63452

94-003/2212

VAX/DS Diagnostic Supervisor User's Guide

Order Number: AA-FK66A-TE

April 1989

This manual describes how to use the VAX/DS Diagnostic Supervisor and the VAX/DS command language.

Revision/Update Information	This revised document supersedes the VAX Diagnostic Supervisor User's Guide, Order No. EK-VXDSU-U1-001.
Software Version	VAX/DS Version 11.6

digital equipment corporation, maynard, massachusetts

First Printing, April 1989

DEC	ULTRIX	VMS
DECnet	UNIBUS	VT
MASSBUS	VAX	XMI
RSX	VAXBI	
SBI	VAXcluster	digital™

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA*	CANADA	INTERNATIONAL
Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061	Digital Equipment of Canada Ltd. 100 Herzberg Road Kanata, Ontario K2K 2A6 Attn: Direct Order Desk	Digital Equipment Corporation PSG Business Manager c/o Digital's local subsidiary or approved distributor

In Continental USA, Alaska, and Hawaii call 800-DIGITAL.

In Canada call 800-267-6215.

* Any order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).

Internal orders should be placed through the Software Distribution Center (SDC). Digital Equipment Corporation, Westminster, Massachusetts 01473.

This document was prepared using VAX DOCUMENT, Version 1.1.

Contents

Preface

Intended Audience

This manual is intended for all users of the VAX/DS Diagnostic Supervisor.

Document Structure

This manual contains five chapters:

- Chapter 1 discusses VAX/DS concepts.
- Chapter 2 explains how to run the VAX/DS Diagnostic Supervisor and VAX/DS diagnostic programs.
- Chapter 3 details the VAX/DS command language.
- Chapter 4 describes how to create command files for use with the VAX/DS Diagnostic Supervisor.
- Chapter 5 provides information regarding automated quality assurance (AUTO-QA) of VAX/DS diagnostic programs.

Associated Documents

- *VAX Diagnostic Design Guide*
- *VAX Diagnostic Software Handbook*
- *VAX Architecture Reference Manual*

Conventions Used in This Document

Convention	Meaning
BOLD	Introduces new terms.
italics	Used for emphasis and indicates the title of a manual.
KEYWORD	Keywords are capitalized.
[ ]	Square brackets indicate that the enclosed element is optional.
DS> LOAD EVXYZ.EXE	Command examples show the VAX/DS prompt DS> .

Note: **The acronyms for the VAX Diagnostic Supervisor are VAX/DS and VAX/VDS.**

1 VAX/DS Concepts

The VAX Diagnostic Supervisor (VAX/DS) provides a common set of user commands for VAX diagnostic programs. All diagnostic programs designed to run under the VAX/DS respond to the same set of commands and have the same operating behavior.

The VAX/DS is a monitor that controls operation of a diagnostic program and allows the user to:

- Select which device to test
- Select which program to run
- Specify various run-time parameters
- Run all or only parts of the selected program
- Specify whether the program should continue, halt, or loop when it detects a fault
- Obtain a summary of the program's operating statistics

For a more complete discussion of VAX diagnostic concepts and VAX/DS concepts, see the *VAX Diagnostic Design Guide*.

1.1 Run-time Environments

The VAX/DS can run under the VMS operating system (in the same manner as any image in VMS), and it can run without an operating system. When the VAX/DS is executing under VMS, it is running in **user mode**. When it is executing without the presence of an operating system, the VAX/DS is running in **standalone mode**. For standalone operation, the VAX/DS and diagnostic programs can be booted and run from media created under either VMS or ULTRIX-32.

Operation of the VAX/DS, from a user's point of view, is nearly identical in each of these modes. The slight differences that do exist will be pointed out in appropriate sections of the text. The one major difference between the two modes is the procedure used to start execution of the VAX/DS. This will be explained in Section 2.1, Running the VAX/DS.

The VAX/DS can also be used in automated environments in which there is no user present at the system being tested, such as the Automated Product Test (APT) environment and the Automated Product Test/Remote Diagnosis (APT/RD) environment.

Not all VAX diagnostic programs can be used in all run-time environments.

1.2 Program Levels

All VAX diagnostic programs are assigned to a **level**. Levels are numbered 1, 2R, 2, 3, 4, and 5. A program's level indicates its run-time environment.

Level 1 diagnostic programs run in user mode. They do not use the VAX/DS.

Level 2R diagnostic programs run in user mode, under the control of the VAX/DS.

Level 2 diagnostic programs run in both user mode and standalone mode, under the control of the VAX/DS.

Level 3 diagnostic programs run only in standalone mode, under the control of the VAX/DS.

Level 4 and level 5 diagnostic programs run in standalone mode. They do not use the VAX/DS.

This guide is concerned only with programs of levels 2R, 2, and 3.

For a detailed discussion of VAX diagnostic program levels, refer to the *VAX Diagnostic Design Guide.*

1.3 Sections, Tests, and Subtests

All diagnostic programs that run under the VAX/DS are composed of one or more **sections**. Each section has a name, and it is possible to specify, by name, which section of the program to execute. A program section can have one of two functions, depending on its design:

- Test a certain major portion of the unit under test.
- Facilitate a certain mode of testing. For example, there might be a **quick-verify** section, used to give the device a quick (but not thorough) test.

In order to determine the number and names of the sections in a particular diagnostic program, use the SHOW SECTIONS command. To fully understand the programmer's intended purpose of a diagnostic program's specific sections, it is necessary to read the documentation for that program (see Section 1.11). However, every program contains a **default** section called DEFAULT, which will execute automatically if you do not specifically select a section. A section is selected with the /SECTION qualifier on the RUN or START command.

Each section is composed of a set of **tests**; each test verifies either a particular function of the hardware or a particular subset of the component logic being tested. The VAX/DS allows you to specify which tests within a section to run. Tests are selected with the /TEST qualifier on the RUN or START command.

Tests within a diagnostic program are numbered from 1 to n, where n is the total number of tests. Sections are made up of groups of these tests. A test can belong to more than one section. For example, a program might contain three sections and 20 tests, where the first section is composed of tests 1 through 10, the second section of tests 11 through 20, and the third section of tests 1, 5, 7, 9, 10, 11, and 15.

Each test can be partitioned into a set of **subtests**. Subtests are used to divide the operations performed in a given test into a series of small suboperations useful for diagnosing faults. Subtests are used by the VAX/DS to create program loops (see Section 1.7).

Refer to the descriptions of the RUN and START commands for details on how to select the various sections, tests, and subtests of a diagnostic program.

1.4 Serial Tests and Parallel Tests

In general, a diagnostic program is written to test a certain type of device. If the system under test has several units of a particular device type, the diagnostic program will generally be capable of testing all of them.

There are two methods by which this testing of multiple units can be accomplished: **serial testing** and **parallel testing**. Serial testing involves testing each unit of the device individually and sequentially. Parallel testing is the testing of all units simultaneously.

The execution sequence for all diagnostic programs that perform serial testing is as follows:

1. Execute initialization code for next selected unit.
2. Execute selected tests on unit.
3. If all units have not been tested, go to step 1.
4. Execute cleanup code for all units.

The execution sequence for all diagnostic programs that perform parallel testing is as follows:

1. Execute initialization code for all selected units.
2. Execute selected tests on all selected units.
3. Execute cleanup code for all selected units.

For more detailed discussions of these testing methods, refer to the *VAX Diagnostic Design Guide*.

1.5 Initialization and Cleanup

Besides the actual tests, two other important sections of code exist in diagnostic programs. These sections are called the **initialization code** and the **cleanup code**.

The initialization code is code that is executed before a device unit is tested. It performs the operations necessary for creating a data link to the unit and prepares the unit for testing.

When all testing of devices has been completed, there must be a means for guaranteeing that each device is left in a known, initialized, static state. The "cleanup code" will perform any functions necessary to place each device unit into such a state. This code resides in the diagnostic program, delimited by the macros $DS_BGNCLEAN and $DS_ENDCLEAN.

1.6 Program Passes

When a diagnostic program is run under the VAX/DS, the units that are to be tested and the tests that are to be executed must be specified. The number of **program passes** can also be specified. A program pass is the execution of all selected tests on all selected units. Thus, after one pass, every selected test has been run on each selected unit once. The number of passes is specified with the /PASSES qualifier on the RUN or START command.

1.7 Program Loops

A **program loop** can be created by the VAX/DS. This loop will be the repeated execution of a block of code that allows the user to examine logic signals with an oscilloscope. (Hence, these program loops are often called **scope loops**.)

There are two methods in the VAX/DS which generate loops:

- Loop execution can automatically begin whenever a hardware fault is detected, which is discussed in Section 1.9.
- A loop can be based on a particular test or subtest. This method is described in the discussions of the START and RUN commands and involves the use of the /TEST, /SUBTEST, and /PASSES qualifiers.

1.8 Flags

Two sets of flags are provided to control certain aspects of a diagnostic program's execution. These flags may be set or cleared before the diagnostic program is started.

One set of flags is called the **VAX/DS control flags**. The flags in this set are interpreted by the VAX/DS and their meanings are the same in all diagnostic programs. The VAX/DS control flags are described in the discussion of the SET FLAGS command.

The other set of flags is referred to as the **event flags**. This set consists of 23 flags whose purposes are defined by the diagnostic programs. The user must read the documentation for specific diagnostic programs to determine if a definition has been assigned to any of the event flags. See Section 1.11 and the description of the SET EVENT FLAGS command.

1.9 What Happens When a Hardware Fault is Detected?

A fault is an erroneous state of hardware or software resulting from failures of components, environmental interference, operator error, or incorrect design. An error is the manifestation of a fault within a program or data structure. The error may occur some distance away from the fault site.

When a diagnostic program detects a fault in the unit under test, several choices are available for controlling the events that take place. These choices involve how the error should be reported and what actions the VAX/DS should take after the report is made.

When a fault is detected, the diagnostic program will provide a message detailing characteristics of the fault or error. The user may choose to have all, some, or none of this message displayed on the terminal.

Error messages are divided into three levels. Message level 1 identifies the error by listing the test, subtest, and pass numbers, and a short statement describing the type of error. Message level 2 provides a more detailed description of the error. Message level 3 generally includes detailed, lengthy information, such as register dumps. The specific contents of each message level is dependent on the particular diagnostic program; sometimes the second and/or third message level is not provided for a particular error.

It is possible to have levels 1, 2, and 3 (the entire message) displayed; only levels 1 and 2; only level 1; or no message at all. Another option is to have the user terminal issue an audible signal when a fault is detected. All these options are controlled with VAX/DS control flags (see SET FLAGS).

After the fault has been detected and the error reported, the VAX/DS takes one of three actions:

- HALT-ON-ERROR. The VAX/DS will stop execution of the diagnostic program and issue its prompt. Halt-on-error is enabled with the VAX/DS control flag HALT (see SET FLAGS).
- LOOP-ON-ERROR. The diagnostic will begin executing a scope loop. The loop will include the code that caused the fault to be detected and will be perpetual until the user types a CTRL/C. Loop-on-error is enabled by setting the VAX/DS control flag LOOP (see SET FLAGS).
- Normal Execution. The program will continue executing as it would have if the fault had not been detected. This mode of execution will occur if the HALT and LOOP flags are both cleared.

Halt-on-error and loop-on-error can be used together; a loop is created that will stop each time the error is encountered. The user can then type the CONTINUE command and the loop will continue its execution.

1.10 Presence of an Operator

It is not always practical or necessary for an operator to be present during the entire execution of a diagnostic program. For example, it may be desirable to run a program for several hours to ensure that no intermittent failures exist. Also, the APT and APT/RD run-time environments are meant to provide system testing without the presence of an operator.

Conversely, it is sometimes necessary that a user be present when certain diagnostic programs execute certain tests. For example, it may be necessary for a cable to be attached, or for a write-enable switch to be set during a program's execution. (These types of actions by the user during program execution are referred to as **manual intervention**.)

Therefore, a means has been provided for indicating to the VAX/DS whether or not an operator is going to be present during the diagnostic program's execution. Setting the OPERATOR flag indicates that a user or operator will be present. (See the description of the SET FLAGS command.) If this flag is cleared, the VAX/DS will not allow execution of any parts of a diagnostic program that require a user to be present if any such parts exist. If any of the tests being executed would normally ask the user to provide answers to questions displayed on the user's terminal, default values will be automatically taken for those answers.

1.11 Diagnostic Program Documentation

Documentation provided for each diagnostic program includes:

- Help file

  The help file is referenced with the HELP command (see the discussion of the HELP command). It contains any operating instructions that may be unique to the diagnostic program. The help file for diagnostic program EVXYZ is EVXYZ.HLP; the help file for the VAX/DS is EVSAA.HLP:

- Documentation file

  The documentation file provides a detailed description of a diagnostic program's function, run-time requirements, operating instructions, fault detection, and maintenance history, plus explanations of each test's function and execution flow. The documentation file for diagnostic program EVXYZ is EVXYZ.DOC.

- Source Code listings

  Source code listings of the diagnostic programs are available. All VAX/DS diagnostic programs are structured in a standard format. For example, the initialization code for every diagnostic program is always located in the same relative area within the listing. To fully understand the source code listings of a VAX/DS diagnostic program, refer to the *VAX Diagnostic Design Guide*.

The help file for a diagnostic program is shipped with the diagnostic program. The documentation file and source code listings are available by ordering a Maintenance Documentation Service Option on microfiche.

1.12 Concepts of Multiprocessor Diagnostic Programs

Some diagnostic programs are designed to run concurrently on multiple processors. A **multiprocessor diagnostic** is a diagnostic program which, while executing on the primary processor, will cause one or more of the other processors and the primary CPU to execute test code simultaneously. The processor on which the VAX/DS is booted is referred to as the **primary processor**, and the secondary (other) processors are called **attached processors**. (For information on booting the VAX/DS in a multiprocessor environment, see Chapter 2).

In order to fully understand the operation of a multiprocessor diagnostic program, read the documentation for the program and the discussion of the VAX/DS in a multiprocessor environment in the *VAX Diagnostic Design Guide*.

Some restrictions apply to certain VAX/DS commands when multiprocessor diagnostic programs are running. Refer to the descriptions of SET BREAKPOINT, NEXT, EXAMINE, DEPOSIT, and SET/SHOW CPU in this guide for details of those restrictions.

2 How to Run a VAX/DS Diagnostic Program

This chapter discusses the steps involved in running a diagnostic program under the VAX Diagnostic Supervisor. These steps include:

- Loading and starting the VAX/DS
- Attaching and selecting devices so they can be tested
- Loading and starting a diagnostic program

2.1 Running the VAX/DS

The method used to load and begin execution of the VAX/DS depends on the run-time environment. The following sections describe how to start the VAX/DS in user mode and standalone mode. For standalone operation, booting from both VMS media and ULTRIX media is discussed.

