EK-DS725-UG-001

VAX-11/725 Diagnostic System Overview Manual

digital™

EK-DS725-UG-001

VAX-11/725 Diagnostic System Overview Manual

Prepared by Educational Services
of
Digital Equipment Corporation

1st Edition, February 1984

Printed in U.S.A.

This document was set on a DIGITAL DECset Integrated Publishing System.

CONTENTS

CONTENTS (Cont)

CONTENTS (Cont)

CHAPTER 1
INTRODUCTION

1.1 MANUAL SCOPE

This manual describes the use of the VAX diagnostic system with the VAX-11/725 computer system. It covers the following topics:

- Console commands
- Self test
- Microdiagnostics
- Levels 4 and 3 macrodiagnostics
- Diagnostic supervisor
- Customer runnable diagnostics
- Maintenance disk

This manual serves as a reference for Field Service engineers and as a resource for field service, manufacturing, and customer training programs. To use this manual, the user should be familiar with VAX architecture, VAX/VMS software, and VAX-11/725 hardware.

This manual is part of a four-level documentation set, as shown in Figure 1-1.

These four levels of documentation form a progression from general to specific. The general level being the *VAX Diagnostic System User's Guide* and the specific being the *Diagnostic Program Listings*. To apply the VAX diagnostic system effectively, the user should become familiar with each documentation level. The *VAX Diagnostic System User's Guide* contains stable information that applies to all VAX computer systems. It explains the VAX diagnostic system structure and strategy and the various uses of the diagnostic supervisor. The *VAX Diagnostic System User's Guide* is available in hardcopy and microfiche.

This manual, the *VAX-11/725 Diagnostic System Overview Manual*, is available in hardcopy, microfiche, and ASCII files. The manual may be revised periodically. Revisions will be available on microfiche and magnetic media and distributed with other diagnostic system microfiche updates.

Program documentation files and program listings are available on microfiche and magnetic media. These are revised and distributed periodically. The program documentation files give information that helps the user to effectively use the diagnostic programs.

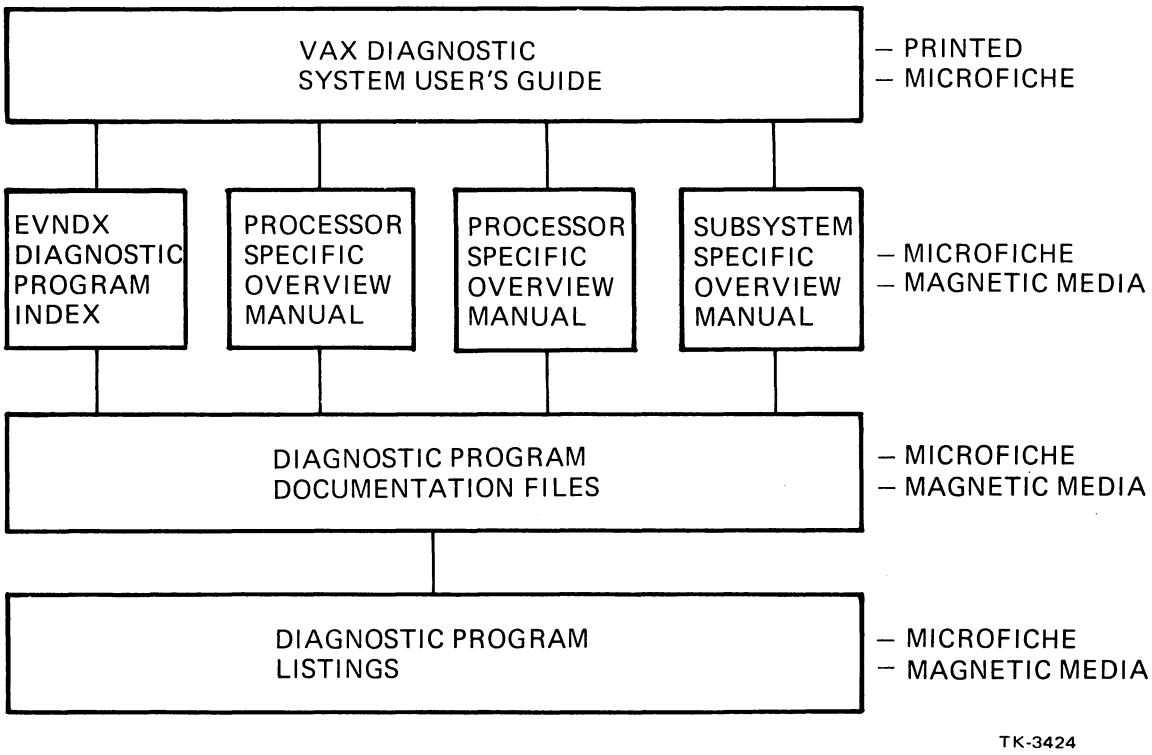

Figure 1-1 VAX-11 Diagnostic System User Documentation

Digital Equipment Corporation also distributes EVNDX, the VAX Diagnostic Index, on microfiche and magnetic media. EVNDX enables users to keep up with the periodic changes and additions to the VAX-11 diagnostic system.

Table 1-1 lists related documentation.

1.2 VAX-11/725 DIAGNOSTIC SYSTEM STRUCTURE

The VAX-11 diagnostic structure consists of the following six program levels.

Level 1 – VAX/VMS-based Operating system diagnostic program using logical or virtual queue I/O, such as:

VAX System Diagnostic (exerciser)

Level 2R – Diagnostic supervisor-based diagnostic programs restricted to running under VMS only using physical queue I/O, such as:

Certain peripheral diagnostic programs

Level 2 – Diagnostic supervisor-based diagnostic programs that can be run either under VAX/VMS (on-line) or in the standalone mode using physical queue I/O, such as:

Formatter and reliability level peripheral diagnostic programs

Level 3 – Diagnostic supervisor-based diagnostic programs that can be run in standalone mode only using physical queue I/O, such as:

Functional level peripheral diagnostic programs
Repair level peripheral diagnostic programs
CPU cluster diagnostic programs

Level 4 – Standalone macrodiagnostic programs that run without the supervisor, such as:

Hard-core instruction set

Level 5 – Console-based diagnostic programs that can be run in the standalone mode only, such as:

Microdiagnostics
Microdiagnostic monitors
Console diagnostics

Table 1-1 VAX-11/725 Related Documentation

Title	Order Number
VAX-11/725 User's Guides:	
Diagnostic System	EK-VX11D-UG
DMF32 Multi-Function Communications Interface	EK-DMF32-UG
Hardware	EK-11730-UG
CRD Users	EK-CR725-UG
VAX-11/725 Installation Guide	EK-725CI-IN
DEUNA User Guide	EK-DEUNA-UG
11725 Field Maintenance Print Set	MP-01728-01
RC25 Disk Subsystem Operation Guide	EK-ORC25-OP
RC25 Disk Subsystem Installation Guide	EK-ORC25-IN
VAX-11/725 Technical Descriptions:	
Central Processor Unit	EK-KA730-TD
Memory System	EK-MS730-TD
Power System	EK-PS730-TD
FP730 Floating Point Accelerator	EK-FP730-TD
KA730 Remote Diagnostics Option	EK-KC730-TM
BA11-A Mounting Box and Power System	EK-BA11-A-TM
DMR11 Sync Controller	EK-DMR11-TM
VAX-11 Handbooks:	
Architecture	EB-19580
Hardware	EB-21710
Software	EB-21812
VAX-11 Reference Manual	EK-VAXAR-RM

Overlapping four levels (levels 5, 4, 3, and 2) of the diagnostic system structure is the Customer Runnable Diagnostic (CRD). CRD is a special program which simplifies the execution of these diagnostics with a single command. Refer to Chapter 7 for further information on CRD.

Most diagnostic programs that test peripheral devices are not processor-specific. They run on the VAX-11/725 processor as well as on other VAX processors. These programs are called transportable diagnostics. They are identified by the letter V as the second character of the five-character program (for example, EVREA).

Of the diagnostic programs that test a VAX-11/725 system, some are transportable and some are processor-specific. VAX-11/725 processor-specific diagnostic programs are identified by the letter N as the second character of the five-character program code (for example, ENSAA).

See the VAX Diagnostic Index, EVNDX, for a complete list of VAX diagnostic programs. Refer to the appropriate program documentation file to answer questions concerning the use of any specific diagnostic program.

The VAX-11/725 diagnostic system provides flexibility concerning the load paths and execution control of different program levels. For example, the diagnostic supervisor and level 2 and level 3 programs can be loaded from either the console load media (TU58 tape drive) or from the system disk. If the primary mass storage load path does not work properly, the TU58 can load diagnostic programs, which help to repair the load path problem. (Refer to Chapter 5.) Although level 2 programs are flexible and run in the user mode or in the standalone mode, levels 2R, 3, and 4 programs are less flexible.

1.3 DIAGNOSTIC STRATEGY

CRD forms the basis of the DIGITAL strategy for VAX-11/725 system maintenance.

The SYE utility can be used to generate an error log report. Analyze this report to isolate the problem to a failing subsystem or option. Further testing can be done using on-line diagnostics (levels 2 and 2R) for fault isolation. If the system is off-line, CRD can be used to identify the failing subsystem or option. Individual microdiagnostics and level 3 macrodiagnostics then can be used to isolate to a field replaceable unit.

Implementation of a specific diagnostic strategy is area-dependent (US, GIA, EUROPE).

1.4 VAX-11/725 REMOTE DIAGNOSIS OPTION

The Remote Diagnosis Option (KC730) is installed in the VAX-11/725 system to allow an operator at a remote terminal (at a remote support center) to perform the same operations as the local operator at the system console terminal. These operations include booting the operating system, controlling the system and communicating with the user programs, and loading and running the diagnostics. These operations are enabled only when the local operator sets the VAX-11/725 keyswitch to the REMOTE or REMOTE DISABLE position.

The local DIGITAL Field Service representative initiates any request for remote support; customer site personnel are not involved in this procedure.

NOTE

Normal system operation for the VAX-11/725 system is not affected by the installation of the KC730 Remote Diagnostic option.

CHAPTER 2
CONSOLE SUBSYSTEM

2.1 INTRODUCTION

The VAX-11/725 console subsystem is an important diagnostic tool. It enables the operator to perform the following functions:

- Starts and stops the CPU instruction set processor
- Examines and deposits information in locations in main memory I/O registers, processor registers, and internal registers
- Controls CPU execution

The console subsystem hardware consists of a dedicated console terminal, switches and lights on the front panel of the CPU cabinet, dual integral TU58 tape cartridge drives, and remote access port.

The VAX-11/725 console runs in two modes: the program and console I/O modes. These modes are mutually exclusive. One of these modes is always enabled while power is applied to the system.

In the program I/O mode, the console terminal functions like a user terminal on the VAX-11/725 system. It passes characters between itself and the program running in the CPU.

In the console mode, the CPU is in the idle loop and receptive to console commands.

The CPU enters the console I/O mode when the front panel keyswitch is set to either the LOCAL or REMOTE position and one of the following occurs:

- Power is cycled with the AUTO RESTART switch in the OFF position.
- Power is cycled with the AUTO RESTART switch in the ON position, if the restart attempt fails.
- Power is cycled with the AUTO RESTART switch in the ON position, if the cold start fails.
- Power is cycled with the AUTO RESTART switch in the ON position, if both the warm and the cold start attempts fail.
- A boot, continue, or start console command fails.
- The operator types CTRL/P on the console terminal.
- The instruction set processor executes a HALT MACRO-32 instruction. (See Table 2-1 for HALT codes and their meanings.)

Table 2-1 CPU Halt Codes and Halt Conditions

Code	Halt Conditions
00	Halt command – given by the operator while the processor is in the Console mode.
01	Self-test was successful.
02	CPU Halt – operator typed CTRL/P while the machine was in the Program mode.
03	Typed by the console on a power fail restart. Does not appear in the halt message.
04	Invalid interrupt stack (IS) or unable to read system control block (SCB).
05	CPU double error. A second machine check occurred before a previous one could be reported.
06	CPU Halt – CPU executed a halt instruction in the kernel mode.
07	Invalid system control block (SCB) vector (bits <01:00> of the SCB vector =3).
08	No user WCS – SCB vector bits <01:00> = 2 with no user WCS installed.
09	Error Pending on HALT – processor was halted by a CTRL/P before an error halt could be performed.
0A	Change Mode from internal stack instruction was executed when the PSL <IS> bit was set.
0B	Change Mode to interrupt stack instruction was executed when the exception vector for a change mode had bit <0> set.
0C	SCB Read Error – uncorrectable memory error occurred when CPU tried to read an exception of interrupt vector.

When the CPU enters the console mode, it prints on the console terminal:

- A question mark followed by a two-digit halt code (Table 2-1).
- The address contained in the program counter (PC).
- The console prompt symbol, >>>.

For example:

```
^P                ! CTRL/P typed at the console terminal
?02 PC=nnnnnnnn   ! Halt code followed by the PC address
>>>               ! Console prompt
```

2.2 FRONT PANEL CONTROLS AND INDICATORS

The front panel of the VAX-11/725 system has two switches and four indicator lights. These switches, indicator lights, and their functions are identified below.

2.2.1 Six-Position Keylock Switch

The six positions of keylock switch and the functions of each are shown in Table 2-2.

Table 2-2 Functions of the Six-Position Keylock Switch

Position	Function
OFF	No power is applied to the CPU or to memory.
STD BY	Power is applied to main memory, the WCS module and to the TU58 tape drives.
LOCAL	Power is applied to the CPU and to memory allowing the CPU to respond to console commands. The remote access port is disabled.
LOCAL/DISABLE	The CPU does not respond to console commands, however the console terminal functions as a user terminal. The remote access port is disabled. The BOOT position of the AUTO RESTART switch is ignored.
REMOTE/DISABLE	The console terminal is disabled. The remote access port is enabled, allowing a terminal connected to it to be used as a user terminal. The CPU does not respond to console commands from this remote terminal.
REMOTE	The remote terminal functions as the console terminal. The console terminal is disabled unless enabled by the remote terminal.

2.2.2 AUTO/RESTART Switch

This three-position switch controls the machine in a power-up sequence, power restoration and software crash. The three positions of the AUTO/RESTART switch and the functions of each are shown in Table 2-3.

Table 2-3 The Functions of the Three-Position AUTO/RESTART Switch

Position	Function
OFF	The system halts and prints the console prompt ">>>" after loading the microcode or executing a HALT instruction.
ON	The system loads the microcode and bootstraps the system disk using the bootstrap command procedure (DEFBOO.CMD) automatically, if one of the following conditions occur: • During initial power up • After a brief power failure • After the CPU microcode detects an error condition (HALT instruction is executed in the kernel mode)
BOOT	This momentary contact position reboots the system (using DEFBOO.CMD) when the system is in the console mode (>>> is displayed on the console terminal).

2.2.3 State Indicator Lights

The functions of the state indicator lights on the front panel are shown in Table 2-4.

Table 2-4 Front Panel State Indicator Lights

Light	Function
RUN	The processor CPU is executing instructions.
DC ON	The dc voltages are applied to the system.
R/D	Remote diagnostics procedures are being performed on the system.

2.3 POWER-UP SELF-TEST

On initial powerup, the VAX-11/725 console subsystem goes through a sequence of events that is shown in Figure 2-1. If the self-enable switch (SW1) on E47 of the WCS module (M8394, slot 3) is OFF (high), the console microprocessor performs a test of the +15 V and −15 V power and the PROM resident self-test. If SW1 is ON (low), the PROM resident self-test is bypassed.

