform data compares on read operations, using a standard data pattern. Data compare operations are random. Enter NO to disable data pattern verifications.

```
USE VARIABLE LENGTH TRANSFERS [(Yes), No]
```

Enter YES to enable variable-length data transfers. Enter NO to enable fixed-length data transfers.

```
MAXIMUM TRANSFER SIZE IN BLOCKS [(16), 1-256 (D)]
```

Enter the maximum number of sectors to be read/written during one I/O operation. Select the default to vary the sector size randomly between the minimum and maximum.

```
ENABLE ERROR RETRIES [(Yes), No]
```

Enter YES to enable retries. Enter NO to disables error retries. The error is then logged as a hard error.

```
ENABLE ECC DATA CORRECTION [(Yes), No]
```

Enter YES to enable ECC error correction. Enter NO to enable ECC errors to be treated as hard errors and top disable retries.

```
RANDOMLY ACCESS DRIVE [(Yes), No]
```

Enter NO to enable sequential data transfers starting from the lowest LBN and continuing to the highest LBN. Enter YES to enable random access data transfers.

```
DATA PATTERN - 0 FOR RANDOM SELECTION [(0), 0-16 (D)]
```

This applies to write-enabled drives only. You can select 15 data patterns with an optional 16th operator-defined pattern capability. Enter the default (zero) to randomly use any one of the 15 data patterns. (Refer to <REFERENCE>(ept) for available data patterns.)

```
MODIFY DATA PATTERN 16 [(No), Yes]
```

This question is displayed only if zero or sixteen patterns are chosen from the previous query. Enter YES to modify the pattern or NO to default to a standard data pattern.

```
NUMBER OF WORDS IN DATA PATTERN 16 [(1), 1-16 (D)]
```

This question is asked only if the MODIFY DATA PATTERN 16 function is selected. Enter the number of words to be used in the pattern. The maximum number of words per data pattern is 16.

```
DATA WORD [(00000000), 00000000-0000FFFF (X)]
```

This question is asked if data pattern 16 is selected for modification. The question is repeated until all words in the pattern have been defined.

11.3.1 Device-Specific Test Parameters

The following questions are device-specific parameter questions. These are disk-dependent parameters.

```
DO YOU WISH TO CHANGE THE  DEVICE  SPECIFIC  TEST  PARAMETERS [(No), Yes]
```

Enter NO to enable device default parameters. The system disables all other prompts. Enter YES to generates a series of device-specific parameter questions.

```
THESE QUESTIONS REFER TO CONTROLLER AT NODE ID XX (H), DRIVE DUxn
```

This is device identifier message ensuring the proper unit has been selected for parameter modification.

```
DO YOU WISH TO WRITE TO THE CUSTOMER DATA AREA [(No), Yes]
```

Answer YES to write-enable customer data areas of the selected disk drive.

Caution: Customer data will be destroyed. Ensure proper back-up methods have been employed before continuing.

Enter NO to write-protect the selected drive and disable write operations.

```
ARE YOU SURE CUSTOMER DATA CAN BE DESTROYED [(No), Yes]
```

This question protects customer data areas from accidental corruption.

```
TEST OVER THE ENTIRE DISK [(Yes), No]
```

Enter YES to test the entire customer data area. Enter NO to enable the selection of the LBN range. A series of questions follow, allowing you to specify the range.

```
NUMBER OF BEGIN/END SETS TO USE [(1), 1-4 (D)]
```

Select a number to begin one of the four BEGIN/END sets.

```
STARTING LBN [(0), 0-4294967295 (D)]
END LBN [(0), 0-4294967295 (D)]
```

Select starting and ending LBN ranges for each BEGIN/END set selected. The valid range starts at zero and ends at the selected drive's upward LBN boundary. The ending LBN must be greater than or equal to the starting LBN.

Event flag 23 can be set to cause this program to default to write testing on all selected disk drives. If an operator is present (the operator flag is set) and event flag 23 is set, the following message will be printed before asking the drive specific questions:

```
WARNING!!!
Event flag 23 is set so all writable disk drives will be written!
Are you sure customer data can be destroyed [(No), Yes]
```

Enter YES to write-enable all selected disk drives. Enter NO to clear this event flag and to disable writes to selected drives.

The following is a runtime example of EVRLJ after booting VAX/DS in the standalone mode:

Example 11–1 EVRLJ Runtime Example

```
DS> ATTACH KDM70 HUB DUx 2 5
DS> ATTACH RA60 DUx DUx6
DS> ATTACH RA90 DUx DUx4
DS> SELECT DUx6
DS> SELECT DUx4
DS> SET TRACE
DS> RUN EVRLJ

..Program: EVRLJ - Level 3 Subsystem Exerciser, rev 1.0, 5 tests,
at 10:06:17.65. Testing: _DUx _DUx16

Testing: _DUx6, _DUx4

Diagnostic started at: 21-JUN-1989 12:59:20.30

Unknown interrupt (error 500) errors can occur, if running this
diagnostic on an 8200/8300 CPU module (T1001) below revision E.
This can also occur on an 8250/8350 CPU module (T1001-YA) below
revision B.  If you are below these revisions, install one of the
following FCOs.

               CPU      Module         FCO      Updated/Revision
        8200     T1001       82XBX-I002         E
        8250     T1001-YA    82XBX-I002         B
        8300     T1001       82XBX-I002         E
        8350     T1001-YA    82XBX-I002         B

Do you wish to change the global test parameters [(No), Yes] yes
Drive hard error limit [(1), 1-65535(D)]
Minutes between summary reports - 0 for no reports [(15), 0-65535 (D)]
Exerciser time limit in minutes - 0 for no limit [(60), 0-65535 (D)]
Print soft errors [(Yes), No]
Do data pattern verification on reads [(No), Yes]
Use variable length transfers [(Yes), No]
Maximum transfer size in blocks [(16), 1-256 (D)]
Enable error retries [(Yes), No]
Enable ECC data correction [(Yes), No]
Randomly access drives [(Yes), No]
Data pattern - 0 for random selection [(0), 0-16 (D)]
Modify Data Pattern 16 [(No), Yes]

Do you wish to change the device specific test parameters [(No), Yes] yes


The questions refer to controller at node id 02 (H), drive
Do you wish to write to the customer data area [(No), Yes] yes
Are you sure the customer data area can be destroyed [(No), Yes] yes
Test over the entire disk [(Yes), no]
```

Example 11–1 Cont'd on next page

Example 11–1 (Cont.) EVRLJ Runtime Example

```
Test 1: Controller Functional Test
        Subtest 1: Bus Address Subtest
        Subtest 2: Diagnostic wrap mode subtest
 Subtest 3: Controller Interrupt Subtest
        Subtest 4: Controller RAM Verification Subtest
Test 2: Subsystem Functional Test
Test #2 in progress    Time now is: 21-JUN-1989 12:59:20.30
Diagnostics runtime:     0 00:02:13.82

Subsystem I/O Summary:

Device        Unique       Bytes     Bytes    Bytes    Bytes     ECC
Name        Identifier     Written   Read   Accessed  Compared   Data
---------------------------------------------------------------------
_DUx    2 ** 00018476468782 16384     512     65536        0        0
_DUx6   2  4 00092874467348 16384     512     65536        0        0
_DUx4   2 19 00013439879874 16384     512     65536        0        0

Device Error Summary:

Device       Device       Volume             Hard/Fatal        Soft
Name          Type     Serial Number           Errors         Errors
---------------------------------------------------------------------
_DUx    KDM70                                     0                0
_DUx6   RA60          000000198373932             0                0
_DUx4   RA90          000000907844987             0                0

Test #4 in progress   Time now is: 21-JUN-1989 12:59:20.30
Diagnostics runtime:     0 00:003:31.73

Subsystem I/O Summary:

Device        Unique       Bytes     Bytes    Bytes    Bytes     ECC
Name        Identifier     Written   Read   Accessed  Compared   Data
---------------------------------------------------------------------
_DUx    2 ** 00018476468782    1M   800256   369152        0        0
_DUx6   2  4 00092874467348    1M   800256   369152        0        0
_DUx4   2 19 00013439879874    1M       1M   224789        0        0

Device Error Summary:

Device       Device       Volume             Hard/Fatal        Soft
Name          Type     Serial Number           Errors         Errors
---------------------------------------------------------------------
_DUx    KDM70                                     0                0
_DUx6   RA60          000000198373932             0                0
_DUx4  RA90          000000907844987             0                0
```

The following is an EVRLJ sample error report:

Example 11–2 EVRLJ Error Report

```
***  EVRLJ - Level 3 VAX UDA50-A/KDB50/KDM70 Subsystem Exerciser - 1.1  ***
 Pass 1, test 3, subtest 0, error 422, 16-JUL-1986 10:23:56.09
 Soft error while testing DUx2:

 SDI Error error log message received

 For controller at address 772150 (O), drive type RA81
 Operation status: [status code 00EB (H)]
        Disk drive error - drive detected error
 Operation continuing.
 Controller model type: 6, device class: 1.
 Controller hardware version: 0, software version: 5.
 Drive model type: 5, device class: 2.
 Drive hardware version: 6, software version: 7.
 The drive volume serial number is 1.
 The disk block where the failure occurred is 0.
 Drive status (right to left (H)): 19 4D 0D 04 66 00 0A 00 00 80 04 1B
 Message buffer contents (right to left (H))    Byte offset (D)
         00EB4103  00000002  00000013  00100038  :0
         00000CBB  00010005  01064141  414141C5  :16
         00000000  00000001  00000607  02050000  :32
         00000000  194D0D04  66000A00  0080041B  :48
***  End of soft error number 422  ***
```

All error message reports follow the same format.

12 EVRLM Functional Code Update Utility

12.1 EVRLM Overview

EVRLM is an operating system-independent, code update utility for VAX-based processors. It is used to update KDM70 functional code in the field and manufacturing environment.

Note: All functions noted in this chapter can now be done On-Line and this is the "preferred method". These On-Line functions are done via a host operating system (VMS or ULTRIX) dup connection. Most functions are done via "PATCH.KDM", On-Line code loads are done via dup-supplied-programs. Forced Code Update procedure must still be done via VDS. Refer to section 12.7 for this procedure.

EVRLM is also a menu-driven utility that uses a DISPLAY function for viewing device internal error logs, bad page lists, software revision levels, and other device-relevant information.

The following sections provide a basic description of EVRLM and detail its use when updating code and when using DISPLAY functions.

12.2 Invoking EVRLM

EVRLM is run from the diagnostic supervisor in standalone mode. The procedure is slightly different than the customer update function. To run EVRLM, use the following procedure. In this example, the load media is the TK50:

```
>>> B CSA1/R5:10

DS> ATTACH
Device type? KDM70
Device link? HUB
Device name? DUx
XMI node # (1,2,3,4,B,C,D,E) ? 2
Bus request Level (4 - 7) ? 5
DS> SELECT DUx
DS> RUN EVRLM

..Program: EVRLM -- LEVEL 3 KDM70 EEPROM UPDATE UTILITY,
revision 1.0, 1 test,
   at 12:00:00
Testing: _DUx

KDM70 Code Update Utility Menu
Functions That May be Performed:

 1) Perform Code Update
 2) Display Bad Page List
 3) Display EEPROM Suspect Page List
 4) Display SRAM Suspect Page List
 5) Display MIST Internal Error Log
 6) Display Real Time Error Log
 7) Display up-Time Count
 8) Display Software Revision Number
 9) Display Unique Identifier
 10) Display This Menu
 11) Exit This Program

 Enter your choice by number: [(10), 1-11(D)]
```

The boot command boots to the diagnostic supervisor prompt (DS>). By running EVRLM from the diagnostic supervisor, the menu and selection functions are enabled. The following sections document the display functions.

12.3 Using the EVRLM Help Facility

Use the following command, at the diagnostic supervisor prompt to invoke the EVRLM HELP facility:

```
DS>  HELP EVRLM
```

The help command invokes EVRLM to display informational text describing EVRLM in general terms. Help is available on the following topics:

- Device: Information on supported devices
- Data_structures: Data structures supported by this help facility
- Code_update: The function of the code update procedure
- Attach: How to use the ATTACH command
- Sections: Menu options
- Default: Defaults associated with EVRLM
- Quick: Quick flag
- Event: No event flags are supported
- Summary: A summary of hard/soft errors

Information on data structures can be obtained by typing the following command:

```
DS> HELP EVRLM DATA_STRUCTURES name_of_data_structure
```

Where *name_of_data_structure* refers to the name of the data structure to be displayed. Information can be obtained on the following data structures:

- Bad page list
- EEPROM suspect page list
- SRAM suspect page list
- MIST internal error log
- Real-time error log
- Software revision number
- Unique identifier

Information is presented in the form of the DISPLAY function for the selected data structure. This includes a description of the fields of the displayed data structure.

12.4 KDM70 SOFTWARE REVISION NUMBERING SCHEME

It is reported as a hex byte. The hex byte is converted to decimal, with the LSD reflecting the MINOR revision and the MSD's showing the MAJOR revision.

Note: If the software rev byte is 80(Hex) or greater, then the KDM70 software is proto-type software and is not supported by Customer Services.

For example, after loading V2.4 KDM70 software image, version 2.4 would report as software version of 24(18 hex) in XMI XDEV register and in host error logs. The 24 should be read as 2.4.

The software version reported via SSP (during initialization) will only reflect the MAJOR revision number (ie, if the KDM70 software were at revision 2.4, SSP will report it as a Ucode revision of 2)

New major releases increment the high order digit(s) and resets the the low order digit (ie, 2.x -> 3.0). New minor releases increment the low order digit by 2 (2.0 -> 2.2 -> 2.4 -> 2.6 -> 2.8). If any patches get applied (via PATCH.KDM), the full version number will be ORed with 1. Thus, all unpatched versions will be even and all patched versions will be odd.

Note: Prior to V3.0, if a patch were applied, the updated version number was only reflected in the host error logs. Use of the PATCH Display List was required to show if any patch had been applied.