Once the VAX/DS has been loaded and started, the banner will be displayed and the user prompt will be issued:

```
                    VAX DIAGNOSTIC SOFTWARE
                          PROPERTY OF
                 DIGITAL EQUIPMENT CORPORATION

               ***CONFIDENTIAL AND PROPRIETARY***

Use Authorized Only Pursuant to a Valid Right-to-Use License
Copyright, Digital Equipment Corporation, 1988.  All Rights Reserved.

DIAGNOSTIC SUPERVISOR.  ZZ-EnSAA-XX.Y-NNN   3-FEB-1989 12:55:32
DS>
```

2.1.1 Starting the VAX/DS in User Mode

In user mode, the VAX/DS is started when you give the VMS command:

```
$ RUN EnSAA
```

where

n = An alphabetic character unique to each type of VAX processor. Table 2-1 shows the name of the VAX/DS programs for each VAX processor.

Table 2-1 Names of the VAX/DS Program for Each VAX Processor

PROCESSOR	PROGRAM
VAX 780/785	ESSAA
VAX 8600/8650	EDSAA
VAX 8200/8250/8300/8350	EBSAA
VS8000	EBSAA
VAX 8530/8550/8700/8800/8820N	EZSAA
VAX 8820/8830/8840	EJSAA
VAX 6210/6220/6230/6240 6310/6320/6330/6340/6350/6360	ELSAA

2.1.2 Starting the VAX/DS in Standalone Mode Using VMS Media

This section outlines the steps involved in bootstrapping the VAX/DS for each type of VAX processor.

For more detailed information, see the following documents:

- *VAX-11/780 Diagnostic System User's Guide*
- *VAX 8600/8650 Diagnostic System User's Guide*
- *VAX 8530/8550/8700/8800/8830/8840 Diagnostic User's Guide*
- *VAX 8200/8300 Owner's Manual*
- *VAX 6200/6300 Owner's Manual*
- *VAXSTATION 8000 Owner's Manual*
- *VAX Hardware Handbook*

2.1.2.1 Optional R5 Arguments to the Booting Process

You can select certain bootstrap options by setting various bits of general-purpose register R5. R5 bits relevant to booting the VAX/DS are described in Table 2-2.

Refer to processor-specific console documentation to determine the correct method for setting R5 bits before initiating the boot procedure.

Table 2-2 R5 Bits Used to Boot the VAX/DS

Bit	Function
Bit 0	Conversational boot. If set, the bootstrap code will solicit input parameters from the operator during the bootstrap process.
Bit 2	Initial breakpoint. If the VAX/DS has been linked with XDELTA, an initial BPT will be executed.
Bit 4	Diagnostic boot. Secondary bootstrap file will be [SYS0.SYSMAINT]:DIAGBOOT.EXE. (This bit should always be set for booting the VAX/DS.)
Bit 7	Memory test. If set, memory will not be tested.
Bit 15	Autocom. The VAX/DS will automatically run a command file, VDSSCRIPT.COM upon booting. This is supported on all VAX systems except 11/7XX and 86XX systems, and 82XX/83XX (console). See Chapter 4.

2.1.2.2 Booting a VAX-11/780, 11/785

- If booting from a non-HSC-based disk, insert the VAX 11/78X standard console floppy diskette and the disk from which you are booting. Type the following commands from console mode:

```
>>>BOOT Sgn
```

where

S = Diagnostic boot

g = Generic drive type character:

B = RP04/RP05/RP06/RP07
M = RK06/RK07
R = RM03/RM05/RM80

n = Drive number (0-7)

For example:

```
>>>BOOT SB0          ! Boots from drive unit 0 of an RP06.
```

- If booting from an HSC-based disk, modify the VAX 11/78X standard console floppy file SCIBOO.CMD. Initialize R2 to be the desired station of the remote CI port and R3 to be the desired disk unit number. Insert the floppy and type the following commands from console mode:

```
>>> BOOT SCI
```

- If booting from the console floppy, insert the Diagnostic Supervisor floppy and type the following commands from console mode:

```
>>> BOOT
```

2.1.2.3 Booting a VAX 8200/8250/8300/8350

- If booting a non-HSC-based disk, type the following commands from console mode:

```
>>> H                 ! Halt
>>> I                 ! Initialize
>>> B/R5:10 ddnu      ! Boot
```

where

dd = Boot device type of the controller for the specific boot device
  DU = RA type disk connected via KDB50 or UNIBUS adapter
  CS = console floppy or HSC-based disk
n = controller's VAXBI node number (hex)
  A = boot device type CS
u = Unit number of the boot device

- If booting from an HSC-based disk, copy the VAX 82XX/83XX console floppy file CIBOO.CMD to SCIBOO.CMD. Modify SCIBOO.CMD as directed in the comments within the file, and set bit <4> in R5 to cause a VAX/DS boot. Insert the floppy and type the following commands from console mode:

```
>>> H                 ! Halt
>>> I                 ! Initialize
>>> B/R5:800 CSA1     ! Boot
```

At the BOOT58 prompt, type:

```
BOOT58> @SCIBOO
```

- If booting from a console diskette, insert the VAX 82XX/83XX Diagnostic Supervisor console diskette into console drive 1 or 2 (CSA1 or CSA2), and type the following commands from console mode:

```
>>> H                 ! Halt
>>> I                 ! Initialize
>>> B CSA1 or B CSA2 ! Boot
```

2.1.2.4 Booting a VAX 8600/8650

- If booting from a nonconsole disk, insert the VAX 86XX console RL02 disk (with or without diagnostics) and the disk from which you are booting, and type the following command from console mode:

```
>>> BOOT [ /switches ] [ device ]
```

where

switches = Any combination of the following:

/R5:10	= Indicates diagnostic supervisor boot
/NOSTART	= Specifies that the boot command file does not start the operating system at its completion. If your system disk configuration is different from that supported by the standard boot command files, you may alter register values prior to starting the operating system. To start the operating system, type the following command at the console prompt:

```
>>> START 10000
```

device = one of the following:

- A device mnemonic (1 to 3 characters) such as DU0 or CS1, that is appended automatically with a BOO.COM suffix. The resulting file name is used to locate the xxxBOO.COM file to boot the operating system.
- The file name with the default extension of .COM; the file name must be greater than 3 characters in length. This file is then located and used to boot the operating system.

If "device" is not specified, the file DEFBOO.COM is used.

For example:

```
>>>BOOT/R5:10 DU0        ! Loads R5 with 10 and invokes DU0BOO.COM
>>>BOOT/R5:10/NOSTART    ! Loads R5, runs DEFBOO.COM, waits for input
```

- If booting from a console disk, type the following command from console mode:

```
>>> @EDSAA
```

2.1.2.5 Booting a VAX 8530/8550/8700/8800/8820N

- If booting a single processor system, type the following command from console mode:

  ```
  >>> BOOT ddd
  ```

  where

  ddd = device type code of the boot device

  SCI = HSC disk

  SDA = KDB50 disk

  DIA = console Winchester disk

  This causes the command file, dddBOO.COM, to be executed. If no device code is specified, DEFBOO.COM is executed.

- If booting a dual processor system, type:

  ```
  >>> SET NEXT_PRIMARY <left or right>
  ```

  Then follow the above steps for booting a single processor system.

- In order to reboot a loaded VAX/DS on a different processor, type:

  ```
  DS> EXIT                      ! Exit VAX/DS
  >>> SET CPU <left or right>   ! Set new processor
  >>> INITIALIZE
  >>> UNJAM
  >>> S 10000                   ! Start
  ```

2.1.2.6 Booting a VAX 8820/8830/8840

- To boot a VAX 8820/8830/8840 system, type the following commands from console mode:

```
PS-HW-0> @CPUALL or
         @CPUn[...n]        ! Selects the CPUs from those present and
                            ! enabled; n can be a decimal number
                            ! between 0 and 3 and identifies the
                            ! processor or processors.

PS-HW-0> @SYSINT            ! Load microcode and initialize memory

PS-CIO-0> BOOT ddd
```

where

ddd = device type code of the boot device

DIA = console Winchester disk

BCI = HSC disk

BDA = KDB50 disk

This causes the command file, dddBOO.COM, to be executed. If no device code is specified, DEFBOO.COM is executed.

- In order to reboot a loaded VAX/DS on a different processor, type:

```
DS> EXIT

PS-CIO-0> SET PRIMARY n       ! Where n is the number of the
                              ! new CPU.

PS-CIO-n> SET NEXT_PRIMARY n  ! This ensures reboot of CPU n
                              ! if power fails.

PS-CIO-n> ST 10000            ! Start.
```

2.1.2.7 Booting a VAX 6210/6220/6230/6240/6310/6320/6330/6340/6350/6360

- If booting from the TK50/TK70 drive, type either of the following commands from the console prompt:

  ```
  >>> BOOT/R5:10 CSA1
  ```

  or

  ```
  >>> BOOT/R5:10/BI:i/XMI:j MUO
  ```

  where

  i = BI node number of the TBK50/TBK70 tape controller

  j = XMI node number of the DWMBA

- If booting from a KDB50 disk, type:

  ```
  >>> BOOT/R5:10/BI:i/XMI:j DUn
  ```

  where

  i = BI node number of the KDB50 disk controller

  j = XMI node number of the DWMBA

  n = disk unit number

Note: **If the system is configured with a CIBCA-Ax (where x = A or B), the system console tape which contains VMB.EXE and CIBCA.BIN must be mounted on the TK50/TK70 drive. For help regarding the system configuration, type the following command at the console prompt:**

```
>>> SHOW CONFIG
```

- If booting from a CI disk, type:

  ```
  >>> BOOT/R5:10/BI:i/XMI:j/NODE:k DUn
  ```

  where

  i = BI node number of the CIBCA adapter

  j = XMI node number of the DWMBA

  k = CI node number of the HSC disk controller

  n = disk unit number

Note: **If the system is configured with a CIBCA-Ax (where x = A or B), the system console tape which contains VMB.EXE and CIBCA.BIN must be mounted on the TK50/TK70 drive. For help regarding the system configuration, type the following command at the console prompt:**

```
>>> SHOW CONFIG
```

2.1.2.8 Booting a VS8000

- If booting from the TK50 tape drive, insert the diagnostic tape cartridge (labeled VAXSTATION 8000 HDW FILES). Make sure the cartridge is write-protected. Type the following commands from the console prompt:

```
>>> H               ! Halt

>>> I               ! Initialize

>>> B/R5:10 MU60    ! MU = boot device type code
                    !  6 = tape controller's node number
                    !  0 = drive number
```

- If booting from a nonremovable disk (RD5x) from console mode, type:

```
>>> H               ! Halt

>>> I               ! Initialize

>>> B/R5:10 DU5n    ! DU = boot device type code
                    !  5 = KFBTA controller's node number
```

where

```
n = 1, 2 OR 2       ! drive number
```

2.1.2.9 Booting from a VAX/VMS Shadow Set

If booting from a VAX/VMS shadow set, R3 must be set up with the following values: bit <31> must be set, bits <30:16> must be the virtual unit number, and bits <15:0> must be the member unit number. The virtual unit number corresponds to the unit number assigned to the shadow set. The member unit number is the physical drive number of any member of the shadow set.

Note: For more information regarding volume shadowing, see the *VMS VAXcluster Manual*.

If the shadow set has not yet been established, type the following commands at the VMS DCL prompt:

```
$ INITIALIZE $HSC50$ggan1: Vol_Label_0
$ MOUNT $HSC50$ggan2:/SHADOW=($HSC50$ggan0: $HSC50$ggan1:) Vol_label_1
```

where

```
    $HSC50$ggan0 and $HSC50$ggan1 are shadow set members
    $HSC50$ggan2 is the shadow set volume
```

```
>>> DEPOSIT/G 3 80020001    ! Deposit to GPR 3 (R3)

>>> BOOT xyz
```

where

xyz is the processor-specific boot parameter (see Sections 2.1.2.1 to 2.1.2.7).

2.1.3 Starting the VAX/DS in Standalone Mode Using ULTRIX Media

To boot the VAX/DS from ULTRIX-formatted media, follow the steps used for booting from VMS-formatted media for the particular processor (see Section 2.1.2). The exception is the VAX 82XX/83XX.

To boot the VAX/DS from Ultrix-formatted media for the VAX 82XX/83XX, type the following command:

```
>>> B/R5:60000010
```

2.2 Attaching Devices

Before a device can be tested, its device characteristics and location must be given to the VAX/DS. The information about the device is stored in the VAX/DS so that it is available to the diagnostic program.

2.2.1 Hardware Parameter Tables (P-Tables)

Because some device characteristics depend on the specific system to which the device is connected, it is impossible to completely specify device characteristics before the VAX/DS is loaded into the system under test. Hence, it is necessary to provide a means for specifying system-specific hardware characteristics at run time.

These variable characteristics of hardware devices are referred to as **hardware parameters**. The VAX/DS stores these parameters in a set of tables known as the **hardware parameter tables**, or **p-tables**. A p-table must be constructed for each device unit that will be tested. The p-table contains the unique characteristics of the device. During execution, the diagnostic program references the p-tables to obtain these device characteristics.

Specifying device characteristics to the VAX/DS is defined as **attaching** the device and is based on the fact that every hardware device on a system is connected to another piece of hardware. For example, a disk drive is connected to a disk controller, which is connected to a bus, which is in turn connected to a processor.

2.2.2 Using the ATTACH Command

Devices can be attached manually with the ATTACH command. Every time the ATTACH command is used, the VAX/DS constructs a p-table. Attaching a device entails specifying the device, its unique characteristics, and its connection. A device's connection is referred to as its **link**. All of the specified device characteristics are stored in the device's p-table.

To attach a device, the device's link must have previously been attached. For example, attaching a disk drive requires first attaching the bus, then the controller, and finally the drive to create a complete hardware path from the processor to the device.

To obtain a display of the devices that may be attached, type:

```
DS> HELP DEVICE
```

Device characteristics for a specific device type can be obtained with the command:

```
DS> HELP DEVICE device-type
```

where

device-type = The type of device, such as "RK06" or "TU58."

Examples of attaching devices can be found in Example 2–1 and under the description of the ATTACH command.

It is sometimes useful to create a command file containing all of the ATTACH commands you frequently use on your VAX system. Refer to Chapter 4 for a discussion of command files.

2.2.3 The Autosizer

The **autosizer** is a program that runs under the VAX/DS. It determines what hardware devices exist on the system under test. It then automatically generates the p-table entries needed to describe the devices it finds, eliminating the need to issue ATTACH commands. *This is the preferred method for generating p-tables.* The autosizer can only be used in standalone mode. Its file name is EVSBA. Refer to the documentation file EVSBA.DOC for a detailed discussion of the autosizer. (There may be some cases when the autosizer cannot identify a device, such as when the device is not sold by DIGITAL. Therefore, the user must issue ATTACH commands manually.)

2.3 Running a Diagnostic Program

To run a diagnostic program under the VAX/DS, the following operations must be performed:

1 Attach the devices to be tested (with the autosizer or the ATTACH command). Select them for testing (with the SELECT command).

2 Set or clear the various VAX/DS control flags, event flags, and other run-time variables. The SET and CLEAR commands are used for these purposes.

3 Load and start the desired diagnostic program by using the LOAD and START commands or the RUN command. You can select the tests and the number of program passes to be executed.