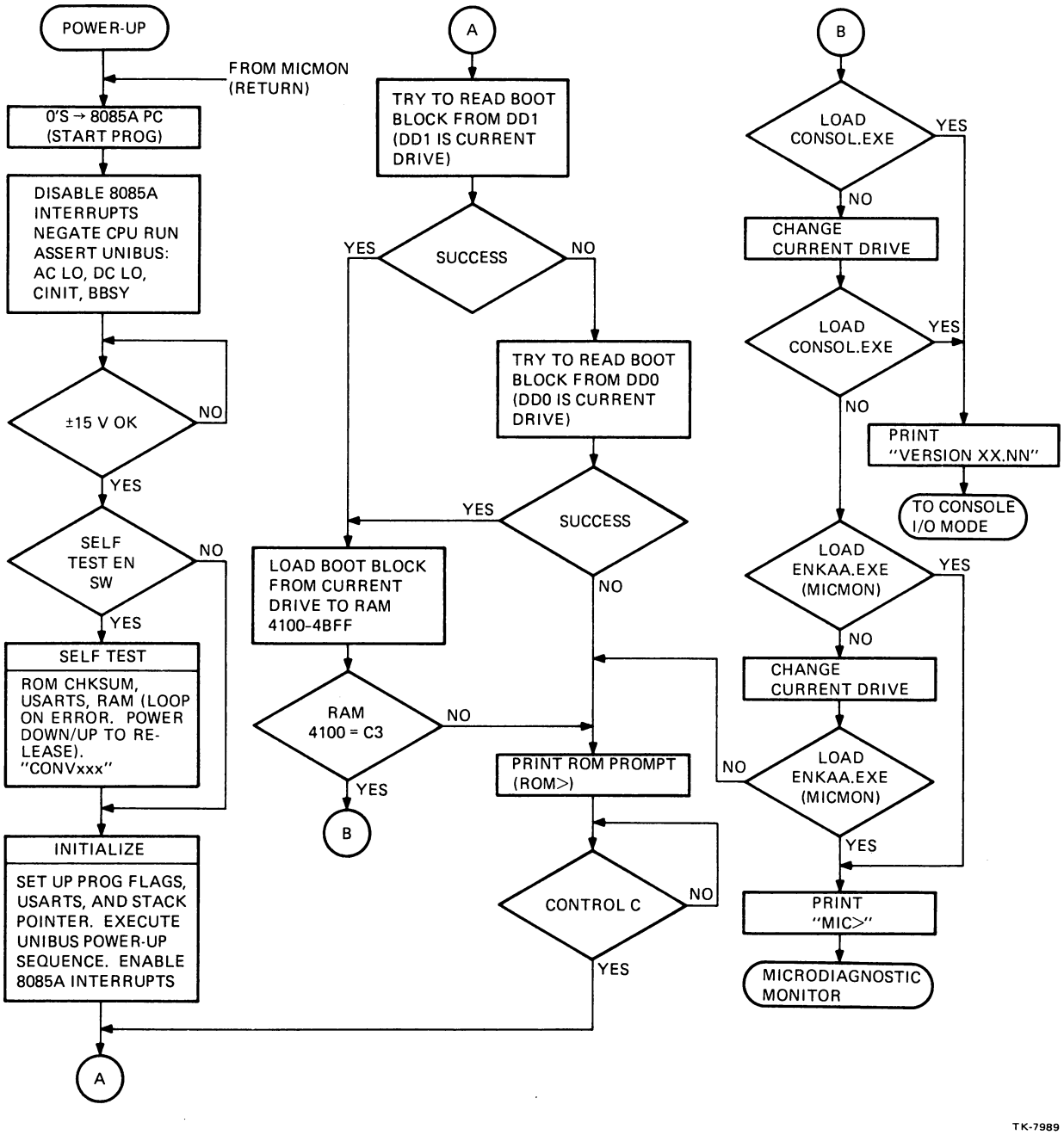

Figure 2-1 Power-Up Sequence of Events

During the self-test the word "CONVxyz" is printed on the console terminal. The "xyz" designation will vary according to the version of the console program. Each letter and digit of this word is a series of subtest completion flags. One letter or digit of the word "CONVxyz" is printed between each subtest and the last two digits are printed at the end of the test (Example 2-1). The self-test sequence is as follows:

```
Print <LF><CR>"C"          ; Start Self-Test (power-up)

Print "0"                  ; PROM CHECKSUM Test

Print "N"                  ; USARTs Loopback Test

Print "V"                  ; USARTs Dual Address Test

Print "x"                  ; RAM Float 1 Test

Print "yz"                 ; RAM March Test
```

```
CONVxyz  (1)
VERSION xx.nn  (2)

>>>@POWER.CMD  (3)
>>>L/C CONSLE.CPU                    !VERSION yy     (3a)
>>>L/C/S:0800 MMIE.CPU               !VERSION yy     (3b)
>>>L/C/S:0E00 POWER.CPU              !VERSION yy     (3c)
>>>S/C 0B (3d)
>>>W    (3e)
>>>@CODE00.CMD  (4)
>>>L/C/S:0E00 FP.CPU                  !VERSION yy     (4a)
>>>L/C/S:1A00 BITFLD.CPU              !VERSION yy     (4b)
>>>L/C/S:1D00 CM.CPU                  !VERSION yy     (4c)
>>>L/C/S:3300 BASIC.CPU               !VERSION yy     (4d)
>>>L/C/S:4B00 QUEUE.CPU               !VERSION yy     (4e)
>>>I    (5)
>>>     (6)
```

Example 2-1 Printout Produced by the Console Terminal as the CPU Enters the Console I/0 Mode

NOTE

The console message and the subtests performed may change as later versions of the program microcode are released.

If an error is detected while checking the 15 V power, or during a subtest, the console microprocessor loops on that failing subtest. The two RAM subtests have a failing subtest error message. The other subtests do not print a failure message.

The PROM resident self-test is as follows:

- PROM CHECKSUM Test – The PROM code is verified by calculating a checksum on the PROM and comparing it to a known checksum stored in the last location of the PROM.

- USARTs Loopback Test – The three USARTs are verified by sending out data and reading it back via a wraparound function in the USARTs.

- USARTs Dual Address Test – The three USARTs address response is verified by writing different data to each USART command register, then reading the data back. This ensures that each USART responds only to its unique address and no other address.

- RAM Float 1 Test – The RAM data test checks the RAM data lines, the first location of RAM (4000 hex) is read/write tested with a pattern of ones (1) floating through a field of zeros (0).

  Failing Test Printout:

Expected	**Received**
XXXXXXXX	XXXXXXXX

- RAM March Test – The RAM march test writes a background of 0s, then marches a 1 through it. This leaves a background of 1s through which a 0 is marched. This subtest has a delay inserted in it so that the RAM refresh hardware is also tested.

  Failing Test Printout:

Expected	**Received**
XXXXXXXX	XXXXXXXX

2.4 CONSOLE MODES

Immediately following the completion of the self-test, the CPU enters one of three modes depending upon whether a tape cartridge is inserted into a TU58 tape drive unit and whether that tape contains the console or microdiagnostic files. The mode entered is indicated by the console terminal. It prints either the ROM idle loop prompt (ROM>), the console I/O mode prompt (>>>), or the micromonitor mode prompt (MIC>).

If no tape cartridge is inserted into either tape drive unit, the CPU enters the ROM idle loop mode and the console terminal prints the ROM> prompt. The console subsystem stays in ROM idle mode until a CTRL/C is typed. When CTRL/C is typed, the console subsystem performs the self-test again and then looks for a tape cartridge in one of the drives.

If the console file (CONSOL.EXE) is found on the tape cartridge inserted into a tape drive unit, the console terminal produces a printout similar to Example 2-1 and the CPU enters the console I/O mode.

The steps indicated in Example 2-1 are outlined below:

1. These are console self-test flags.

2. This is the console program release version.

3. The boot block instructions are executed by the console microprocessor. They load the file CONSOL.EXE from the console tape cartridge in the TU58 drive unit into the CPU writable control store (WCS). The console prompt (>>>) is printed. The indirect command file POWER.CMD is accessed on the cartridge. The commands within this file are performed.

a. Load the basic console microcode from the file CONSLE.CPU into the CPU WCS starting at microaddress 0.

b. Load the memory management, interrupts, and exceptions microcode from the file MMIE.CPU into the CPU WCS starting at microaddress 0800 (hex).

c. Load the initializing microcode from the file POWER.CPU into the CPU WCS starting at microaddress 0E00 (hex).

d. Start the CPU microcode running at microaddress 0B (hex). This microcode jumps to the INIT sequence at microaddress O E00 (hex). The CPU is initialized, the first 64K bytes of good main memory is found, and the presence of the internal disk controller (IDC) and the floating-point accelerator (FPA) is noted.

e. The console microprocessor waits for the CPU initialization sequence to be completed.

4. The indirect command file CODE01.CMD is accessed on the cartridge. The commands within this file are now performed.

   These commands load the microcode from the files on the cartridge into CPU WCS starting at the addresses shown below:

Step	File Name	CPU WCS Address (hex)
4a	FP.CPU	0E00
4b	BITFLD.CPU	1A00
4c	CM.CPU	1D00
4d	BASIC.CPU	2200
4e	QUEUE.CPU	3B00
4f	IDC.CPU	4000

   The indirect command file CODE01.CMD is one of four possible command files that can load selected microcode into the CPU WCS. The file used depends on the presence of the FPA and the IDC. (The IDC is not used in the VAX-11/725 system.) This presence is determined by a value passed to the console microprocessor from the execution of the microcode at 0E00 (POWER.CPU). The values passed to the console microprocessor to indicate which command file is used are shown below.

	Value Returned	
File Used	**FPA**	**IDC**
CODE00.CMD	0	0
CODE01.CMD	0	1*
CODE02.CMD	1	0
CODE03.CMD	1	1

   *1 indicates presence

5. The CPU is initialized to a known state.

6. The console prompt is printed. The console subsystem is now ready to perform any console command.

If the CONSOLE file is not found and the microdiagnostic file (ENSAA.EXE) is found on the tape cartridge inserted into a tape drive unit, the CPU enters the microdiagnostic monitor mode (see Chapter 4), and the console terminal prints the MIC> prompt.

2.5 CONSOLE COMMAND LANGUAGE

When the CPU is not executing instructions, it is in the console I/O mode idle loop. In this mode, the CPU is receptive to console commands. These commands enable the user to communicate with the VAX-11/725 firmware from the console terminal. A console command is specified either as a control character (CTRL/C) or as a single letter or word with optional modifiers.

This section covers commands that are VAX-11/725 processor-specific.

2.5.1 Control Characters

Table 2-5 lists the control characters and their functions for the console I/O mode.

Table 2-5 Control Characters for the Console I/O Mode

Control Character	Function

CTRL/P (program I/O mode)	Aborts the current command, prints halt message, and returns the console to the console I/O mode idle loop
	
	The console prints:
	
	`^P  !CTRL/P.`
	`?02PC=nnnnnnnnn  !Halt message.`
	`>>>  !Console idle`
	`!loop.`

CTRL/C	Aborts the current operation and returns the console to the console I/0 mode idle loop
CTRL/P (console I/O mode)	Same as CTRL/C
CTRL/U	Aborts acceptance of the current input line; instructs the console program to return to the idle loop and reissue the console prompt
CTRL/S	Stops the terminal from printing the current output; only control characters are recognized while in this state
CTRL/Q	Resumes printing on the teminal after CTRL/S is issued
CTRL/O	Enables and disables the console printout

2.5.2 BOOT Command

The BOOT command is:

B[<space><TU58-select>][<space><device-name>k <CR>

This command loads and executes a bootstrap command file from a drive device (TU58).

```
>>>B            ! Loads and executes DEFBOO.CMD from
                ! the default tape drive.

>>>B DUO        ! Loads and executes DUOBOO.CMD from
                ! the default tape drive.
```

2.5.3 CONTINUE Command

The CONTINUE command is:

C<CR>

If the CPU clock is running, the CONTINUE command restarts execution of a halted program at the address currently in the PC.

If the CPU clock is not running because of a microcode break point or the CPU is in the microstep mode, the CONTINUE command restarts the clock and the console remains in the console I/O mode.

2.5.4 DIRECTORY Command

The DIRECTORY command is:

DIR<CR>

This command prints the directory of tape cartridge that is either inserted into the default TU58 tape drive or inserted into a specified TU58 tape drive, as shown below:

```
>>>DIR          ! Prints the directory of the tape
                ! inserted in the default TU58 drive.

>>>DIR DDn:     ! Prints the directory of the tape
                ! inserted in the specified TU58 drive
                ! (DD0: or DD1:).
```

NOTE

If only one tape cartridge is inserted, then the DIR command prints the directory on that tape. However, if one tape cartridge is inserted into each TU58 tape drive, then the DIR command prints the directory of the tape in the default drive DD1:.

2.5.5 EXAMINE and DEPOSIT Commands

The EXAMINE and DEPOSIT commands are, respectively:

E/[<qualifier>]<space><address><CR> and
D/[<qualifier>]<space><address><space><data><CR>

The EXAMINE AND DEPOSIT commands are explained together because their formats are similar. Both commands require definition of the address space and size of the operand in addition to the address.

The EXAMINE command reads and the DEPOSIT command writes <data> at the <address> specified. The address and data lengths used depend upon the qualifier or qualifiers specified with the command. If no address qualifier is specified, the default is the last used address and data length; following another EXAMINE or DEPOSIT, the same address as that of the previous command will be used as the default.

If no data length qualifier is used (Table 2-6), the default for a physical or virtual EXAMINE or DEPOSIT is whatever the data length was in the previous EXAMINE or DEPOSIT.

Table 2-6 EXAMINE and DEPOSIT Qualifiers and Definitions

Qualifier	Definitions
Data Length Qualifers	
/B	Byte
/W	Word
/L	Longword
Repetition Qualifiers	
/N:<count>	Executes the EXAMINE and DEPOSIT <count>+1 times (>>>E/P/L/N:3 1000), and examines the 4-hexadecimal address in physical address space, starting in location 1000
Address Space Qualifiers	
/V	Virtual Address: This does not work unless mapping is set up for the virtual address reference also. Virtual address examines the display of the translated physical address.
/P	Physical address
/I	Internal Processor Register (IPR) (Listed in Appendix A)
/G	General Processor Register (GPR)
/M	Machine-Dependent Internal Register
/U	Console Microprocessor (ROM or RAM): A console I/O device may be examined using this command (>>>E/U FFaa where aa is an I/O device address 00 through FF).
/C	CPU Writable Control Store (WCS)

Table 2-6 EXAMINE and DEPOSIT Qualifiers and Definitions (Cont)

Qualifier	Definitions
Internal Register Address Specification	
PSL or PS	Processor status longword
PC	CPU program counter (GPR F)
SP	Currently active stack pointer (GPR E)
Rn	General purpose register (GPR) n=0 to F (Hex)
Symbolic Address Specification	
*	Last location
+	Next location
−	Previous location
@	Last data used as address

The <address> information must be either a one-to-eight hexadecimal digit, a register address specification (Table 2-6), or a symbolic address specification (Table 2-6). The initial default is zero; however, the default is unpredictable when the address space is changed.

Following another virtual or physical EXAMINE or DEPOSIT, the default is the sum of the address from the last EXAMINE/DEPOSIT plus the data length from the last EXAMINE/DEPOSIT. Typing a plus sign (+) or minus sign (−) for <address> (for DEPOSIT only) gets this default. Following another IPR or GPR EXAMINE/DEPOSIT, the default is the sum of the address from the last EXAMINE/DEPOSIT plus one. Using PSL for the <address> performs a longword reference of the Processor Status Longword, independent of the address and data length.

The <data> information must be represented by one-to-eight hexadecimal digits.

If more digits than specified by the data length qualifier are supplied, the extra digits to the left are ignored; if fewer digits are supplied, zeros are appended to the left. Examples of EXAMINE and DEPOSIT commands are shown below.

```
>>>E*                    ! Examines the last location examined
                         ! or deposited into.

>>>E                     ! Examines the next location.

>>>E/G <address>         ! Examines GPR register number
                         ! <address>.

>>>E/I <address>         ! Examines IPR register number
                         ! <address>.

>>>E/P                   ! Examines physical <address>.

>>>E/V <address>         ! Examines virtual <address>.
```

```
>>>E PSL                      ! Examines PSL.

>>>E/P/W <address>            ! Examines word at physical <address>.

>>>E/U FF20                   ! Examines console I/O device 20.

>>>E/C 3B00                   ! Examines WCS 3B00.

>>>E/M 12                     ! Examines machine internal register
                              ! 12.

>>>D+ <address>               ! Deposits <data> in the next
                              ! sequential address.

>>>D/G <address> <data>       ! Deposits <data> into GPR <address>.

>>>D/V/W <address> <data>     ! Deposits a word of <data> in
                              ! virtual <address>.

>>>D PSL <data>               ! Deposits <data> into PSL.

>>>D/U 5177 CA                ! Deposits CA into 5177 into console
                              ! microprocessor RAM.

>>>D/C 3B00 43FF11            ! Deposits 43FF11 into WCS 3B00.
```

2.5.6 HALT Command

The HALT command is:

H<CR>

The HALT command does not halt the CPU. HALT causes the console to print the contents of the PC when the CPU is already halted (CTRL/P is used to halt the CPU).

2.5.7 Indirect Command

The indirect command is:

@<file specification><CR>

The @ command loads and executes the indirect command file specified on the power-up sequence or by the operator.

2.5.8 INITIALIZE Command

The INITIALIZE command is:

I<CR>

The INITIALIZE command performs the following:

- Initializes the TU58 controller
- Initializes the internal machine constants
- Initializes the processor to a known state
- Generates new processor status longword (PSL)
- Deposits starting address of first good 64K bytes + 200 to SP

2.5.9 LOAD Command

The LOAD command is:

L/[<qualifier>]<space><file specification><CR>

The LOAD command reads data from the specified file on the console load device to main memory, console microprocessor memory, or to the WCS. If no qualifier is given with the command, then physical main memory is loaded.

LOAD qualifiers are shown in Table 2-7.

Table 2-7 LOAD Qualifiers and Definitions

Qualifier	Definitions
/S:<address>	This start qualifier specifies a starting address for the load. If this qualifier is not given, then the console starts loading at address 0.
/P	This qualifier forces physical main memory to be the destination of the load.
/U	This qualifier forces the console microprocessor memory to be the destination of the load.
/C	This qualifier forces the WCS to be the destination of the load.