The KDM70 Software Revisions show up in the following places:

XMI XDEV register (bits 31-24)

Read the register via host CPU command, O/S read. etc. The Most significant byte will be the KDM70 software revision (displayed in hex).

TMSCP datagrams (errlogs)

Reported via a (T)MSCP error log message in the "csvrsn" field This is normally displayed in hex, with the error log format text output converting the hex number to decimal.

SSP driver initialization (Known by many names; uqssp, pudriver, puport)

At step 4, the SA register (bits 3-0) contains the MOD 16 value of the KDM70 software version.

DUP connection via PATCH.KDM

Using the PATCH command, Display version, the KDM70 software number is displayed as a three digit decimal number. Starting with KDM70 software V3.0, use of the PATCH display image command will display detailed information about the software build location and time.

12.5 Using the DISPLAY Option

To use the DISPLAY option, enter the menu option after the following prompt:

```
Enter your choice by number: [(10), 1-11(D)]
```

EVRLM further prompts for module selection through the following prompt:

```
ARE YOU REFERRING TO BOARD 1 OR 2: [(1), 1-2(D)]
```

Select either board 1 (T2022) or board 2 (T2023). EVRLM displays the requested information.

The following sections describe the individual DISPLAY options.

12.5.1 Display Bad Page List

The bad page list is displayed in the following manner:

```
Displaying Bad Page List

Byte     Byte Address &       Expected     Returned     Time
Code    Page Frame Number     Data         Data         Stamp
-------------------------------------------------------------------------------
 3          00100000          05234871     05234870     00000010
 3          1FFFFFFF          FFFFFFFF     FFFFFFFF     FFFFFFFF
 3          1FFFFFFF          FFFFFFFF     FFFFFFFF     FFFFFFFF
 3          1FFFFFFF          FFFFFFFF     FFFFFFFF     FFFFFFFF
 3          1FFFFFFF          FFFFFFFF     FFFFFFFF     FFFFFFFF
 3          1FFFFFFF          FFFFFFFF     FFFFFFFF     FFFFFFFF
 3          1FFFFFFF          FFFFFFFF     FFFFFFFF     FFFFFFFF
```

Information is presented in the same format for board 1 or 2.

```
31 30  29                               9  8                      0
------------------------------------------------------------------------
|          |                                   |                       |
| Byte Code|         Page Frame Number         |      Byte Address     |
|          |                                   |                       |
------------------------------------------------------------------------
```

Byte code is a code which designates the parity bit that is in error. This code is valid only if the actual contents field is equal to the expected contents field. Listed below are the valid byte codes:

```
    00 - Byte 0 parity bit
    01 - Byte 1 parity bit
    10 - Byte 2 parity bit
    11 - Byte 3 parity bit
```

12.5.2 Display EEPROM Suspect Page List

The EEPROM suspect page list is displayed in the following manner:

```
Suspect      Expected        Actual       Number of           Time
Address      Data            Data         Bytes in Error      Stamp
-------------------------------------------------------------------------------
FFFFFFFF       23              FF               FF           FFFFFFFF
FFFFFFFF       FF              FF               FF           FFFFFFFF
FFFFFFFF       FF              FF               FF           FFFFFFFF
```

Information is presented in the same format for board 1 or 2.

12.5.3 Display SRAM Suspect Page List

The SRAM suspect page list is displayed in the following manner:

```
Displaying SRAM Suspect Page List

Suspect                     Error                             Time
Address                     Count                             Stamp
-------------------------------------------------------------------------------
FFFFFFFF                  FFFFFFFF                           FFFFFFFF
FFFFFFFF                  FFFFFFFF                           FFFFFFFF
FFFFFFFF                  FFFFFFFF                           FFFFFFFF
```

Information is presented in the same format for board 1 or 2.

12.5.4 Display MIST Internal Error Log

The MIST internal error log is displayed in the following manner:

```
Displaying Internal Error Log

Test #        00              00              00
Error ID      00EC            0000            0000
Test Addr     8020AF5D        00000000        00000000
ESL 1         00008204        00000000        00000000
ESL 1         00000000        00000000        00000000
ESL 2         00000000        00000000        00000000
ESL 3         00000000        00000000        00000000
ESL 4         00000000        00000000        00000000
ESL 5         00000000        00000000        00000000
ESL 6         00000000        00000000        00000000
ESL 7         00000000        00000000        00000000
Time Stamp    00000008        00000008        00000008
```

Information is presented in the same format for board 1 or 2.

The following table may be used to provide further detail on a specific failure. Only the error identifier is described. The additional information provided in the MIST internal error log is intended primarily for use by manufacturing to assist in the module repair process. In the previous example, the error ID of EC is described as a “Code Transfer Failure”. (See Table 12–1.) This failure was caused by excessive EEPROM errors detected when trying to copy the KDM70 code image from EEPROM to SRAM.

Table 12–1 Internal Error Log — Error Identifiers

Error Identifier	Description
F000	Failure during testing of Bd1 SRAM
F001	Failure during testing of Data Buf
F002	Failure of only 1 CVAX cache set
F003	EEPROM code byte(s) ECC corrected
0004	Unexpected CVAX restart occurred
0008	Unexpected interrupt/exception
000C	Unexpected NXM or bus timeout (ERR)
0010	Unexpected MEMERR interrupt
0014	Access fail of Bd1 Read Reg - (Fs)
0016	Error bit causing Read Reg to lock up
0018	Stuck-at-0 fault in Bd1 Read Reg
001C	Stuck-at-1 fault in Bd1 Read Reg
0020	Stuck-at-0 fault in Bd1 Write Reg
0024	Stuck-at-1 fault in Bd1 Write Reg
0028	Attempted bus timeout failed (VIC)
002C	Attempted NXM timeout failed (all)
0030	Failure of B1 RD Reg to clock after

Table 12–1 (Cont.) Internal Error Log — Error Identifiers

Error Identifier	Description
0034	Failure of single NXM address bit
0038	Failure to generate CVAX PERR L
003C	Failure to generate MEMERR interrupt
003E	Failure of a PERR X L line
0040	Failure with a byte mask bit going
0044	Contents of SSC TO Control invalid
0048	Invalid data read from Err Addr Reg
0054	Unexpected mach chk in bus err test
0058	Read of Read Reg failed to clear
0059	Failure of RDY PAL or Processor
005A	Failure of RDY PAL or XIC/Processor
005B	Failure of RDY PAL or Board 2
005C	Failure in attempt to read or write
0060	Failure during VIC registers test
0064	VIC failure to initialize properly
0068	VIC IRQ line is stuck-at-0
006C	Failure of VIC to recognize request
0070	VIC IRQ line is stuck-at-1
0072	CVAX IPL failure
0074	Failure during programmable timers
0078	Failure of programmable timer
007C	Failure of interval timer interrupt
0080	Failure of interval timer to occur
0084	Failure of byte mask to SRAM Bank 2
0088	Failure of byte mask to Scratchpad
008C	Failure of byte mask to Data Buffer
0090	CVAX Failure to execute Bd1 word
0094	CVAX Failure to execute Bd1 word
0098	CVAX Failure to execute Bd1 quadword
009C	CVAX Failure to execute Bd1 quadword
00A0	CVAX Failure to execute Bd2 word
00A4	CVAX Failure to execute Bd2 word
00A8	CVAX Failure to execute Bd2 quadword
00AC	CVAX Failure to execute Bd2 quadword
00B4	Failure detected on virtual write
00B8	Translation Buffer failure
00BC	Failure detected on virtual read

Table 12–1 (Cont.) Internal Error Log — Error Identifiers

Error Identifier	Description
00C0	Failure on forced TNV exception
00C4	Failure on forced ACV exception
00C8	Failure on forced ACV exception
00CC	Failure during soft interrupts test
00D0	Failure in attempt to init the KSP
00D2	Failure of Bd2 EEPROM Checkerboard
00D4	Failure of a Bd2 EEPROM read
00D8	Bad data after Bd2 EEPROM read
00DC	Overflow of Bd1 Bad Page List
00E0	Overflow of Bd2 Bad Page List
00E4	Voting test of B1 BPL failed
00E8	Voting test of B2 BPL failed
00EC	Code Transfer Failed
00F0	Failure of INVR ADDR H signal
0104	HIB address error
0108	HIB address MUX error
010C	HIB byte mask error
0110	CVAXII address MUX error
0114	HIB register chip select error
0118	HIB bus register init error
011C	HIB Write signal error
0120	No DPC parity error interrupt
0122	No CVAX parity error
0124	Unexpected CVAX parity error
0126	No DPC parity error
0128	Unexpected DPC parity error
012C	Bad HIB address parity
0130	Bad HIB control parity
0140	Bd2 Diagnostic Write Register error
0144	HIB Bus error on write operation
0148	HIB Bus error on read operation
014C	Bd2 data value error on read
0150	No HIB Address parity error
0154	No HIB Control parity error
0158	No HIB Data parity error
015C	Bd2 Bus Error Register error
0160	Bd2 Diagnostic Read Reg Parity error

Table 12–1 (Cont.) Internal Error Log — Error Identifiers

Error Identifier	Description
0164	Bd2 Diagnostic Write Reg Parity error
0200	CVAX port transaction failure
0204	CVAX port transaction not complete error
0208	CVAX port transaction did not fail on forced error
020A	CXHIC Byte Mask failure
0210	Proper bit not set in Port Activity Register
0214	CXHIC register initialization failure
0215	DPC Idle failure
0216	No DPC Idle after pre-empt
0217	VIC did not see xmism intrrpt
0218	Unexpected workblock failure
0219	CVAX did not see sm interrupt
0224	XMISM timeout
0225	SM either didn’t fail or failed unexpectedly
0226	XMI Workblock error
0227	XMI Workblock data error
0228	DPC register initialization error
0230	Inconsistent signature after DPC self-test
0234	CXHIC/HIB interface failure
0238	DPC register failure
0240	XMI Control RAM location failure
0242	XMISM microcode verify failure
0244	DPC microcode parity checker failure
0248	HIB/DPC interface failure
024C	DPC/Control RAM interface failure
0250	Stuck-at-0 fault in DPC register
0254	CXHIC auto-incremented unexpectedly
0258	CXHIC did not auto-increment when expected
025C	CXHIC did not load next page frame when expected
0264	IBPE parity error failure
0268	Data compare fail on write transaction
026C	Data compare fail on read transaction
0270	Failure of CXHIC to send MAR interrupt
0274	Failure of VIC to request MAR int
0278	Failure of CXHIC to send MES interrupt
027C	Failure of VIC to request MES int
0280	Failure during initial access

Table 12–1 (Cont.) Internal Error Log — Error Identifiers

Error Identifier	Description
0284	HIB bus error detected during XMI
0288	Failure of XBE Error Summary interrupt
0300	SISM interrupt timed-out
0301	SISM WB remote status error
0302	SISM WB remote status error
0304	SI ucode too big for CRAM
0308	Bad lower SI CRAM value
030C	Bad upper SI CRAM value
0310	SISM CRAM counter failure
0314	No parity error detected
0318	No interrupt from parity error
031C	Parity error at wrong location
0320	Interrupt with no parity error detect
0324	No CRAM counter overflow
0328	SISM CRAM parity error detected
032C	No SISM error in BER
0330	CSIC calibration counter failure
0334	SIECL calibrate failed
0340	Workblock Address Register pattern error
0344	SISM internal register error
0348	SI pulse error
034A	SI Workblock remote status error
034C	No SI pulse error
0350	SI data compare error
0354	SI address error
0358	SI port error
035A	SI header error
035B	SI transfer EDC error
035C	SI transfer ECC error
0400	XMI port transaction command failed
0404	XMI port transaction data error
0408	XMI DMA transaction failure
040C	XMI DMA byte mask failure
0410	XMI port transaction command failed
0500	RAM error detected
0E00	Purge/poll test failed

12.5.5 Display Real-time Error Log

The real-time error log contains a copy of the most recent last crash error log packet(s). This error log is duplicated on board 1 (T2022) and board 2 (T2023). It will be used primarily by manufacturing to assist in the repair process. In most cases this information will be available on line in the system error log. If an error log is not available, the real-time error log may be used for fault isolation.

The values displayed in the internal error log are determined by the type of bug check that occurred. In the following example, a bug check code of “C858C12A”, is defined as a SISM control ROM parity error. The next eight longwords are defined in Section 4.9 and are based on the MSCP event. In this case, the event is a device interface hardware error. The real-time internal error log is displayed in the following manner:

```
Displaying Internal Error log

Bug Check Code C858C12A          00000000           00000000
ESL 1          00000000          00000000           00000000
ESL 2          00000000          00000000           00000000
ESL 3          00000000          00000000           00000000
ESL 4          00000000          00000000           00000000
ESL 5          00000000          00000000           00000000
ESL 6          00000000          00000000           00000000
ESL 7          00000000          00000000           00000000
ESL 8          00000000          00000000           00000000
Time Stamp     11211050          00000000           00000000
```

12.5.6 Display Up-Time Count

The up-time count is displayed in the following manner:

```
Displaying Up Time Count

For node B the Up-Time count for board 2 is: 9 days
```

Information is presented in the same format for board 1 or 2.

12.5.7 Display Software Revision Number

The software revision number is displayed in the following manner:

```
Software Revision level : 00000001
```

Information is presented in the same format for board 1 or 2.

12.5.8 Display Unique Identifier

The unique identifier is displayed in the following manner:

```
Unique Identifier: 012345099899
```

The unique identifier consists of board one (T2022) serial number and board two (T2023) serial number as follows:

```
63       56 55       48 47                   24 23                     0
-------------------------------------------------------------------------
|          |           |                        |                       |
|  CLASS   |  MODEL    |  T2022 Serial Number   |  T2023 Serial Number  |
|          |           |                        |                       |
-------------------------------------------------------------------------
```

12.6 Running EVRLM

The code update utility is used to install new versions of KDM70 controller software. New versions of software are periodically released to improve KDM70 controller performance or to add KDM70 controller features after KDM70 controller's initial introduction.