The program can be stopped during execution with CTRL/C, and the flags can be modified. The CONTINUE command can then be used to cause the program to resume execution at the point at which it was stopped.

When the program completes execution, the user can obtain a summary printout of the program results by using the SUMMARY command.

Example 2-1 shows a sample execution of a diagnostic program. In the sample, a KA62A processor is attached and selected. The TRACE flag is set (causing test titles to be displayed), and then program ELKAX is executed (only tests 1, 2, and 3 are chosen). When the tests are completed, a summary display is obtained. Detailed descriptions of the commands used in the example are in Chapter 3.

Example 2-1 Sample Execution of a Diagnostic Program

```
DS> ATT KA62A HUB KA0 2
DS> SELECT KA0
DS> SET FLAGS TRACE
DS> RUN/TEST=1:3

.. Program: ELKAX - VAX 6200 Manual Tests, revision 1.0, 22 tests,
at 17:02:43.93.
Testing: _KA0

Test 1: Good Restart Parameter Block Test
Test 2: Valid RPB Search Test
Test 3: Bad RPB Checksum Test


End of run, 0 errors detected, pass count is 1,
   time is 9-AUG-1988 04:19:51.24

DS> SUMMARY
Summary of ELKAX - VAX 6200 Manual Tests, Rev 1.0:
0 program detected errors (0 Hard, 0 Soft, 0 System, 0 Device).
0 Supervisor detected errors.

DS>
```

3 The VAX/DS Command Language

The VAX/DS command language consists of a set of English-like commands. Most commands require one or more parameters and some accept qualifiers.

3.1 Introduction to the Commands

The VAX/DS commands can be divided into several groups, depending on their function.

Commands used to specify the hardware devices to be tested:

ATTACH
DEATTACH
BOOT
DESELECT
SELECT

Commands to control the execution of diagnostic programs:

ABORT
CONTINUE
LOAD
RUN
START
SUMMARY

Commands to set, clear, and determine the current state of various run-time parameters that affect the operation of diagnostic programs:

CLEAR
SET
SHOW

Commands designed to help programmers develop and debug new programs:

DEPOSIT
EXAMINE
NEXT

Miscellaneous commands:

DIRECTORY
EXIT
HELP
RECALL

3.2 Command Summary

```
ABORT
ATTACH uut-type link-name generic-name [device-specific]
BOOT node-number
CONTINUE
DEATTACH/ADAPTER=link-name [generic-name]
DEPOSIT address contents
DESELECT device-list
DIRECTORY file-spec
EXAMINE address
EXIT
HELP [subject]
LOAD file-spec
NEXT
RECALL
RUN[/PASSES= /SECTION= /TEST= /SUBTEST= /TIME= /QA] file-spec
SELECT device-list

SHOW    SET    CLEAR    BASE address
                        BREAKPOINT address
                        CALLS
                        CPU processor-name
                        DEFAULT display-mode[,display-mode]
                        DEVICE
                        ENFORCE
                        EVENT FLAGS event-flag[,event-flag,...]
                        FLAGS BELL
                              HALT
                              IE1
                              IE2
                              IE3
                              IES
                              LOOP
                              OPERATOR
                              PROMPT
                              QUICK
                              SEARCH
                              TRACE
                              VERIFY

                        LOAD [device-name:][directory-name]
                        MEMORY pages
                        MM mm-status
                        PAGE lines
                        QACKLOOPLOOPS number
                        QADEFAULTS
                        QAERRORPRINTS number
                        QAMULTIPLEPASS number
                        QASUBTESTLOOPS number
                        QATESTLOOPS number
                        SECTIONS
                        SELECTED
                        STATUS
                        SUPPORT
                        TESTS
                        VERIFY
                        WIDTH number

START[/PASSES= /SECTION= /TEST= /SUBTEST= /TIME= /QA]
SUMMARY
```

3.3 Control Characters

Under the VAX/DS, certain control characters have special meaning. The characters and their meanings are as follows:

- CTRL/C

  Stops execution of the current function and causes the VAX/DS to issue a user prompt. Functions that can be aborted include executing a diagnostic program and processing an ATTACH command. After a CTRL/C has been typed, a CONTINUE command can be used to cause the diagnostic program to resume execution.

- CTRL/U

  Deletes any characters previously typed in a command. Used to erase a typing error.

- CTRL/S

  Halts display of messages to the user terminal until a CTRL/Q, CTRL/C, or CTRL/Y is typed.

- CTRL/Q

  Resumes display of messages on user terminal.

- CTRL/R

  Retypes the current command line.

- CTRL/Y

  In user mode, forces immediate exit of VAX/DS and return to VMS.

- CTRL/Z

  Emulates EXIT command (see EXIT command description).

3.4 Conventions Used in the Command Descriptions

In the command descriptions, the following conventions will be used:

- Keywords (those parts of the example command line that must be typed exactly as shown) are capitalized.
- Parameters (those parts of the command line for which you must provide a specific name or value) are in lowercase letters.
- Optional arguments are enclosed in brackets.
- In the examples, user input is underlined.
- All VAX/DS commands and qualifiers can be abbreviated to the least number of characters that uniquely identify the command or qualifier. For example, the DIRECTORY command can be specified as DI and the START command as ST.

3.5 File Specifications

Certain VAX/DS commands, such as LOAD and RUN, require that a file be specified. A file specification may be in FILES-11, RT-11, or ULTRIX-32 format.

3.5.1 FILES-11 Format

Generally, storage devices formatted under VMS make use of FILES-11 format. A FILES-11 file specification has the following format:

```
device:[directory]filename.type;version
```

where

device =	The generic name of a hardware device existing on the system.
directory =	The name of a directory existing on the specified device. Subdirectories are allowed.
filename =	The name of the desired file. Wildcards are sometimes allowed, depending on the command being used.
type =	The file's 3-character file type. Wildcards are sometimes allowed, depending on the command being used.
version =	The file's version number. If not specified, the file name with the highest version number will be used.

All fields of the specification except "filename" can be omitted and default values used. (For the DIRECTORY command, the "filename" field can also be omitted.)

Device names can have up to 15 characters, directory names up to 39 characters (with a maximum of seven nested subdirectories), file name and type up to 39 characters, and version number up to 5 characters. The overall full file specification can be up to 252 characters, including punctuation as shown:

```
DEVICE:[DIRECTORY]FILENAME.TYPE;VERSION
|_15__|__39 each_|___39___|_39_|___5__|
```

A sample file specification is:

```
DMA0:[SYS0.SYSMAINT]EVRAD.EXE;3
```

The example refers to version number three of file EVRAD.EXE, which exists in subdirectory SYSMAINT of the directory named SYS0. located on device DMA0.

In user mode, logical names or logical name searchlists can be used for file specifications. An equivalence string may include all or only part of a directory specification. In the latter case you must specify the missing fields; otherwise, default values will be applied. For example, if the logical name SYS$SYSROOT is equivalent to $1$DUS0:[SYS3.], the missing subdirectory name must be specified either in the command line (for example, SYS$SYSROOT:[SYSMAINT]) in or the default directory (by way of the SET LOAD command).

Note: The VAX/DS does not support network nodes in file specifications.

3.5.2 RT-11 Format

RT-11 format is used for console device media. An RT-11 file specification has the following format:

```
device:filename.type
```

where

device =	The generic name of a hardware device existing on the system.
filename =	The name of the desired file. Wildcards are sometimes allowed, depending on the command being used. This field has a maximum length of 6 characters. Names greater than 6 characters are truncated.
type =	The file's 3-character file type. Wildcards are sometimes allowed, depending on the command being used.

All fields of the specification except "filename" can be omitted and default values used. (For the DIRECTORY command, the "filename" field can also be omitted.)

A sample file specification is:

```
CSA0:EVRLD.EXE
```

The example refers to file EVRLD.EXE located on device CSA0.

3.5.3 ULTRIX-32 Format

ULTRIX-32 format is used on storage devices created under the ULTRIX-32 operating system. An ULTRIX-32 file specification has the following format:

```
device:(x,y)pathname
```

where

device =	The generic name of a hardware device existing on the system
x =	The unit number of the device
y =	A partition on the device
pathname =	The path name of the file you want to reference

Path names are of the form dir/dir/dir.../filename, where each /dir is the path name to a subdirectory of the previous /dir. Wildcards are allowed. All fields except "filename" are optional. (For the DIRECTORY command, "filename" is also optional.)

An example of an ULTRIX-32 file specification is:

```
dma0:(0,6)usr/field/evrad.exe
```

Note: Upper and lowercase characters are not interchangeable in ULTRIX-32 path names.

3.6 Indicating Input Radix

All numeric values used in command lines are interpreted in the current default radix, unless specified otherwise. (See SET DEFAULT, which controls the current radix.) The default radix can be overridden when you specify a radix with the numeric value.

To specify a radix, precede the number with a percent sign and either O, D, or X to indicate octal, decimal, or hexadecimal, respectively. For example, %X200 indicates a hexadecimal 200, and %D1234 means 1234 in decimal radix.

3.7 VAX/DS Command Descriptions

Following are detailed descriptions of the VAX/DS commands.

ABORT

The ABORT command causes the diagnostic program's cleanup code to be executed. (See Section 1.5.)

Use this command if a program's execution is stopped during a program pass (by CTRL/C, halt-on-error, unexpected exception fielded by the VAX/DS, and so on), and you do not wish to type CONTINUE to allow the pass to complete. In this case, using the ABORT command will ensure that the device being tested is not left in an indeterminate state.

FORMAT **ABORT**

PARAMETERS *None.*

QUALIFIERS *None.*

EXAMPLE

```
DS> ABORT
```

ATTACH

The ATTACH command is used to specify a device on the system under test. A device cannot be accessed by a diagnostic program unless it has first been attached.

Attaching a device consists of issuing a set of ATTACH commands so that a series of links can be established from the processor to the device. (The autosizer can be used to automatically attach all existing devices.) The link that is closest to the processor is a pseudo-connection known as "HUB." All devices must be ultimately linked to HUB. This is illustrated by the example below.

FORMAT

ATTACH *uut-type link-name generic-name [device-specific-params]*

PARAMETERS

uut-type

Product name for the type of device being attached, such as RA90, KDB50, or DWMBA. (A list of valid device types can be obtained with the HELP DEVICE command.)

link-name

Generic name (see below) of this device's link. If the link-name is not HUB, it must have been previously defined as the generic-name parameter of another ATTACH command. See the example.

generic-name

Generic name of the device being described. Future references to this device will be by its generic name. Most devices have standard generic names that should be adhered to. The HELP command may be used to determine proper generic names for the various device types. (SET ENFORCE and CLEAR ENFORCE can be used to enable and disable enforcement of naming conventions for generic names.)

prompts

If the ATTACH command is issued without parameters, the user will be prompted for parameter values. For all devices, the first three prompts are:

Device Type?	uut-type
Device Link?	link-name
Device Name?	generic-name

Prompts that follow the first three differ for each device.

EXAMPLE

In this example, the device to be tested is a RA90 disk drive. The drive (DUA0) is linked to KDB50 controller (DUA), which is connected to DWMBA 0 (DWMBA0), which is in turn connected to HUB.

```
DS> ATTACH
Device Type? DWMBA
Device Link? HUB
Device Name? DWMBA0
XMI node # (1,2,3,4,B,C,D,E) ? 2
BI Node Number (HEX)? 5
DS> ATTACH
Device Type? KDB50
Device Link? DWMBA0
Device Name? DUA
BI Node Number (HEX)? 4
DS> ATTACH
Device Type? RA90
Device Link? DUA
Device Name? DUA0
```

The same example is now repeated, but this time all command parameters are specified on the ATTACH command line, so no prompting occurs:

```
DS> ATTACH DWMBA HUB DWMBA0 2 5
DS> ATTACH KDB50 DWMBA0 DUA 4
DS> ATTACH RA90 DUA DUA0
```

BOOT

The BOOT command boots another processor such that all subsequently invoked diagnostic programs will execute on the specified processor. This service is only available for use with the VAX 82XX/83XX series and the VAX 62XX/63XX series, and only in standalone mode.

On a VAX 82XX/83XX series system, after the BOOT command is issued, the target processor will run all the subsequently loaded diagnostics. However, the primary processor will execute routines to handle console I/O operations. Control is returned to the primary processor with the EXIT command, at which point the VAX/DS and diagnostic programs will again be executed by the primary processor.

On a VAX 62XX/63XX series system, the target processor of the boot command actually becomes the primary processor. Diagnostics need not be reloaded, but just started, to be run on the primary processor. All memory state is preserved. To return to the former primary, simply boot back.

FORMAT

BOOT *node-number*

PARAMETERS

node-number

The node number of the processor to be started. The default radix is used. For VAX 82XX/83XX systems, this is the BI node number; for VAX 62XX/63XX systems, this is the XMI node number.

QUALIFIERS

None.

EXAMPLES

```
DS> BOOT 4
```

```
DS> BOOT E
```

CLEAR

The CLEAR command is used to clear or reset parameters that have been previously set with the SET command (or were set by default).

Each parameter is discussed separately in the following pages.

FORMAT

CLEAR *run-time-param*

PARAMETERS

run-time-param

The run-time parameter to be cleared. Following is a list of the parameters:

BREAKPOINT
ENFORCE
EVENT FLAGS
FLAGS

QUALIFIERS

None.

CLEAR BREAKPOINT

CLEAR BREAKPOINT is used to cancel a previously set breakpoint.

FORMAT

CLEAR BREAKPOINT *address*

PARAMETERS

address

A numeric value representing the address at which the diagnostic program should break.

If the keyword ALL is used as the parameter, all set breakpoints will be cleared.

QUALIFIERS

/BASE=number

A numeric value representing the desired base address. If a radix is not specified, the default radix is used.

/NOBASE

Indicates that no base value should be added to the address specified.

EXAMPLES

```
DS> CLEAR BREAKPOINT 34E0

DS> CLEAR BREAK 44C4

DS> CLE BRE ALL

DS> CLEAR BRE/BASE=3000 4E0
```

CLEAR ENFORCE

CLEAR ENFORCE disables enforcement of VMS device naming conventions, allowing the user to specify generic names that do not follow VMS conventions for device names.

FORMAT

CLEAR ENFORCE

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLES

```
DS> CLEAR ENFORCE
DS> CLE ENF
```

CLEAR EVENT FLAGS

CLEAR EVENT FLAGS causes the specified event flags to be cleared. One or several flags can be specified.

FORMAT

CLEAR EVENT [FLAGS] *event-flag[,event-flag,...]*

PARAMETERS

event-flag

One of the event flags. Flags are numbered from 1 to 23. The keyword ALL can be used to indicate all event flags.

QUALIFIERS

None.

EXAMPLES

```
DS> CLEAR EVENT FLAGS 5,6

DS> CLEAR EVENT 8

DS> CLE EVE ALL
```

CLEAR FLAGS

CLEAR FLAGS clears the specified VAX/DS control flags. One or several flags can be specified.

FORMAT

CLEAR [FLAGS] *flag[,flag,...]*

PARAMETERS

flag

One of the VAX/DS control flags. The flags and their meaning are discussed under the description of the SET FLAGS command.