Examples of the LOAD command are shown below:

```
>>>L/P/S:nnnn DDn:filename    ! Loads the specified file to
                              ! physical main memory,
                              ! starting at address nnnn
                              ! (hex).

>>>L/P/S:@                    ! Loads data from the default
                              ! TU58 drive device to main
                              ! memory, starting at the
                              ! address specified by the
                              ! last data deposited or
                              ! examined.
```

2.5.10 MICROSTEP Command

The MICROSTEP command is:

M[<space><count>]<CR>

The MICROSTEP command stops the CPU clock and allows the CPU to execute the number of microinstructions indicated by <count>. The console then prints the address of the next microinstruction to be performed, starts the clock and enters the idle loop.

If <count> is not specified the CPU clock is stopped and one instruction is executed, then the console enters the space-bar STEP mode. In space-bar STEP mode, the next microinstruction is executed when the space-bar is depressed. The clock is started and the idle loop is entered by typing <CR>.

NOTE

Stepping through instructions that access the console will not work.

Examples of the MICROSTEP command are shown below.

```
>>>M 2<CR>                    ! The console stops the CPU
                              ! clock and executes two micro-
                              ! instructions.
            UPC=nnnnnnnn      ! Address of next instruction.
            UPC=nnnnnnnn      ! Address of next instruction.
            UPC=nnnnnnnn      ! Address of next instruction.
>>>                           ! Start CPU clock and enter
                              ! idle loop.


>>>M<CR>                      ! Stop CPU clock, execute
                              ! one microinstruction
                              ! and enter space-bar STEP mode.
            UPC=nnnnnnnn      ! Address of next instruction.
<SPACE>                       ! Space-bar depressed.
            UPC=nnnnnnnn      ! Execute instruction at last
                              ! address.
<CR>                          ! Start CPU clock and enter
>>>                           ! idle loop.
>>>
```

2.5.11 NEXT Command

The NEXT command is:

N[<space><COUNT>]<CR>

The NEXT command executes the number of macroinstructions indicated by <COUNT>. The console enters the program I/O mode to perform the instruction, prints the halt code and current contents of the PC, then returns to the console I/O mode idle loop after the <COUNT> completes.

If no number is given for the <COUNT>, one instruction is executed and console enters the space-bar STEP mode. In space-bar STEP mode, the next microinstruction is executed when the space-bar is depressed. The console enters the console I/O mode idle loop from the space-bar STEP mode when a <CR> is typed.

NOTE

During execution of the NEXT command, interrupts are blocked.

Examples of the NEXT command are shown below.

```
>>>N 2<CR>                          ! The console executes two macro-
                                    ! instructions.
?nn          PC=nnnnnnnn            ! Halt code and next address.
?nn          PC=nnnnnnnn            ! Halt code and next address.
?nn          PC=nnnnnnnn            ! Halt code and next address.
>>>                                 ! Enters idle loop.


>>>N<CR>                            ! Executes one macroinstruction
                                    ! and enter space-bar STEP mode.
?nn          PC=nnnnnnnn            ! Halt code and next address.
<SPACE>                             ! Space-bar depressed.
?nn          PC=nnnnnnnn            ! Halt code and next address.
<CR>                                ! Enters idle loop.
>>>
>>>
```

2.5.12 REPEAT Command

The REPEAT command is:

R<space><console command><CR>

The REPEAT command causes the console to repeatedly execute the <console command> (either DEPOSIT, EXAMINE or INITIALIZE) specified until execution is terminated by CTRL/C, CTRL/P or the BREAK key.

2.5.13 START Command

The START command is:

S[<space><address>]<CR> or S/C<space><address><CR>

The START command starts execution of a program that is loaded into memory (see LOAD command). The command starts execution of the program that begins at the specified address in the CPU PC. If no address is given then the data last examined or deposited is used as the CPU PC starting address.

The START command performs the following functions:

1. Initializes the CPU.
2. Deposits the specified address into the PC. If no address is specified, then the current value in the PC is used.
3. Performs the CONTINUE command to begin program CPU execution.

```
>>>S 1000<CR>                       ! Starts the program that begins at
                                    ! address 1000 (Hex).

>>>S @<CR>                          ! Uses the data last deposited or
                                    ! examined as the starting address.
```

2.5.14 TEST Command

The TEST command is:

T<qualifier><CR>

The TEST command performs the self-test then loads and starts one of three diagnostic programs from a console drive device, depending on whether a qualifier is used and, if so, which one. The qualifiers used are:

None	Loads and starts the customer-runnable diagnostic AUTO Mode package (ENSAB.EXE) (See Chapter 7)
/M	Loads and starts the customer-runnable diagnostic MENU Mode package (See Chapter 7)
/C	Loads and starts the microdiagnostic monitor package (ENKAA.EXE) (See Chapter 4)

The operating state of the machine is altered whether or not the program is found. If the program is not found, the console subsystem returns to the console I/O mode to reload itself from the console tape. If the program is found, the console subsystem loads that program into memory and transfers control to it. For example:

```
>>>T                                    ! Loads and start AUTO Mode
                                        ! customer runnable
                                        ! diagnostics.
CONVxyz                                 ! Performs the self-test.
?01   FILE NOT FOUND DD1:ENSAB.EXE      ! The AUTO Mode CRD
                                        ! file was not found on DD1.
            CONTINUING
?02 READ ERROR DD0:
            CONTINUING
?02 READ ERROR DD0:
            CONTINUING

            (Load files from CONSOLE cartridge
            and enter console I/O mode idle
            loop.)
```

2.5.15 WAIT Command

The WAIT command is:

W<CR>

The WAIT command is used by the console subsystem to detect the completion of POWER.CPU on a power-up sequence.

CHAPTER 3
COLD AND WARM START FUNCTIONS

3.1 INTRODUCTION

Whether the VAX-11/725 console subsystem performs a cold start of the operating system or diagnostic supervisor, or a warm start of the operating system, is determined by the following conditions:

- The setting of the front panel AUTO/RESTART switch during a power-up sequence, power restoration, or a system HALT
- The command typed at the console terminal while the CPU is in the console I/O mode wait loop
- The failure of a warm start
- The setting of the AUTO/RESTART switch while the CPU is in the console I/O mode wait loop

At most, the VAX-11/725 attempts one cold start of the operating system and the diagnostic supervisor, and one warm start of the operating system.

3.2 COLD START (BOOT)

Cold start or boot refers to loading either the operating system or the diagnostic supervisor and starting it. Cold start for the VAX-11/725 operating system is initiated by one of the actions listed below:

- A power-up sequence with the AUTO/RESTART switch set to ON
- Typing the BOOT command while the CPU is in the console I/O mode wait loop
- Setting the AUTO/RESTART switch to the momentary contact position of BOOT
- Setting the AUTO/RESTART switch to the ON position and executing a HALT instruction while the CPU is in the kernel mode
- Failure of a warm start while the AUTO/RESTART switch is set to the ON position

Cold start for the diagnostic supervisor is initiated by typing the BOOT supervisor command while the system is in the console I/O mode (Chapter 5).

3.2.1 Console Subsystem Action on a Cold Start

When cold starting either the operating system or the diagnostic supervisor, the console subsystem initiates the sequence of events shown in Figure 3-1.

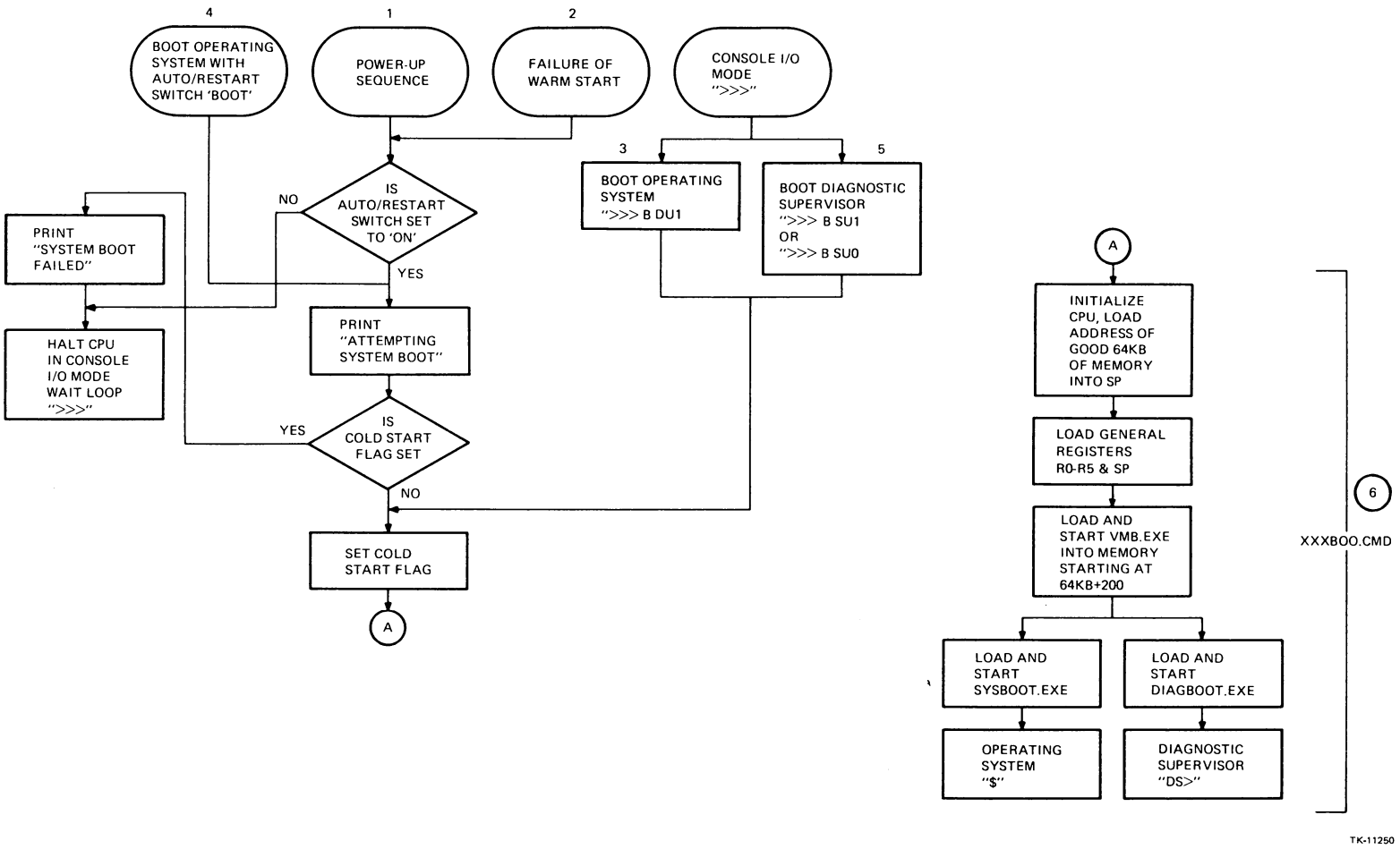

Figure 3-1 Cold Start of Either the Operating System or the Diagnostic Supervisor

As shown in Figure 3-1, the sequence of events is essentially the same for all actions initiating a cold start. The sequence of events for each action is outlined in the following paragraphs.

(1) Power-Up Cold Start Sequence – A power-up cold start of the operating system causes the console subsystem to perform the following sequence of events:

- Execute the self-test, load the console program (Chapter 2), and check the setting of the AUTO/RESTART switch. If the switch is set to the OFF position, the CPU enters the console I/O mode wait loop. However, if the switch is set to the ON position, the console subsystem prints the message ATTEMPTING SYSTEM BOOT on the console terminal.

- Examine the cold start flag. If the flag is set, indicating a cold start has previously been attempted, the message SYSTEM BOOT FAILED is printed on the console terminal. The CPU then enters the console I/O mode wait loop. However, if the cold start flag is not set, indicating that a cold start has not been attempted, the console subsystem sets the cold start flag.

- Execute the indirect command file (XXXBOO.CMD) associated with the type of cold start to be performed. This file executes the functions necessary to cold start the operating system.

(2) Warms Start Failure Cold Start Sequence – The failure of a warm start causes the console subsystem to perform the following sequence of events:

- Check the setting of the AUTO/RESTART switch. If the switch is set to the OFF position, the console I/O mode wait loop is entered. However, if the switch is set to the ON position the console subsystem prints the message ATTEMPTING SYSTEM BOOT on the console terminal.

- Examine the cold start flag. If the flag is set, indicating a cold start has previously been attempted, the message SYSTEM BOOT FAILED is printed on the console terminal. The CPU then enters the console I/O mode wait loop. However, if the cold start flag is not set, indicating that a cold start has not been attempted, the console subsystem sets the cold start flag.

- Execute the indirect command file (XXXBOO.CMD) associated with the type of cold start to be performed. This file executes the functions necessary to cold start the operating system.

(3) Console I/O Mode BOOT Command Sequence – Typing the BOOT command at the console terminal while the CPU is in the console I/O mode wait loop causes the console subsystem to perform the following sequence of events:

- Set the cold start flag.

- Execute the indirect command file (XXXBOO.CMD) associated with the type of cold start to be performed. This file executes the functions necessary to cold start the operating system.

(4) **AUTO/RESTART Switch Set to BOOT Sequence** – Setting the AUTO/RESTART switch to the BOOT position, while the CPU is in the console I/O mode wait loop, causes the console subsystem to perform the following sequence of events:

- Print the message ATTEMPTING SYSTEM BOOT on the console terminal.
- Examine the cold start flag. If the flag is set, indicating a cold start has previously been attempted, the message SYSTEM BOOT FAILED is printed on the console terminal. The CPU then enters the console I/O mode wait loop. However, if the cold start flag is not set, indicating that a cold start has not been attempted, the console subsystem sets the cold start flag.
- Execute the indirect command file (XXXBOO.CMD) associated with the type of cold start to be performed. This file executes the functions necessary to cold start the operating system.

(5) **Console I/O Mode Diagnostic Supervisor BOOT Command Sequence** – Typing the console I/O mode BOOT command for the diagnostic supervisor causes the console subsystem to perform the following sequence of events:

- Set the cold start flag.
- Execute the indirect command file (XXXBOO.CMD) associated with the type of cold start to be performed. This file executes the functions necessary to cold start the diagnostic supervisor.

(6) **XXXBOO.CMD Execution Sequence** – The indirect command file executed by the cold start sequence performs the appropriate functions to cold start either the operating system or diagnostic supervisor. The name of this file reflects the instructions contained within it.

The name of this file is in the following format:

```
XXXBOO.CMD
```

where XXXBOO describes the cold start device and whether the file cold starts the operating system or diagnostic supervisor. For example:

```
SUOBOO.CMD          ;  Cold starts the diagnostic supervisor from
                    ;  DUA0.
SUOBOO.CMD          ;  Cold starts the diagnostic supervisor from
                    ;  DUA0.
DEFBOO.CMD          ;  Cold starts the operating system from the
                    ;  default system device (DUA0).
```

The instructions contained within these indirect command files causes the console subsystem to perform the following sequence of events (Figure 3-1):

- Initialize the CPU to a known state and load the address of the first 64K bytes of good memory plus 200, which was located by the console program, into the stack pointer (SP which is GPR E).
- Load parameters into the general registers R0 through R5. These parameters inform the primary bootstrap program what to load (either the operating system or diagnostic supervisor), what device to load from and how to load it. The parameters that can be loaded into the general registers are shown in Table 3-1.

Table 3-1 Parameters Loaded into the General Registers at BOOT

Register	Parameters	Description
R0	<07:00>	Cold start device type code
	1 (hex)	RK06/7
	2 (hex)	RL01/2
	3 (hex)	IDC (R80, RL02, LESI)
	17 (hex)	UDA-50 (RA80, RA81, RA60)
	32 (hex)	HSC on CI
	64 (hex)	TU58
	<15:08>	Reserved for future expansion
	<31:16>	Device class dependent (RPB$WROUBVEC). UNIBUS-optional vector address; 0 implies use the default vector.
R1		Cold start device bus adapter address
	<31:04>	MBZ
	<03:00>	TR number of adapter
R2		UNIBUS bootstrap device code
	<31:18>	MBZ
	<17:00>	UNIBUS address of cold start device's CSR
R3		Cold start device controller unit number
R4		Cold start block logical block number (LBN)
R5	<09:00>	Software cold start control flags
	0 (hex)	Conversational cold start
	1 (hex)	Debug
	2 (hex)	Initial Breakpoint
	3 (hex)	Boot block
	4 (hex)	Diagnostic monitor cold start
	5 (hex)	Bootstrap breakpoint
	6 (hex)	Image header
	7 (hex)	Memory test inhibit
	8 (hex)	File name (query)
	9 (hex)	Halt before transfer

- Examine the stack pointer (SP) for the address of the first 64K bytes of good memory. This address is placed in a console RAM location to be used when the @ symbol replaces this address.