The following is an example for completing the code update using the VAX diagnostic supervisor and the EVRLM utility booted from the VAX 6000 system console TK50/70 tape drive.

Note: The CPU key switch must be in the UPDATE position to run the code update.

1 Shutdown the system by using the System Shutdown command.

2 Install the supplied tape, containing the new KDM70 controller code image, into the TK50/70 tape drive.

3 At the system boot prompt, type the following command:

Example 12–1 Code Update Utility—Partial Screen Display

```
>>> B CSA1/R5:10

Initializing system.

(An initialization chart is displayed....)

Loading system software.

DIAGNOSTIC SUPERVISOR.  ZZ-ExSAA-01.x-xxxx 10-June-1989 12:00:00

DS> ATTACH
Device type? KDM70
Device link? HUB
Device name? DUx
XMI node # (1,2,3,4,B,C,D,E) ? 2
Bus request Level (4 - 7) ? 5
DS> SELECT DUx
DS> RUN EVRLM

..Program: EVRLM -- LEVEL 3 KDM70 EEPROM UPDATE UTILITY,
revision 1.0, 1 test,
   at 12:00:00
Testing: _DUx

KDM70 Code Update Utility Menu
Functions That May be Performed:
```

Example 12–1 Cont'd on next page

Example 12–1 (Cont.) Code Update Utility—Partial Screen Display

```
 1) Perform Code Update
 2) Display Bad Page List
 3) Display EEPROM Suspect Page List
 4) Display SRAM Suspect Page List
 5) Display MIST Internal Error Log
 6) Display Real Time Error Log
 7) Display Up-time Count
 8) Display Software Revision Number
 9) Display Unique Identifier
 10) Display This Menu
 11) Exit This Program

Enter your choice by number: [(10), 1-11(D)] 1

Updating . . . . . . . . . . . . . . . . . . . . . . . . .

CODE UPDATE SUCCESSFULLY COMPLETED

KDM70 Code Update Utility Menu
Functions That May be Performed:

 1) Perform Code Update
 2) Display Bad Page List
 3) Display EEPROM Suspect Page List
            .
            .
            .
 9) Display Unique Identifier
 10) Display This Menu
 11) Exit This Program

Enter your choice by number: [(10), 1-11(D)] 11

.. End of run, 0 errors detected, pass count is 1,
   time is day-month-year  time
```

12.7 EVRLM Errors

Generally, the code update utility is an error-free process. However, if controller memory has been degraded or if the code loaded from the tape has been corrupted, problems may arise. If the code update utility fails, the KDM70 controller operates the same as before the code update was attempted. Actual KDM70 code is not overwritten with new code until testing ensures the probability of success. The following sections outline EVRLM error handling.

If the KDM70 code update process fails and leaves the KDM70 controller in an indeterminate state, refer to Section 12.8 for error recovery procedures.

If an operation fails, the program displays the time of the failure, the failing device, and information relevant to the failure.

This section lists the errors that EVRLM can return. And explanation of the error and possible causes are included.

Error 1 — Get Hardware Parameter Table Failed

```
Error 1 - The Get Hardware Parameter Table system service failed.

***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY 1.0 ***
     Pass "pass", Initialization section, error 1, "date" "time"
     System fatal error while testing DUA: $DS_GPHARD failed

     Media load error.

     The Get Hardware Parameter Table system service failed.

     Try new load media.

***  End of System error number 1  ***
```

This error indicates a bad load device, such as the TK tape. The GET HARDWARE PARAMETER TABLE command provides information on controller type and its XMI node number. If the information is not available, no update can occur and the update is aborted.

Error 2 — Allocation of Host Communication Area Failed

```
Error 2 - Allocation of the host communication area failed

***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY 1.0 ***
     Pass "pass", Initialization section, error 2, "date" "time"
     System fatal error while testing DUA: "device": $DS_GETBUF FAILED

     Media load error.

     Allocation of host communication area failed

     Return status from $DS_GETBUF: 00000009


***  End of System fatal error number 2  ***
```

Buffer space was not allocated for the host communication area. The host communication area is used for communication between EVRLM and the KDM70 controller.

Recovery: The media is suspect, retry the update procedure. If the same failure occurs, use a different update tape or load device.

Error 3 — Allocation of Scratch Pad Failed

```
Error 3 - Allocation of controller scratch pad area failed

***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY 1.0 ***
     Pass "pass", Initialization section, error 3, "date" "time"
     System fatal error while testing DUA: "device": $DS_GETBUF FAILED

     Media load error.

     Allocation of scratch pad area failed

     Return status from $DS_GETBUF: 0000000A


***  End of System fatal error number 3  ***
```

Buffer space for the controller scratchpad area was not allocated.

Recovery: The media is suspect, retry the update procedure. If the same failure occurs, use a different update tape or load device.

Error 4 — Node Reset Failed

```
Error 4 - Node reset failed

***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY 1.0 ***
     Pass "pass", Initialization section, error 4, "date" "time"
     System fatal error while testing DUA: "device": NODE RESET FAILED

     type_2_message

***  End of soft error number 4  ***
```

A node reset is used to initialize the KDM70 controller and invoke the controller self-test sequence. The error message depends on the return status from the system service call. There are three possible error messages:

1 The logical unit selected is too large.

Recovery: The media is suspect, retry the update procedure. If the same failure occurs, use a different update tape or load device.

2 An attempt to set/clear a register adapter bit failed.

Recovery: The T2022 or current controller EEPROM image is suspect. Retry the update procedure. If the same failure occurs, use the forced-code update procedure. If the same failure occurs again, replace the T2022 module.

3 Node self-test failed.

Recovery: The T2022 or current controller image is suspect; retry the update procedure. If the same failure occurs, use the forced-code update procedure. If the failure occurs again, replace T2022 module. NOTE: The module LED codes may give additional information.

The error message is printed, and EVRLM returns the following prompt:

```
Controller failed self-test, do you wish to continue? [(No), Yes] YES
```

You can continue with the code update or abort.

Error 100 — Allocation of Controller Image File Buffer Failed

```
Error 100 - Error while allocating buffer space for controller image file

***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY - 1.0  ***
     Pass "pass", test 1, subtest 0, error 100, "date" "time"
     System fatal error while testing "device": $DS_GETBUF FAILED

     Media load error.

     Allocation of controller image file buffer failed

     Return status from $DS_GETBUF: 00000009


***  End of System fatal error number 100  ***
```

Before the EEPROM code update is performed, the controller image file is stored in a buffer located in host memory. This error indicates that the allocation of the controller image file buffer failed.

Recovery: The media is suspect, retry the update procedure. If the same failure occurs, use a different update tape or load device.

Error 110 — Controller Failed to Set Step Bit

```
Error 110 - Controller failed to set step bit during initialization

***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY - 1.0  ***
     Pass "pass", test 1, subtest 0, error 110, "date" "time"
     Hard error while testing "device":

     STEP BIT DID NOT SET IN SA REGISTER DURING INITIALIZATION

     Expected Data : 9C0 (H)

     Returned Data : 8010 (H)

***  End of Hard error number 110  ***
```

Initialization of the controller is accomplished by sequencing through steps 1 through 4 of the SSP initialization. The sequencing for one step to the next is accomplished by setting a particular step bit in the SA register. This error indicates that a step bit did not set within 10 seconds. After the error message is printed, the code update is aborted.

Recovery: This is a KDM70 controller error. Retry the code update. Refer to Appendix B for SA error codes.

Error 111 — Controller Fatal Error Detected

```
Error 111 - Controller fatal error detected during initialization

***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY - 1.0  ***
     Pass "pass", test 1, subtest 0, error 111, "date" "time"
     Hard error while testing "device":

     CONTROLLER DID NOT RETURN CORRECT STATUS IN SA
     REGISTER DURING INITIALIZATION

     Expected data:  9C0 (H)

     Returned data:  9CF (H)

***  End of Hard error number 111  ***
```

The controller returned an incorrect value in the SA register.

Recovery: This is a KDM70 controller error. Retry the code update. Refer to Appendix B for SA Error Codes.

Error 112 — Call to $SETIMR Failed

```
Error 112 - Call to $SETIMR failed

***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY - 1.0  ***
     Pass "pass", test 1, subtest 0, error 112, "date" "time"
     Hard error while testing "device": UNABLE TO START TIMER

     Media load error.

     $SETIMR returned error status of : 00000011

***  End of Hard error number 112  ***
```

The VAX/DS system service call returned an error status.

Recovery: The media is suspect, retry the update procedure. If the same failure occurs, use a different update tape or load device.

Error 113 — Error While Initializing Controller

```
Error 113 - Error while initializing controller

***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY - 1.0  ***
     Pass "pass", test 1, subtest 0, error 113, "date" "time"
     Device fatal error while testing "device":

     CONTROLLER INITIALIZATION FAILURE

     Diagnostic will be aborted because of previous error

***  End of Device fatal error number 113  ***
```

The controller failed to initialize because of error messages 110 through 112. This is an informational error message indicating that the controller could not be initialized. The code update is aborted.

Error 120 — Error While Opening Controller Image File

```
Error 120 - Error while opening controller image file

***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY 1.0  ***
     Pass "pass", test 1, subtest 0, error 120, "date" "time"
     System fatal error while testing "device": $OPEN FAILED

     Media load error.

     Error processing controller image file: "filename".

     The system service call $OPEN returned the following error status:

     type_2_message

     Try new load media.

***  End of System fatal error number 120  ***
```

The OPEN EXISTING FILE command makes files available for processing. If a file cannot be opened, one of the following messages is displayed and the update aborted:

- File access error
- Dynamic memory error
- Bad device error
- Fab_error
- File not found error
- Bad file name error
- Invalid file organization error
- File read error

Error 120 — Error While Opening Controller Image File

Recovery: The media is suspect, retry the update procedure. If the same failure occurs, use a different update tape or load device.

Error 121 — Error Connecting to the Controller Image File

```
Error 121 - Error while connecting to the controller image file

***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY 1.0  ***
     Pass "pass", test 1, subtest 0, error 121, "date" "time"
     System fatal error while testing "device": $CONNECT FAILED

     Media load error.

     Error processing controller image file: "filename".

     The system service call $CONNECT returned error.

     Try new load media.

***  End of System fatal error number 121  ***
```

The code update is aborted if no connection can be made to the controller image file.

Recovery: The media is suspect, retry the update procedure. If the same failure occurs, use a different update tape or load device.

Error 122 — Error While Reading Image File

```
Error 122 - Error while reading in image file

 ***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY 1.0 ***
     Pass 1, test 1, subtest 0, error 122,"date" "time"
     System fatal error while testing DUA: $GET FAILED

     Media load error.

     Error processing controller image file: "filename".

     Unable to read a record from "filename" into host memory.

     Try new load media.

***  End of system fatal error number 122  ***
```

An error occurred while trying to read the controller image file from load media to host memory.

Recovery: The media is suspect, retry the update procedure. If the same failure occurs, use a different update tape or load device.

Error 123 — Disconnect Error

```
Error 123 - Disconnect error

 ***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY 1.0 ***
     Pass 1, test 1, subtest 0, error 123,"date" "time"
     System fatal error while testing DUA: $DISCONNECT FAILED

     Media load error.

     Error processing controller image file "filename".

     $DISCONNECT returned error status of "status".

     Try new load media.

***  End of system fatal error number 123  ***
```

The program is aborted if I/O buffers and structures cannot be deallocated.

Recovery: The media is suspect, retry the update procedure. If the same failure occurs, use a different update tape or load device.

Error 124 — Close Error

```
Error 124 - Close error

 ***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY 1.0 ***
     Pass 1, test 1, subtest 0, error 124,"date" "time"
     System fatal error while testing DUA: $CLOSE FAILED

     Media Error

     Error processing controller image file "filename".

     $CLOSE returned error status of "status".

     Try new load media.

***  End of system fatal error number 124  ***
```

If an open file cannot be closed after all processing of files has been completed, the code update is aborted.

Recovery: This is a media load error. Retry the code update. If the update still fails, try a new load media and retry the code update.

Error 130 — Error Reading Update Flag

```
Error 130 - Error reading update flag

 ***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY 1.0 ***
     Pass 1, test 1, subtest 0, error 130,"date" "time"
     Device fatal error while testing "device":

     Controller error.

     Error reading update flags.

     READ DATA command failed with a status of: "status"

***  End of Device fatal error number 130  ***
```

The KDM70 controller must set its controller update flag before code can be updated. This error indicates the controller could not read the update flag. The update is aborted.

Recovery: This is a KDM70 controller error. Retry the code update. If the update still fails, refer to error recovery procedure.

Error 132 — Error Writing Update Flag

```
Error 132 - Error writing update flag

 ***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY 1.0 ***
     Pass 1, test 1, subtest 0, error 132,"date" "time"
     Device fatal error while testing "device":

     Controller error.

     Error writing update flag.

     WRITE DATA command failed with a status of: "status"

***  End of Device fatal error number 132  ***
```

This error indicates that update flag could not be written.

Recovery: This is a controller error. Retry the code update. If the update still fails, refer to error recovery procedure. retry the code update.

Error 133 — Controller Update Flag Incorrectly Written

```
Error 133 - Controller update flag incorrectly written

 ***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY 1.0 ***
     Pass 1, test 1, subtest 0, error 133,"date" "time"
     Device fatal error while testing "device":

     Controller error.

     Update flag incorrectly written.

***  End of Device fatal error number 133  ***
```

Once the update flag has been written, it is read to verify that it contains the specified value. If after five retries the flag is still incorrect, the update is aborted.

Recovery: This is a KDM70 controller error. Retry the code update. If the update still fails, refer to error recovery procedure.

Error 134 — Unable to Set EEPROM Address

```
Error 134 - Unable to set EEPROM address

 ***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY 1.0 ***
     Pass 1, test 1, subtest 0, error 134,"date" "time"
     Device fatal error while testing "device":

     Controller error.

     Error setting EEPROM address

     SET NEW ADDRESS command failed with a status of: "status"

***  End of Device fatal error number 134  ***
```

If the EEPROM address cannot be selected, then the code update is aborted.