The keyword ALL can be used to indicate all control flags.

QUALIFIERS

None.

EXAMPLES

```
DS> CLEAR FLAGS IE2, IE3

DS> CLEAR HALT

DS> CLE ALL
```

CONTINUE

The CONTINUE command is used to resume execution of the diagnostic program after it has been stopped because of an error halt, a breakpoint, or the typing of a CTRL/C on the terminal. Execution will resume at the point at which it was stopped.

FORMAT

CONTINUE

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLE

```
DS> CONTINUE
```

DEATTACH

The DEATTACH command is used to reverse the ATTACH command; that is, the p-table for the specified device is deallocated.

FORMAT

DEATTACH/ADAPTER = link-name *[generic-name]*

PARAMETERS

generic-name

The generic name of the device to be deattached. It is the name defined by the generic-name parameter of the ATTACH command used to attach the device.

If this parameter is not specified, *all* devices attached to the link indicated by ''link-name'' including the link itself, will be deattached.

QUALIFIERS

/ADAPTER = link-name

The qualifier *must* be specified. It is used to indicate the link of the device to be deattached. This assures that the correct device is removed. The ''link-name'' parameter is the device's link as defined in the ATTACH command used to attach the device being removed.

EXAMPLES

To deattach device DUA from dwmba adapter DWMBA0, type:

```
DS> DEATTACH/ADAPTER=DWMBA0 DUA
```

To deattach DWMBA adapter DWMBA0 and all devices attached to that adapter, type:

```
DS> DEATTACH/ADAPTER=DWMBA0
```

DEPOSIT

The DEPOSIT command allows you to change the contents of a segment of memory. Locations whose contents can be modified include main memory addresses and processor registers.

If memory management is disabled, all locations in main memory can be modified. Addresses specified are interpreted as physical addresses.

If memory management is enabled (as is always the case in user mode), the user may be prohibited from accessing some areas of memory. However, all addresses into which the diagnostic program has been loaded will be accessible. Addresses specified are interpreted as virtual addresses.

Processor registers that can be modified include all general-purpose registers, and (in standalone mode only) the internal processor registers (IPRs). See Appendix A, Accessing Processor Registers.

The size of the memory segment to be modified and the format in which the specified new contents will be interpreted are determined by the command qualifiers. If qualifiers are not used, the default cell size and radix are used.

In a multiprocessing environment, use the SET CPU command to select the processor you wish to reference. You cannot reference an attached processor unless you have set a breakpoint for that processor and the breakpoint has been executed. You cannot modify the PC of an attached processor using the DEPOSIT command.

FORMAT

DEPOSIT *address contents*

PARAMETERS

address

The first address of the segment of memory whose contents are to be modified.

This address is expressed as a numeric value in the current default radix (see SET DEFAULT). Alternately, it can be a register name (see Appendix A, Accessing Processor Registers.)

contents

The new contents of the specified memory segment. This value is interpreted according to the qualifiers if they are specified, or else according to the default radix.

QUALIFIERS

/BYTE
/WORD
/LONGWORD

The /BYTE, /WORD, and /LONGWORD qualifiers are used to indicate the memory cell size to be used when a cell of memory is modified. If one of these qualifiers is not specified, the default cell size is used (see SET DEFAULT).

/OCTAL
/DECIMAL
/HEXADECIMAL

The /OCTAL, /DECIMAL, and /HEXADECIMAL qualifiers are used to indicate the radix in which the "address" and "contents" parameters are to be interpreted. If one of these qualifiers is not specified, the default radix is used (see SET DEFAULT).

/ASCII

Indicates that the "contents" parameter is to be interpreted as a string of ASCII characters.

/BASE = number

A numeric value representing the desired base address. If a radix is not specified, the default radix is used.

/NOBASE

Indicates that no base value should be added to the address specified.

/NEXT = number

Indicates the size of the memory segment, which succeeds the one at the specified location, to be modified. "Number" is the number of memory cells in the succeeding segment. For example, EXAMINE/NEXT=5/BYTE 200 12 will store 12 in each of 5 bytes starting at location 200.

If the /NEXT qualifier is not used, one memory cell is modified.

EXAMPLES

```
DS> DEPOSIT 1000 FFFFFFFF

DS> DEPOSIT/ASC 500 ABCD

DS> DEP/N=200/BYTE 1000 0

DS> DEP 4000 %D256
```

DESELECT

Removes devices from the list of devices to be tested. A diagnostic program will not test devices that are deselected. Devices specified in the DESELECT command must have been previously selected.

FORMAT

DESELECT *device-list*

PARAMETERS

device-list

List of generic names of devices to be deselected in the format

```
generic-name[,generic-name,...]
```

The device-list parameter can also be ALL, which causes all attached devices to be deselected.

QUALIFIERS

/ADAPTER = link-name

If the /ADAPTER qualifier is used, only those devices connected to the link specified by "link-name" will be deselected. This qualifier is only meaningful if the keyword ALL is used as the command's "device-list" parameter, as in the following example, which will deselect all RA90 drive units attached to the KDB50 controller named DUA:

```
DESELECT/ADAPTER=DUA ALL
```

EXAMPLES

```
DS> DESELECT DUA
DS> DESELECT DUA0,DUA1
DS> DESELECT/ADAPTER=DWMBA0 ALL
```

DIRECTORY

DIRECTORY displays the names of the files in the specified (or default) device directory.

In standalone mode, the device containing the directory to be displayed must have been previously attached (see ATTACH).

FORMAT

DIRECTORY *[file-spec]*

PARAMETERS

file-spec

Specifies the device, directory on the device, and, optionally, files within the directory. Wildcard characters are allowed in the file name, but not in the device or directory names. Defaults are taken for all unspecified fields of "file-spec." See Section 3.5, File Specifications.

QUALIFIERS

/WIDE

Prints one full file name per line. If you omit this qualifier, DIRECTORY lists file names in four columns.

EXAMPLES

```
DS> DIRECTORY DRB2:[SYS0.SYSMAINT]

DS> DIR [SYSMAINT]

DS> DIR SYS$MAINTENANCE:

DS> dir /usr/field  (ULTRIX-32 format)

DS> DIR
```

EXAMINE

The EXAMINE command allows the user to examine a segment of memory. Locations that can be examined include main memory addresses, processor registers, and I/O device registers.

If memory management is disabled, all locations in main memory can be examined. Specified addresses are interpreted as physical addresses.

If memory management is enabled (as is always the case in user mode), the user may be prohibited from accessing some areas of memory. However, all addresses into which the diagnostic program has been loaded will be accessible. Specified addresses are interpreted as virtual addresses.

Processor registers that can be examined include all general-purpose registers, and (in standalone mode only) the internal processor registers (IPRs). See Appendix A, Accessing Processor Registers.

The size of the memory segment to be displayed and the format in which the memory contents will be displayed are determined by the command qualifiers. If qualifiers are not used, the default cell size and radix will be used.

In a multiprocessing environment, use the SET CPU command to select the processor you wish to reference. You cannot reference an attached processor unless you have set a breakpoint for that processor and the breakpoint has been executed. You can examine only the PSW portion of an attached processor's PSL.

FORMAT

EXAMINE *address*

PARAMETERS

address

The first address of the segment of memory whose contents are to be displayed.

This address is expressed as a numeric value in the current default radix (see SET DEFAULT). Alternately, it can be a register name (see Appendix A, Accessing Processor Registers.)

QUALIFIERS

/BYTE
/WORD
/LONGWORD

The /BYTE, /WORD, and /LONGWORD qualifiers indicate the memory cell size to be used when the memory segment is displayed. If one of these qualifiers is not specified, the default cell size will be used (see SET DEFAULT).

/OCTAL
/DECIMAL
/HEXADECIMAL

The /OCTAL, /DECIMAL, and /HEXADECIMAL qualifiers indicate the radix in which the memory segment's addresses and contents are to be displayed. If one of these qualifiers is not specified, the default radix is used (see SET DEFAULT).

/ASCII

Indicates that the memory segment's contents are to be interpreted as ASCII characters.

/BASE = number

A numeric value representing the desired base address. If a radix is not specified, the default radix is used.

/NOBASE

Indicates that no base value should be added to the address specified.

/NEXT = number

Indicates the size of the memory segment, which succeeds the one at the specified location, to be displayed. "Number" is the number of memory cells in the succeeding segment. For example, EXAMINE/NEXT = 5/BYTE 200 displays a memory segment of five bytes starting at location 200.

If the /NEXT qualifier is not used, only the memory cell at the specified location is displayed.

EXAMPLES

```
DS> EXAMINE 400

DS> EXAMINE/BYTE 244

DS> E/ASCII/NEXT=10 346

DS> E/ASC/N=4 346

DS> E/OCT/WO %O400

DS> EXAM/BASE=200 200
```

EXIT

EXIT causes execution of the VAX/DS to halt. In user mode, control is returned to the VMS command line interpreter. In standalone mode, the console command interpreter takes control (see the *VAX Hardware Handbook*).

When you use the VAX/DS command BOOT to start an attached processor, use the EXIT command to halt the attached processor and return control to the primary processor.

FORMAT

EXIT

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLES

```
DS> EXIT
```

```
DS> EXI
```

HELP

HELP causes the VAX/DS help file and help files relating to specific diagnostic programs to be displayed. If the parameter is omitted, a list of available subjects will be displayed. In order to use this command, the VAX/DS help files must reside on the default load device and directory (see SET DEFAULT). The help file for the VAX/DS is ESSAA.HLP; the help file for a diagnostic program EVXYZ is EVXYZ.HLP.

FORMAT

HELP *[subject]*

PARAMETERS

subject

The subject for which help is desired. If no subject is specified, a list of available subjects will be displayed. For example, to obtain help for the diagnostic program EVXYZ, type HELP EVXYZ.

QUALIFIERS

None.

EXAMPLES

```
DS> HELP

DS> HELP DEVICE RP07

DS> HELP SET FLAGS

DS> HELP EVRAD
```

LOAD

The LOAD command causes the file specified by "file-spec" to be loaded into memory. LOAD clears all event flags.

FORMAT

LOAD *file-spec*

PARAMETERS

file-spec

The location (device and directory) and name of a diagnostic program. If the file type is omitted, .EXE is assumed. Wildcard characters are not allowed.

If the file location is omitted, the default load device and directory are assumed (see SET LOAD). In standalone mode, the load device must have been previously attached (see ATTACH). See Section 3.5 for a detailed description of file specifications.

QUALIFIERS

None.

EXAMPLES

```
DS> LOAD EVRAD

DS> LOAD [SYS0.SYSMAINT]ECXYZ

DS> LOAD SYS$MAINTENANCE:EVRAD

DS> load /field/evrad.exe       (ULTRIX-32 format)
```

NEXT

This command causes one or more instructions in the diagnostic program to be executed. The command can only be used after a breakpoint has been executed (see SET BREAKPOINTS). After the instructions are executed, the VAX/DS prompt is displayed.

In a multiprocessing environment, the NEXT command executes the next instruction only if the breakpoint was executed by the primary processor. You can use the NEXT command only with code in the primary processor.

FORMAT

NEXT *n*

PARAMETERS

n
Number of instructions to execute before stopping. The default is 1, which causes one instruction to be executed ("single-stepping").

QUALIFIERS

None.

EXAMPLES

```
DS> NEXT 4
DS> N 2
DS> N
```

RECALL

Displays previously entered commands so that you can reprocess them. Commands work in standalone mode *only*. If recall is typed without any parameters, the most recently executed command is recalled.

The UP arrow emulates the RECALL command. If the UP arrow is typed once, it is equivalent to RECALL 1; if the UP arrow is typed n times, it is equivalent to typing RECALL n.

The DOWN arrow will allow you to view a command that had already been retrieved by the UP arrow.

The RIGHT and LEFT arrows provide cursor control and can be used for editing a command line that was retrieved by the UP arrow.

FORMAT

RECALL *[n]*

PARAMETERS

n

Specifies the number or leading substring of the command that you wish to recall. Command numbers can range from 1 to 20. The most recently entered command is number 1.

QUALIFIERS

/ALL

Specifies that all the commands currectly stored in the RECALL buffer be displayed, along with their command numbers.

Note: **The RECALL command is never stored in the RECALL buffer.**

EXAMPLES

```
DS> RECALL 5
```

```
DS> RECALL COPY
```

```
DS> RECALL/ALL
```

RUN

The RUN command combines the functions of the LOAD and START commands. The specified diagnostic program will be loaded into memory and its execution will be started. RUN clears all event flags.

FORMAT

RUN *file-spec*

PARAMETERS

file-spec

The location (device and directory) and name of a diagnostic program. If the file type is omitted, .EXE is assumed. Wildcard characters are not allowed. If the file location is omitted, the default load device and directory are assumed (see SET LOAD). In standalone mode, the device must have been previously attached (see ATTACH). See Section 3.5 for a detailed description of file specifications.

QUALIFIERS

/PASSES = pass-count

Indicates the number of program passes the diagnostic program should cycle through before execution is stopped. (See Section 1.6, Program Passes.)

If the /PASSES qualifier is not used, the diagnostic program will execute one pass. (See SET FLAGS SEARCH for an exception.)

If the "pass-count" parameter is set to zero, the diagnostic program will execute an infinite number of passes.

/SECTION = section-name

Indicates which program section to execute. To find the names of sections of a particular program, type SHOW SECTIONS. If this qualifier is not used, the section named DEFAULT will be executed.

/TEST = first-test[:last-test]

Indicates which tests should be executed. It is possible to select all tests within a section, a range of tests, or one test. To find the names of tests of a particular program, type SHOW TESTS.

To execute all tests in the specified (or DEFAULT) section, do not use the /TEST qualifier.

To execute a range of tests, use the qualifier and include the range of tests, such as /TEST=4:9.

If the "last-test" parameter is omitted, it defaults to the last test in the section. For example, /TEST=4 would cause every test from test 4 through the last test in the specified (or DEFAULT) section to be executed.

To execute only one test in its entirety, specify the test number for both the "first-test" and "last-test" parameters, as in /TEST=3:3. If the /PASSES qualifier is also specified, a scope loop will be created for the test.

The specified tests must be components of the specified (or DEFAULT) section.

[/SUBTEST = subtest-num]

Indicates which subtests to execute.

To execute a range of tests, starting at the beginning of one test and stopping at a particular subtest within the last test of the specified range, use the /TEST and /SUBTEST qualifiers together, as in /TEST=5:7/SUBTEST=9. In this case, execution begins at test 5 and continues through subtest 9 of test 7. If the /PASSES qualifier is also specified, a scope loop begins executing once the specified subtest is reached. The loop, consisting of only the specified subtest, continues for the specified number of passes. Typically, if a loop-on-subtest is desired, only the test containing the desired subtest needs to be executed.

You can use the /SUBTEST qualifier alone to stop at a certain subtest in the last of a string of tests, as in /SUBTEST=5. In this case, if the diagnostic contains 4 tests, each test containing 10 subtests, then all 10 subtests in the first 3 tests run and only the first 5 subtests in the fourth test run.

/TIME = [dddd-][hh:mm:ss.cc]

Indicates the amount of time you want the diagnostic program to run. When the time expires, the program's cleanup code executes.