- Load the primary bootstrap program VMB.EXE into memory starting at the address in the SP, then start the program.

- Start processing instructions in VMB.EXE. VMB.EXE takes control and loads the secondary bootstrap program (either SYSBOOT.EXE for cold starting the operating system or DIAGBOOT.EXE for cold starting the diagnostic supervisor) into memory and starts it running.

3.2.1.1 VMB.EXE Operation – The primary bootstrap program VMB.EXE performs the following functions:

1. Creates a temporary System Control Block (SCB) to be used during bootstrapping.

2. Creates and stores warm start data in the operating system data structure called the Restart Parameter Block (RPB).

3. Identifies all bus adapters and memory in the hardware system configuration.

4. Tests all memory to mark each good page in a bit map.

5. Initializes the cold start device's adapter.

6. Prompts for a secondary bootstrap program file specification if the BOOT command specified the solicit flag (bit 0 in R5).

7. Locates the secondary bootstrap program file and loads that file into memory.

8. Transfers control to the secondary bootstrap program.

3.2.1.2 SYSBOOT.EXE Operation – SYSBOOT is the secondary bootstrap program for the operating system. This program loads the operating system into memory, and transfers control to its initialization code.

3.2.1.3 DIAGBOOT.EXE Operation – DIAGBOOT is the secondary bootstrap program for the diagnostic supervisor. It loads the supervisor into memory, and transfers control to its initialization code.

3.3 WARM START (RESTART)

Warm start or restart refers to restarting the operating system without reloading it into memory. When the console subsystem gains control of the VAX-11/725 following a CPU halt or power restoration, it performs the sequence of events shown in Figure 3-2.

The sequence of events performed by the console subsystem for warm starting the operating system is outlined in the following list.

The console subsystem:

1. Examines the AUTO/RESTART switch to determine what action will be taken. If the switch is in the OFF position, the CPU enters the console I/O mode wait loop. If the switch is set to the ON position the console subsystem prints the console terminal message RESTART IN PROGRESS.

2. Examines main memory to determine if data has been preserved. If no data is preserved the warm start fails.

3. Checks for the presence of the restart parameter block (RPB) in memory. If the RPB is not found, the warm start fails. If the RPB is found, then the console subsystem checks to see if it is valid (see Section 3.3.1.) If the RPB is not valid warm start fails. If the RPB is valid the console terminal loads the SP with the address of the RPB plus X200.

4. Loads the AP (GPR C) with a value that indicates the cause of the warm start.

5. Jumps to the location contained in the second longword of the RPB and starts execution of the operating system restart routine (Section 3.3.2).

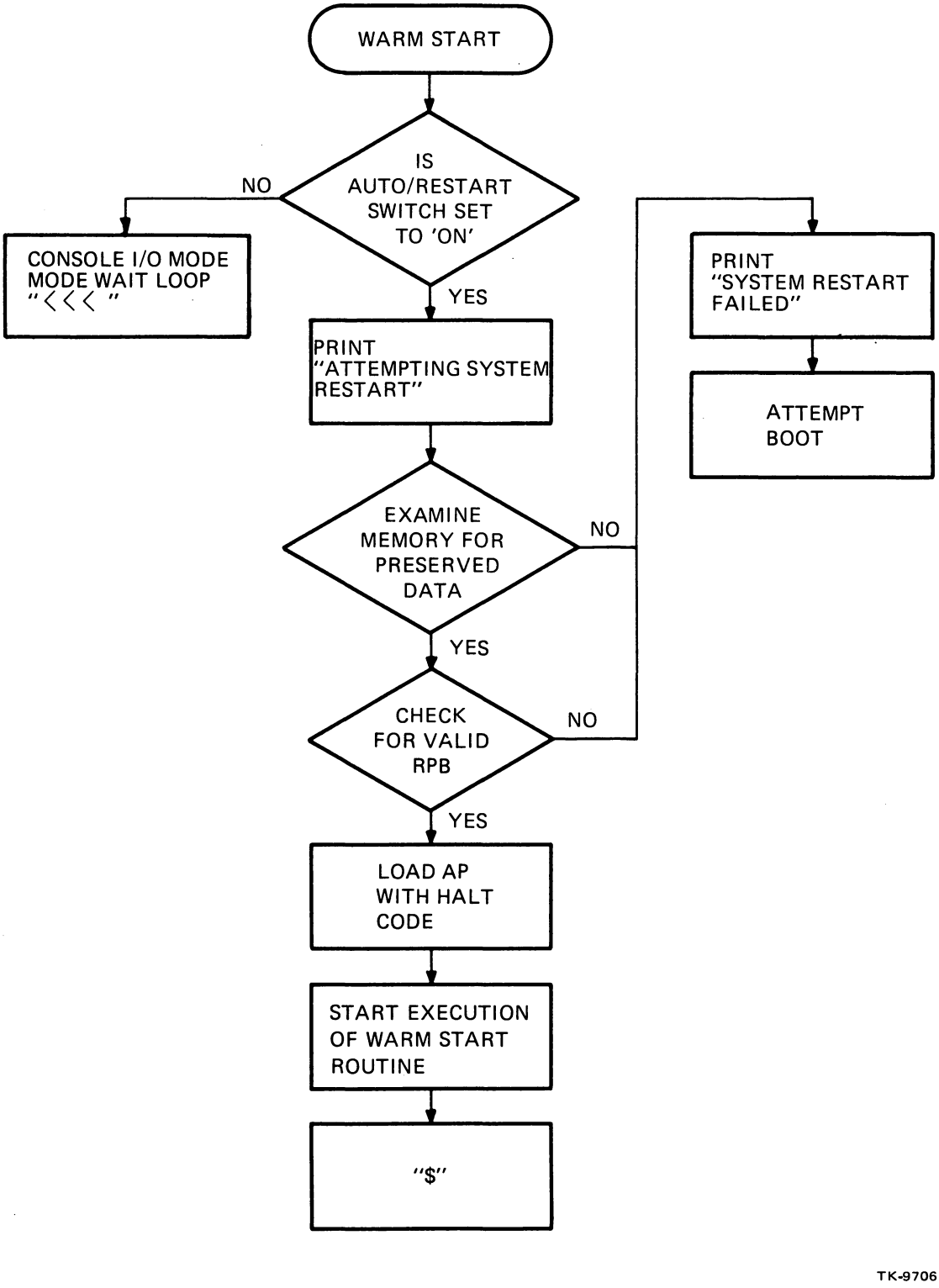

Figure 3-2 Warm Start of the Operating System

If warm start fails, the console subsystem prints the console terminal message RESTART FAILED and attempts to cold start the operating system (Section 3.2).

If the operating system warm starts successfully, it sends a message to the console subsystem, causing the console to clear the cold and warm start flags.

3.3.1 Restart Parameter Block (RPB)

The RPB (Figure 3-3) is a block of four longwords starting on a page boundary. It contains sufficient data for the operating system to warm start itself from the point at which either a power failure or software crash occurred. The console subsystem performs the following actions with the RPB during a warm start sequence:

1. Searches through physical memory for a paged, aligned longword that contains its own address.

2. Compares the first longword to the second longword of the RPB. If these longwords are equal, the RPB is not valid, and the warm start fails. However, if these longwords are unequal, the console subsystem compares the checksum of the first 31 longwords of the restart routine against the contents of the third longword in the RPB. If the checksum is valid, the RPB is valid.

3. Checks the warm start flag in the fourth longword. If the warm start flag is set, the warm start fails. If the warm start flag is not set, the console subsystem starts execution of the restart routine.

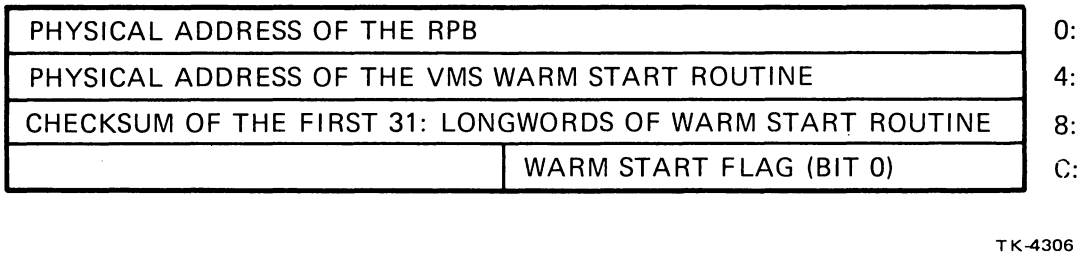

Figure 3-3 Restart Parameter Block (RPB)

3.3.2 Restart Routine

The restart routine attempts to recover from a power failure or software crash by recreating a consistent software environment. The restart routine sets the warm start flag to prevent warm start looping by displaying a WARM START IN PROGRESS message. It then restarts the operating system. If the warm start succeeds, the operating system requests the console subsystem to clear both the cold and warm start flags.

CHAPTER 4
MICRODIAGNOSTICS

4.1 INTRODUCTION

Microdiagnostics are the lowest level of diagnostics (level 5) in the VAX-11/725 diagnostic system. These diagnostics test the system at the console microprocessor and CPU microcode levels, and should be executed when higher level diagnostics (levels 4, 3, 2, 2R and 1) are unable to load or execute. They are the only level of diagnostics that provide for call-out of faulty module(s) for repair.

Digital Equipment Corporation supplies the microdiagnostics programs on a single TU58 tape cartridge, labeled 11725/730 MICRODIAG (TU58 #36), and on the diagnostic distribution media, labeled VAX-725/730 CMPLT DIAG. These microdiagnostic programs are divided into two groups of multiple sections; ENKBx and ENKCx (where x is sections A through G).

NOTE

Throughout the remainder of this chapter, individual microdiagnostic programs will be referred to as microdiagnostic sections.

The microdiagnostic group ENKBx executes from the console microprocessor RAM, and tests the hardware on the WCS (Writable Control Store) and the DAP (Data Path) modules. The console microprocessor group sections are listed in Table 4-1.

Table 4-1 ENKBx Microdiagnostic

Section	Number of Tests
ENKBA	1-8
ENKBB	9-F
ENKBC	10-1D
ENKBD	1E-26
ENKBE	27-2D
ENKBF*	2E
ENKBG*	

* Sections ENKBF and ENKBG are only executed when specifically called out by the section qualifier. These tests are not executed when the diagnose command, MIC>DI, is given with no section selected.

ENKBF tests the WCS RAM memory thoroughly using a "march" pattern, which takes approximately 5 minutes to execute.

ENKBG tests the remote diagnosis (RD) port on the M8394 board. It requires a modem loopback connector (H3248 may be used) on the RD port.

The ENKCx group executes from the WCS RAM (which writes over any system microcode that may be loaded) and uses the CPU micromachine to test the parts of the system listed in Table 4-2.

NOTE

There is no microdiagnostic for the DMF32 (COMBO board), as it is a UNIBUS device. The DMF32 contains a ROM board hard-core self-test which is evoked by either the power-up self-test (Chapter 2) or by commands from the execution of macrodiagnostics (Chapter 5).

Table 4-2 ENKCx Microdiagnostic Group Sections

Section	Number of Tests	Parts Tested
ENKCA	1-20	CPU
ENKCB	1-B	CPU
ENKCC*	1-44	Memory Controller (MCT)
ENKCD	1-10	CPU/MCT
ENKCE	1-45	Floating-Point Acc. (FPA)
ENKCF**	1-35	Internal Disk Cont. (IDC)
ENKCG**		Internal Disk Cont. (IDC)
ENKCH**		Internal Disk Cont. (IDC)

* Tests 1-33 of section ENKCC are executed automatically when the MICMON diagnose command MIC>DI is used alone. Tests 34-43 of ENKCC are executed automatically if a UNIBUS exerciser (UBE) module is present in the system. Test 44 of ENKCC thoroughly tests the main memory array boards using a march test pattern. It is executed only when explicitly called out by the command, DS>DI TE 44.

**These tests are not used by the VAX-11/725 processor.

One other diagnostic program is the microdiagnostic monitor or MICMON (ENKAA.EXE). The MICMON provides the code to load, control, and monitor all microdiagnostic sections. This program loads the microdiagnostic sections from the TU58 tape cartridge into the proper test area. It is also responsible for reporting any errors that occur during its loading, and for controlling and monitoring the microdiagnostic tests.

NOTE

The MICMON and the microdiagnostics are also loaded and executed under Customer Runnable Diagnostics (CRDs). Refer to Chapter 7.

4.2 LOADING THE MICMON

The console microprocessor loads the MICMON from the TU58 tape cartridge into the console RAM by one of two methods. The MICMON can be loaded either before or after the system microcode is loaded.

Method 1 loads the MICMON prior to loading the system microcode. Perform the procedures below:

1. Insert TU58 tape cartridge #36 (labeled 11725/730 MICRODIAG) into TU58 tape drive 1.

CAUTION
Make sure that the stabilizer foot is extended from the bottom of the system cabinet before the CPU mounting box is extended.

2. Set the AUTO-RESTART/BOOT switch to the OFF position.

3. Turn the keyswitch to the LOCAL position.

The VAX-11/725 powers up and the CPU enters the microdiagnostic monitor (MICMON), causing the console to produce a printout as shown in Example 4-1.

```
CONxyz
?40      FILE NOT FOUND   DD1:CONSOL.EXE

     CONTINUING

?27      READ ERROR       DD0:

     CONTINUING

?27      READ ERROR       DD0:

     CONTINUING

VER  X.Y  MMM.YY

MIC>
```

Example 4-1 Console Printout at Powerup as the CPU Enters MICMON

Method 2 loads the MICMON after the console microcode. Perform the following procedure:

1. Insert the TU58 tape cartridge #34 (labeled VAX 11725/730 CONSOLE) into TU58 tape drive 1.

CAUTION
Make sure that the stabilizer foot is extended from the bottom of the system cabinet before the CPU mounting box is extended.

2. Set the AUTO-RESTART/BOOT switch to the OFF position.

3. Turn the keyswitch to the LOCAL position.

   The VAX-11/725 powers up into the console I/O mode and the console terminal produces a printout as shown in Example 4-2.

4. Insert the TU58 tape cartridge #36 (labeled 11725/730 MICRODIAG) into TU58 tape drive 0 and type T/C in response to the console prompt, >>>, to load and execute the microdiagnostic monitor.

   The console terminal produces a printout as shown in Example 4-3.

Once the CPU is in MICMON, the microdiagnostics are executed by commands typed in response to the MICMON prompt (MIC>).

```
CONSOLE
VERSION XX.nn
>>>@POWER.CMD
>>>L/C CONSLE.CPU                !VERSION yy
>>>L/C/S:0800 MMIE.CPU           !VERSION yy
>>>L/C/S:0EC0 POWER.CPU          !VERSION yy
>>>S/C 0B
>>>W
>>>@CODE00.CMD
>>>L/C/S:0E00 FP.CPU             !VERSION yy
>>>L/C/S:1A00 BITFLD.CPU         !VERSION yy
>>>L/C/S:1D00 CM.CPU             !VERSION yy
>>>L/C/S:2200 BASIC.CPU          !VERSION yy
>>>L/C/S:3B00 QUEUE.CPU          !VERSION yy
>>>I
>>>
```

Example 4-2 Console Printout at Powerup as the CPU Enters the Console I/O Mode

```
>>>T/C

CONSOLE
?40     FILE NOT FOUND  DD1:ENKAA.EXE

     CONTINUING

VER  XX.XX

MIC>
```

Example 4-3 Console Printout of the CPU Entering the MICMON from the Console I/O Mode

4.3 MICRODIAGNOSTIC MONITOR COMMANDS

The following is a list of MICMON commands to execute the microdiagnostics:

DIRECTORY	RETURN	S/U
T/E	CONTINUE	INITIALIZE
REPEAT	LOAD	DEPOSIT/EXAMINE
DIAGNOSE	SHOW	SET/CLEAR
START		

These commands are defined and illustrated with examples in Appendix B.

4.4 MICRODIAGNOSTIC ERRORS

Microdiagnostic errors are detected and error messages are displayed under two conditions:

- Any time the CPU is under the control of the MICMON.
- When the individual microdiagnostic sections are executed. When errors are detected while the CPU is under control of the MICMON, due to conditions such as testing timeout or WCS parity, the MICMON prints an error message such as those shown in Examples 4-4 and 4-5. Example 4-5 is the error message MICMON prints if the SE TR command was entered.

```
MIC>DI SE ENKCC

ENKCC Vxx.xx

UPC FFFF
?xx ERROR

MIC>
```

Example 4-4 MICMON Error Printout

```
MIC>DI SE ENKCC

ENKCC V00.05
ENKCC TEST 01
ENKCC TEST 02
ENKCC TEST 03
ENKCC TEST 04
ENKCC TEST 05
ENKCC TEST 06
ENKCC TEST 07
ENKCC TEST 08

(1)     (2)    (3)    (4)        (5)        (6)     (7)        (8)
SECT    TST    ERR    EXP        REC        OTHER   MSK        MODULE
ENKCC   08     01     0000000C   0000000A   N/A     FFFFFF00   M8391

MIC>
```

Example 4-5 MICMON Error Printout After SE TR Command

The xx in the error message shown in Example 4-4 is the hex error code shown in Table 4-3.