Recovery: This is a KDM70 controller error. Retry the code update. If the update still fails, refer to the error recovery procedure. Retry the code update.

Error 135 — Unable to Write EEPROM

Note: This error may corrupt the current EEPROM code image. The KDM70 code was written to recover from these failures. Use the error recovery procedure. If the update still fails, replace the T2022 module.

```
Error 135 - Unable to write EEPROM.

 ***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY 1.0 ***
     Pass 1, test 1, subtest 0, error 135,"date" "time"
     Device fatal error while testing "device":

     Controller error.

     Error while writing to EEPROM

     WRITE DATA command failed with a status of: "status"

***  End of Device fatal error number 135  ***
```

An error occurred while writing the controller image to EEPROM.

Recovery: This is a KDM70 controller error. Retry the code update. If the update still fails, replace the T2022 module and retry the code update.

Error 136 — Unable to Read EEPROM

Note: This error may corrupt the current EEPROM code image. The KDM70 code was written to recover from these failures. Use the error recovery procedure. If the update still fails, replace the T2022 module.

```
Error 136 - Unable to read EEPROM.

 ***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY 1.0 ***
     Pass 1, test 1, subtest 0, error 136,"date" "time"
     Device fatal error while testing "device":

     Controller error.

     Error while reading EEPROM

     READ DATA command failed with a status of: "status"

***  End of Device fatal error number 136  ***
```

EEPROM written with new code is read and compared to the controller image file stored in host memory. This error indicates a read failure of EEPROM.

Recovery: This is a KDM70 controller error. Retry the code update. If the update still fails, replace the T2022 module and retry the code update.

Error 137 — One Byte Verified Written Incorrectly

```
Error 137 - One byte verified written incorrectly
 ***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY 1.0 ***
     Pass 1, test 1, subtest 0, error 137,"date" "time"
     Soft error while testing "device":

     Controller error.

     One byte verified written incorrectly.

***  End of soft error number 137  ***
```

A byte-by-byte comparison is performed after KDM70 EEPROM is written. This error indicates one byte of EEPROM was found to be in error. The code update process will continue

Recovery: This is an informational message only. No operator action is required.

Error 138 — Excessive Bytes Written Incorrectly

Note: This error may corrupt the current EEPROM code image. The KDM70 code was written to recover from these failures. Use the error recovery procedure. If the update still fails, replace the T2022 module.

```
Error 138 - Excessive bytes written incorrectly
 ***  EVRLM - Level 3 KDM70 EEPROM UPDATE UTILITY 1.0 ***
     Pass 1, test 1, subtest 0, error 138, "date" "time"
     Device fatal error while testing "device":

     Controller error.

     More than eight bytes verified written incorrectly within a 1K boundary.

     ABORTING...

***  End of Device fatal error number 138  ***
```

A byte-by-byte comparison is performed after KDM70 EEPROM is written. This error indicates more than 8 bytes of EEPROM are in error. The code update process aborts.

Recovery: This is a KDM70 controller error. Retry the code update. If the update still fails, replace the T2022 module and retry the code update.

12.8 Error Recovery Procedures

Normally the code update utility requires support from the current code image to assist in the update process. The forced update procedure will use code written in non-volatile ROM for this support.

1 Power down the system.

2 Place a jumper from pin E13 to E15 (T2022 slot). (See Figure 12–1.)

3 Power up the system and verify the KDM70 LED code is 1011. This indicates that the KDM70 controller is in forced update mode.

4 Boot the diagnostic supervisor in standalone mode.

5 Attach and select the KDM70 controller to be updated, as shown in the following example:

```
DS> ATTACH KDM70 HUB DUx  4 5

DS> SELECT DUx

DS> RUN EVRLM
```

The variables are explained in the following table:

Variable	Meaning
x	An alphabetic designation. Generally, the boot device controller is assigned an A.
4	XMI node number — T2022 slot number
5	Bus request level

The following EVRLM error message is displayed:

```
..Program: EVRLM - LEVEL 3 code update UTILITY FOR KDM70 CONTROLLERS, revision
1.0, 1 test, at 15:08:22.73.
Testing _KDMA

********  EVRLM - LEVEL 3 code update UTILITY FOR KDM70 CONTROLLERS - 1.0 *****
    Pass 0, Initialization section, error 2, 20-SEP-1989 15:20:48:56
    System fatal error while testing KDMA: NODE RESET FAILED

Node self-test failed.

************* End of System fatal error number 2  **********

CONTROLLER FAILED SELF-TEST, DO YOU WISH TO CONTINUE? [(No), Yes]
```

6 To continue from this point, answer YES. Otherwise, the code update procedure aborts. The code update program menu is displayed.

7 Select the perform code update option.

A message is printed indicating that the disaster mode is being invoked.

The code update procedure completes with the following message:

```
EEPROMS updated
code update SUCCESSFULLY COMPLETED
```

8 Press Return to return to the code update program menu.

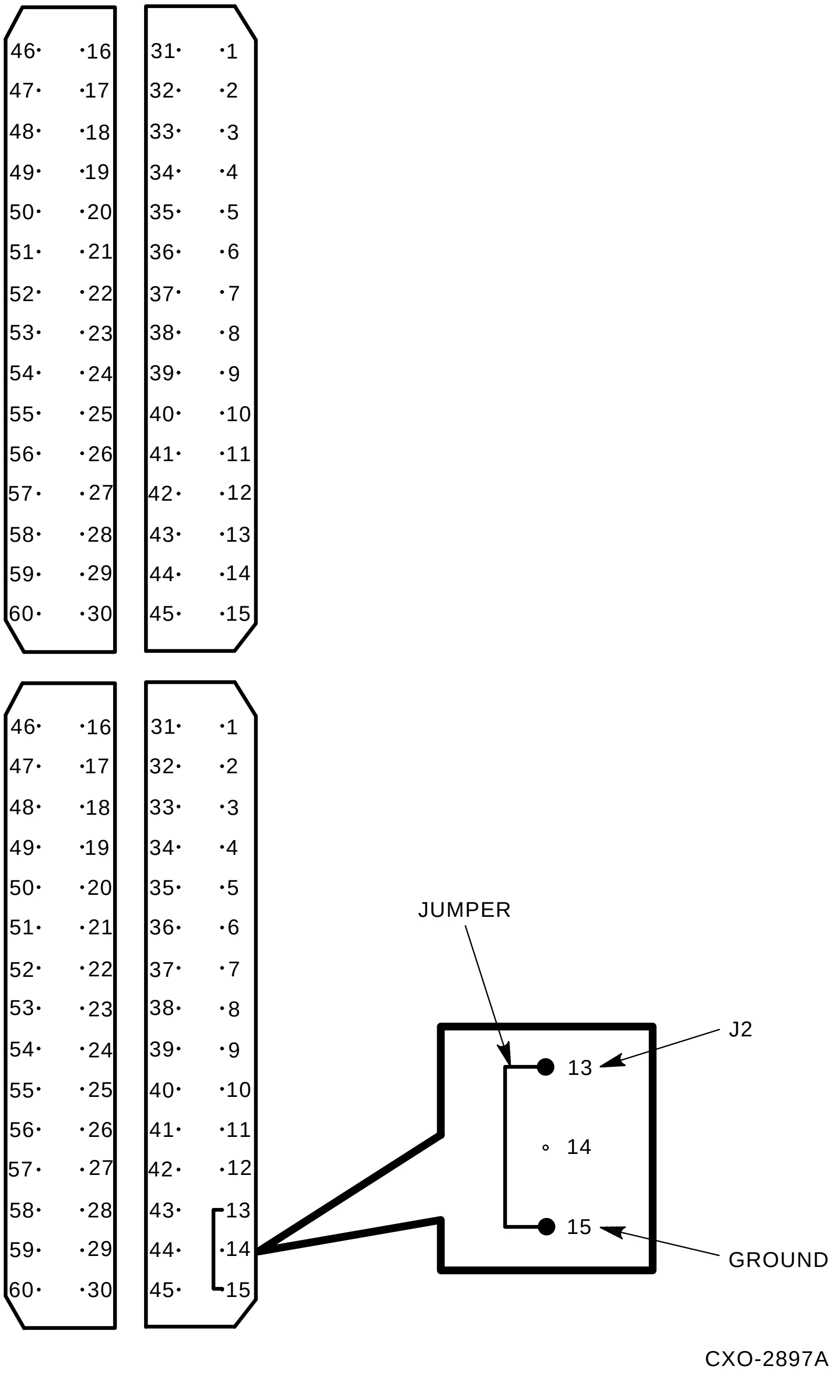
9 Remove the backplane jumper. (If the jumper is left in place, a self-test failure error message is displayed after exiting from the update.)

10 Type EXIT to return the DS> prompt.

11 Exit from the VAX diagnostic supervisor.

If no errors occurred during the forced-code update, reboot the system. Ensure that the KDM70 controller is functional by running on-line exerciser programs, such as ILEXER or EVRAE.

Figure 12–1 Backplane Pin Location



13 KDM70 PATCH Utility

The PATCH utility provides a method of fixing KDM70 software problems between major code updates. It allows modification of KDM70 program memory which is stored in EEPROM. Although PATCH release instructions are included with each patch release, this chapter provides an overview of PATCH commands and modifiers.

Note: The PATCH utility will write the current code image in EEPROM. Verify that a backup code image is available before applying a PATCH.

13.1 Invoking PATCH

The PATCH utility is a supplied program. It requires a DUP connection in order to execute. A DUP connection may be supplied under VMS by the FYDRIVER or standalone under the VAX diagnostic supervisor by EVRLN.

13.1.1 Invoking PATCH On Line from VMS

Use the following commands to invoke PATCH on line from VMS:

```
$ RUN SYS$SYSTEM:SYSGEN

SYSGEN> CONNECT FYA0/NOADAPTER   ; Load VMS DUP Driver

SYSGEN> EXIT

$ SET DEFAULT SYS$MAINTENANCE

$ SET HOST/DUP/SERVER=DUP/LOAD=PATCH.KDM PUA0/DEV

***
*** PATCH (KDM70 Patch Utility) V 001  *** 27-NOV-1856 11:43:20 ***
***

PATCH>
```

13.1.2 Invoking PATCH Standalone from the VAX Diagnostic Supervisor

Use the following commands to invoke PATCH using VAX DS.

```
DS> ATTACH KDM70 HUB DUx 3 5

DS> SELECT DUx

DS> RUN EVRLN

EVRLN> RUNS PATCH

***
*** PATCH (KDM70 Patch Utility) V 001  *** 27-NOV-1856 11:43:20 ***
***

PATCH>
```

13.2 Software Revision Numbering Scheme

Refer to page 12-3 in the EVRLM Chapter.

13.3 Running PATCH

Use the following commands to run PATCH:

```
PATCH> DISPLAY LIST

*** CURRENT PATCHES: NONE

PATCH> PATCH 1

PATCH> RELOCATE 80000100

PATCH> MODIFY/LONG    1C0 C0AB3FE0

PATCH> MODIFY/LONG    1E0 D0A12FC0

PATCH> DISPLAY  Patch

*** PATCH 001 Description ***

 Address  Current  Changed
 -------- -------- --------
 800002C0 04049FD0 C0AB3FE0
 800002E0 C1522014 D0A12FC0

PATCH> UPDATE E02AC1A0   ;Checksum supplied with PATCH release

PATCH> EXIT
```

13.3.1 PATCH Commands

Table 13–1 shows a summary of all PATCH commands.

Table 13–1 PATCH Commands

Command	Parameter	Qualifier	Description
DELETE	Address		Delete first occurrence of address in PATCH buffer. Use to correct errors in PATCH buffer.
DISPLAY	PATCH		Display current changes in PATCH buffer
	LIST		Display PATCH numbers currently installed

Table 13–1 (Cont.) PATCH Commands

Command	Parameter	Qualifier	Description
	VERSION		Display current KDM70 software version
	RELOCATE		Display current relocation address
EXIT			Exit from PATCH
RELOCATE	Relocation Address		Sets base address for the image being patched
MODIFY	Address	/BYTE	Modify a byte location in the PATCH buffer
MODIFY	Address	/WORD	Modify a word location in the PATCH buffer
MODIFY	Address	/LONG	Modify a longword location in the PATCH buffer
PATCH	Patch number		Identifies the PATCH number being applied
UPDATE	Checksum		Checksum supplied with the PATCH release. This command will verify the correct checksum and copy the changes to KDM70 EEPROM

13.4 New PATCH Commands

Table 13–2 Shows a summary of new PATCH commands that are supported withg Version 3.0 of the KDM70 software.

Table 13–2 New PATCH Commands

Command	Parameter	Qualifier	Description
DISABLE	CTRL_P		Requires XMI_UPDATE to be enabled. Disables CTRL_P from console to break into the MONITOR.
DISABLE	SIL		Requires XMI_UPDATE to be enabled. Disables last crash packet for spontaneous INIT.
ENABLE	CTRL_P		Requires XMI_UPDATE to be enabled. Enables CTRL_P from console to break into the MONITOR.
ENABLE	SIL		Requires XMI_UPDATE to be enabled. Enables last crash packet for spontaneous INIT.

Table 13–2 (Cont.) New PATCH Commands

Command	Parameter	Qualifier	Description
RESET			KDM70's write a fatal error code of x(8)380 in its SA register. The Operating system will then reset the controller upon detection of a fatal SA code. This feature is also used to force the KDM70 to reload its code from EEPROM to SRAM. This allows any patches or options to take effect immediately.
DISPLAY	BOARD	/B1	Displays all relevant information for T2022.
DISPLAY	BOARD	/B2	Displays all relevant information for T2023.
DISPLAY		/FULL	Causes entries in "lists to be dumped even if they are "null".
DISPLAY	BPL		Bad Page List.
DISPLAY	BPL2		Bad Page List Copy 2.
DISPLAY	BPL3		Bad Page List Copy 3.
DISPLAY	COMMENT		Program Comments, applies only to the loaded image.
DISPLAY	ESPL		EEPROM Suspect Page List.
DISPLAY	HWRN		Hardware Revision Number.
DISPLAY	IMAGE		Information about the loaded image.
DISPLAY	MIEL		MIST Internal Error Logs.
DISPLAY	OPTIONS		Program Special Options.
DISPLAY	RTEL		Real Time Error Log.
DISPLAY	SERIAL		Serial Number of the board.
DISPLAY	SSPL		SRAM Suspect Page List.
DISPLAY	SWRN		Software Revision Number.
DISPLAY	UPTIME		Up Time.