If you also specify the number of passes, the diagnostic programs stops either when the specified time elapses or when it reaches the number of passes, whichever occurs first.

If you interrupt the program with a CTRL/C or a CONTINUE command, the diagnostic program continues executing until it reaches the time you specify.

If you omit the /TIME qualifier or enter a value of zero, the program does not have a time limit.

dddd

Number of 24-hour days, (0–9999)

hh

Number of hours, (0–23)

mm

Number of minutes, (0–59)

ss

Number of seconds, (0–59)

cc

Number of hundredths of seconds, (0–59)

You can omit any time field provided you supply the punctuation marks. If you omit a field, it defaults to zero.

/QA

Invokes automated quality assurance (Auto-QA). (See Chapter 5 for details.)

EXAMPLES

To execute all tests in the default section for one pass, type:

```
DS> RUN EVXYZ
```

To execute all tests in the default section for five passes, type:

```
DS> RUN/PASSES=5 EVXYZ
```

To execute tests 1 through 8 of the section called NOWRITE, type:

```
DS> RUN/SECTION=NOWRITE/TEST=1:8 EVXYZ
```

To create a perpetual loop on test 4, type:

```
DS> RUN/TEST=4:4/PASS=0 EVXYZ
```

To run all tests for one and one-half hours, type:

```
DS> RUN/TIME=1:30
```

SELECT

The SELECT command selects devices for testing. A diagnostic program will test only devices that have been selected. Devices specified in the SELECT command must have been previously attached (see ATTACH).

FORMAT

SELECT *device-list*

PARAMETERS

device-list

List of generic names of devices to be selected in the format:

```
generic-name[,generic-name,...]
```

The device-list parameter can also be ALL, which causes all attached devices, with the exception of the load device, to be selected. (This prevents accidental testing and media destruction.)

QUALIFIERS

/ADAPTER = link-name

If the /ADAPTER qualifier is used, only those devices connected to the link specified by "link-name" will be selected. This qualifier is only meaningful if the keyword ALL is used as the command's "device-list" parameter, as in the following example, which will select all RA90 drive units attached to the KDB50 controller named DUA:

/NOALLOCATE

Prevents the selected device(s) from being allocated.

EXAMPLES

```
DS> SELECT DUA

DS> SELECT DUA0,DUA1

DS> SELECT/ADAPTER=DWMBA0 ALL

DS> SELECT/NOALLOCATE/ADAPTER=DWMBA0 ALL
```

SET

The SET command is used to set any of a group of run-time parameters affecting the execution of diagnostic programs. These parameters relate to execution options, debugging facilities, and quality assurance aids.

Each parameter is discussed separately in the following pages.

FORMAT

SET *run-time-param*

PARAMETERS

run-time-param

The run-time parameter to be set. Following is a list of the parameters:

BASE
BREAKPOINT
CPU
DEFAULT
ENFORCE
EVENT FLAGS
FLAGS
LOAD
MEMORY
MM
PAGE
QACHKLOOPLOOPS
QADEFAULTS
QAERRORPRINTS
QAMULTIPLEPASS
QASUBTESTLOOPS
QATESTLOOPS
VERIFY
WIDTH

QUALIFIERS

None.

SET BASE

SET BASE stores a base address value that will be automatically added to every address specified in an EXAMINE, DEPOSIT, SET BREAKPOINT, or CLEAR BREAKPOINTS command.

The default base address is 0.

FORMAT

SET BASE *address*

PARAMETERS

address
A numeric value representing the desired base address. If a radix is not specified, the default radix is used.

QUALIFIERS

None.

EXAMPLES

```
DS> SET BASE 200
```

```
DS> SET BASE %D512
```

SET BREAKPOINT

SET BREAKPOINT can be used to set a breakpoint within the diagnostic program. Once a breakpoint is set, a START or CONTINUE command will cause the diagnostic program to execute only up to the breakpoint address, at which point the VAX/DS prompt (DS>) will be displayed.

Up to 15 breakpoint settings can exist at one time.

When running a multiprocessor diagnostic program, you may set breakpoints in only one processor at a time. Setting breakpoints in more than one processor can cause unpredictable system behavior.

FORMAT

SET BREAKPOINT *address*

PARAMETERS

address
A numeric value representing the address at which the diagnostic program should break.

QUALIFIERS

/BASE = number
A numeric value representing the desired base address. If a radix is not specified, the default radix is used.

/NOBASE
Indicates that no base value should be added to the address specified.

EXAMPLE

```
DS> SET BREAKPOINT 34AC

DS> SET BRE/BASE=3000  4AC
```

SET CPU

SET CPU can be used when you run diagnostic programs that test multiprocessor systems. The command is used to select an attached processor for subsequent EXAMINE and DEPOSIT commands. If you want to examine or deposit the general-purpose registers (GPRs) or internal processor registers (IPRs) of an attached processor, you must first use the SET CPU command to point to the desired processor.

This command can only be used in a standalone, multiprocessing environment with diagnostic programs written specifically to make use of the multiprocessing services available in the VAX/DS.

If the SET CPU command is not used, all references to GPRs and IPRs will be to the primary processor by default. Examines and deposits to memory and I/O space are always performed by the primary CPU.

FORMAT

SET CPU *processor-name[:]*

PARAMETERS

processor-name

The device name of the processor whose registers are to be referenced. If you want to reference an attached processor, this argument is the name that you assigned to the processor with the ATTACH command. If you want to reference the primary processor, use the keyword PRIMARY.

QUALIFIERS

None.

EXAMPLES

```
DS> SET CPU KA0:

DS> SET CPU KA4

DS> SET CPU PRIMARY
```

SET DEFAULT

SET DEFAULT is used to set the default values for the two display mode parameters: **cell size** and **radix**. The display mode parameters are used to format numeric values of addresses and their contents when displayed with the EXAMINE and DEPOSIT commands. Cell size refers to the size of the memory units displayed. Radix refers to the radix of numeric values typed and displayed representing both location addresses and contents.

You can specify values for one or both parameters in one SET DEFAULT command. If both are given, their order does not matter.

FORMAT

SET DEFAULT *display-mode[,display-mode]*

PARAMETERS

display-mode

Indicates a display mode parameter. Values for "display-mode" can be:

Cell Size	Radix
BYTE	OCTAL
WORD	DECIMAL
LONGWORD	HEXADECIMAL

QUALIFIERS

None.

EXAMPLES

```
DS> SET DEFAULT BYTE

DS> SET DEFAULT HEXADECIMAL

DS> SET DEF LONG, OCT

DS> SET DEF DEC, BYTE
```

SET ENFORCE

SET ENFORCE causes the VAX/DS to enforce VMS device-naming conventions. An error message will be displayed if an attempt is made to assign to a device a generic name that does not conform to the VMS generic naming convention for the device (during use of the ATTACH command). For example, the VMS generic naming format for terminals is TTan, where "a" is an alphabetic character and "n" is a numeric character.

ENFORCE is set by default.

FORMAT

SET ENFORCE

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLE

```
DS> SET ENFORCE
```

SET EVENT FLAGS

SET EVENT FLAGS causes the specified event flags to be set. One or several flags can be specified.

Event flags have meaning only if the meanings have been defined by a diagnostic program. Refer to documentation for specific diagnostic programs to determine which event flags have been implemented, and what their meanings are.

The LOAD and RUN commands clear all event flags.

FORMAT

SET EVENT [FLAGS] *event-flag[,event-flag,...]*

PARAMETERS

event-flag

One of the event flags. Flags are numbered from 1 to 23. The keyword ALL can be used to indicate all event flags.

QUALIFIERS

None.

EXAMPLES

```
DS> SET EVENT FLAGS 1,2
DS> SET EVENT 8
DS> SET EVENT ALL
```

SET FLAGS

SET FLAGS sets the specified VAX/DS control flags. One or several flags can be specified.

FORMAT

SET [FLAGS] *flag[,flag,...]*

PARAMETERS

flag

One of the VAX/DS control flags. The flags and their meaning are as follows.

- BELL

  Bell-on-error. If set, an audible signal is emitted from the user's terminal whenever the diagnostic program detects an error.

- HALT

  Halt-on-error. If set, execution of the diagnostic program stops and the VAX/DS prompt (DS>) is displayed whenever the program detects a hardware failure.

- IE1

  Inhibit error message displays at message level 1 (the highest level). If set, messages identifying hardware failures are completely inhibited. (See Section 1.9.)

- IE2

  Inhibit error message displays at message level 2 (only print the first level). (See Section 1.9.)

- IE3

  Inhibit error message displays at message level 3 (only print levels 1 and 2). (See Section 1.9.)

- IES

  Inhibit automatic display of program summary. (Summary can still be displayed via SUMMARY command.)

- LOOP

  Loop-on-error. If set, the diagnostic program enters a scope loop within the failing subtest whenever a hardware failure is detected.

- OPERATOR

  Operator (user) present. Indicates to the VAX/DS that a user is present who can respond to requests made at the user terminal by the diagnostic program.

- PROMPT

  Provide extended user prompting messages. For all requests for user input, display the range of legal input values and the default value.

- QUICK

  Quick-verify. Indicates that the diagnostic program should execute in quick-verify mode. Quick-verify mode is not intended to be a thorough testing method. Refer to program documentation to see if a program supports this mode.

- SEARCH

  Error-search. If set and a test detects an error, subsequent program passes will execute only those selected tests having a lower test number than the failing test. (Note that if this flag is used, the default value for the /PASSES qualifier [see START or RUN] is changed from 1 to infinity.)

- TRACE

  Trace tests. Print the number and title of each test before it is executed.

- VERIFY

  If executing commands contained in a VAX/DS command file (script), print each command line on the user terminal as it is executed.

The keyword ALL can be used to indicate all control flags.

QUALIFIERS

None.

EXAMPLES

```
DS> SET FLAGS BELL, HALT, LOOP

DS> SET OPERATOR

DS> SET TRA, VER

DS> SET ALL
```

SET LOAD

SET LOAD sets the default load device and directory to the device and/or directory name specified.

When the VAX/DS loads a file into memory, if the device and directory are not specified explicitly, the file is read from the default directory name of the default device. If one or the other is specified, the VAX/DS will use the default for the unspecified parameter.

The SET LOAD command can be used to change either the device or the directory, or both. If only one parameter is specified, the other is not affected.

In standalone mode, a file cannot be loaded from a device unless the device has first been attached (see ATTACH).

The directory name is relevant only for those devices that support directory names.

When the VAX/DS is first loaded from a VMS-formatted device and started, the default directory is set to the directory named [SYS0.SYSMAINT]. (If this directory does not exist, [SYSMAINT] is used.) When the VAX/DS is loaded from an ULTRIX-32 device, the default directory is /usr/field.

In standalone mode, the default load device is the device from which the VAX/DS was loaded. In user mode, the default loadpath is SYS$MAINTENANCE.

FORMAT

SET LOAD *[device-name:][directory-name]*

PARAMETERS

device-name

The generic name of a device. The device can be any VAX/DS-supported file-oriented device existing on the system under test.

For ULTRIX-32 format, this field has the form device:[x, y], where x is the unit number and y is the partition.

directory-name

The name of a directory existing on the specified device.

For ULTRIX-32 format, this is a path name.

QUALIFIERS

None.

EXAMPLES

```
DS> SET LOAD DRB2:[SYSMAINT]

DS> SET LOAD [SYS0.SYSMAINT]

DS> SET LOAD DUA1:
```

SET MEMORY

SET MEMORY enables you to change the amount of physical memory available to the VAX/DS and diagnostic programs. SET MEMORY can be used to verify that a diagnostic program will run in a smaller amount of memory than is currently present on the system.

SET MEMORY deattaches and deselects all devices. However, p-tables for the console load device and the boot path devices are saved.

You cannot use SET MEMORY in user mode.

FORMAT

SET MEMORY *pages*

PARAMETERS

pages
Number of pages of memory. This value must be greater than or equal to zero and less than or equal to the total number of memory pages actually available on the system.

If you enter a value greater than the actual memory size, the command is ignored.

To reset the memory to the actual size, specify zero.

QUALIFIERS

None.

EXAMPLES

```
DS> SET MEMORY 262144      ! 262, 144 pages is equivalent to 128 Mg
```

SET MM

SET MM is used to enable and disable the memory management hardware.

In standalone mode, memory management is disabled unless enabled with the SET MM command.

In user mode, the user is not allowed to change memory management, which is permanently enabled. Thus, the SET MM command is invalid in user mode.

In a multiprocessing environment, if you issue a SET MM command after an attached processor is booted and a diagnostic is started and stopped with CTRL/C, the attached processor's memory management state does not change until you issue a CONTINUE command.

FORMAT

SET MM *mm-status*

PARAMETERS

mm-status
The state to which the memory management hardware is to be set. The status must be specified with one of the keywords ON or OFF.

QUALIFIERS

None.

EXAMPLES

```
DS> SET MM ON
```

```
DS> SET MM OFF
```

SET PAGE

SET PAGE is used to control the number of lines of terminal output (referred to as "page size") that will be displayed at one time. This command makes it possible to temporarily stop terminal output after a specified number of lines have been displayed. This ensures that no text will be scrolled out of view before it has been read.

Once the specified number of lines have been displayed, a prompting message will appear. If, for example, the page size has been set to 20 lines, a prompt will appear after 20 lines have been displayed and after you respond to the prompt, another 20 lines will be displayed.

FORMAT

SET PAGE *lines*

PARAMETERS

lines

Number of lines to be displayed before the prompting message appears.

A value of zero sets the page size to infinity, which has the effect of displaying all text immediately without prompting.

The default page size is 0.

Generally, page size for video terminals should be set to 20 lines.

QUALIFIERS

None.

EXAMPLES

```
DS> SET PAGE 20
DS> SET PAGE 0
```

SET QACHKLOOPLOOPS

Used only during the quality assurance phase of diagnostic program development. Refer to Chapter 5, Automated Quality Assurance (Auto-QA).

SET QADEFAULTS

Used only during the quality assurance phase of diagnostic program development. Refer to Chapter 5, Automated Quality Assurance (Auto-QA).

SET QAERRORPRINTS

Used only during the quality assurance phase of diagnostic program development. Refer to Chapter 5, Automated Quality Assurance (Auto-QA).

SET QAMULTIPLEPASS

Used only during the quality assurance phase of diagnostic program development. Refer to Chapter 5, Automated Quality Assurance (Auto-QA).

SET QASUBTESTLOOPS

Used only during the quality assurance phase of diagnostic program development. Refer to Chapter 5, Automated Quality Assurance (Auto-QA).

SET QATESTLOOPS

Used only during the quality assurance phase of diagnostic program development. Refer to Chapter 5, Automated Quality Assurance (Auto-QA).

SET VERIFY

Causes commands in a command file to be printed as they are executed.

FORMAT

SET VERIFY

QUALIFIERS

None.

EXAMPLE

```
DS> SET VERIFY
```

SET WIDTH

Indicates the number of characters to be printed per line of display on the user's terminal.