The microdiagnostic error printout in Example 4-5 is explained below:

(1) Diagnostic section in which an error occurred

(2) Test number within that diagnostic section

(3) Error number. Within a particular test, several pieces of hardware may be tested, each by a subtest. The error number identifies the unique subtest that failed.

(4) Expected correct data

(5) Received data

(6) OTHER field will contain N/A (not applicable) or pertinent data. When data is present, the errors section of the test listing on microfiche will define the data.

(7) MASK is the error mask used to check the result. Bits set to a 1 in this mask correspond to bits in the result that are not checked.

(8) MODULE is the suspected failing module.

Table 4-3 MICMON Error Code List (Hex)

Error Code	Definition
01	Board not found
02	No test number found
03	No pass count found
04	Continue not available at this point
10	No address or no data input with command
11	No WCS image for exam or deposit routine
22	Examine or Deposit main memory reports error
23	Sequence error in WCS testing or nonexistent test
24	Timeout error in WCS testing
25	TU58 error
26	Parity error in WCS
27	Checksum error in MICMON
28	DI TE attempted with possible invalid test code loaded (do DI SE or DI BO)
29	UPC does not match on verify of 32 bit data write
2A	INIT done without file loaded
30	Checksum error in X command
31	No X command address found
32	No X command count found

CHAPTER 5
VAX-11/725 DIAGNOSTIC SUPERVISOR AND LOAD PATH

5.1 INTRODUCTION

The VAX-11/725 diagnostic supervisor may be loaded off-line from the RC25 diagnostic distribution disk, the RC25 system disk, or the TU58 tape drive. If the user disk drive is not used by VMS, the supervisor is loaded from the distribution disk while the system is operating in the user mode. However, if the user disk drive is unavailable, the supervisor is loaded from the system disk. The supervisor is loaded from the TU58 only if it cannot be loaded from the user or the system disk drives.

NOTE

The [SYSMAINT] directory on the system disk must contain the diagnostic supervisor file ENSAA.EXE before the supervisor can be booted from that disk.

Loading the supervisor from the disk drive is considered loading through the primary load path. In contrast, loading the supervisor from the TU58 is considered loading through the secondary load path. The primary path is used first, but if the system has a hardware failure in the disk subsystem (LESI or its disk drives) or in the CPU cluster, the supervisor may be loaded through the secondary load path. However, before the supervisor is loaded from the TU58, level 4 diagnostics are executed. If the level 4 diagnostics execute without error, the supervisor is executed, followed by the execution of level 3 disk subsystem diagnostics.

5.2 LOADING THE SUPERVISOR THROUGH THE PRIMARY LOAD PATH

The VAX-11/725 computer system uses low end storage interface (LESI) based disk drives to provide the primary load path for the diagnostic supervisor program.

5.2.1 Loading the Supervisor Off-Line from the Diagnostic Distribution Disk

The diagnostic supervisor is loaded off-line from the diagnostic distribution disk by booting the supervisor from that disk. To boot the diagnostic supervisor from the diagnostic distribution disk, perform the following procedures:

1. Press the EJECT switch to open the cartridge receiver door.

2. Load the diagnostic distribution disk labeled "VAX 725/730 CMPLT DIAG" into the disk drive, close the receiver door, and press the RUN switch to place the drive on line.

3. When the RUN indicator on the disk drive is continuously lit, type the following command at the console prompt (>>>) to boot the supervisor.

```
>>>B SUO
```

NOTE

When you type the command ">>>B SU0", approximately 1.75 minutes elapses before any messages are printed on the console terminal.

The console responds by printing a boot message and the diagnostic supervisor prompt (DS>) as shown in Example 5-1. The DS> prompt indicates that the diagnostic supervisor is loaded and executing.

```
>>>B SU0
>>>@DD1:SU0B00.CMD
>>>I
>>>D/G/L 0 11
>>>D/G 1 3
>>>D/G 2 3F468
>>>D/G 3 0
>>>D/G 4 0
>>>D/G 5 10
>>>D/G D 0
>>>E SP
G 0000000E 00000200
>>>L/P/S:@ VMB.EXE
>>>S @

DIAGNOSTIC SUPERVISOR. ZZ-ENSAA-x.x-xxx  DD-MMM-YYYY HH:MM:SS
DS>
```

Example 5-1 Booting the Diagnostic Supervisor from the RC25 Diagnostic Distribution Disk

5.2.2 Loading the Diagnostic Supervisor Off-Line from the System Disk

The diagnostic supervisor is loaded off-line from the system disk by booting the supervisor from that disk. To boot the diagnostic supervisor from the system disk, perform the following procedures:

1. Press the RUN switch to place the system drive on-line.

2. When the RUN indicator on the disk drive is continuously lit, type the following command at the console prompt (>>>) to boot the supervisor.

   ```
   >>>B SU1
   ```

   The console responds by printing a boot message and the diagnostic supervisor prompt (DS>) as shown in Example 5-2. The DS> prompt indicates the diagnostic supervisor is loaded and executing.

```
>>>B SU1
>>>@DD1:SU1B00.CMD
>>>I
>>>D/G/L 0 11
>>>D/G 1 3
>>>D/G 2 3F468
>>>D/G 3 0
>>>D/G 4 0
>>>D/G 5 10
>>>D/G D 0
>>>E SP
G 0000000E 00000200
>>>L/P/S:@ VMB.EXE
>>>S @

DIAGNOSTIC SUPERVISOR. ZZ-ENSAA-x.x-xxx  DD-MMM-YYYY HH:MM:SS
DS>
```

Example 5-2 Booting the Diagnostic Supervisor from the RC25 System Disk

5.2.3 Loading the Supervisor On-Line from the Distribution Disk

To load and execute the diagnostic supervisor in the on-line mode (with VMS) from the distribution disk, perform the following procedures:

1. Press the EJECT switch to open the cartridge receiver door.

2. Load the diagnostic distribution disk labeled "VAX 725/730 CMPLT DIAG" into the disk drive, close the receiver door, and press the RUN switch to place the drive on-line.

3. When the RUN indicator on the disk drive is continuously lit, type the following command at the VMS prompt ($) to mount the distribution disk.

   ```
   $MOUNT DUA0:  CRDPACK
   ```

   The terminal responds with the following message to indicate that the disk is mounted properly.

   ```
   %MOUNT-I-MOUNTED, CRDPACK        mounted on DUA0:

   $
   ```

4. In response to the VMS prompt, type the following command to set default to the directory containing the diagnostic supervisor program (ENSAA.EXE).

   ```
   $ SET DEFAULT DUA0:[SYSMAINT]
   ```

5. Type the following command to load and execute the diagnostic supervisor.

   ```
   $ RUN ENSAA.EXE
   ```

When the diagnostic supervisor starts, it prints the following message:

```
DIAGNOSTIC SUPERVISOR. ZZ-ENSAA-x.x-xxx  DD-MMM-YYYY HH:MM:SS
DS>
```

5.2.4 Loading the Supervisor On-Line from the System Disk

To load and execute the diagnostic supervisor in the on-line mode (with VMS) from the system disk, perform the following procedures:

1. In response to the VMS prompt, type the following command to set default to the directory containing the diagnostic supervisor program (ENSAA.EXE):

   ```
   $ SET DEFAULT SYS$MAINTENANCE
   ```

2. Type the following command to load and execute the diagnostic supervisor:

   ```
   $ RUN ENSAA.EXE
   ```

   When the diagnostic supervisor starts, it prints the following message:

```
DIAGNOSTIC SUPERVISOR. ZZ-ENSAA-x.x-xxx  DD-MMM-YYYY HH:MM:SS
DS>
```

5.3 LOADING THE DIAGNOSTIC SUPERVISOR THROUGH THE SECONDARY LOAD PATH

If the diagnostic supervisor cannot be booted from the distribution or system disk, use the secondary load path (TU58 tape drive) to load programs which test the hardware in the primary load path. These programs are called load path diagnostics.

Digital Equipment Corporation supplies load path diagnostics for the VAX-11/725 system on the following TU58 cassette tapes:

- VAX 11 HARDCORE INSTR (TU58 #7)
- 11725/730 DIAG SUPER (TU58 #35)
- VAX RC25 SUBSYSTEMS (TU58 #58)

TU58 tape cartridge #7, labeled VAX 11 HARDCORE INSTR, is inserted into a TU58 tape drive and the level 4 diagnostic (EVKAA.EXE) is loaded and executed using the console command language. This diagnostic program functionally verifies the kernel instruction set used by the diagnostic supervisor. If the level 4 diagnostic program executes without error, the diagnostic supervisor (ENSAA.EXE) is loaded and executed from the TU58 tape cartridge #35, labeled 11725/730 DIAG SUPER. The functional integrity of the disk controller (LESI) and its attached disk drives are then tested with the disk subsystem diagnostic programs that reside on TU58 tape cartridge #39, labeled VAX 725/730 IDC DIAG and on TU58 tape cartridge #58, labeled VAX RC25 SUBSYSTEMS.

The diagnostic programs on the VAX RC25 SUBSYSTEMS (TU58) tape cartridge are listed below:

```
EVRMA.EXE       ! VAX RC25 Disk Exerciser
EVRMA.HLP       ! Help File for the EVRMA Diagnostic
EVRMB.EXE       ! VAX RC25 Front End Test
EVRMB.HLP       ! Help File for the EVRMB Diagnostic
EVRMC.EXE       ! VAX RC25 Level 3 Disk Formatter
EVRMC.HLP       ! Help File for the EVRMC Diagnostic
```

5.3.1 Executing Diagnostics for the Load Path

The level 4 diagnostic is loaded from the tape cartridge and executed using the console command language.

To load the level 4 diagnostic from the tape cartridge, follow the procedures below:

1. Insert the TU58 tape cartridge #7, labeled VAX 11 HARDCORE INSTR, into the TU58 tape drive 0.

2. In response to the console prompt, >>>, type the following commands to load and execute the level 4 diagnostic EVKAA.EXE:

```
>>>I
>>>D/P/L FE00 0
>>>L/P/S:0 DD0:EVKAA.EXE
>>>S 200
```

   If EVKAA executes successfuly, at the end of each 16 passes the console produces a printout as shown below and rings the terminal bell.

```
EVKAA Vn.n PASS #nn DONE!
EVKAA Vn.n PASS #nn DONE!
EVKAA Vn.n PASS #nn DONE!
```

   This program continues to execute and produce the above printout until a CTRL/P is typed. At that time the system returns to the console I/O mode and prints the console prompt, >>>.

   If the level 4 diagnostic executes without error, the diagnostic supervisor (ENSAA.EXE) is loaded from TU58 #35 and executed using the console command language. Then, the disk subsystem diagnostic is loaded and executed.

3. Remove TU58 #7 from TU58 tape drive 0.

4. Insert TU58 tape cartridge #35, labeled 11725/730 DIAG SUPER, into TU58 tape drive 0.

5. In response to the console prompt, type the following commands to load and execute the diagnostic supervisor from the tape cartridge:

```
>>>I
>>>L/P/S:FE00 DD0:ENSAA.EXE
>>>S 10000
```

When the diagnostic supervisor starts, it prints the following message:

```
DIAGNOSTIC SUPERVISOR. ZZ ENSAA-x.x-xx  DD-MMM-YYYY HH:MM:SS
DS>
```

6. Use the ATTACH command to define the system hardware configuration, as shown below:

```
DS>ATTACH KA730 HUB KA0 YES 03FF 1 2048 NO NO
DS>ATTACH DW730 HUB DW0
DS>ATTACH LESI DW0 DAA 772150 0154 05 02
DS>ATTACH RC25 DAA DAA0
DS>ATTACH RCF25 DAA DAA1
```

7. Select the disk drives for testing with the following command:

```
DS>SELECT DAA0, DAA1
```

8. Remove TU58 #35 from TU58 tape drive 0.

9. Insert the TU58 tape cartridge #58, labeled "VAX RC25 SUBSYSTEMS," into TU58 tape drive 0.

10. Use the HELP command to obtain system set-up information on the diagnostic programs EVRMA.EXE, EVRMB.EXE, and EVRMC.EXE.

```
DS>HELP EVRMA
DS>HELP EVRMB
DS>HELP EVRMC
```

11. Perform the set-up procedures, then run the diagnostics with the RUN command:

```
DS>RUN EVRMA.EXE
DS>RUN EVRMB.EXE
```

5.4 DIAGNOSTIC SUPERVISOR COMMANDS

Commands for the diagnostic supervisor are obtained by using the HELP utility. If the supervisor was loaded from a disk device, the HELP utility is invoked by typing the HELP command (for example, DS> HELP). This command used in conjunction with a particular topic (for example, DS> HELP SHOW) displays general diagnostic environment and supervisor command information.

Most diagnostics programs also have HELP files to assist in their execution. These HELP files have the same filename as the diagnostic program file, but the file extension is named .HLP (EVRMA.HLP). Help files are accessed by typing the HELP command followed by the filename (DS> HELP EVRMA). Using this command in conjunction with the diagnostic program name displays diagnostic information specific to that diagnostic.

System configuration is bypassed to the diagnostic supervisor either with individually typed ATTACH commands or by executing the autosizer program on the diagnostic distribution disk.

The autosizer program (EVSBA.EXE) sizes the system on which it is executed (under the diagnostic supervisor in the console mode) and passes the system configuration to the diagnostic supervisor with a single command.

To execute the autosizer, perform the following procedures:

1. Load and execute the autosizer in its self-test mode by typing the following commands:

```
DS>SET FLAG QUICK
DS>RUN EVSBA/SECTION:SELFTEST
```

   The autosizer is loaded and executed and produces the prompt: COMMAND?. At this prompt, type the following command to instruct the autosizer to determine the hardware configuration of the system. Then, pass this information to the diagnostic supervisor:

```
COMMAND?  SIZE
```

   The autosizer prints ATTACH commands as they are passed to the diagnostic supervisor (Example 5-3).

2. Type ATTACH to the COMMAND? prompt to actually build the database in the diagnostic supervisor.

3. Type EXIT to the COMMAND? prompt to exit the autosizer program. The autosizer prints the completion message shown below (if no errors were encountered) on exiting. It then returns console control to the diagnostic supervisor, DS>.

```
COMMAND?  EXIT

...End of run, 0 errors detected, pass count is 1,
time is dd-mmm-yyyy HH:MM:SS
DS>
```

4. Select the devices to be tested. Then use the RUN command to load and execute the diagnostic program.

See Chapters 4 and 5 of the *VAX Diagnostic System User's Guide* for details on the diagnostic supervisor commands.

```
! AUTOMATIC SIZING PROGRAM.
! CONFIGURATION FILE FOR SYSTEM.
! COMPUTER GENERATED CPU TYPE 3 MICROCODE REVE LEVEL = xx
! NUMERIC VALUES WITH A LEADING ZERO ARE ASSIGNED.
! FILE VALID ONLY FOR STANDARD HARDWARE CONFIGURATION.
! TIME IS dd-mmm-yyyy hh:mm:ss
!
! ***QUICK FLAG SET. NO CHECKS MADE FOR TERMINALS ON DZ11'S***
!
! DEFINE PROCESSOR...
!
DS> ATTACH KA730 HUB KA0 YES 03FF 1 0148 NO NO
!
! DEFINE UNIBUS ADAPTERS...
!
DS> ATTACH DW730 HUB DW0
!
DS> ATTACH LESI DW0 DAA 772150 0154 05 02
DS> ATTACH RC25 DAA DAA0
DS> ATTACH RCF25 DAA DAA1
DS> ATTACH DMF32S DW0 XGA0 760340 0300 05 NONE
DS> ATTACH DMF32A DW0 TXA 760340 0300 05 0377 9600 INTERNAL YES
DS> ATTACH DMF32P DW0 LCA 760340 0300 05 OTHER
COMMAND?
```

Example 5-3 Typical Autosizer Printout of a VAX-11/725 System

CHAPTER 6
EXECUTING THE LEVEL 4 DIAGNOSTIC PROGRAM

6.1 LOADING AND EXECUTING THE LEVEL 4 DIAGNOSTIC PROGRAM

Only one level 4 diagnostic program applies to the VAX-11/725 – EVKAA, a hardcore instruction test.

DIGITAL ships this diagnostic program for the VAX-11/725 on the following diagnostic distribution media:

- TU58, #7 labeled VAX 11 HARDCORE INSTR
- RC25, #1 labeled VAX 725/730 CMPLT DIAG

The method for loading the level 4 diagnostic into memory from the TU58 tape cartridge is different from loading it from the disk pack.

The level 4 diagnostic is loaded into memory from the TU58 tape cartridge by using the console command language. However, the diagnostic is loaded into memory from the disk pack with the diagnostic supervisor.

After the diagnostic is loaded into memory, it is executed from the console I/O mode with the START command.