13.4.1 PATCH Error Messages

The following error messages may be reported under the PATCH utility. Error messages are generally procedural in nature; however, some errors may result in fatal conditions. Suggested error recovery procedures are included in this section.

Table 13–3 PATCH Error Messages

Error Message	Action
*** "Item" is an invalid command	Re-enter a valid command.
*** Input value is out of range	Re-enter a valid-range value.
*** Invalid PATCH address; range is 80201200 - 802597FE	Re-enter an address within the valid range.
*** PATCH aborted by user!	Results from a CTRL/C, CTRL/Y, or a CTRL/ \. Restart the program.
*** No patch has been opened	Issue the PATCH command before modifying a location.
*** "Item" is an invalid option for DISPLAY	Re-enter a valid DISPLAY option.
*** "Item" is an invalid modifier	Re-enter an appropriate modifier for the MODIFY command.
*** Checksum does not match	Re-enter modified values; ensure correct values have been input by using the DISPLAY command.
*** Missing parameter	Supply missing parameter; use DISPLAY to identify missing parameters.

A KDM70 Controller Bug Check Codes

A.1 Introduction

This appendix contains KDM70-specific bug check codes. There are software bug check codes and hardware bug check codes. Software bug check codes always have a zero as the most significant bit (MSB).

A.2 KDM70 Bug Check Codes

KDM70 bug checks are the result of an error detected during functional code. There are two types of bug checks: software and hardware detected. A description and recommended user action is provided for each bug check code. The most likely solution is listed first.

A.2.1 Software Bug Checks

Software bug checks are inconsistencies detected by the KDM70 controller software during normal controller operation. Software bug checks may be caused by either hardware or software.

Software inconsistencies have two possible causes:

1. An internal data structure was corrupted due to an undetected hardware error. This type of failure would typically cause many different types of bug checks. Look for other error symptoms to assist in fault isolation.
2. An unexpected condition occurred in the KDM70 controller software. A true software design error is unlikely. If the same bug check code is seen repeatedly, and this is the only failure, it should be reported to KDM70 engineering by submitting a PRISM report.

00006001: DCM_INCONSISTENT_STATUS

Facility: Disk communications management

Explanation: Bad communication status. When attempting to translate an SI state machine code into an MSCP sub-event code, an unknown SI status code was detected.

MSCP Event: Software logic error

User Action: Check the T2023, drive, and KDM70 controller software.

0000A002: DER_ERROR_ANAL_FAILURE

Facility: SDI Disk Data Error Recovery

Explanation: Error Recovery could not determine the type of error to retry and report.

MSCP Event: Software logic error

User Action: Check the T2023 and KDM70 controller software.

0000A003: RCT IS CORRUPT

Facility: SDI Disk Error Recovery

Explanation: Test Descriptor resulted in a return code of "RCT is corrupt"

MSCP Event: Software Logic Error

User Action: This is not a KDM70 controller error. It indicates that an attempt was made to mount a disk with a corrupted RCT.

0000E003: DPX_WRONG_DATA_TYPE

Facility: Disk physical transfer management

Explanation: The wrong data type was received as input. The “Type” field in the structure indicates the structure is not a transfer block (TB).

MSCP Event: Software logic error

User Action: If other types of bug checks are also reported, the problem could be hardware, T2022 or T2023. If no other bug checks are reported, the KDM70 software is most likely at fault.

0000E004: DPX_INTERPRET_HANDLE_FAILURE

Facility: Disk physical transfer management

Explanation: The interpret handle routine failed. An invalid PTE was found during the buffer handle resolution for a specific portion of a transfer.

MSCP Event: Software logic error

User Action: T2022, KDM70 controller software, Host SCS/Port software. Examine error logs for indications of host memory or XMI errors. Corruption in the communications area is also a possibility.

0000E005: DPX_INCONSISTENT_PCB

Facility: Disk physical transfer management

Explanation: An inconsistent port control block exists. Error recovery was called, but no error recovery status was supplied.

MSCP Event: Software logic error

User Action: Check the SI state machine (T2023) and KDM70 controller software.

0000E007: DPX_INVALID_RESUME_CASE

Facility: Disk physical transfer management

Explanation: The resume case specified in TB is invalid. Transfer management called routines to allocate the TB, TF, CF and buffer for a disk data transfer. Upon return from the allocation routines, the resume case provided in the TB was not as expected.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

0000E009: DPX_INCONSISTENT_BYTE_COUNT

Facility: Disk physical transfer management

Explanation: An inconsistent byte count was found during a compare operation. Compare operations do a block-for-block comparison, which requires the byte count to be 512. A value other than 512 causes a bug check.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

0000E00A: DPX_UNDEFINED_TB_STATE

Facility: Disk physical transfer management

Explanation: The TB state code value is undefined. An abort command was issued, but the TB state could not be determined. Abort processing attempts to locate the TB (if allocated) on one of several queues after examining the ATD or TMD state. If the ATD or TMD state implies that the TB may be found on one of these queues, a case instruction is used to locate the TB. If the TB cannot be located, invoke a bug check.

MSCP Event: Software logic error

User Action: Unless other bug checks or errors are also being reported, this is most likely a KDM70 controller software problem.

0000E00B: DPX_RBN_NOT_VALID

Facility: Disk physical transfer management

Explanation: The facility tried to transfer an RBN on ESE. There are no replacement blocks on ESEs. A replacement block number was given to the transfer manager for an ESE.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

0000E00C: DPX_INVALID_BYTE_COUNT

Facility: Disk physical transfer management

Explanation: The byte count for multicopy transfer was not equal to 512. Multicopy transfers are internal controller writes of the four copies of the RCT.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

00010001:DSM_NO_UCBS

Facility: SDI disk/tape state machine management

Explanation: No UCBs exist.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

0001E001: TCM_INCONSISTENT_STATUS

Facility: Tape communications management

Explanation: The facility found bad communication status and could not translate an SI state machine code into a meaningful MSCP event sub-code.

MSCP Event: Software logic error

User Action: Check the SI state machine (T2023), drive, and KDM70 controller software.

00024001: TER_INVALID_ERROR_CODE

Facility: STI tape data error recovery

Explanation: The facility found an invalid error code. STI data error recovery was called, but the error code was invalid.

MSCP Event: Software logic error

User Action: Check the tape formatter, SI state machine (T2023), and KDM70 controller software.

00024002: TER_ERR_RECOV_INCONSISTENCY

Facility: STI tape data error recovery

Explanation: The direction of the tape error recovery retry is not the original direction. The retry must be attempted in the same direction as the error.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

00028001: TPX_MEM_ALLOC_FAILED

Facility: Tape physical transfer management

Explanation: Memory allocation failed. Tape pipeline management could not allocate memory buffers for a read reverse operation.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

00028002: TPX_INVALID_OPERATION

Facility: Tape physical transfer management

Explanation: The opcode in TMD is illegal. This indicates an invalid tape operation.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

00028003: TPX_INVALID_RESUME_CASE

Facility: Tape physical transfer management

Explanation: The resume case specified in TB is invalid. Transfer management called routines to allocate the TB, TF, CF, and buffer for a disk data transfer. Upon return from the allocation routines, the resume case in the TB was not as expected.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

00028004: TPX_INTERPRET_HANDLE_FAILURE

Facility: Tape physical transfer management

Explanation: The interpret handle routine failed. An invalid PTE was found during the buffer handle resolution for a specific portion of a transfer.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

00028005: TPX_INVALID_ERROR_CODE

Facility: Tape physical transfer management

Explanation: The error code in the buffer list is invalid. The status of the tape error recovery is not success, but the status of the command does not indicate that the operation ever started. The SI state machine reported an error before the operation started.

MSCP Event: Software logic error

User Action: Check the SI state machine (T2023) and KDM70 controller software.

0002A001: TSM_NO_UCBS

Facility: Tape state machine management

Explanation: No UCBs exist. The tape state management tried a Get Unit Characteristics operation on a newly discovered unit, but no UCBs were available.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

0002E081: ILX_MAIN_NOWORK

Facility: ILEXER

Explanation: No work returned from the HLDI. There should always be work outstanding for ILEXER once it is called.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

0002E082: ILX_MAIN_BOGUS

Facility: ILEXER

Explanation: An invalid end message was returned.

MSCP Event: Software logic error

User Action: Check the SI state machine (T2023) and KDM70 controller software.

0002E083: ILX_MAIN_TSTNUM

Facility: ILEXER

Explanation: The facility found an illegal device test number. The test is not applicable for selected device.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

0002E084: ILX_MAIN_STOPPING

Facility: ILEXER

Explanation: A command was received in a stopping state. The number of outstanding commands did not reach zero.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

0002E085: ILX_MAIN_DEAD

Facility: ILEXER

Explanation: A command was received in a dead state. ILEXER received a command after it had been stopped.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

0002E086: ILX_MAIN_BADSTATE

Facility: ILEXER

Explanation: The device is in wrong state to run the exerciser. HLDI should have caught this before starting ILEXER.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

0002E087: ILX_MAIN_NULLCB

Facility: ILEXER

Explanation: No device CB was supplied with the command.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

0002E088: ILX_MAIN_NOCMD

Facility: ILEXER

Explanation: No CMDs exist on the free queue.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

0002E101: ILX_PARM_NOSTORE

Facility: ILEXER

Explanation: No storage is available.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

0002E102: ILX_PARM_WHOA

Facility: ILEXER

Explanation: The wrong command/unit was returned.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

0002E103: ILX_PARM_NOCOMP

Facility: ILEXER

Explanation: The command never completed.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

00032001: DLS_ALLOC_HANDLE

Facility: Diagnostic subroutine library

Explanation: The allocate buffer handle failed; an invalid buffer handle was found.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

00032002: DLS_DEALLOC_HANDLE

Facility: Diagnostic subroutine library

Explanation: The deallocate buffer handle failed; an invalid buffer handle exists.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

00182001: KERNEL STACK INVALID

Facility: EXEC

Explanation: The kernel stack is not valid.

MSCP Event: Software logic error

User Action: Check the T2022 and KDM70 controller software.

00182002: RESERVED INSTR

Facility: EXEC

Explanation: Reserved instruction.

MSCP Event: Software logic error

User Action: Check the T2022 and KDM70 controller software.

00182003: EXTENDED INSTR

Facility: EXEC

Explanation: Extended instruction.

MSCP Event: Software logic error

User Action: Check the T2022 and KDM70 controller software.

00182004: RESERVED OPERAND

Facility: EXEC

Explanation: Reserved operand.

MSCP Event: Software logic error

User Action: Check the T2022 and KDM70 controller software.

00182005: RESERVED ADDRESSING

Facility: EXEC

Explanation: Reserved addressing mode.

MSCP Event: Software logic error

User Action: Check the T2022 and KDM70 controller software.

00182006: ACCESS VIOLATION

Facility: EXEC

Explanation: Access control violation.

MSCP Event: Software logic error

User Action: Check the T2022 and KDM70 controller software.

00182007: TRANSLATION NOT VALID

Facility: EXEC

Explanation: The translation is not valid.

MSCP Event: Software logic error

User Action: Check the T2022 and KDM70 controller software.

00182008: TRACE PENDING

Facility: EXEC

Explanation: Trace pending.

MSCP Event: Software logic error

User Action: Check the T2022 and KDM70 controller software.

00182009: BREAKPOINT INSTRUCTION

Facility: EXEC

Explanation: Breakpoint instruction.

MSCP Event: Software logic error

User Action: Check the T2022 and KDM70 controller software.

0018200A: ARITHMETIC OVERFLOW

Facility: EXEC

Explanation: Arithmetic overflow.

MSCP Event: Software logic error

User Action: Check the T2022 and KDM70 controller software.

0018200D: UNUSED VECTOR

Facility: EXEC

Explanation: Unused vector.

MSCP Event: Software logic error

User Action: Check the T2022 and KDM70 controller software.

00188081: DCP_BBR_ATD

Facility: Disk MSCP server

Explanation: The structure returned by DPX is not the ATD that was sent.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

00188082: DCP_BBR_SCR

Facility: Disk MSCP server

Explanation: The structure returned by DPX is not the SCR that was sent.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

00188101: DCP_DSP_GATE

Facility: Disk MSCP Server

Explanation: The gatekeeper found an invalid port state.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

00188102: DCP_DSP_NUCB

Facility: Disk MSCP server

Explanation: No UCB was found after an update, although the port previously had a UCB.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

00188103: DCP_DSP_ISC

Facility: Disk MSCP server

Explanation: An invalid port state transition exists.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

00188181: DCP_MSC_AK

Facility: Disk MSCP server

Explanation: The Make-Unit-Known routine was called when the unit was already known.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

00188182: DCP_MSC_SLQ

Facility: Disk MSCP server

Explanation: The Proc-Slow-Q queue was called for a null item.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

00188183: DCP_MSC_CTMO

Facility: Disk MSCP server

Explanation: A host connection timeout was found. This could indicate a host hardware or software problem.

MSCP Event: Software logic error

User Action: Examine the error logs for other XMI or XMI-based controller problems. Look for other error symptoms. Check the T2022 or KDM70 software.

For KDM70 Software Versions 3 and greater, this bug check no longer occurs. It has been changed to an informational datagram, which will have the following attributes.

COMMAND REFERENCE NUMBER = Zero.

SEQUENCE NUMBER = Zero.

FORMAT = Zero; an MSCP Controller error format error log message.

FLAGS = 83 (hex); Sequence number reset, non-error/informational event, operation successful

EVENT CODE = A (hex); controller timeout.