Note: **For versions of VAX/DS earlier than Version 6.11, this command is meaningful only in standalone mode. In user mode, use the DCL command SET TERMINAL/WIDTH = number.**

FORMAT

SET WIDTH *number*

PARAMETERS

number
A decimal numeric value in the range of 1 to 132. The default line width is 80.

QUALIFIERS

None.

EXAMPLE

```
DS> SET WIDTH 80
```

SHOW

The SHOW command displays the current value of parameters that can be set with the SET command.

Each parameter is discussed separately in the following pages.

FORMAT

SHOW *run-time-param*

PARAMETERS

run-time-param

The run-time parameter to be displayed. Following is a list of the parameters:

BASE
BREAKPOINTS
CALLS
CPU
DEFAULT
DEVICE
ENFORCE
EVENT FLAGS
FLAGS
LOAD
MEMORY
MM
QACHKLOOPLOOPS
QADEFAULTS
QAERRORPRINTS
QAMULTIPLEPASS
QASUBTESTLOOPS
QATESTLOOPS
SECTIONS
SELECTED
STATUS
SUPPORT
TESTS
WIDTH

SHOW BASE

SHOW BASE displays the currently set base address.

FORMAT

SHOW BASE

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLE

```
DS> SHOW BASE
```

SHOW BREAKPOINTS

SHOW BREAKPOINTS displays all set breakpoints.

FORMAT

SHOW BREAKPOINTS

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLES

```
DS> SHOW BREAKPOINTS
DS> SH BR
```

SHOW CALLS

SHOW CALLS displays information about all currently active call frames. The FP, saved PC, saved AP, and saved PSW of all frames are displayed.

FORMAT

SHOW CALLS

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLE

```
DS> SHOW CALLS
```

SHOW CPU

SHOW CPU can be used when running diagnostic programs that test multiprocessor systems. It is used to determine which processor's GPRs and IPRs will be referenced when the EXAMINE and DEPOSIT commands are used. Use the SET CPU command to select a particular processor for EXAMINEs and DEPOSITs.

FORMAT

SHOW CPU

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLE

```
DS> SHOW CPU
```

SHOW DEFAULT

SHOW DEFAULT displays the current default cell size and radix.

FORMAT

SHOW DEFAULT

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLES

```
DS> SHOW  DEFAULT
DS> SH DEF
```

SHOW DEVICE

SHOW DEVICE displays a list of the devices for which p-tables exist. These are the devices that were previously specified to the VAX/DS in ATTACH commands, were automatically attached by the VAX/DS at boot time, or were automatically attached by the autosizer. The display includes all device parameters that were included in the ATTACH command.

The parameter list is optional. If included, only those devices specified will be displayed. If no list is specified, all attached devices are displayed.

FORMAT

SHOW DEVICE *[generic-name[,generic-name,...]]*

PARAMETERS

generic-name
The generic name of an attached device.

QUALIFIERS

/ADAPTER = link-name
Displays only those devices connected to the specified link.

/BRIEF
Displays only the "generic-name" and "uut-type" device parameters for each device.

EXAMPLES

```
DS> SHOW DEVICE/ADAPTER=DWMBA0

DS> SH DEV

DS> SH DEV/BRIEF

DS> SHOW DEVICE DMA1
```

SHOW ENFORCE

SHOW ENFORCE displays the current status of the flag that enforces adherence to VMS device naming conventions. (See SET ENFORCE for details.)

FORMAT

SHOW ENFORCE

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLES

```
DS> SHOW ENFORCE
DS> SH ENF
```

SHOW EVENT FLAGS

SHOW EVENT FLAGS displays the current status of all event flags.

FORMAT

SHOW EVENT [FLAGS]

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLES

```
DS> SHOW EVENT FLAGS
```

```
DS> SH EVE
```

SHOW FLAGS

SHOW FLAGS displays the status of all VAX/DS control flags.

FORMAT

SHOW FLAGS

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLES

```
DS> SHOW FLAGS
DS> SH FL
```

SHOW LOAD

SHOW LOAD displays the current default load device and directory name.

FORMAT

SHOW LOAD

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLES

```
DS> SHOW LOAD
DS> SH LO
```

SHOW MEMORY

SHOW MEMORY displays the memory allocations made to various major portions of the VAX/DS and the diagnostic program. This display indicates the size and location of the major components of the diagnostic software. Items listed include:

- The diagnostic program
- The VAX/DS
- Buffer space available to the diagnostic program
- Various buffers allocated to the VAX/DS

In standalone mode, the display includes such system-level data structures as the page tables and the system control block. In user mode, these tables are controlled by VMS and are not available for display. If memory management is disabled, the addresses displayed are physical addresses. If memory management is enabled (as in user mode), the addresses are virtual.

FORMAT

SHOW MEMORY

PARAMETERS

None.

QUALIFIERS

/ALL
Displays all available information, including buffer space and data structures.

/BUFFER
Displays only buffer space allocations.

/DATASTRUCTURE
Displays only system-level data structures.

/MAP
Displays all available information except buffers and data structures. Issuing the SHOW MEMORY command without any qualifiers is equivalent to using only the /MAP qualifier.

EXAMPLES

```
DS> SHOW MEMORY/ALL

DS> SH MEM/BUFFER

DS> SH MEM
```

SHOW MM

Displays the current status of the memory management hardware (whether memory management is enabled or disabled).

This command is valid only in standalone mode. In user mode, memory management is always enabled.

FORMAT

SHOW MM

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLES

```
DS> SHOW MM
```

```
DS> SH MM
```

SHOW QACHKLOOPLOOPS

Used only during the quality assurance phase of diagnostic program development. Refer to Chapter 5, Automated Quality Assurance (Auto-QA).

SHOW QADEFAULTS

Used only during the quality assurance phase of diagnostic program development. Refer to Chapter 5, Automated Quality Assurance (Auto-QA).

SHOW QAERRORPRINTS

Used only during the quality assurance phase of diagnostic program development. Refer to Chapter 5, Automated Quality Assurance (Auto-QA).

SHOW QAMULTIPLEPASS

Used only during the quality assurance phase of diagnostic program development. Refer to Chapter 5, Automated Quality Assurance (Auto-QA).

SHOW QASUBTESTLOOPS

Used only during the quality assurance phase of diagnostic program development. Refer to Chapter 5, Automated Quality Assurance (Auto-QA).

SHOW QATESTLOOPS

Used only during the quality assurance phase of diagnostic program development. Refer to Chapter 5, Automated Quality Assurance (Auto-QA).

SHOW SECTIONS

SHOW SECTIONS displays a list of the names of all program sections in the currently loaded diagnostic program.

FORMAT **SHOW SECTIONS**

PARAMETERS *None.*

QUALIFIERS *None.*

EXAMPLES

```
DS> SHOW SECTIONS
DS> SH SEC
```

SHOW SELECTED

SHOW SELECTED displays a list of all devices that have been selected for testing by the use of the SELECT command. The display includes all device parameters specified when the device was attached.

FORMAT

SHOW SELECTED

PARAMETERS

None.

QUALIFIERS

/BRIEF

Displays only the "generic-name" and "uut-type" device parameters.

EXAMPLES

```
DS> SHOW SELECTED/BRIEF
DS> SH SEL
```

SHOW STATUS

Displays the current execution status of the currently loaded diagnostic program. Information displayed includes current section, test, subtest, and program counter, along with current pass and number of errors detected.

This command is useful primarily after a CTRL/C has been typed during the execution of a diagnostic program.

FORMAT

SHOW STATUS

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLES

```
DS> SHOW STATUS
DS> SH STA
```

SHOW SUPPORT

SHOW SUPPORT displays a list of device types that the currently loaded diagnostic program is capable of testing.

FORMAT

SHOW SUPPORT

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLES

```
DS> SHOW SUPPORT
DS> SH SUP
```

SHOW TESTS

SHOW TESTS displays the number and title of each test in the currently loaded diagnostic program.

FORMAT

SHOW TESTS

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLES

```
DS> SHOW TESTS

DS> SH T
```

SHOW WIDTH

SHOW WIDTH displays the number of characters to be printed per line on the user's terminal.

Note that for versions of the VAX/DS earlier than Version 6.11, this command is meaningful only in standalone mode. In user mode, the DCL command SHOW TERM/WIDTH should be used.

FORMAT

SHOW WIDTH

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLES

```
DS> SHOW WIDTH

DS> SH WI
```

START

The START command causes the current diagnostic program (loaded with the most recent LOAD command) to begin execution.

FORMAT

START

PARAMETERS

None.

QUALIFIERS

/PASSES = pass-count

Indicates the number of program passes the diagnostic program should cycle through before execution is stopped. (See discussion of program passes, Section 1.6.)

If the /PASSES qualifier is not used, the diagnostic program will execute one pass. (See SET FLAGS SEARCH for an exception.)

If the "pass-count" parameter is set to zero, the diagnostic program will execute an infinite number of passes.

/SECTION = section-name

Indicates which program section to execute. To find the names of a particular program's sections, type SHOW SECTIONS. If this qualifier is not used, the section named DEFAULT will be executed.

/TEST = first-test[:last-test]

Indicates which tests should be executed. It is possible to select all tests within a section, a range of tests or one test. To find the names of a particular program's tests, type SHOW TESTS.

To execute all tests in the specified (or DEFAULT) section, do not use the /TEST qualifier at all.

To execute a range of tests, use the qualifier and include the range of tests, such as /TEST=4:9. If the "last-test" parameter is omitted, it defaults to the last test in the section. For example, /TEST=4 would cause every test from test 4 through the last test in the specified (or DEFAULT) section to be executed.

To execute only one test in its entirety, specify the test number for both the "first-test" and "last-test" parameters, as in /TEST=3:3. If the /PASSES qualifier is also specified, a scope loop will be created for the test.

The specified tests must be components of the specified (or DEFAULT) section.

[/SUBTEST = subtest-num]
Indicates which subtests to execute.

To execute a range of tests, starting at the beginning of one test and stopping at a particular subtest within the last test of the specified range, use the /TEST and /SUBTEST qualifiers together, as in /TEST = 5:7/SUBTEST = 9. In this case, execution begins at test 5 and continues through subtest 9 of test 7. If the /PASSES qualifier is also specified, a scope loop begins executing once the specified subtest is reached. The loop, consisting of only the specified subtest, continues for the specified number of passes. Typically, if a loop-on-subtest is desired, only the test containing the desired subtest needs to be executed.

You can use the /SUBTEST qualifier alone to stop at a certain subtest in the last of a string of tests, as in /SUBTEST = 5. In this case, if the diagnostic contains 4 tests, and each test contains 10 subtests, all 10 subtests in the first 3 tests run and only the first 5 subtests in the fourth test run.

/QA
Invokes automated quality assurance (Auto-QA). (See Chapter 5 for details.)

/TIME = [dddd-][hh:mm:ss.cc]
Indicates the amount of time you want the diagnostic program to run. When the time expires, the program's cleanup code executes.

If you also specify the number of passes, the diagnostic program stops either when the specified time elapses or when it reaches the number of passes, whichever occurs first.

If you interrupt the program with a CTRL/C and then issue a CONTINUE command, the diagnostic program continues executing until it reaches the time you specify.

If you omit the /TIME qualifier or enter a value of zero, the program does not have a time limit.

dddd

Number of 24-hour days, (0-9999)

hh

Number of hours, (0-23)

mm

Number of minutes, (0-59)

ss

Number of seconds, (0-59)

cc

Number of hundredths of seconds, (0-59)

You can omit any time field provided you supply the punctuation marks. If you do omit a field, it defaults to zero.

EXAMPLES

To execute all tests in the default section for one pass, type:

```
DS> START
```

To execute all tests in the default section for five passes, type:

```
DS> START/PASSES=5
```

To execute tests 1 through 8 of the section called NOWRITE, type:

```
DS> START/SECTION=NOWRITE/TEST=1:8
```

To create a perpetual loop on test 4, type:

```
DS> START/TEST=4:4/PASS=0
```

To execute all tests for one hour and twenty minutes, type:

```
DS> START/TIME=1:20
```

SUMMARY

The SUMMARY command is used to obtain a terminal display of the diagnostic program's execution history. This execution history will always include the total number of errors detected on the devices that were tested. Additional information contained in the display is unique to each diagnostic program. (Some programs may not provide additional information.)

FORMAT

SUMMARY

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLE

```
DS> SUMMARY
```

4 Command Files

It is often necessary to issue the same set of VAX/DS commands repeatedly. For example, running a particular diagnostic program requires issuing a set of ATTACH and SELECT commands, perhaps issuing the SET FLAGS and SET EVENT FLAGS commands, and then issuing the RUN command. To avoid typing the entire command list every time the program is run, you can place the list of commands in a command file, sometimes called a **script file**.

4.1 Creating and Running Command Files

A command file contains a set of VAX/DS commands and is created with an editor. Commands are placed in the file exactly as they would appear if they were being typed directly to the VAX/DS. An example of a command file is:

```
DS> ATTACH DWMBA HUB DWMBA0 4 2
DS> ATTACH KDB50 DWMBA0 DUA 3
DS> ATTACH RA90 DUA DUA0
DS> SELECT ALL
DS> RUN/PASS:3 EVRLB
```

To cause the commands in this file to be executed, at the VAX/DS prompt, type the following command where "file-spec" is the name and location of the command file. If the file type is not specified, .COM is the default:

```
DS> @file-spec
```

4.2 Rules for Creating Command Files

The following rules apply to command files:

- The VAX/DS prompt must be included in the command file at the beginning of each command line that is issued to the VAX/DS. The prompt must consist of 4 characters: capital D, capital S, right angle bracket, and space, DS> .
- The command file must contain a response for every question the diagnostic program asks during its execution. This response must also include the prompting message that would normally be displayed on the user terminal. For example, suppose the following dialogue takes place when a program is executed without a command file:

  ```
  Disk write-protected? YES
  How many grandchildren do you have? 14
  ```

  If a command file was created for this program, the command file would have to contain all the text contained in both the prompt strings and the responses.

- To determine exactly what has to be included in a command file for a certain program, run the program without a command file. Then create a command file that contains all the prompts and responses that were needed to run the program.
- It is possible to force the VAX/DS to fetch some responses from the user terminal. To do this, include the prompt in the command file but do not include the response. Instead, follow the prompt with the 2 slash characters (\\). When the VAX/DS sees these two characters after a prompt, it will fetch the response from the user terminal. For example, suppose a command file contained the following line of text:

  ```
  Disk write-protected? \\
  ```

  In this case, the VAX/DS would fetch the response to the prompt from the user terminal. Program execution would not continue until the user had typed a response.
- A few diagnostic programs are coded so that some responses *must* come from the user's terminal. The user cannot alter this requirement. The only way of knowing if a particular response will be forced to come from the user terminal is to create a command file and try it. (Programmers force responses to come from the user terminal by preceding the prompt string with a null character.)
- Command files can contain comments. Comments begin with the exclamation point (!) character and continue to the end of the line.

  ```
  ! This is a comment.
  DS> RUN/PASSES=4 EVXYZ  ! This is another comment.
  ```
- Command files can be "nested" (one file may call another, which in turn calls a third, and so on).
- Command files should be named with a file type of .COM. This is a convention, not a requirement.