6.1.1 Loading EVKAA from the TU58 Tape Cartridge

The level 4 diagnostic EVKAA is loaded and executed with the console command language as shown in the following steps:

1. Insert the TU58 tape cartridge #11, labeled VAX 11 HARDCORE INSTR, into the TU58 drive 0.

2. At the console prompt, >>>, type the following commands to load and execute EVKAA:

```
>>>I                          ! Initializes the CPU.
>>>D/P/L FE00 0               ! Zero memory location FE00.
>>>L/P/S:0 DD0:EVKAA.EXE      ! Loads the diagnostic into memory.
>>>
```

After EVKAA is loaded into memory, it is executed using the procedure in Section 6.1.3.

6.1.2 Loading and Executing EVKAA from the Disk Pack

EVKAA is loaded and executed from the disk pack as described in the following:

1. Boot the diagnostic supervisor from the disk pack as described in Chapter 5.

2. When the DS> prompt is displayed, load EVKAA from the disk pack into memory by using the LOAD command:

```
DS>LOAD EVKAA                ! Loads the diagnostic into memory.
```

3. Exit from the diagnostic supervisor to the console I/O mode by typing the EXIT command:

```
DS>EXIT                      ! Terminates the diagnostic
                             ! supervisor.
?02 PC=0001A5D8
>>>                          ! Enters the console I/O mode.
```

6.1.3 Executing EVKAA

After EVKAA is loaded into memory it is executed at memory location 200 with the following command:

```
>>>S 200                    ! Executes the diagnostic program
                            ! at memory location 200.
```

If EVKAA executes successfully, at the end of each 100 passes the console produces one line of the printout shown below and rings the terminal bell.

```
EVKAA Vx.x PASS #nn DONE!
EVKAA Vx.x PASS #nn DONE!
EVKAA Vx.x PASS #nn DONE!
```

This program continues to execute and produce the above printout until a CTRL/P is typed. At that time the system returns to the console I/O mode and prints the console prompt, >>>.

If EVKAA does not execute successfully, it prints the error message shown below and returns control to the console I/O mode.

```
???ERROR TEST #nn, SUBTEST #nn (instruction) FAILED
(one line description of failure)
EXPECTED DATA: XXXXXXXX
RECEIVED DATA: XXXXXXXX

?06 00009301
>>>
```

6.2 EVKAA ERROR INTERPRETATION AND LOOP CONTROL

When EVKAA detects an error and the halt code is 06, it indicates that the processor executed a HALT instruction at the error. Other halt codes indicate that the program is not executing properly. See Table 2-1 for a list of halt codes and their meanings.

If the error message was printed and the halt code is 06, the user can look in the listing for the algorithm used to determine the failure.

NOTE

If the base address for the program is not 0, the user must find the base address and subtract it from the PC in order to find the corresponding HALT instruction in the listing.

General register 10 contains the base address and is examined by typing the following command:

```
>>>E/G 10                          ! Examines register R10.
   G   00000010    00000000        ! Base address is 00000000.
```

Consider the case where the user executes the hardcore instruction test and gets the console output in Example 6-1.

```
>>>I                               ! Initializes the CPU.
>>>D/P/L FE00 0                    ! Zero memory Location FE00.
>>>L/P/S:0 DD0:EVKAA.EXE           ! Loads the diagnostic into memory.
>>>S 200                           ! Starts the diagnostic program.
???ERROR TEST #nn, SUBTEST #nn (instruction) FAILED
(one line description of failure)
EXPECTED DATA: XXXXXXXX
RECEIVED DATA: XXXXXXXX

?06 00009301
>>>
```

Example 6-1 HALT in Hardcore Instruction Test

Look up the location 9300 (the PC minus 1) in the program listing. Read the test description and analyze the code. If the fault is a hard error, the user can cause the program to loop (on the next pass) by replacing the HALT instruction with a NOP instruction.

NOTE

The user should examine the location first to make sure that it contains a HALT instruction.

By replacing the HALT with a NOP, the error message printing is cancelled.

Example 6-2 shows an EXAMINE and DEPOSIT to the location that contains the HALT instruction.

```
>>>E/P/B 9300                      ! Examines location 9300.
   P    00009300    00
>>>D/P/B 9300   01                 ! Deposits 01 in the byte at.
                                   ! location 9300.
>>>C                               ! Continue. The program should
                                   ! loop on error.
```

Example 6-2 Level 4 Diagnostic Program Set Up to Loop on Hard Error

However, if the error is intermittent, the program may progress out of the loop on the first success. In order for the program to loop every time, the user must find the instruction which checks for success or failure and replace it with a NOP instruction (see Example 6-3).

```
92C1    12591    T12_S55:
92C1    12592             MOVL    #^X55,SUBNUM
92CC    12593    5$:      MOVW    #-1,TEMP0
92D5    12594             BISPSW  #15
92D7    12595             TSTW    TEMP0
92DD    12596             MOVPSL  R1
92DF    12597             BICL    #NZVC,R1
92E6    12598             MOVL    #8,R0
92E9    12599             XORL3   R0,R1,R2          ; Compare expected
                                                    ; and received data.
92ED    12600             BEQL    10$               ; Test for success.
92EF    12601             MOVAB   B^1045$,^X44(R11)
92F4    12602             MOVW    #1,^X40(R11)
92F8    12603             TSTL    (R11)
92FA    12604             BEQL    1045$
92FC    12605             JSB     (R12)
92FE    12606             BRB     10$               ; Branch to
                                                    ; next test.
9300    12607    1045$:   HALT                      ; Halt.
9301    12608             BRB     5$                ; Loop PC printed.
9303    12609    10$:     CLRL    R3                ; Next part of test.
```

Example 6-3 Level 4 Program Listing Sample EVKAA, Test 12, Subtest 55

In this case, the BELQ 10$ instruction at line 12600 checks for success of the test operation. Also, the BRB 10$ instruction at line 12606 branches to the next test after completion of another operation. Two bytes correspond to each instruction. Both the instruction and displacement in the instructions are replaced with two NOP instructions (0101) as shown in Example 6-4.

```
>>>E/P/W 92ED                    ! Examine the two bytes at 92ED.
   P    000092ED      1413
>>>D/P/W 92ED 0101               ! Deposit two NOP codes at 92ED.
>>>E/P/W 92FE                    ! Examine the two bytes at 92FE.
   P    000092FE      0311
>>>D/P/W 92FE 0101               ! Deposit two NOP codes at 92FE.
>>>E/P/B 9300                    ! Examine the HALT location.
   P    00009300      00
>>>D/P/B 9300 00                 ! Replace the HALT with a NOP.
>>>C                            ! Continue. The program should loop
                                 ! on the error.
```

Example 6-4 Setting Up a Loop on an Intermittent Error

CHAPTER 7
CUSTOMER RUNNABLE DIAGNOSTICS

7.1 INTRODUCTION

Customer runnable diagnostics (CRD) is a special control program that simplifies the execution of micro- and macrodiagnostics in the VAX-11 diagnostic system on the VAX-11/725 system.

The CRD package operates in two modes: AUTO and MENU. In the AUTO mode, off-line bottom-up testing is performed on the system without user intervention. In the MENU mode, the user is offered, with menu prompts, the opportunity to further off-line test (beyond AUTO mode testing) any device, including added options, on the system.

NOTE

Both TU58 tape drives, the removable media disk drive, and the console terminal are used for CRD, so they must be functional.

For more information on customer runnable diagnostics, refer to the documentation entitled *Customer Runnable Diagnostics User Guide.*

7.2 CRD MINIMUM SYSTEM CONFIGURATION

The minimum VAX-11/725 system requirements on which CRD testing is performed consist of the following:

1. VAX-11/725 CPU cluster with 1M bytes of memory
2. Disk controller (LESI)
3. Dual TU58 tape drive
4. Console terminal
5. RC25 disk drive
6. TU58, #34 "VAX 11725/730 CONSOLE" tape cartridge
7. TU58, #36 "VAX 11725/730 MICRODIAG" tape cartridge
8. RC25, #1 "VAX 725/730 CMPLT DIAG" disk cartridge

7.3 AUTO MODE

CRD AUTO mode performs the following tests in the order shown:

1. Test the CPU, memory, and FPA with microdiagnostics.
2. Test the CPU with level 4 diagnostics.
3. Boot the Diagnostic Supervisor from the disk drive.
4. Test the LESI and RC25 disk drives with macrodiagnostics.

NOTE

The DMF32 module is not tested in AUTO Mode; this testing must be performed manually in the MENU Mode. Refer to Paragraph 7.4 for this procedure.

Once AUTO mode is invoked, device testing is completed without user intervention in approximately 15 minutes. An exception to this would be when the system is improperly set up. In this instance, the AUTO mode program displays an informational message and halts (Example 7-1). The operator has a chance to properly prepare the system, and resume execution of the AUTO mode by typing a <CR>.

```
***       BEGIN VAX-11/730,725 AUTO TEST      ***
VERSION x.x RUN TIME APPROXIMATELY 15:00 MINUTES

CPU PART 1          TESTING STARTED -- RUN TIME = 3:30     PASSED

MEMORY              TESTING STARTED -- RUN TIME = 1:30     PASSED

CPU PART 2          TESTING STARTED -- RUN TIME = 0:45     PASSED

FPA                 TESTING STARTED -- RUN TIME = 1:45     PASSED

CPU PART 3          TESTING STARTED -- RUN TIME = 4.00

** PROPER SETUP REQUIRED -- TYPE <CR> TO CONTINUE **
** CHECK FOR RC25 DIAGNOSTIC PACK 'CRDPACK'        **
** INSERTED AND SPINNING IN DRIVE DAA0:            **
** IF SETUP CORRECT, POSSIBLE RC25 DRIVE PROBLEMS **

RC25 PART 1    DAA0 TESTING STARTED -- RUN TIME = 0:30     PASSED

RC25 PART 2      DAA0 TESTING STARTED -- RUN TIME 1:00     PASSED

RCF25          DAA1 TESTING STARTED -- RUN TIME = 1:00     PASSED

AUTO TEST COMPLETE -- NO ERRORS
**** END OF VAX-11/730,725 AUTO TEST ****

MENU MODE is entered at this point.
```

Example 7-1 AUTO Mode Testing with Improper System Setup

Upon successfully completion of AUTO mode CRD enters MENU mode (see Section 7.4).

7.3.1 AUTO Mode Evocation

AUTO mode is invoked after system preparation is performed. The system is prepared for testing and AUTO mode is invoked by performing the following procedures:

1. Insert TU58 tape cartridge #34, labeled VAX 11725/730 CONSOLE, into TU58 tape drive 1.

2. Insert TU58 tape cartridge #36, labeled 11725/730 MICRODIAG, into TU58 tape drive 0.

3. Press the EJECT switch to open the cartridge receiver door.

4. Load the diagnostic distribution disk labeled VAX 725/730 CMPLT DIAG into the disk drive, close the receiver door, and press the RUN switch to place the drive on-line.

5. When the RUN indicator on the disk drive is continuously lit, proceed with the next step. This drive and the system disk drive may or may not be WRITE PROTECTed.

NOTE

CRD AUTO mode performs nondestructive disk testing on VAX/VMS disk cartridges. If the disk which is inserted into the disk drive is formatted under systems other than VAX/VMS, WRITE PROTECT the disk to protect that data.

6. Obtain the console prompt, >>>, as described in Chapter 2.

NOTE

After the console prints VERSION xx.yy, a CTRL/C may be typed since all microcode needed to execute CRDs is loaded.

7. Invoke AUTO mode by typing T to the console prompt as shown below:

```
>>>T <CR>
```

NOTE

When the command, >>>T, is typed, there is a time span of approximately 35 seconds before the first message is printed on the console terminal.

If no errors are encountered, the console terminal produces a printout similar to that in Example 7-2.

7.3.2 AUTO Mode Messages

Messages normally seen while executing the diagnostic programs are replaced in the AUTO mode with plain English messages. These consist of progress messages, warning messages, and error messages.

Progress messages are printed to assure the user that the AUTO mode CRD is progressing properly. These messages are printed when device testing is begun and completed (Example 7-2).

```
****       BEGIN VAX-11/730,725 AUTO TEST      ****
VERSION x.x RUN TIME APPROXIMATELY 15:00 MINUTES

CPU PART 1              TESTING STARTED -- RUN TIME = 3:30     PASSED

MEMORY                  TESTING STARTED -- RUN TIME = 1:30     PASSED

CPU PART 2              TESTING STARTED -- RUN TIME = 0:45     PASSED

FPA                     TESTING STARTED -- RUN TIME = 1:45
FPA NOT AVAILABLE
 PASSED

CPU PART 3              TESTING STARTED -- RUN TIME = 4:00     PASSED

RC25 PART 1     DAA0 TESTING STARTED -- RUN TIME = 0:30     PASSED

RC25 PART 2     DAA0 TESTING STARTED -- RUN TIME = 1:00     PASSED

RCF25           DAA1 TESTING STARTED -- RUN TIME = 1:00     PASSED

AUTO TEST COMPLETE -- NO ERRORS
**** END OF VAX-11/730,725 AUTO TEST ****

MENU MODE is entered at this point.
```

Example 7-2 AUTO Mode Testing Sequence (No Errors Encountered)

Warning messages are printed to inform the user that the system is improperly set up (Example 7-1). These messages are printed in the following situations:

1. The user disk drive does not contain the proper diagnostic disk cartridge.
2. The user (removable) and system (fixed) disk drives are off-line (RUN switch not activated).
3. The system disk drive is WRITE PROTECTed.
4. The diagnostic tape in the TU58 tape drive has missing or corrupted diagnostic program files.

After printing warning messages, AUTO mode handles each of the above situations differently. AUTO mode pauses for the first two situations above to allow the user to properly set up the disk drive (Example 7-1). However, for the third and fourth situations, AUTO mode continues testing after printing a warning message indicating that testing is incomplete. AUTO mode executes the appropriate macrodiagnostic in NOWRITE mode for the third situation (Example 7-3). AUTO mode handles the fourth situation as an error and aborts further testing (Example 7-4).

Error messages are printed when an error occurs during diagnostic testing. These messages identify the device that caused the error and the next action to be taken by the user. For more error information, refer to Section 7.3.3.

```
****      BEGIN VAX-11/730,725 AUTO TEST     ****
VERSION x.x RUN TIME APPROXIMATELY 15:00 MINUTES

CPU PART 1           TESTING STARTED -- RUN TIME = 3:30    PASSED

MEMORY               TESTING STARTED -- RUN TIME = 1:30    PASSED

CPU PART 2           TESTING STARTED -- RUN TIME = 0:45    PASSED

FPA                  TESTING STARTED -- RUN TIME = 1:45    PASSED

CPU PART 3           TESTING STARTED -- RUN TIME = 4:00    PASSED

RC25 PART 1    DAA0 TESTING STARTED -- RUN TIME = 0:30    PASSED

RC25 PART 2    DAA0 TESTING STARTED -- RUN TIME = 1:00    PASSED

RCF25          DAA1 TESTING STARTED -- RUN TIME = 1:00

**        UNIT IS WRITE-PROTECTED      **
** DAA1: DISK DRIVE TESTING INCOMPLETE **

AUTO TEST INCOMPLETE -- NO ERRORS
CHECK DEVICE PREPARATION -- IF OK, THEN CALL FIELD SERVICE
**** END OF VAX-11/730,725 AUTO TEST ****
```

Example 7-3 AUTO Mode Testing Sequence (DAA1 WRITE-PROTECTed)

```
****      BEGIN VAX-11/730,725 AUTO TEST     ****
VERSION x.x RUN TIME APPROXIMATELY 15:00 MINUTES


CPU PART 1           TESTING STARTED -- RUN TIME = 3:30    PASSED

MEMORY               TESTING STARTED -- RUN TIME = 1:30    PASSED

CPU PART 2           TESTING STARTED -- RUN TIME = 0:45    PASSED

** PROPER SETUP REQUIRED -- CANNOT CONTINUE       **
** CHECK DIAGNOSTIC TAPE INSERTED IN TU58 DRIVE 0 **
** IF SET CORRECT, POSSIBLE TAPE OR DRIVE PROBLEM *

AUTO TEST ABORTED
**** END OF VAX-11/730, 725 AUTO TEST ****
```

Example 7-4 AUTO Mode Testing Sequence (Missing a Microdiagnostic Program File)

7.3.3 AUTO Mode Errors

Errors detected during AUTO mode cause an error message to be printed and the current diagnostic test to be aborted. Also, depending on which diagnostic test detected the error, AUTO mode aborts. If errors are detected during microdiagnostic or level 4 diagnostic testing, the program that detected the error along with the remaining test programs are aborted (Example 7-5). Also, if microdiagnostics attempt to test a device that is missing (other than the FPA), AUTO mode is aborted. By contrast, errors detected during macrodiagnostic testing of a device abort only that program and not any other programs that test the device. AUTO mode then proceeds with execution of the remaining diagnostic programs (Example 7-6). However, there are some situations where AUTO mode will abort:

- Errors encountered in the diagnostic disk drive
- Improper system setup (for example, if the programs to be executed cannot be found on the diagnostic disk)

When AUTO mode testing is aborted due to microdiagnostic errors or AUTO mode completes with macrodiagnostic errors, the user has the option of restarting the AUTO mode testing or returning to the console I/O mode (Example 7-5). This is announced to the user after completion of testing.

```
****       BEGIN VAX-11/730,725 AUTO TEST      ****
VERSION x.x RUN TIME APPROXIMATELY 15:00 MINUTES

CPU PART 1          TESTING STARTED -- RUN TIME = 3:30     PASSED

MEMORY              TESTING STARTED -- RUN TIME = 1:30     PASSED

CPU PART 2          TESTING STARTED -- RUN TIME = 0:45     PASSED

FPA                 TESTING STARTED -- RUN TIME = 1:45     PASSED

CPU PART 3          TESTING STARTED -- RUN TIME = 4:00    *FAILED*

AUTO TEST ABORTED -- CPU BAD -- CALL FIELD SERVICE

ENTER 1 TO RETURN TO CONSOLE, 2 TO RESTART AUTO TEST 1

**** END OF VAX-11/730,725 AUTO TEST ****
```

Example 7-5 AUTO Mode Testing Sequence (Error Detected during Microdiagnostic Testing)

7.4 MENU MODE

MENU mode is invoked for system testing by one of the following methods:

1. Automatically at the successful completion of AUTO mode
2. Manually with the console I/O mode command, T/M
3. Manually with the diagnostic supervisor command, CRD

In the first of the above methods, MENU mode is invoked once AUTO mode testing has successfully completed. In this method, MENU mode testing is started without operator intervention, but further system preparation may be required if options are added to the minimum system configuration.

```
****       BEGIN VAX-11/730,725 AUTO TEST      ****
VERSION x.x RUN TIME APPROXIMATELY 15:00 MINUTES

CPU PART 1            TESTING STARTED -- RUN TIME = 3:30     PASSED

MEMORY                TESTING STARTED -- RUN TIME = 1:30     PASSED

CPU PART 2            TESTING STARTED -- RUN TIME = 0:45     PASSED

FPA                   TESTING STARTED -- RUN TIME = 1:30     PASSED

CPU PART 3            TESTING STARTED -- RUN TIME = 4:00     PASSED

RC25 PART 1     DAA0  TESTING STARTED -- RUN TIME = 0:30     PASSED

RC25 PART 2     DAA0  TESTING STARTED -- RUN TIME = 1:00    *FAILED*

RCF25           DAA1  TESTING STARTED -- RUN TIME = 1:00     PASSED

AUTO TEST COMPLETE -- DAA0 BAD -- CALL FIELD SERVICE

**** END OF VAX-11/730,725 AUTO TEST ****
```

Example 7-6 AUTO Mode Testing Sequence (Error Detected during Microdiagnostic Testing)

7.4.1 MENU Mode Evocation

MENU mode is invoked after the "base" system is fully prepared. Prepare the "base" system by performing the following steps:

1. Insert TU58 tape cartridge #34, labeled VAX 11725/730 CONSOLE, into the TU58 tape drive 1.
2. Insert TU58 tape cartridge #36, labeled 11725/730 MICRODIAG, into the TU58 tape drive 0.
3. Press the EJECT switch to open the cartridge receiver door.
4. Load the diagnostic distribution disk labeled VAX 725/730 CMPLT DIAG into the disk drive, close the receiver door, and press the RUN switch to place the drive on-line.
5. When the RUN indicator on the disk drive is continuously lit, proceed to the next step. This drive and the system disk drive may or may not be WRITE-PROTECTed.

NOTE

CRD MENU mode performs nondestructive disk testing on VAX/VMS disk cartridges. If the disk that is inserted into the disk drive is formatted under systems other than VAX/VMS, WRITE PROTECT the disk to protect the data.

6. Invoke MENU by one of the following methods:

Console I/O Mode Method

a. Obtain the console prompt, >>>, as described in Chapter 2.

b. Type T/M in response to the console prompt as shown below:

```
>>>T/M<CR>
```

NOTE

When the command >>>T/M is typed there is a time span of approximately 1.5 minutes before the first message is printed on the console terminal.

Diagnostic Supervisor Method

a. Obtain the diagnostic supervisor prompt DS> as described in Chapter 5.

b. Type CRD in response to the supervisor prompt as shown below:

```
DS>CRD<CR>
```

Whichever method is used to evoke CRD MENU mode, user control is passed to the Main Menu which produces a printout similar to Example 7-7.

```
**** BEGIN VAX-11/730,725 MENU TEST ****

Type the CTRL key and the C key (together) at any time
to interrupt Menu Test Processing...

VAX-11/730,725 HARDWARE IDENTIFICATION -- RUN TIME = 00:30 Minutes
PLEASE WAIT...

VAX-11/730,725 HARDWARE IDENTIFICATION COMPLETE


MAIN MENU -- Functional Test


    A = Exit Menu Test

    B = Print list of identified system hardware and support status

    C = Print preparation required for test of supported hardware

    D = Select and test hardware from TEST MENU

    E = TEST ALL supported hardware

    Type one of the above (for example, A), and press RETURN,

    Enter MAIN MENU choice:
```

Example 7-7 MENU Mode Main Menu

In Main Menu, selections are provided to allow the user to perform the following functions:

- Exit from MENU mode.
- Print a list of devices on the system and determine whether they are supported.
- Print the preparation required to test the available devices.
- Select either a single device or all devices on the system for diagnostic testing.

Choice A:

If choice A is selected, MENU mode is aborted and user control is returned to the console I/O mode (Example 7-8).

```
!
!
Type one of the above (for example, A), and press RETURN,

Enter MAIN MENU choice: A


MENU TEST STOPPED BY OPERATOR
**** END OF VAX-11/730,725 MENU TEST ****

RETURNING TO VAX-11/730,725 CONSOLE, WAIT FOR '>>>' PROMPT...

?06 PC=0001E822
>>>
```

Example 7-8 MENU Mode: Choice A Selected from Main Menu

Choice B:

If choice B is selected, MENU mode prints a list of devices available on the system (Example 7-9). This list also includes the device generic name and determines if the device is supported for testing. After this list is printed, the Main Menu is printed again.

Choice C:

If choice C is selected, MENU Mode prints a list of the system preparations required to test the supported devices on the system (Example 7-10). After this list is printed, the Main Menu is printed again.

```
!
Type one of the above (for example, A), and press RETURN,

Enter MAIN MENU choice: B

     Hardware          Generic          Support
     Name              Name             Status

     Console           CSA0             No Support
     DW730             DW0              No Support
     LESI              DAA              No Support
     DMF32S            XGA0             Supported
     DMF32A            TXA              Supported
     DMF32P            LCA              Supported
     RC25              DAA0             Supported
     RCF25             DAA1             Supported
                    [End of list]

     !
     !
     !
(return to Main Menu)
```

Example 7-9 MENU Mode: Choice B Selected from Main Menu

```
!
Type one of the above (for example, A), and press RETURN,

Enter MAIN MENU choice: C

     Hardware:        KA730, DMF32A, DMF32P, DMF32S

     Preparation:     No hardware preparation required

     Hardware:        RC25, RCF25

     Preparation:     1.  Insert removable disk in drive
                      2.  Push in RUN switch
                      3.  Pull out Write-protect REMOVE and FIXED
                          switch (not lighted)
                      4.  Wait for RUN indicator to to stop blinking
                          [End of list]
          !
          !
     (return to Main Menu)
```

Example 7-10 MENU Mode: Choice C Selected from Main Menu

Choice D:

If choice D is selected, user control is passed to Test Menu, which produces a printout similar to Example 7-11. This menu lists the devices that can be tested. To test a device, type the menu choice that corresponds to the device to be tested (that is, typing choice D1 causes the MENU mode to test all supported devices). See Example 7-12.

```
!
Type one of the above (for example, A), and press RETURN,

Enter Main Menu choice: D


TEST MENU

A1 =  KA730   CPU        KA0

B1 =  RC25    DISK       DAA0
B2 =  RCF25   DISK       DAA1

C1 =  DMF32S  COMM.      XGA0
C2 =  DMF32A  COMM.      TXA
C3 =  DMF32P  COMM.      LCA

D1 =  TEST ALL of the above supported hardware

Type one of the above (for example, A1), and press
RETURN to start testing.

Enter TEST MENU choice:
```

Example 7-11 MENU Mode: Choice D Selected from Main Menu

Example 7-12 is a typical functional test printout produced by MENU mode. This printout indicates to the user which device is currently being tested, the time spent to test that device, and at the end of the test, whether the device passed or failed. At the completion of testing, MENU mode returns control to the Main Menu.

```
!
Type one of the above (for example, A1), and press RETURN to
start testing.

Enter TEST MENU choice: D1

Press the CTRL key and type the C key (together)
at any time to interrupt Functional Test...

********** BEGIN FUNCTIONAL TESTING **********

           RUN TIME = 15:00 MINUTES

KA730   PART 1  KA0 TESTING STARTED -- RUN TIME = 00:45 MINUTES  PASSED
KA730   PART 2  KA0 TESTING STARTED -- RUN TIME = 02:00 MINUTES  PASSED
KA730   PART 3  KA0 TESTING STARTED -- RUN TIME = 02:45 MINUTES  PASSED
KA730   PART 4  KA0 TESTING STARTED -- RUN TIME = 02:00 MINUTES  PASSED
KA730   PART 5  KA0 TESTING STARTED -- RUN TIME = 03:00 MINUTES  PASSED
RC25    PART 1 DAA0 TESTING STARTED -- RUN TIME = 00:30 MINUTES  PASSED
RC25    PART 2 DAA0 TESTING STARTED -- RUN TIME = 01:00 MINUTES  PASSED
RCF25          DAA1 TESTING STARTED -- RUN TIME = 01:00 MINUTES  PASSED
DMF32S  COMM.  XGA0 TESTING STARTED -- RUN TIME = 01:00 MINUTES  PASSED
DMF32A  COMM.   TXA TESTING STARTED -- RUN TIME = 00:30 MINUTES  PASSED
DMF32P  COMM.   LCA TESTING STARTED -- RUN TIME = 00:30 MINUTES  PASSED
FUNCTIONAL TESTING COMPLETE -- NO ERRORS

        !
        !
(return to Main Menu)
```

Example 7-12 MENU Mode Successful Testing Sequence: Choice D Selected from Test Menu

MENU Mode can be interrupted or suspended at any time by typing CTRL/C. This causes a menu to be printed (Example 7-13), giving the user a choice of three actions to take. If choice A is selected, the Exit message is printed (Example 7-8) and MENU mode exits, returning control to the console I/O mode. Selecting choice B aborts whatever MENU Mode process was being performed at the time of the CTRL/C, and returns control to the Main Menu (Example 7-7). Conversely, if choice C is selected, the MENU Mode process interrupted by the CTRL/C continues.

```
Press the CONTROL key and type the C key together

     !
     !
     !
     !

     CONTROL-MENU

          A = Exit Menu Test
          B = Abort the current process, and return to MAIN MENU
          C = Resume the process interrupted by CONTROL-C

          Type one of the above (for example, A), and press RETURN.

          Enter CONTROL-C MENU choice:
```

Example 7-13 CONTROL-C Menu

7.4.2 MENU Mode Errors

Errors detected during device testing are listed on the functional test printout (similar to Example 7-14) as having failed the test (FAILED). Also, at the end of the functional test printout, the failed device is called out by its generic name.

When the device diagnostic test detects an error, that test is aborted (printing FAILED) and the next test is executed. Once all the selected devices are tested, control is passed back to Main Menu.

```
     !
     Type one of the above (for example, A1), and press RETURN to
     start testing.

     Enter TEST MENU choice: D1

     Press the CTRL key and type the C key (together)
     at any time to interrupt Functional Test...

     ********** BEGIN FUNCTIONAL TESTING **********

                RUN TIME = 15:00 MINUTES

KA730    PART 1  KAO TESTING STARTED -- RUN TIME = 00:45 MINUTES  PASSED

KA730    PART 2  KAO TESTING STARTED -- RUN TIME = 02:00 MINUTES  PASSED

KA730    PART 3  KAO TESTING STARTED -- RUN TIME = 02:45 MINUTES  PASSED

KA730    PART 4  KAO TESTING STARTED -- RUN TIME = 02:00 MINUTES  PASSED

KA730    PART 5  KAO TESTING STARTED -- RUN TIME = 03:00 MINUTES  PASSED

RC25     PART 1 DAAO TESTING STARTED -- RUN TIME = 00:30 MINUTES  PASSED

RC25     PART 2 DAAO TESTING STARTED -- RUN TIME = 01:00 MINUTES  PASSED
```

```
RCF25          DAA1 TESTING STARTED -- RUN TIME = 01:00 MINUTES *FAILED*
DMF32S  COMM.  XGA0 TESTING STARTED -- RUN TIME = 01:00 MINUTES *FAILED*
DMF32A  COMM.   TXA TESTING STARTED -- RUN TIME = 00:30 MINUTES  PASSED
DMF32P  COMM.   LCA TESTING STARTED -- RUN TIME = 00:30 MINUTES  PASSED
** FUNCTIONAL TESTING INCOMPLETE -- ERRORS
** FAILED HARDWARE TESTING: XGA0, DAA1
** CALL FIELD SERVICE **
   (return to Main Menu)
```

Example 7-14 Unsuccessful MENU Mode Testing Sequence: Choice D1 Selected from Test Menu

CHAPTER 8
BUILDING AND UPDATING THE SYSTEM DISK DIAGNOSTIC AREA

8.1 INTRODUCTION

The VAX-11/725 is shipped with an RC25 diagnostic distribution disk cartridge. This FILES-11 structured diagnostic distribution media is labeled VAX 11725/730 CMPLT DIAG. This media contains all microdiagnostic programs appropriate to the VAX-11/725 system.

Because the diagnostic area of a newly created system disk contains no diagnostic programs, selected diagnostic programs from the distribution media are copied to this area. These selected diagnostic programs are those programs that test the devices on the VAX-11/725 system on which the programs reside. The process of copying diagnostic programs to the system disk diagnostic area is referred to as building the diagnostic area.

NOTE

If the system disk does not contain sufficient room for the selected diagnostic programs, the diagnostic programs should remain on the distribution media.

Digital Equipment Corporation periodically releases new versions of existing diagnostic programs and, in some cases, may release entirely new diagnostic programs. When new versions or entirely new diagnostic programs are released for devices on a VAX-11/725 system, then these diagnostic programs are copied to the diagnostic area of the system disk (if diagnostic programs reside there). The process of copying these diagnostic programs to the diagnostic area is referred to as updating.

8.2 BUILDING AND UPDATING THE DIAGNOSTIC AREA

Building the diagnostic area of the system disk is accomplished under control of the operating system using one of two methods:

- Using the COPY command
- Using the Diagnostic Update Configuration Tool (DUCT)

Updating the diagnostic is accomplished under control of the operating system using DUCT.

8.2.1 Building and Updating the Diagnostic Area with the COPY Command

To build the diagnostic area on the system disk perform the following procedure:

1. Ensure that VMS is executing properly and the DCL prompt, $, is displayed on the terminal.
2. Ensure that the diagnostic distribution media, labeled VAX 11725/730 CMPLT DIAG, is loaded or mounted onto the drive.
3. Place the drive containing the diagnostic distribution media on-line by pressing the RUN switch on that drive.

4. When the RUN indicator stops blinking, type the following command at the "$" prompt to mount the diagnostic distribution cartridge.

```
$MOUNT DAA0:  CRDPACK
```

5. Type one of the following commands at the "$" prompt to copy all selected diagnostic program files to the system disk.

```
$COPY DAA1:[SYSMAINT]filename.ext SYS$MAINTENANCE
```

Filename.ext is the name of the diagnostic program file to be copied to the system disk maintenance area.

```
$COPY DAA1:[SYSMAINT]*.*; SYS$MAINTENANCE
```

This copies all diagnostic programs file to the system disk maintenance area.

6. Type the following commands to make the system disk maintenance area bootable:

```
$COPY/CONTIG DAA1:[SYSMAINT]DIAGBOOT.EXE DAA1:[SYSMAINT]*.*;0

$PURGE DUA1:[SYSMAINT]DIAGBOOT.EXE
```

NOTE
SYS$MAINTENANCE is the logical name for the diagnostic area on the system disk drive.