CONTROLLER IDENTIFIER =

- The unique identifier is the controller serial number.
- The controller "class" is 02 (disk class device 166)
- The controller "model" is 1B (KDM70)

CSVRSN = Controller software version number.

CHVRSN = Controller hardware revision number.

PORT ERROR CODE = The port error code of 14 (hex) which is defined as "High level protocol incompatibility error".

CONTROLLER DEPENDENT INFORMATION =

The KDM70 bugcheck code of 00188183, which indicates that the host has stopped communicating with the KDM70. After a host specified timeout interval, the KDM70 declares the connection to the host is lost via this bugcheck code.

00188184: DCP_MSC_SLII

Facility: Disk MSCP Server

Explanation: An invalid item was received on the miscellaneous queue.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

00188185: DCP_MSC_NK

Facility: Disk MSCP server

Explanation: The unit was not known.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

0018C181: DIM_FWB

Facility: Disk/tape state machine management

Explanation: The facility failed to find an XMI work block in error.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

0018C182: DIM_OWB

Facility: Disk/tape state machine management

Explanation: Orphaned SI work block. A work block was started but did not finish. Neither SI state machine was operating on the same port as the work block. Invoke a bug check.

MSCP Event: Software logic error

User Action: Check the SISM (T2023) and KDM70 controller software.

0018C183: DIM_BWO

Facility: Disk/tape state machine management

Explanation: Bad work block opcode.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

0018C184: DIM_INC

Facility: Disk/tape state machine management

Explanation: The XMI state machine did not complete its work and is considered trapped in the work block.

MSCP Event: Software logic error

User Action: Check the XMI State Machine (T2022) and KDM70 controller software.

00192001: DUP_PROG_TMO

Facility: Diagnostic utility

Explanation: Diagnostic utility time out.

MSCP Event: Software logic error

User Action: Look for other error symptoms, T2022, KDM70 software

00192002: DUP_HOST_TMO

Facility: Diagnostic utility

Explanation: The host failed to respond or poll within the timeout period.

MSCP Event: Software logic error

User Action: Look for other error symptoms. T2022, KDM70 Software

0019C000: SCA_SYS_SIZ

Facility: SCA

Explanation: An attempt to allocate SYS handle failed because its size is not 512.

MSCP Event: Software logic error

User Action: Look for other error symptoms. T2022, KDM70 Software

0019C001: SCA_MIS_BUF

Facility: SCA

Explanation: An attempt to allocate handle failed because insufficient buffers were supplied.

MSCP Event: Software logic error

User Action: Look for other error symptoms. T2022, KDM70 Software

0019C002: SCA_INV_HND

Facility: SCA

Explanation: An attempt to deallocate an invalid handle failed.

MSCP Event: Software logic error

User Action: Look for other error symptoms. T2022, KDM70 Software

001A6081: TIM_HQE

Facility: Tape state machine management

Explanation: The holding queue on the port control block is not consistent. A TB should be on the PCB holding queue.

MSCP Event: Software logic error

User Action: Look for other error symptoms. If no other errors exist, it is most likely a KDM70 controller software problem. Other symptoms may indicate hardware.

A.2.2 Hardware Bug Checks

Hardware bug checks are the result of errors detected by hardware during execution of functional code. Refer to "User Action" for the specific bug check code.

Note: Hardware bug checks do not necessarily mean that hardware must be replaced. Check the "User Action" under the specific bug check code encountered.

C048202B: EXC_MEM_ERR

Facility: EXEC

Explanation: A memory error was detected by memory error logic.

MSCP Event: Controller memory error

User Action: The KDM70 controller will report memory parity errors via this bug check code. The page containing the parity error will be logged to the Bad Page List and will no longer be used by functional code. Once the threshold of bad pages has been reached, a fault management event (1EA) errorlog will be generated. This indicates that a call should be scheduled to replace either board 1 (T2022) or board 2 (T2023). The failing module is indicated in the fault management event error log.

C0604029: XIM_CVAX_FAIL

Facility: XMI state machine management

Explanation: An error occurred while the CVAX channel was doing an XMI transfer. The CVAX will retry the transfer. If it still fails, perform a bug check.

MSCP Event: Host interface hardware error

User Action: Look for other error symptoms. T2022, KDM70 Software

C068202C: EXC_BUS_ERR

Facility: EXEC

Explanation: A memory parity error was detected by HIB bus logic.

MSCP Event: Internal bus error

User Action: This error indicates a problem in the T2022/T2023 internal bus. This bugcheck has been resolved with hardware REV E02 of the T2022 module.

C070202D: EXC_MC_ERR

Facility: EXEC

Explanation: Machine check.

MSCP Event: Policy processor error

User Action: Check the T2022.

C458C12A: DIM_SST

Facility: Disk/tape state machine management

Explanation: The SISM status has an unrecoverable error code.

MSCP Event: Device interface hardware error

User Action: Check the T2023 and KDM70 controller software.

C458C3AA: DIM_ISE

Facility: Disk/tape state machine management

Explanation: The send work block status has an invalid error code.

MSCP Event: Device interface hardware error

User Action: Check the T2023 and KDM70 controller software.

C4604029: XIM_CXH_FAIL

Facility: Disk/tape state machine management

Explanation: The CXHIC transaction timed out.

MSCP Event: Host interface hardware error

User Action: Check the T2022 and KDM70 controller software.

C460C1A9: DIM_XST

Facility: Disk/tape state machine management

Explanation: The XISM status has an unrecoverable error.

MSCP Event: Host interface hardware error

User Action: Check the T2022 and KDM70 controller software.

C858C12A: DIM_CPE

Facility: Disk/tape state machine management

Explanation: SISM control ROM parity error.

MSCP Event: Device interface hardware error

User Action: Check the T2023.

C858C3AA: DIM_IRE

Facility: Disk/tape state machine management

Explanation: An invalid error code was found in the receive work block. status

MSCP Event: Device interface hardware error

User Action: Check the T2023 and KDM70 controller software.

C8604029: XIM_CMES_NA

Facility: XMI state machine management

Explanation: CMES was not asserted on a CXHIC error.

MSCP Event: Host interface hardware error

User Action: Check the T2022 and KDM70 controller software.

C860C1A9: DIM_XES

Facility: Disk/tape state machine management

Explanation: The XISM is idle, but the DPC error summary is zero.

MSCP Event: Host interface hardware error

User Action: Check the T2022 and KDM70 controller software.

CC58C12A: DIM_SMS

Facility: Disk/tape state machine management

Explanation: SISM failed to start on a work block.

MSCP Event: Device interface hardware error

User Action: Check the T2023 and KDM70 controller software.

CC58C3AA: DIM_IDE

Facility: Disk/tape state machine management

Explanation: An invalid error code was found in the disk buffer status result.

MSCP Event: Device interface hardware error

User Action: Check the T2023 and KDM70 controller software.

CC604029: XIM_CXH_NBE

Facility: XMI state machine management

Explanation: A CXHIC transaction failed with no bus error.

MSCP Event: Host interface hardware error

User Action: Check the T2022 and KDM70 controller software.

D058C12A: DIM_SMH

Facility: Disk/tape state machine management

Explanation: SISM failed to HALT.

MSCP Event: Device interface hardware error

User Action: Check the T2023 and KDM70 controller software.

D058C3AA: DIM_ITE

Facility: Disk/tape state machine management

Explanation: An invalid error code was found in the tape buffer status result.

MSCP Event: Device interface hardware error

User Action: Check the T2023 and KDM70 controller software.

D0604029: XIM_CXH_DE

Facility: XMI state machine management

Explanation: CXHIC failed with both DTNOK and CTNOK asserted.

MSCP Event: Host interface hardware error

User Action: Check the T2022 and KDM70 controller software.

D4604029: XIM_CXH_OK

Facility: XMI state machine management

Explanation: A CXHIC transaction timed out.

MSCP Event: Host interface hardware error

User Action: Check the T2022 and KDM70 controller software.

D8604029: XIM_DPC_NI

Facility: XMI state machine management

Explanation: DPC failed to enter the idle mode.

MSCP Event: Host interface hardware error

User Action: Check the T2022 and KDM70 controller software.

B KDM70 Controller Error Codes

B.1 SA Error Codes

Table B–1 lists standard SA event codes associated with the KDM70 controller. SA codes 352(x) through 376(x) are written by module self-test if an error occurs. SA codes 37A through 37F are used to identify fatal controller errors detected during functional code.

The field replaceable unit (FRU) is called out whenever possible. In cases where multiple FRUs are called out, the most likely FRU is listed first.

Table B–1 KDM70 SA Error Codes

Decimal	Hex	Description	FRU Callout
850	352	Bus error test failure	T2022
851	353	Board 2 EEPROM test failure	T2023
852	354	Vectored intrpt. cntlr. test failure	T2022
853	355	VIC init routine failure	T2022
854	356	Board 1 (internal bus) HIB test failure	T2022
855	357	Data path controller RIF test failure	T2022
856	358	CXHIC register interface test failure	T2022
857	359	XMI CRAM test failure	T2022
858	35A	XMI CRAM load test failure	T2022
859	35B	XMI CRAM verify test failure	T2022
860	35C	XMI internal loopback test failure	T2022
861	35D	XMI external loopback test failure	T2022, XMI interface
862	35E	Board 2 internal bus (HIB) test failure	T2023
863	35F	Board 2 SRAM test failure	T2023
864	360	XMI workblock test failure	T2022, XMI interface
865	361	Board 1 SRAM test failure	T2022
866	362	CVAX cache test failure	T2022
867	363	Memory map test failure	T2022
868	364	Memory management test failure	T2022
869	365	Virtual space transfer failure	T2022
870	366	Code transfer routine failure	T2022
871	367	Timers test failure	T2022
872	368	Software interrupts test failure	T2022
873	369	SISM CRAM load routine failure	T2023
874	36A	SISM CRAM verify test failure	T2023

Table B–1 (Cont.) KDM70 SA Error Codes

Decimal	Hex	Description	FRU Callout
875	36B	SISM CRAM read test failure	T2023
876	36C	SIECL calibrate routine failure	T2023
877	36D	SISM microBIST test failure	T2023
878	36E	SISM sector test failure	T2023
879	36F	SISM RTCS test failure	T2023
880	370	SISM OPCODE valid test failure	T2023
881	371	SISM diagnostic read test failure	T2023
882	372	Waiting for update init	N/A
883	373	Soft initialization failure	N/A
884	374	Purge/poll test failure	T2022, XMI interface
885	375	Extended ECC recovery: EEPROM	T2022
886	376	Unexpected restart	T2022, T2023
890	37A	Software logic error	N/A
891	37B	Host interface hardware error	T2022
892	37C	Drive interface hardware error	T2023
893	37D	Controller memory error	T2022, T2023
894	37E	Internal bus hardware error	T2022, T2023, HIB Cable
895	37F	Policy processor error	T2022

B.2 MSCP Controller Event Codes

Table B–2 lists the MSCP event codes for controller detected errors.

Table B–2 KDM70 Controller Event Codes

HEX Code	Description
00A	Last-crash error log packet (LCELP)
02A	Serdes overrun error
04A	External EDC error
06A	Software logic error
08A	Internal EDC error
10A	Controller overrun or underrun error
12A	Controller memory error
16A	Device interface hardware error
18A	Host interface hardware error
1AA	Internal bus error
1CA	Policy processor error
1EA	Fault management analysis event
20A	Self-test error

B.3 Known Tape Errors

Protocol errors when drive not online

If a drive fails and drops offline (i.e., losses vacuum, check errors, etc), KDM error recovery code will still attempt to issue commands to the tape drive that are not legal to an offline drive. These commands will be rejected thus causing a protocol error. This results in error logs that are misleading.

Use the tape device error logs or device visual indicators to troubleshoot the problem.

TA78/79

DATA LATE's

Data late errors intermittently occur (Testing indicates approximately 10%) on one TA7X drive when another TA7X drive connected to the same formatter is performing a Rewind command with the Erase modifier set. There are no data integrity issues, the error is recoverable, and at best is a possible performance impact. The error occurs with either the KDM or the HSC.

This is a TS78 Formatter Ucode bug. Based on the infrequent use of Rewind-Erase and the fact that there have been no known customer complaints, CLD's, Prisms, etc., it has been recommend that the TS78 Ucode not be changed to fix this problem and thus there will not be an FCO to correct this problem.

WRITE-LENGTH

Write-length errors intermittently occur on TA7X drives when writing the ID burst at 1600 BPI. This problem occurs on both the KDM and HSC (although the HSC has a different error code).

The KDM hardware declares a "00CC" error (write length) which means that Data Ready was dropped (by the drive) in the middle of the transfer. This is a fatal error, and reported as such to the host. The Error Number low and high bytes from the Get Extended Drive Status are "0414" (Could not find gap after ID code was written correctly).

There is no work ongoing to fix this problem and thus there will not be an FCO to correct this problem.

TA81

The KDM70 will deliver an error log (event code = FF6B) after a TA81 unload. This is a normal condition for TA81's attached to a KDM70, and should not be considered as a failure.

If an error occurs during normal TA81 tape operation, two error logs will be delivered for each error (event code = FF6B).

B.4 Tape TMSCP Event Codes

Table B–3 shows new TMSCP event codes related to the KDM70 controller.

Table B–3 TMSCP Event Codes

Event Code	Description
02C	Drive/Formatter timeout
04C	Controller-detected transmission error
06B	Data Ready Timeout
08B	Acknowledge not asserted at start of transfer
0CB	Receiver Ready not asserted at start of transfer
0CC	Write length error
0EC	Drive/Formatter-detected error
10C	Controller-detected pulse/parity error
16C	Drive/Formatter failed initialization
18C	Drive/Formatter ignored initialization
1AC	Receiver ready collision
1CB	TA90 Micro-code not at minimum revision (2.3)
F6C	Formatter requested error log

C XMI Register Summary

C.1 Introduction

Table C–1 lists the XMI nodespace base addresses for XMI registers. The register offsets are listed in Table C–2. The base address is determined by the physical location of the KDM70 controller processor module (T2022). For example, a T2022 module located in XMI slot 3 would be seen as node number 3. The XMI registers would begin at base address 21980000.