4.3 Autocom

A command file can be executed automatically at boot time. The command file must be named VDSSCRIPT.COM, and bit <15> must be set in R5 (see Table 2-1). This is supported on all VAX systems except 11/78X and 86XX systems, and 82XX/83XX console booting.

5 Automated Quality Assurance (Auto-QA)

To help the programmer determine the quality of a diagnostic program, the VAX/DS provides an automated quality assurance feature called Auto-QA. This feature will automatically perform some (but not all) of the quality assurance checks listed in the *VAX Diagnostic Design Guide*.

5.1 Quality Assurance Checks Performed by Auto-QA

Following is a list of the quality assurance checks performed by Auto-QA. Note that Auto-QA only checks the DEFAULT program section. Quality assurance of other program sections must be performed manually.

1 Normal Start Check

This check performs a normal load and execution of the diagnostic program with the TRACE flag set.

The program must make an error-free pass, printing out the normal trace messages and terminating with End-of-Pass. If the program does not execute an error-free pass, an appropriate QA error message is printed. The trace messages must be visually checked by the user.

This check also makes sure that the DEFAULT section does not request input from the user. (The OPERATOR flag is cleared.)

This check is equivalent to the following sequence of VAX/DS commands:

```
DS> CLEAR FLAG ALL
DS> SET FLAG TRACE
DS> RUN diagnostic-program-name
DS> CLEAR FLAG TRACE
```

2 Multiple Passes Check

This check executes 10 passes (by default) of the diagnostic program. The program must make ten error-free passes and terminate after the tenth pass. If this does not happen, an error message is printed.

The number of passes executed by the diagnostic program can be changed by the user.

This check is equivalent to the following VAX/DS command:

```
DS> START/PASS:10
```

3 Infinite Loop-on-Test Check

This check executes each test in the diagnostic program's DEFAULT section 100 times (by default). The diagnostic must execute each test the given number of times. If the diagnostic does not execute properly, an error message will be printed.

The number of times each test is executed can be changed by the user.

This check is equivalent to the following VAX/DS commands:

```
DS> START/PASS:100/TEST:1:1
DS> START/PASS:100/TEST:2:2
      .
      .
      .
DS> START/PASS:100/TEST:n:n
```

where n is the highest numbered test in the DEFAULT section. The tests are executed in ascending order.

4 Infinite Loop-on-Subtest Check

This check executes each subtest in each of the tests in the diagnostic program's DEFAULT section 100 times (by default). The program must loop on each subtest the given number of times. If the program does not execute properly, an error message is printed. The number of times each subtest is executed can be changed by the user.

This check is equivalent to the following Supervisor commands:

```
DS> START/PASS:100/TEST:1:1/SUBTEST:1
DS> START/PASS:100/TEST:1:1/SUBTEST:2
      .
      .
      .
DS> START/PASS:100/TEST:1:1/SUBTEST:m1
DS> START/PASS:100/TEST:2:2/SUBTEST:1
DS> START/PASS:100/TEST:2:2/SUBTEST:2
      .
      .
      .
DS> START/PASS:100/TEST:2:2/SUBTEST:m2
      .
      .
      .
DS> START/PASS:100/TEST:n:n/SUBTEST:1
DS> START/PASS:100/TEST:n:n/SUBTEST:2
      .
      .
      .
DS> START/PASS:100/TEST:n:n/SUBTEST:mx
```

where n is the highest numbered test in the DEFAULT section, and mx is the number of subtests in test x.

The tests and subtests are executed in ascending order.

5 Run Individual Tests in Reverse Order Check

This check executes the tests in the diagnostic program's DEFAULT section in reverse order. This check ensures that a test does not depend on results from a previous test, and that each test is a standalone entity. If the diagnostic program does not execute properly, an error message is printed. This check is equivalent to the following VAX/DS commands:

```
DS> START/TEST:n:n
DS> START/TEST:n-1:n-1
      .
      .
      .
DS> START/TEST:1:1
```

where n starts at the highest numbered test in the DEFAULT section and descends to the first test.

5.2 Execution of Diagnostic Programs Using Auto-QA

This section describes how to execute diagnostic programs using the Auto-QA feature of the VAX/DS.

5.2.1 VAX/DS Control Flags Under Auto-QA

The state of the VAX/DS control flags is saved when the START/QA or RUN/QA command is executed. During the Auto-QA sequence, some flags may be set or cleared by the Auto-QA feature. The diagnostic program should not change these flags.

When Auto-QA exits, it restores the state of the VAX/DS control flags to what they were before the START/QA (or RUN/QA) command.

5.2.2 Auto-QA Commands

Following is a list of commands that pertain to Auto-QA:

- RUN /QA [/TEST:first[:last]]#diagnostic-program-name

  The RUN command allows a /QA qualifier. This causes Auto-QA to first load the diagnostic program and then perform the Auto-QA checks on it.

  The /TEST qualifier is also allowed on the RUN command with the /QA qualifier. This causes only those tests within the range specified by "first" and "last" to be included in the Auto-QA.

  When the /QA qualifier is given on the RUN command, the /SUBTEST, /SECTION, and /PASSES qualifiers are ignored.

- START /QA [/TEST:first[:last]]

  The START command allows a /QA qualifier. This causes Auto-QA to perform the Auto-QA checks on the previously loaded diagnostic program.

  The /TEST qualifier is also allowed on the START command with the /QA qualifier. This causes only those tests within the range specified by "first" and "last" to be included in the Auto-QA.

  When the /QA qualifier is given on the START command, the /SUBTEST, /SECTION, and /PASSES qualifiers are ignored.

- SET QAMULTIPLEPASS n

  This command sets the number of passes that the Multiple Pass check performs to the value of "n." The value must be greater than zero.

- SET QASUBTESTLOOPS n

  This command sets the number of times that each test is executed in the Infinite Loop-on-Test check to the value of "n." The value must be greater than 1.

- SET QATESTLOOPS n

  This command sets the number of times that each subtest is executed in the Infinite Loop-on-Subtest check to the value of "n." The value must be greater than zero.

- SET QADEFAULTS

  This command sets the above three Auto-QA symbols to their default values. These defaults are as follows:

Symbol	Default Value
QAMULTIPLEPASS	10
QASUBTESTLOOPS	100
QATESTLOOPS	100

- SHOW QAMULTIPLEPASS

  This command shows the number of passes that the Multiple Pass check performs.

- SHOW QASUBTESTLOOPS

  This command shows the number of times that each test is executed in the Infinite Loop-on-Test check.

- SHOW QATESTLOOPS

  This command shows the number of times that each subtest is executed in the Infinite Loop-on-Subtest check.

- SHOW QADEFAULTS

  This command shows the default values for the three Auto-QA symbols.

- HELP QA

  This command prints help for the various Auto-QA-related Diagnostic Supervisor commands. Help on various Auto-QA subtopics is available with the command

  ```
  DS> HELP QA subtopic-name
  ```

5.2.3 Auto-QA Output Messages

Following are the formats of the various messages displayed by Auto-QA.

- Auto-QA Header Messages

  Each Auto-QA check will start on a new page with the following as header:

```
************************** QUALITY ASSURANCE **************************
         PROGRAM: prog-name  REV: revision-number.update-number
                                    date

                              Auto-QA-check-name
```

The "prog-name," "revision-number," and "update-number" will be taken from the diagnostic header (the information given on the $DS_HEADER macro). The "date" will be the system date (for standalone systems, this is not necessarily the actual date). The "Auto-QA-check-name" is the name of the current Auto-QA check being performed.

For example, here is a printout of the Normal Start Check header output for a diagnostic program (the example shown is not a "real" diagnostic program):

```
************************** QUALITY ASSURANCE **************************
          PROGRAM: Jack's Test Diagnostic Program REV: 5.7
                       21-JAN-1983 10:42:39.10

                                Normal Start

.. Program: Jack's Test Diagnostic Program, revision 5.7, 3 tests,
   at 10:42:39.11.

Test 1: ***** Test One
Test 2: ***** Test Two
Test 3: ***** Test Three

.. End of run, 0 errors detected, pass count is 1,
   time is 21-JAN-1983 10:42:39.21
```

- Auto-QA Error Messages

  Error messages output by Auto-QA will be in the following format:

```
**QA ERROR** error-message
```

  Error messages are displayed in pairs; the first message of the pair describes the error, and the second message indicates the type of check that was performed. For example, consider the following:

```
**QA ERROR** Error reported in cleanup code
**QA ERROR** Infinite loop-on-test check failed
```

  This pair of error messages indicates that the diagnostic program tried to report a hardware fault from within the cleanup code (this is not allowed). The Auto-QA test that was running at the time was the "infinite loop-on-test check."

  Each such set of error messages is followed immediately by the Error Dump. This dump is described in the next section.

  Table 5-1 lists all the error messages provided by Auto-QA.

Table 5-1 Auto-QA Error Messages

Error Name	Error Description
QA ERROR	Error reported (See Note 1.)
QA ERROR	Error reported in cleanup code (See Note 2.)
QA ERROR	Program requested operator intervention (See Note 3.)
QA ERROR	Normal start check failed
QA ERROR	Multiple pass check failed
QA ERROR	Infinite loop-on-test check failed
QA ERROR	Infinite loop-on-subtest check failed
QA ERROR	Run individual tests in reverse order check failed

Note 1: **This message indicates that the diagnostic program attempted to report a hardware fault when all hardware was assumed to be functioning properly.**

Note 2: **Indicates that the diagnostic program attempted to report a hardware fault from within the cleanup code.**

Note 3: **Indicates that the program attempted to request operator intervention when it should not have (such as from within the DEFAULT section).**

- Overall Error Summary Table

  The following shows the format of the Overall Error Summary Table (and the resulting "passed" message) for a diagnostic program that did not have any Automatic Quality Assurance errors.

```
           Overall Error Summary Table
           ---------------------------
QA Checklist Item                Number of Errors
-----------------                ----------------
Normal Start............................0
Multiple Pass (10 passes)...............0
Infinite Loop-On-Test (100 passes).......0
Infinite Loop-On-Subtest (100 passes)....0
Run Tests Backwards.....................0

** QA ** Jack's Test Diag passed Auto-QA portion of Quality Assurance
Checklist
```

  The following shows the format of the Overall Error Summary Table for a diagnostic program that had an error in the Infinite Loop-on-Test check. Note that Auto-QA aborts at the first error it detects, and the subsequent checks are not executed.

```
           Overall Error Summary Table
           ---------------------------
QA Checklist Item                Number of Errors
-----------------                ----------------
Normal Start............................0
Multiple Pass (10 passes)...............0
Infinite Loop-On-Test (100 passes).......1
```

A Accessing Processor Registers

You can access all general-purpose registers (GPRs) using the EXAMINE and DEPOSIT commands. However, examining or depositing into a GPR is not meaningful unless a diagnostic program that was executing is stopped by a CTRL/C or breakpoint.

Note that the VAX/DS does not refer to the GPRs directly, but to locations where the contents of the registers are saved when the diagnostic program is stopped.

GPR names recognized by the VAX/DS are:

R0	R4	R8	R12	or	AP	PSL
R1	R5	R9	R13	or	FP	
R2	R6	R10	R14	or	SP	
R3	R7	R11	R15	or	PC	

The VAX/DS can also refer to internal processor registers (IPRs). You can access IPRs only in standalone mode because you must use the privileged instructions MFPR and MTPR.

IPRs are in the format P followed by a number in the current radix. For example, the P0 base register is P8; the system ID register is P3E (hex). Note, as a special case, that you cannot access IPR C (hex) by PC, which refers to general register 15. To access IPR C, you can use P12 when the default radix is decimal, P0C when the default radix is hexidecimal, or P%XC to force hexadecimal interpretation when the default is any radix.

Table A–1 lists the IPRs by name, their respective numbers (in decimal and hexadecimal), the type of access allowed, and which processors contain them.

Table A–1 Internal Processor Registers

Register Name	Mnemonic	Dec	Hex	Access	Processor
Kernel stack pointer	KSP	0	0	R/W	All
Executive stack pointer	ESP	1	1	R/W	All
Supervisor stack pointer	SSP	2	2	R/W	All
User stack pointer	USP	3	3	R/W	All
Interrupt stack pointer	ISP	4	4	R/W	All
P0 base register	P0BR	8	8	R/W	All
P0 length register	P0LR	9	9	R/W	All
P1 base register	P1BR	10	A	R/W	All
P1 length register	P1LR	11	B	R/W	All
System base register	SBR	12	C	R/W	All
System limit register	SLR	13	D	R/W	All
Process control block base	PCBB	16	10	R/W	All
System control block base	SCBB	17	11	R/W	All
Interrupt priority level	IPL	18	12	R/W	All
AST level	ASTLVL	19	13	R/W	All
Software interrupt request	SIRR	20	14	W	All
Software interrupt summary	SISR	21	15	R/W	All
Interprocessor interrupt request	IPIR	22	16	W	82XX/83XX
Interval clock control	ICCS	24	18	R/W	All
Next interval count	NICR	25	19	W	All
Interval count	ICR	26	1A	R	All
Time of year	TODR	27	1B	R/W	All (option)
Console receiver control/status	RXCS	32	20	R/W	All
Console receiver data buffer	RXDB	33	21	R	All
Console transmit control/status	TXCS	34	22	R/W	All
Console transmit data buffer	TXDB	35	23	W	All
Translation buffer group disable	TBDR	36	24	R/W	82XX/83XX
Cache disable	CADR	37	25	R/W	82XX/83XX/62XX/63XX
Machine check error summary	MCESR	38	26	R/W	82XX/83XX/87XX/88XX
Accelerator control/status	ACCS	40	28	R/W	All
Console Saved PC	SAVPC	42	2A	R	62XX/63XX
Console Saved PSL	SAVPSL	43	2B	R	62XX/63XX
WCS address	WCSA	44	2C	R/W	78X/82XX/83XX
WCS data	WCSD	45	2D	R/W	78X/82XX/83XX