8.2.2 Building and Updating the Diagnostic Area with DUCT

DUCT is a copying utility that copies diagnostic files from the distribution media to the diagnostic area on the system disk. DUCT copies diagnostic files from tapes or disks. To use DUCT, perform the following procedure:

1. Ensure that VMS is executing properly and that the DCL prompt $ is displayed on the terminal.

2. Ensure that the diagnostic distribution media labeled VAX 11725/730 CMPLT DIAG, is loaded or mounted onto the drive.

3. Place the drive containing the diagnostic distribution media on-line by pressing the LOAD button on that drive.

4. At the $ prompt, type the following command to mount the diagnostic distribution cartridge:

```
$MOUNT DAA0:  CRDPACK
```

5. Type the following command at the $ prompt to execute DUCT from the distribution media:

```
$RUN DAA0:[SYSMAINT]DUCT.EXE
```

6. When DUCT prints the menu below (shown in Example 8-1), type the HELP command and follow the instructions to either build or update the system diagnostic area.

```
Select one of the following entries:

I     for     Initialize a diagnostic account

N     for     New option add on

U     for     Update a diagnostic account

Q     for     Query the diagnostic database

V     for     Verify directory against DUCT database

X     for     Execution menu

E     for     Exit

H     for     Help
```

Example 8-1 DUCT Menu Diagnostic Update Configuration Tool

APPENDIX A
VAX-11/725 INTERNAL PROCESSOR REGISTERS

Listed on the following pages are the addresses, mnemonics, and names of the VAX-11/725 internal processor registers.

Table A-1 Internal Processor Registers

Address Hex	Address Dec	Mnemonic	Access	Name
0	0	KSP		Kernel Stack Pointer
1	1	ESP		Executive Stack Pointer
2	2	SSP		Supervisor Stack Pointer
3	3	USP		User Stack Pointer
4	4	ISP		Interrupt Stack Pointer
8	8	POBR		P0 Base Register
9	9	POLR		P0 Length Register
A	10	P1BR		P1 Base Register
B	11	P1LR		P1 Length Register
C	12	SBR		System Base Register
D	13	SLR		System Length Register
10	16	PCBB		Process Control Block Base
11	17	SCBB		System Control Block Base
12	18	IPL		Interrupt Priority Level
13	19	ASTL		Asynchronous System Trap Level
14	20	SIRR	WO	Software Interrupt Request Register
15	21	SISR		Software Interrupt Summary Register
18	24	ICCS		Interval Clock Control and Status
19	25	NICR	WO	Next Interval Count Register
1A	26	ICR	RO	Interval Count Register
1B	27	TODR		Time of Year Register
1C	28	CSRS		Console Storage Receive Status
1D	29	CSRD	RO	Console Storage Receive Data
1E	30	CSTS		Console Storage Transmit Status
1F	31	CSTD	WO	Console Storage Transmit Data
20	32	RXCS		Console Receiver Control and Status
21	33	RXDB	RO	Console Receiver Data Buffer
22	34	TXCS		Console Transmit Control and Status
23	35	TXDB	WO	Console Transmit Data Buffer
26	38	MCESR	RO	Any write clears "machine check in-progress" flag
28	40	ACCS		Accelerator Control and Status *
37	55		WO	Initialize UNIBUS
38	56	MAPEN		Memory Management Enable
39	57	TBIA		Translation Buffer Invalidate All
3A	58	TBIS		Translation Buffer Invalidate Single
3D	61	PMR		Performance Monitor Enable
3E	62	SID	RO	System Identification Register

WO = Write only
RO = Read only
* = VAX-11/730-Specific Register

Table A-2 Machine-Dependent Internal Registers

Address Hex	Dec	Register Mapping
0	0	Processor Status Longword
1	1	CPU Microprogram Counter (UPC)
2	2	Memory Controller CSR 1 Register
3	3	Memory Controller CSR 2 Register
4	4	Set Breakpoint @ Microaddress
5	5	Enable/Disable WCS Control Store Parity
6	6	WCS Control Store Register
13:10	19:16	CPU Scratch Pad Memory (WR3:WR0)
20	32	CPU Quotient Register
IFF:100	511:256	CPU Scratch Pad Memory (Local Store)

APPENDIX B
MICRODIAGNOSTIC MONITOR COMMANDS

The following are the commands used with the MICMON.

DIR[ECTORY] — The directory of the TU58 being used (DD1 or DD0) is printed at the console terminal.

```
MIC>DIR
```

RET[URN] — The microprocessor program counter is forced to zero and the power-up routine is started again. If a console cassette (CONSOLE) is present, the system will power-up into the console I/O mode and the console terminal prints the prompt, >>>.

```
MIC>RET
>>>
```

S/U — The console microprocessor program counter is forced to <address> and operation is continued at that new address.

```
S/U <address>
MIC>S/U 0
```

This command would force zero to the console microprocessor program counter; the function is identical to "RETURN".

T/E — This command operates like the "RETURN" or "S/U 0" commands.

```
MIC>T/E
```

CON[TINUE] — The current diagnostic continues execution from the last error halt, SOMM, single step termination, or CTRL/C.

```
MIC>CON
```

LOAD — Loads a diagnostic section from the microcassette.

```
LD <section>
```

The following section names can be used with this command.

```
ENKBA       ENKBB       ENKBC
ENKBD       ENKBE       ENKBF
ENKBG       ENKCA       ENKCB
ENKCC       ENKCD       ENKCE
ENKCF       ENKCG
```

This command loads microdiagnostic section ENKBC from the microcassette.

INIT[IALIZE] — Initializes a WCS section. This command must be issued after a section is loaded by the LD command before tests are executed. It is automatically executed in a DI SE or DI command.

```
MIC>LD ENKBC
MIC>INT
```

X/C or X/U — Allows the remote diagnostic terminal to downline load a file into the console microprocessor memory (C for console memory) or WCS RAM (U for microcode). It is never executed from a console terminal.

DI[AGNOSE] — The microdiagnostic unitspecified is loaded and run. If no unit is specified, then testing starts at the first test and continues to the last test.

```
DI [<keyword>]
MIC>DI
```

The following keywords may be used to modify the command.

```
BO[ARD]        SE[CTION]     TE[ST]
CO[NTINUE]     PA[SS]        SH[ORTEN]
```

BOARD <type> — This keyword loads and executes all nonoptional sections that are associated with either the WCS, DAP, CPU, FPA, MCT or IDC board types.

NOTE
This command is incompatible with the SECTION and TEST keywords.

```
MIC>DI BO DAP
```

Run those tests that exercise the DAP (M8394) board.

SECTION <section> — This keyword loads and executes the specified section from the microcassette.

```
MIC>DI SE ENKBC
```

Load and execute the tests that are included in section ENKBC.

TEST <number> — This keyword executes the specified test from the currently loaded section.

```
MIC>DI TE 3
```

Run test number 3 of the section that is currently in RAM memory.

```
MIC>DI TE 3 5
```

Run tests 3 through 5 of the section that is currently in RAM memory.

CONTINUE — This keyword is used only after the TEST keyword. It executes the tests from the specified <number> to the end of the currently loaded section.

```
MIC>DI TE 8 CO
```

Run test number 8 through to the end of the section that is currently in RAM memory.

PASS — This keyword is used in conjunction with the TEST keyword to indicate the number of times a test(s) is performed.

```
MIC>DI SE ENKBD TE 2 5 PA 3
```

Load section ENKBD from the cassette into RAM memory and execute tests 2 through 5 three times.

```
MIC>DI TE 2 5 PA -1
```

Run tests 2 through 5 of the section that is currently in RAM memory and repeat forever (until interrupted by CTRL/C or CTRL/P).

SHORTEN — This keyword is used in conjunction with the TEST keyword. If an error occurs during testing loops from the first test specified to the test in which that error occured.

```
MIC>DI TE 1 15 SH
```

Run tests 1 through 15 of the section that is currently in RAM memory. If an error occurs, shorten the loop to execute tests 1 through the test that caused the error.

R[EPEAT] — This command repeats a command, such as a DEPOSIT or EXAMINE. The command is terminated by a CTRL/C or CTRL/P.

```
R <command>
MIC>R EX RA 0
```

This command repeats the command to examine four bytes of data at the console memory location 0.

SE[T]/CL[EAR] — The SET CLEAR command is used to enable/disable flags used in the execution of microdiagnostics.

```
SE <function>
CL <function>
```

The following functions can be set/cleared:

```
HA[LT]      LO[OP]      NE[R]
BE[LL]      SE[R]       TR[ACE]
SO[MM]      DE[FAULT]   PA[RITY]
ST[EP]*     BR[EAK]
```

HALT — Halt on error. SETting this flag causes the MICMON to return to the command level after reporting an error.

```
MIC>SE HA
```

LOOP — Loop on error. SETting this flag causes the MICMON to loop on the smallest piece of test code needed to reporduce the error.

*This flag is only used with the SET command.

NOTE
HALT has a higher priority than LOOP. Thus, if both are set, halt on error occurs first.

```
MIC>SE LO
```

NER — No error reports. SETting this flag causes the MICMON to skip the printout of error reporting.

```
MIC>SE NE
```

BELL — Bell on error. SETting this flag causes the MICMON to ring the terminal bell each time an error is reported.

```
MIC>SE BE
```

SER — Enable single bit errors to be reported as normal errors. SETting this flag causes the MICMON to print single bit errors.

```
MIC>SE SE
```

TRACE — Trace execution. SETting this flag causes the MICMON to print the test number before starting each test.

```
MIC>SE TR
```

SOMM — Stop on micromatch. SETting this flag causes the MICMON to write bad parity to the WCS location specified. When execution occurs and PARITY is set, the MICMON traps the interrupt caused by the parity error. If the address matches the last SOMM address, MICMON prints "SOMM" and the UPC and returns to the command mode (MIC>). If the parity is not the SOM address, a parity error message is displayed. SETting SOMM a second time clears the previous address.

NOTE
CLear PArity must be issued prior to SEt SOmm.

```
MIC>SE SO xxxx
```

DEFAULT — Set flags to default value. SETting this flag cuases the MICMON to SET HALT, disable SOMM and CLEAR all other flags.

```
MIC>SE DE
```

PARITY — Set bad parity. SETting this flag causes the MICMON to write bad parity to the address specified and to disable stall on parity errors. If no address is given, only disable stall on parity error occurs.

Clear parity is used to remove the bad parity error and reenable stall on parity error. If no address is given, only enable stall on parity error occurs. This command is also useful to write good parity at the location that has data deposited into it. This assures the parity at that location is correct.

```
MIC>SE PA xxxx
```

STEP — Set single step. SETting this flag causes the MICMON to single step the CPU n times. If no valud is given, the CPU single steps once for each time the space bar is typed. Any other character type causes MICMON to exit to the command level (MIC>).

NOTE

The character typed to exit to the command level becomes the first character of the next command. If a clean start is desired for the next command line, a CTRL/C must be typed to exit to the command mode.

```
MIC>SE ST nnnn
```

BREAK — Set breakpoint in console microprocessor code. SETting this flag causes the MICMON to write a branch instruction to the break routine at the address specified. This branch instruction is three bytes long. CAUTION: This instruction overwrites another routine's instructions. If the instruction is written to the program area of MICMON, a checksum error is printed, but this will not prevent the user from executing MICMON. The instruction replaced with the SET BREAK is not restored until a CLEAR BREAK or another SET BREAK to a new address is issued. CLEAR BREAK replaces the original three bytes at the address where SET BREAK was inserted.

```
SET BREAK xxxx
```

There are three special SET/CLEAR functions that affect the registers on the Internal Disk controller (IDC). They are SE AF, SE BR, and CL FI.

NOTE

These commands require that a WCS based microdiagnistic be loaded in WCS RAM.

SE AF — Select a Fifo. Selects Fifo address register "A" to supply the address used when a deposit or examine of the DBUF is executed.

SE BF — Select B Fifo. Selects Fifo address register "B" to supply the address used when a deposit or examine of the DBUF is executed.

CL FI — Clears the Fifo address register currently selected. This command preceeds the SE AF and SE BR commands.

SHOW — The show command causes MICMON to print the names of the test flags that are enabled.

```
SH <flag>
```

The flags are defined under the SET/CLEAR commands.

```
HALT        TRACE       LOOP        SOMM
NER         BELL        SER
```

EX[AMINE]/DE[POSIT] — The EXAMINE/DEPOSIT commands are used to read/write data from/to various registers or memory locations. The following modifiers point to the area being accessed.

```
EX <modifier> <address>
DE <modifier> <address> <value>
```

RA[M] or /U – Read four bytes of data at specified address. Write one byte of data to specified address.

Examples:

MIC>EX RA 4100 — Examine four bytes of data at the console memory location 4100.

MIC>DE RAM 71FF C – Deposit the byte value "C" in console memory location 71FF.

CS[R]– Read/write 24 bits of data from/to the CPU CSR.

Examples:

MIC>EX CSR — Reads 24 bits of data from the CPU control store register.

MIC>DE CSR 185F — Writes 24 bits of data to the CPU control store register.

WR[K] — Read/write data from/to the 2901 working registers.

Examples:

MIC>EX WR 3 — Examines contents of data processor working register 3.

MIC>DE WR 2 18FFCCAA — Deposits the value 18FFCCAA into the data processor working register 2.

MM # - Read/write data from/to main memory.

Examples:

MIC>EX MM 1074 — Examines the data in main memory location 1074.

MIC>DE MM 1074 3F8A — Deposits the value 3F8A in main memory location 1074.

ID[AR]#- Read/write data from/to the IDC's Disk Address Register.

Examples:

MIC>EX ID — Examines the data in the IDC Disk Address Register (DAR).

MIC>DE ID 51AB — Deposits the data 51AB in the IDC DAR.

PO[SIT]*# — Read data from the IDC's ECC position register.

Example:

MIC>EX PO — Examines the contents of the IDC Position Register (read only).

PA[TT]*# - Read data from the IDC ECC pattern register.

Example:

MIC>EX PA — Examines the contents of the IDC Pattern Register (read only).

DB[UF]# – Read/write data from/to the IDC's data buffer.

Examples:

MIC>EX DB — Examines the longword (four bytes) of the current data buffer (A or B) at the current data buffer address. The address is then incremented by four.

MIC>DE DB 18FFCCA1 — The value 18FFCCA1 is deposited in the current data buffer at the current address. The address is then incremented by four.

UP[C] – Read/write 15 bits of data from/to the UPC register.

Examples:

MIC>EX UP — Examine the microprogram counter.

MIC>DE UP 01F7C — Deposit the value 01F7C in the microprogram counter.

WC[S]or/C — Read/write data from/to WCS (without parity calculation).

Examples:

MIC>EX WC 0E00 — Examine writable Control Store (WCS) or EX/C 0E00 through location 0E00.

MIC>DE WC 0800 16574F - Deposit the value 16574F in WCS location 0800.

LS — Read/write data from/to local store locations.

Examples:

MIC>EX LS 200 — Examines the data at local store location 200.

MIC>DE LS 3100 3FF - Deposits the value 3FF in local store location 3100.

OS — Read/write eight bits of data from/to the CPU OS register.

Example:

MIC>EX OS — Examine the eight bits of data in the CPU OS register.

UB[S]# - Read/write data from/to the UNIBUS map portion of the translation buffer.

Examples:

MIC>EX UB 200 — Read the location 200 from the UNIBUS translation buffer.

MIC>DE UB 3FF 1F900 — Writes data 1F900 into location 3FF of the UNIBUS translation buffer.

`MC[T]#` — Read/write data from/to the memory controller's CSR registers.

Examples:

`MIC>EX MC 2` — Examine data in MCT's CSR register 2.

`MIC>DE MC 1 2C000000` — Deposit data 2C000000 into MCT's CSR register 1.

`TB #` — Read/write data from/to the memory controller's translation buffer.

Examples:

`MIC>EX TB 5` — Read data from MCT's translation buffer 5.

`MIC>DE TB 10 7100`- Write data 7100 into MCT's translation buffer 10.

`IC[SR]`– Read/write data from/to the IDC's CSR register.

Examples:

`MIC>EX IC` — Examine the data in the IDC's CSR register.

`MIC>DE IC 10B002BD`- Deposit the value 10B002BD into the IDC's CSR register.

VAX-11/725 Diagnostic System
Overview Manual
EK-DS725-UG-001
(MKO)

Reader's Comments

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to this manual? In your judgement is it complete, accurate, well organized, well written, etc? Is it easy to use? _____

_____

_____

_____

What features are most useful? _____

_____

_____

_____

What faults or errors have you found in the manual? _____

_____

_____

_____

Does this manual satisfy the need you think it was intended to satisfy? _____

Does it satisfy *your* needs? _____ Why? _____

_____

_____

_____

☐ Please send me the current copy of the *Technical Documentation Catalog*, which contains information on the remainder of DIGITAL's technical documentation.

Name _____ Street _____
Title _____ City _____
Company _____ State/Country _____
Department _____ Zip _____

Additional copies of this document are available from:

Digital Equipment Corporation
444 Whitney Street
Northboro, MA 01532

Attention: Printing and Circulation Services (NR2/M15)
Customer Services Section

Order No. **EK-DS725-UG-001** _____

Fold Here

Do Not Tear – Fold Here and Staple

digital™

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD, MA

POSTAGE WILL BE PAID BY ADDRESSEE

Digital Equipment Corporation
Educational Services/Quality Assurance
12 Crosby Drive, BU/E08
Bedford, MA 01730

Digital Equipment Corporation • Bedford, MA 01730