The register address is obtained by adding the base address to the offset for the selected register. For example, to obtain the address of the KDM70 controller SA register for node 2, use the following formula:

```
21900000 + 44 = 21900044
```

Note: The example shown in Table C–1 is for a single XMI. Refer to system-specific documentation for systems with multiple XMIs.

Table C–1 XMI Nodespace Registers

XMI Nodespace Base Address	Node Number
2188 0000	Node 1
2190 0000	Node 2
2198 0000	Node 3
21A0 0000	Node 4
21A8 0000	Node 5
21B0 0000	Node 6
21B8 0000	Node 7
21C0 0000	Node 8
21C8 0000	Node 9
21D0 0000	Node 10
21D8 0000	Node 11
21E0 0000	Node 12
21E8 0000	Node 13
21F0 0000	Node 14

The following table shows XMI node space offsets:

Table C–2 XMI Nodespace Offsets

Register		Offset	Size
(XDEV)	Device type register	00	32 Bits
(XBE)	Bus error register	04	32 Bits
(XFADR)	Failing address register	08	32 Bits
(XFAER)	Failing address extension register	2C	32 Bits
(IP)	Initialization/Polling register, read/write	40	16 Bits
(PE)	Port error, read-only	40	16 Bits
(SA)	Status register	44	16 Bits
(PD)	Port data register	48	32 Bits
(IP)	Initialization/Polling register, read-only	4C	16 Bits
(PE)	Port error register, read/write	4C	16 Bits

C.1.1 Examining Registers

To examine a register, halt the system by running system shutdown procedures. At the console prompt, type:

```
>>> EXAMINE node_register_addr
```

To examine the SA register for node 1, using the offset from Table C–2. Type the following command:

```
>>> EXAMINE 21880044
```

To examine the bus error register (XBE) for node 3, add the nodespace offset to the register address as in the following example:

```
>>> EXAMINE 21980004
```

Using this method, you can examine all the registers listed in Table C–2.

C.1.2 XDEV: Device Type Register

The XDEV register contains node identification information. The XDEV fields are loaded during initialization. A zero value indicates an uninitialized node.

Figure C–1 XDEV

31 24	23 20	19 16	15 00
SOFTWARE REV	BD2 REV	BD1 REV	KDM70 DEVICE TYPE (0C22 HEX)

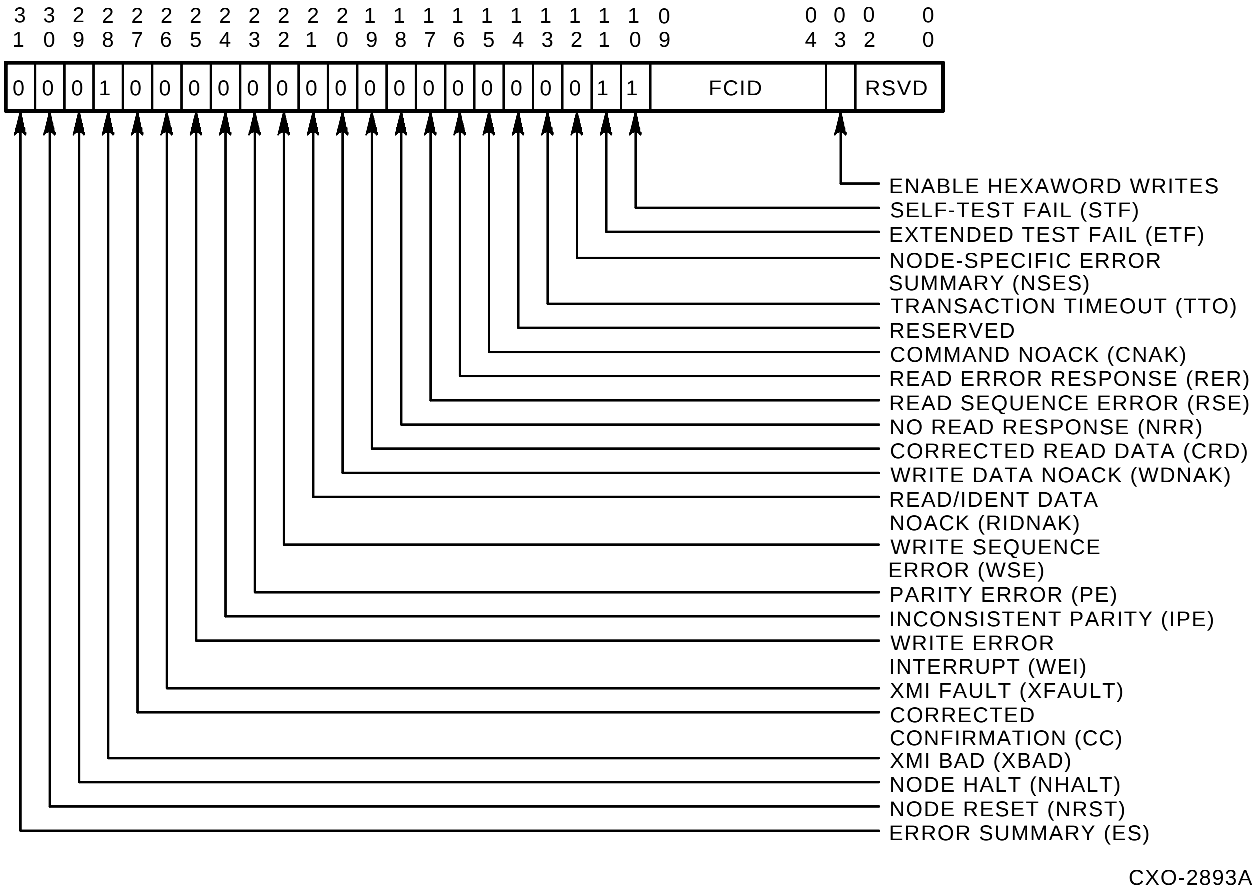
CXO-2892A

The fields are defined as follows:

Field	Description
Software revision	The revision level of the KDM70 software
BD1 revision	The hardware revision level of the T2022 module
BD2 revision	The hardware revision level of the T2023 module.
KDM70 Device type	The KDM70 device type code

C.1.3 XBE: Bus Error Register

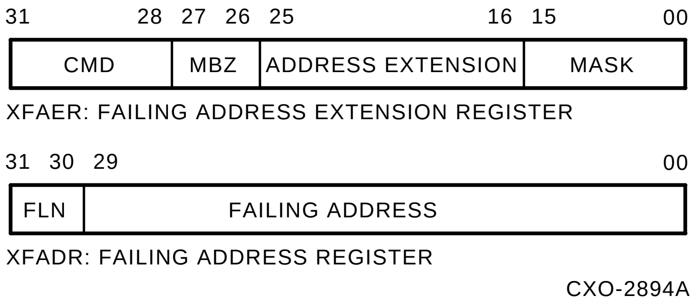
Figure C–2 XBE



C.1.4 XFADR and XFAER Registers

The XFADR and XFAER registers are used to log address, command, and length information associated with a failing transaction. These registers are loaded on every command cycle. On the occurrence of an XMI bus error, loading is disabled, latching bits <63:0> of the XMI D lines.

Figure C–3 XFAR and XFAER Registers



C.1.4.1 XFADR — Failing Address Register

XFADR <31:30>: This field is used to log the value of XMI D <31:30> during the command cycle of a failing transaction.

XFADR <29:0>: This field is used to log the value of XMI D <29:0> during the command cycle of a failing transaction. For read and write transaction, this field contains bit 39 and bits <28:0> of the address specified by the transaction.

C.1.4.2 XFAER — Failing Address Extension Register

XFAER <31:28>: This field is used to log the value of XMI D <63:60> during the command cycle of a failing transaction. During such cycles this field contains the command code of the transaction.

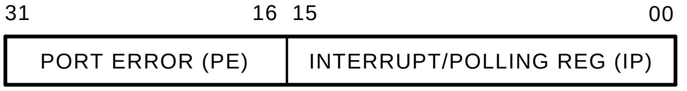
XFAER <25:16>: This field is used to log the value of XMI D <57:48> during the command cycle of a failing transaction. For read and write transactions, this field contains bits <38:29> of the address specified in the transaction.

XFAER <15:0>: This field is used to log the value of XMI D<47:32> during the command cycle of a failing transaction. For write transactions, this field contains the write mask, while for other transactions it is undefined.

C.1.5 PE and IP Registers

The port error and initialization/polling register are each contain 16 bits. The IP is contained in bits <15:0> and the PE is in bits <31:16>.

Figure C–4 PE and IP Registers



CXO-2895A

C.1.5.1 Initialization and Polling Register (IP)

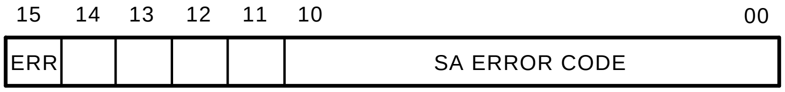
The IP register has three functions for XMI controllers:

1. When read or written by the host with any value and a connection between the host and KDM70 controller exists, causes the controller to examine the current location of the command queue in the communications area for commands. Note the term "polling" is used to describe the controller's access of the command queue. When read by the host and a connection between the host and controller does not exist, the device controller ignores the IP register read.
2. When written with a 1 by the host followed by a node HALT, causes the controller to perform a soft initialization.
3. When written with a 2 by the host followed by a node HALT, causes the controller to perform a maintenance initialization.

C.1.6 SA Register

The SA register is written by self-test (MIST) or functional code to indicate that a failure has occurred. This register contains valid error information only if the "ERR" bit is set. Refer to Table 4–2 for a description of KDM70 SA codes.

Figure C–5 SA Register



CXO-2932A

D Power-Up and MIST Diagnostics

D.1 MIST and DEMON Test Descriptions

This appendix describes the KDM70 controller power-up and module internal self-test (MIST) diagnostics. The appendix is divided into two sections:

- KDM70 core hardware tests
- KDM70 MIST diagnostics

D.2 KDM70 Core Hardware Test Descriptions

KDM70 core hardware diagnostics run from power-up until the MIST diagnostics execute from SRAM. Testing is performed in two steps:

1. Test the processor boot hardware. This is the hardware involved in fetching machine instructions from the boot UVPROM and executing it within the CVAX.
2. Test hardware associated with loading and executing MIST code from SRAM. Core hardware tests are stored and executed out of the Boot UVPROM.

D.2.1 KDM70 Core Hardware Test Descriptions

This section describes KDM70 core hardware tests. Board 1 status LEDS change during testing. They act as visual and readable progress code.

Initialization Test

The following registers are initialized/verified during the initialization test:

CVAX stack pointer (SP)
CVAX system control block base register (SCBB)
CVAX interrupt priority level (IPL) IPR
SSC bus timeout control register
SSC configuration register
SSC channel 0 match register
SSC channel 0 mask register
SSC channel 1 match register
SSC channel 1 mask register

Boot UVPROM Test

An error detection code (EDC) check is performed on each page of core hardware test code in the boot UVPROM.

CVAX PSL Test

The restart contents of the CVAX PSL are tested and all condition flag bits are verified.

CVAX GPR Test

The CVAX GPR test performs a stuck-at-bit test and re-initialization of the stack pointer. This is followed by stuck-at-bit testing of the remaining general purpose registers.

SSC RAM Test

A checkerboard memory test is performed on SSC internal RAM. Additionally, a byte read/write test verifies that the byte mask bits operate correctly between the CVAX and the SSC.

Instructions Test

The VAX hardware instruction set is tested through the execution of a subset of VAX macro instructions. Instruction access modes are also verified with this test.

EEPROM Interface Test

The EEPROM interface test verifies that at least two of three copies of the board 1 bad page list and parity data are equal. It then verifies the board 1 control store EEPROM for proper parity data contents.

CVAX Parity Detector Test

The board 1 diagnostic write register is checked for stuck-at faults by writing and reading different test patterns to/from the register. Additionally, the CVAX bad parity detect logic is tested on longwords and single bytes.

SRAM Test

The SRAM test locates a section of good board 1 memory large enough to hold MIST code, memory for MIST scratchpad, and memory for the code transfer utility.

EEPROM Test and Code Transfer

This is a combination of EEPROM testing and code transfer testing using an EDC/ECC algorithm. Validity testing and correction of bad bytes within MIST physical space is performed. Good code is transferred to board 1 bank 1 SRAM, which was tested in the previous SRAM test.

Transfer to Physical Space MIST

This routine transfers execution control from the core hardware test in boot ROM to the DEMON in board 1 SRAM for execution of the MIST.

D.3 KDM70 Module Internal Self-Tests

Once the core hardware tests have successfully run, the MISTs, which test the KDM70 subsystem, are executed. Tests are executed in the order listed.

Bus Error Test

The following circuitry is tested:

- Board 1 diagnostic read register
- Bus timeout detection logic in the SSC
- RDY and decoder PALs and associated address decode circuitry
- Nonexistent memory (NXM) detection logic
- Read and write parity checkers on the CDAL

- Serial EEPROM data and control bits in the board 1 diagnostic write register.

Board 2 EEPROM Test

This test verifies board 2 serial EEPROM can be read. The test uses two out of three longwords read from EEPROM.

VIC Test

This test verifies the vectored interrupt controller (VIC) chip. VIC reset and enable are tested by writing the board 1 diagnostic write register. A stuck-at test is performed on the read/write registers of the VIC, and the VIC's interrupt handling is tested through the use of the interrupt request bit in the board 1 diagnostic write register.

VIC Init Routine

The internal VIC registers are initialized to the required functional code state with individual interrupts disabled.

Board 1 HIB Test

This test verifies board 1 HIB address/data, parity logic, and HIB bus register. The following circuitry is tested:

- HIB bus register initialization
- HIB bus register address latch
- HIB bus address parity
- HIB bus CVAXII MUX
- HIB bus chip select
- HIB bus byte mask
- DPC data parity detection register

Board 2 HIB Test

The board 2 HIB test verifies board 2 HIB address control and data logic. It also verifies the data path for the board 2 diagnostic write register (DWR), board 2 diagnostic read register (DRR), and board 2 error address register (EAR). The following circuitry is tested:

- Board 2 DWR initialization
- Board 2 DWR read/write path
- Board 2 DRR/EAR data parity
- Board 2 HIB Address parity detect
- Board 2 HIB Control parity detect
- Board 2 HIB Data parity detect
- Board 2 SRAM data path

Board 1 SRAM Test

Board 1 SRAM test performs a checkerboard test of the remaining board 1 SRAM not tested during the core hardware test. The byte mask bits between the CVAX and banks 1 and 2 of board 1 SRAM are also tested through byte-wide writes and reads.

DPC RIF Test

This test verifies the HIB/DPC interface and the internal DPC registers. The DPC internal self-test is used to perform a stuck-at test of the internal DPC registers. The HIB/DPC interface is verified by writing and reading several internal DPC registers. Initial DPC register states are also verified.

CXHIC RIF Test

This test is used to verify the CXHIC/HIB interface and the internal CXHIC read/write registers. A thorough stuck-at test is performed on the read/write registers, and the initial CXHIC register states are verified after a CXHIC reset.

XMI CRAM Test

This routine verifies the DPC control RAMs by performing a checkerboard memory test on them.

XMI CRAM Load Routine

This test loads the XMI microcode from board 1 SRAM into the DPC control RAMs.

XMI CRAM Verify Test

This test verifies the loaded XMI microcode and the ability of the DPC to detect microcode parity errors. Each microword in the CRAM is compared to the corresponding longword in board 1 SRAM. The DPC is then put into a special mode in which it sequences through each of its microcode addresses, reading the microword and checking parity. During the load of XMI microcode, a microword containing bad parity is written into the last location of DPC control RAM. When this location is reached, the test ends.

XMI Internal Loopback Test

This test verifies the ability of the CXHIC to perform CVAX port read and write transactions. All transactions are performed in loopback mode and therefore do not rely on any circuitry on the XMI side of the CXHIC. Also verified during the test is the ability of the CXHIC to properly increment the destination address and to load the next page frame when required.

XMI External Loopback Test

This test is identical to the read/write transactions portion of the XMI internal loopback test, except that loopback is not used. This allows the logic on the XMI side of the CXHIC to be verified.

Board 2 SRAM Test

The board 2 SRAM test performs a checkerboard test of the entire board 2 data buffer memory. The byte mask bits between the CVAX and the board 2 SRAM are also tested by performing byte-wide writes and reads.

CVAX Cache Test

This test verifies the proper functionality of the CVAX internal cache memory. A special diagnostic mode is used to force data patterns into the cache. The patterns are then read back during cache hit cycles.

Memory Map Routine

This routine maps virtual memory into physical memory by setting up the system page table in board 1 SRAM.

MMU Test

This test verifies proper functionality of the CVAX's memory management unit by writing data patterns into physical pages of board 1 SRAM and then enabling mapping and reading back the corresponding virtual addresses.

Virtual Space Transition

This routine initializes the KDM70 environment for virtual space execution and then turns on mapping. All remaining MIST tests execute from virtual space.

EEPROM Test and Code Transfer

This is a combination EEPROM test and code transfer using an EDC/ECC algorithm to check the validity and, if required, correct bad bytes of the MIST/functional code image, while at the same time transferring the good code into board 1 SRAM.

Timers Test

This test is used to verify proper functionality of the two programmable timers and the interval timer, all supplied by the SSC. Both polling and interrupt modes of the programmable timers are checked.

Software Interrupts Test

This test verifies the ability of the CVAX to accept requests for software interrupts and to properly service them. The software interrupt logic is contained entirely within the CVAX.

XMI Workblock Test

This test verifies the interface between the DPC and the CXHIC during XMI workblock transactions. It also verifies the DPC Idle signal, the ability to preempt the XMIC state machine through software, and the ability of the CXHIC and DPC to request interrupts when appropriate. Buffer to buffer compare workblocks and XMI DMA read and write workblocks are used during the testing.

SISM CRAM Load Routine

This routine loads both SI state machine (SISM) control RAMs with the SI microcode instructions. The SISM microinstructions are 40 bits wide with even parity in the most significant bit. The microcode is built with the CVAX MIST image, and resides in instruction SRAM. Each SISM control RAM array is 5 bytes wide, and can accommodate up to 2K microinstruction. Both SISM CRAMs are mapped into two locations of CVAX address space: the first provides access to the lower four bytes of each microword, the latter provides access to the upper byte of each microword. Access of all CRAM locations is controlled via internal counters, which increment after each access. The board 2 diagnostic write register (board 2 DWR) controls CRAM read/write direction and internal counter reset.

The SISM CRAM load routine verifies the internal counter after each microinstruction via the board 2 diagnostic read register (Board 2 DRR). The routine sets up and loads the lower CRAM for SISM A, followed by a load of SISM A upper CRAM. The procedure is repeated for SISM B.

The unused CRAM locations, if any are zero filled. The microcode parity (bit 39) of the last CRAM location (hex 7FF) is inverted, as a safeguard against CRAM location counter error, and is used in a later test.

SISM CRAM Verify Test

This test verifies the contents of both SISM control RAMs against the microcode built with the MIST image. The routine sets up and reads the lower CRAM for SISM A, followed by read of SISM A upper CRAM. The procedure is repeated for SISM B. The SISM CRAM load routine verifies the internal counter after each microinstruction via the board 2 diagnostic read register (board 2 DRR). The unused locations are verified to contain zeros, and the inverted parity at location 7FF is verified.

SISM CRAM Rolling Read Test

This test verifies that the SISMs can correctly access their control RAMs without error. The CVAX starts each SISM on a sequential read of its CRAM, ignoring the microinstruction. The CVAX waits for the SISM to detect a CRAM parity error at location 7FF, then verifies the bad parity location and CRAM overflow condition in the board 2 diagnostic read register (board 2 DRR).

SIECL Calibrate Routine

This routine is designed to calibrate the 12- and 20-nsec delay line circuitry in both SIECLs. The routine is driven by the CVAX, without SISM activity. The routine checks that the calibration counter in the CSIC is functioning properly. The board 2DWR is then used to configure the 12-nsec clock into the CSIC calibration counter. Calibration values are clocked serially into the CYCLE using the board 2DWR; the CSIC calibration counter is then read twice, at a 0.5-millisecond interval. The counter values are compared to determine the actual frequency of the CYCLE clock. The CYCLE calibration value range is 0-255, and a binary search algorithm is used to determine the optimum value. The procedure is repeated for the SIECLs 20 nsec clock.

SISM MicroBIST Test

This test consists of four subtests. The following sections describe each subtest.

External Register Subtest

This subtest verifies the interface between the CVAX and the CSIC external registers. The test writes patterns to the SISM workblock address register (WAR), selects the WAR using the SISM control register (CR), then reads and verifies the pattern through the SISM read register (RR).

Microcode Subtest

The following list describes the SISM functions tested in the microcode subtests:

- CSIC TMUX test 1: This tests the CSIC branch register through the predefined jump function in microcode. This test ensures all microcode branching executes correctly.
- CSIC WAR test: This test waits in a tight loop until the CSIC's WAR is written with an address in data buffer memory. The address is the first location in data buffer memory of all the data for the remaining MicroBIST microcode tests to read/write/etc.
- CSIC BIF test: This test reads a data pattern from buffer memory and writes it into the SISM writable registers. The test also writes the data pattern back to buffer memory.
- CSIC TMUX test 2: This test checks the functions of the primary and secondary test MUXes. There are 18 tests in all and there is special data set up by the CVAX and read in from buffer memory by the microcode to set/clear various flags and to check these flags. The microcode writes a value to indicate the state of the flags in the Test MUXes. The CVAX checks the state written to memory and calls out any discrepancies after the test is complete.
- CSIC SERDES tests 1 and 2: The function of the two tests is the same, except that in the second, the CVAX sets the forced error bit in the diagnostic write register, which causes the CSIC to see pulse/parity errors. The CVAX writes the necessary data to buffer memory. This microcode writes three words, allowing the data to loop in the SERDES register. The microcode reads 17 consecutive words, which allows the ECC and EDC logic to be exercised. The data is looped back through the SERDES and written out to buffer memory, including the final status of the test. The CVAX verifies that the data in buffer memory is correct.
- Internal register subtest: This test verifies the contents of the SISM's internal registers immediately following execution of the microcode test, above. The CSIC register data path is 16 bits, and each register requires two select and read operations. Each register is read by selecting (via Control Register) the upper/lower internal register number, followed by a read (via Read Register) of the upper/lower internal register value.

- SISM data parity error subtest: This test checks the SISM data parity detection logic. The CVAX first loads the SISM Workblock into data memory. parity error interrupts are disabled, and the board 2 data parity invert is set in the board 2 DWR. The CVAX starts the SISM and waits for interrupt. This action directs the SISM to write data into data memory with bad parity. The CVAX restarts the SISM, and again waits for an interrupt. The SISM reads the location with bad parity, and the CVAX verifies that a HIB Bus parity error was detected.

SISM Sector Test

The following steps are performed in this test:

1. CVAX loads workblock into memory and writes WAR to start SISM.
2. CVAX waits for an interrupt (or timeout).
3. SISM clears the sector counter and sets interrupt to synchronize with CVAX.
4. CVAX clears the interrupt and verifies that the sector counter has a zero value.
5. CVAX reads/writes to WAR to restart SISM.
6. SISM reads the desired sector count from workblock.
7. The counter is incremented and an interrupt is sent to CVAX.
8. CVAX clears the interrupt and verifies that the sector counter has the desired value.
9. CVAX reads/writes to WAR to restart SISM.
10. The previous six steps are repeated one more time.

SISM RTCS Test

The following steps are performed in this test:

1. CVAX loads the workblock into memory and writes WAR to start SISM.
2. SISM reads the opcode longword with the port number.
3. SISM sets all possible state bits in the state frame; sets sync last.
4. SISM waits for two cycles of an RTCS reload; checks the state bits in TMUX.
5. SISM checks the state and writes the status of the state bits as reflected in TMUX.
6. SISM clears all state bits, including sync.
7. SISM waits for two cycles of an RTCS reload; checks the state bits in TMUX.
8. SISM writes the status to buffer memory again.

9 CVAX waits for an interrupt/timeout; checks memory against expected data.

SISM Opcode Validate Test

The following steps are performed in this test:

1 CVAX loads the workblock into memory and writes WAR to start SISM.

2 SISM reads the status pointers and opcode with the port number.

3 SISM tests PUNTL inactive:

- Read in any opcode with diagnostic port set
- Read in an opcode for a NOP on a real port
- Read in a real port and level 1 command
- Read in a real port and wait for rw/rdy command
- Read in a real port and wait for sector command
- Read in a real port and a disk read command
- Read in a real port and a tape read command

4 SISM tests for PUNTL active:

- Read in opcode with premature bit set
- Read in opcode with M bit set and IGNORE MARKED WB set in SISM control reg
- Read in opcode with wait-for-sector command
- Test absence of sync
- Test absence of rw/rdy
- Test absence of receiver ready

5 SISM writes status of PUNT signal after each test.

SISM Diagnostic Read Test

This test consists of two subtests:

1 Diagnostic Read Setup: The diagnostic read setup workblock test is used to initialize each SISM as either A or B. The CVAX loads a workblock into memory, sets the appropriate A/B flag, and directs each SISM to execute the workblock.

2 Diagnostic Read: The diagnostic read subtest exercises both SISMs together. One SISM emulates an SI drive by sending data through the CSIC data channel to the other SISM. The second SISM executes functional microcode and reads the data sent by the first SISM. The results are written to data memory. The CVAX loads send and receive workblocks into data memory and starts both SISMs. When microcode execution is complete, the CVAX verifies the data patterns written to memory. The test is repeated with data flow between SISMs reversed.

E ELECTROSTATIC DISCHARGE (ESD) PART LIST

E.1 INTRODUCTION

This appendix lists the part numbers and part descriptions for Electrostatic Discharge (ESD) kits and materials.

Table E–1 lists the part numbers and part description

Table E–1 ELECTROSTATIC DISCHARGE PART NUMBER LIST

PART NUMBER	PART DESCRIPTION
29-25435-00	ESD Demonstration Kit (110-volt)
29-25433-00	ESD Demonstration Kit (220-volt)
29-11762-00	Velostat Field Service Grounding Kit (see contents below)

The Velostat Field Service Grounding Kit (complete) listed in the Table E–1 includes the following items:

- 10-foot coil cord (for wrist strap) (Part Number 29-25492-00)
- Velostat field service work surface (Part Number 29-25493-00)
- Insulated alligator clip (Part Number 29-25494-00)
- Wrist strap, small (Part Number 29-25496-00)
- Wrist strap, medium (Part Number 29-25497-00)
- Wrist strap, large (Part Number 29-25498-00)

Table E–2 lists the part numbers and part description for Static Shielding Bags.

Table E–2 STATIC SHIELDING BAG PART NUMBER LIST

PART NUMBER	PART DESCRIPTION
99-07092-01	Static Shielding Bag, 4 inches x 6 inches i.d.
99-07092-02	Static Shielding Bag, 6 inches x 8 inches i.d.
99-07092-03	Static Shielding Bag, 8 inches x 12 inches i.d.
99-07092-04	Static Shielding Bag, 10 inches x 14 inches i.d.*
99-07092-05	Static Shielding Bag, 12 inches x 18 inches i.d.
99-07092-06	Static Shielding Bag, 14 inches x 18 inches i.d.
99-07092-07	Static Shielding Bag, 18 inches x 24 inches i.d.
99-07092-08	Static Shielding Bag, 15 inches x 20 inches i.d.

Table E–2 (Cont.) STATIC SHIELDING BAG PART NUMBER LIST

PART NUMBER	PART DESCRIPTION

* Minimum size shielding bag required for the each HSX50 board.

Index