Table A-1 (Cont.) Internal Processor Registers

Register Name	Mnemonic	Dec	Hex	Access	Processor
WCS CAM	WCSCAM	46	2E		82XX/83XX
SBI fault/status	SBIFS	48	30	R/W	78X
SBI silo	SBIS	49	31	R	78X
SBI silo comparator	SBISC	50	32	R/W	78X
SBI maintenance	SBIMT	51	33	R/W	78X
SBI error register	SBIER	52	34	R/W	78X
SBI timeout address	SBITA	53	35	R	78X
SBI quadword clear	SBIQC	54	36	W	78X
Initialize UNIBUS	IORESET	55	37	W	62XX/63XX
Memory management enable	MAPEN	56	38	R/W	All
Translation buffer invalid all	TBIA	57	39	W	All
Translation buffer invalid single	TBIS	58	3A	W	All
Translation buffer	TBDATA	59	3B	R/W	
Microprogram breakpoint	MBRK	60	3C	R/W	78X
Performance monitor enable	PME	61	3D	R/W	All except 62XX/63XX
System identification	SID	62	3E	R	All
Translation buffer check	TBCHK	63	3F	W	All
PAMM access	PAMACC	64	40	R/W	86XX
PAMM location	PAMLOC	65	41	R/W	86XX
Cache sweep	CSWP	66	42	R/W	86XX
MBOX data ECC	MDECC	67	43	R/W	86XX
MBOX error enable	MENA	68	44	R/W	86XX
MBOX data control	MDCTL	69	45	R/W	86XX
MBOX MCC control	MCCTL	70	46	R/W	86XX
MBOX error generator	MERG	71	47	R/W	86XX
Console reboot	CRBT	72	48	W	86XX
Diagnostic fault insert	DFI	73	49	W	86XX
Error handling status	EHSR	74	4A	R/W	86XX
Console block storage C/S	STXCS	76	4C	R/W	86XX
Console block storage D/B	STXDB	77	4D	R/W	86XX
EBOX scratchpad address	ESPA	78	4E	W	86XX
EBOX scratchpad data	ESPD	79	4F	R	86XX
Serial-Line Unit1 Rcv CSR	RXCS1	80	50	R/W	82XX/83XX
NMI Interrupt Enable	NMION	80	50	W	87XX/88XX
Serial-Line Unit1 Rcv Dat	RXDB1	81	51	R	82XX/83XX
Interrupt Other Processor	INOP	81	51	W	87XX/88XX
Serial-Line Unit1 Xmt CSR	TXCS1	82	52	R/W	82XX/83XX

Table A–1 (Cont.) **Internal Processor Registers**

Register Name	Mnemonic	Dec	Hex	Access	Processor
NMI Fault/Status Reg	NMIFSR	82	52	R	87XX/88XX
Serial-Line Unit1 Xmt Dat	TXDB1	83	53	W	82XX/83XX
NMI Bus Silo	NMISILO	83	53	R	87XX/88XX
Serial-Line Unit2 Rcv CSR	RXCS2	84	54	R/W	82XX/83XX
NMI Error Address Reg	NMIEAR	84	54	R	87XX/88XX
Serial-Line Unit2 Rcv Dat	RXDB2	85	55	R	82XX/83XX
Cache On Register	COR	85	55	R/W	87XX/88XX
Serial-Line Unit2 Xmt CSR	TXCS2	86	56	R/W	82XX/83XX
Revision Register #1	REVR1	86	56	R	87XX/88XX
Serial-Line Unit2 Xmt Dat	TXDB2	87	57	W	82XX/83XX
Revision Register #2	REVR2	87	57	R	87XX/88XX
Serial-Line Unit3 Rcv CSR	RXCS3	88	58	R/W	82XX/83XX
Clear time out status	CLRTOSTS	88	58	R/W	87XX/88XX
Serial-Line Unit3 Rcv Dat	RXDB3	89	59	R	82XX/83XX
Serial-Line Unit3 Xmt CSR	TXCS3	90	5A	R/W	82XX/83XX
Serial-Line Unit3 Xmt Dat	TXDB3	91	5B	W	82XX/83XX
Receive Console Data	RXCD	92	5C	R/W	82XX/83XX
Cache Invalidate	CACHEX	93	5D	W	82XX/83XX
VAXBI Node Identification	BINID	94	5E	R	82XX/83XX
VAXBI Stop	BISTOP	95	5F	W	82XX/83XX

B Building Bootable Media for Use in Standalone Mode

B.1 Building a Bootable Medium for a Nonconsole Disk

All nonconsole device media must use Files-11 structure level 2 format. This is the default format used by VAX/VMS.

The medium must contain the following files:

- VMB.EXE, the primary bootstrap. This is placed on the boot medium for VAX 82XX/83XX/62XX/63XX systems. On other systems, VMB.EXE exists on the standard console medium.
- DIAGBOOT.EXE, the secondary bootstrap used for booting the VAX/DS.

  Note that VMB and DIAGBOOT must be contiguous files.
- EnSAA.EXE, the VAX/DS where n is specific to the processor type of the system on which the VAX/DS will be executed (see Table 2-2).
- The diagnostic programs.
- EVQxxx.EXE, the device drivers that are needed if level 2 diagnostic programs are to be executed. xxx indicates the device type.

You should also include the following files:

- EVSAA.HLP, the help file for the VAX/DS
- Help files for the diagnostic programs
- EVSBA.EXE, the autosizer program

DIAGBOOT.EXE and EnSAA.EXE must be placed in a directory named either [SYS0.SYSMAINT] or [SYSMAINT]. The former name is preferred. (Versions of the VAX/DS earlier than Version 6.6 must use the directory name [SYSMAINT].) The other files may be placed in any directory you wish. If DIAGBOOT.EXE and EnSAA.EXE are not placed in one of these directories, you must perform a **conversational boot**. See Section 2.1.2.1.

The following command procedure is an example of creating a bootable RA60 disk pack:

```
$ ALLOCATE DJA0                        ! Allocate an RA60 drive
$ INITIALIZE DJA0: DIAG                ! Initialize the disk, setting
                                       ! the volume name to DIAG
                                       ! (This step is unnecessary if the disk
                                       ! has been initialized previously.)
$ MOUNT DJA0: DIAG                     ! Mount the disk.
$ CRE/DIR DJA0:[SYS0.SYSMAINT]         ! Create directory.
$ COPY/CONTIG DIAGBOOT.EXE,-           ! Secondary bootstrap
        VMB.EXE                        ! Primary bootstrap (82XX/83XX/
                                       ! 62XX/63XX systems)
$ COPY E*SAA.EXE,-                     ! VAX/DS
        EVQ*.EXE,-                     ! Level 2 device drivers
        EVKAC.EXE,EVRAD.EXE,-          ! A couple of diagnostic programs
        EVSAA.HLP,-                    ! VAX/DS help file
        EVKAC.HLP,EVRAD.HLP,-          ! Help files for the diagnostic programs
        EVSBA.EXE -                    ! The autosizer
        DJA0:[SYS0.SYSMAINT]           ! Put everything here.
$ DISMOUNT DJA0                        ! Dismount. All done.
```

Note: **Prior to Version 4 of DIAGBOOT, which was released with Version 6.6 of the VAX/DS, the VAX/DS (E*SAA.EXE) had to be written as a contiguous file (using the /CONTIG switch on the COPY command).**

The commands listed above are described in the *VAX/VMS Command Language User's Guide.*

An additional step is required when you create a bootable disk for VAX 82XX/83XX/62XX/63XX and VS8000 systems. You must run the RSX-11M utility WRITEBOOT under VMS to create a pointer to the primary bootstrap program (VMB.EXE), which you have already copied onto the medium. (For all other systems, the primary bootstrap is located on the standard console medium.)

The following example of WRITEBOOT dialogue shows how to create a bootable RA60 disk to be used on a VAX 62XX:

```
$ MCR WRITEBOOT
Target system device (and boot file if not VMB.EXE): DJA0:[SYS0.SYSMAINT]
Enter VBN of boot file code (default is 1): <RET>
Enter load address of primary bootstrap in HEX (default is 200): <RET>
$
```

B.2 Building a Bootable Medium for a Console Device

The console media must contain the following files:

- EnSAA.EXE, the VAX/DS where n is specific to the processor type of the system on which the VAX/DS will be executed (see Table 2-2).
- The diagnostic programs.
- EVQxxx.EXE, the device drivers that are needed if level 2 diagnostic programs are to be executed. xxx indicates the device type.

You should also include the following files:

- EVSAA.HLP, the help file for the VAX/DS
- Help files for the diagnostic programs
- EVSBA.EXE, the autosizer program

B.2.1 Building Console Media for the VAX 11/780, 11/785, and VAX 8600/8650 Systems

Console media for VAX 11/78X and VAX 86XX systems must use RT-11 format. This format can be obtained when you run the RSX-11M utility EXCHANGE under VAX/VMS.

The following command procedure is an example of creating a bootable TU58 console tape:

```
$ ALLOCATE DDA0:                                 ! Allocate a TU58 drive.
$ MOUNT/FOREIGN DDA0:                            ! Mount with non-VMS file
                                                 ! structure.
$ EXCHANGE                                       ! Run the EXCHANGE utility.
 EXCHANGE> INIT DDA0:                            ! Initialize in RT-11 format.
 EXCHANGE> COPY E*SAA.EXE/RECORD=FIXED DD0:      ! Copy the VAX/DS.
 EXCHANGE> COPY DIAG.NAME/RECORD=FIXED DD0:      ! Copy diagnostic programs.
 EXCHANGE> DIR DD0:                              ! Display directory.
^Z                                               ! Leave EXCHANGE.
```

Refer to the *VAX/VMS Exchange Utility Reference Manual* for details on the use of EXCHANGE, the file transfer utility.

B.2.2 Building Console Media for the VAX 8200/8250/8300/8350 Systems

Console media for VAX 82XX/83XX systems must use Files-11 structure level 2 (ODS2) format. The following procedure is an example of creating an RX50 console diskette:

```
$ ALLOC DUC0:                                    ! Allocate RX50 disk drive.
$ INIT DUC0:                                     ! Initialize disk.
$ MOUNT DUC0:                                    ! Mount disk.
$ CREATE/DIR DUC0:[SYSMAINT]                     ! Create a directory.
$ COPY EBSAA.EXE DUC0:[SYSMAINT]                 ! Copy the VAX/DS.
$ COPY DIAGNOSTIC.NAME DUC0:[SYSMAINT]           ! Copy diagnostic programs.
```

Then run the RSX-11M utility WRITEBOOT under VMS to create a pointer to the VAX/DS, which you have already copied onto the medium.

The following example of WRITEBOOT dialogue shows how to create a bootable RX50 console disk a VAX 82XX/83XX:

```
$ MCR WRITEBOOT
Target system device (& boot file if not VMB.EXE): DUC0:[SYSMAINT]EBSAA.EXE
Enter VBN of boot file code (default is 1): 2
Enter load address of primary bootstrap in HEX (default is 200): 10000
$
```

B.2.3 Building Console Media for the VAX 8530/8550/8700/8800/8820N Systems

Console media for VAX 85XX/8700/8800/8820N systems must use Files-11 structure level 1 format (ODS1). The following procedure is an example of creating an RX50/RX53 console diskette:

```
$ ALLOC DUC0:                           ! Allocate RX50 disk drive.
$ INIT/STRUCTURE=1 DUC0:                ! Initialize disk.
$ MOUNT DUC0: DIAG                      ! Mount disk, label DIAG.
$ CREATE/DIR DUC0:[USERFILES]           ! Create a directory.
$ COPY EZSAA.EXE DUC0:[USERFILES]       ! Copy the VAX/DS.
$ COPY *.* DUC0:[USERFILES]             ! Copy diagnostic programs.
$ DISMOUNT DUC0:                        ! Dismount.
```

Remove the floppy from the drive and insert it in drive number 1 (top) of the console. Exit the console program. At the dollar ($) prompt, type:

```
$ COPY DZ1:[USERFILES]*.* LB0:[CONSOLE]*.*
```

B.2.4 Building Console Media for the VAX 8820/8830/8840 Systems

Console media for VAX 8820/8830/8840 systems must use Files-11 formats. The following procedure is an example of creating a TK50 tape cartridge:

```
$ ALLOC MUA0:                           ! Allocate drive.
$ INIT MUA0: DIAG
$ MOUNT/BLOCK_SIZE=512 -                ! Mount, name volume DIAG.
      MUA0: DIAG
$ COPY VMB.EXE,-                        ! Copy DIAGBOOT, VMB, and the VAX/DS.
      DIAGBOOT.EXE,EJSAA.EXE MUA0:
$ DISMOUNT/NOUNLOAD MUA0:
$ MOUNT/BLOCK_SIZE=32768 MUA0: DIAG
$ COPY *.* MUA0:                        ! Copy diagnostics and help files.
$ DISMOUNT MUA0:
```

Insert the TK50 into the microvax console on the 8820/8830/8840 system, and copy the files to the RD53 winchester disk. At the microvax prompt, type the following commands:

```
$ MOUNT DUA0:
$ COPY VMB.EXE,DIAGBOOT.EXE,EJSAA.EXE DUA0:[PSTAR.SYS]
$ COPY *.EXE,*.HLP DUA0:[CONSOLE]
$ DISMOUNT DUA0:
```

B.2.5 Building Console Media for VS8000 System

Console media for VS8000 system must use ANSI format. The following procedure is an example of creating a bootable TK50 tape cartridge:

```
$ ALLOC MUL6:                           ! Allocate RX50 disk drive.
$ INIT MUL6: DIAG                       ! Initialize disk.
$ MOUNT/BLOCK_SIZE=512 MUL6: DIAG       ! Mount disk, label DIAG.
$ COPY VMB.EXE, DIAGBOOT.EXE,-
           EBSAA.EXE MUL6:              ! Copy VMB, DIAGBOOT and the VAX/DS.
$ DISMOUNT/NOUNLOAD MUL6:
$ MOUNT/BLOCK_SIZE=16384 MUL6: DIAG     ! Remount with larger block size.
$ COPY *.* MUL6:                        ! Copy diagnostic programs.
$ DISMOUNT MUL6:                        ! Dismount.
```

B.2.6 Building Console Media for the VAX 6210/6220/6230/6240/6310/6320/6340/6350/6360 Systems

Console media for the VAX 62XX/63XX systems must use ANSI format. The following procedure is an example of creating a bootable TK50/TK70 tape cartridge:

```
$ ALLOC MUB6:                                 ! Allocate TK50/TK70 drive.
$ INIT MUB6: DIAG                             ! Initialize disk.
$ MOUNT/BLOCK_SIZE=512 MUB6: DIAG             ! Mount disk, label DIAG.
$ COPY VMB.EXE,CIBCA.BIN,DIAGBOOT.EXE,-       ! Copy VMB, DIAGBOOT and
          ELSAA.EXE MUB6:                     ! the VAX/DS,
                                              ! (copy CIBCA.BIN if CIBCA-AA or
                                              ! CIBCA-AB on system and
                                              ! you will use CI).
$ DISMOUNT/NOUNLOAD MUB6:
$ MOUNT/BLOCK_SIZE=32768 MUB6: DIAG           ! Remount with larger block size.
$ COPY *.* MUB6:                              ! Copy diagnostic programs.
$ DISMOUNT MUB6:                              ! Dismount.
```

Index

VAX/DS
Diagnostic Supervisor
User's Guide
AA-FK66A-TE

READER'S COMMENTS

Your comments and suggestions help us to improve the quality of our publications.

For which tasks did you use this manual? (Circle your responses.)

(a) Installation	(c) Maintenance	(e) Training
(b) Operation/use	(d) Programming	(f) Other (Please specify.) ________

Did the manual meet your needs? Yes ☐ No ☐ Why? ________

____

Please rate the manual in the following categories. (Circle your responses.)

	Excellent	Good	Fair	Poor	Unacceptable
Accuracy (product works as described)	5	4	3	2	1
Clarity (easy to understand)	5	4	3	2	1
Completeness (enough information)	5	4	3	2	1
Organization (structure of subject matter)	5	4	3	2	1
Table of Contents, Index (ability to find topic)	5	4	3	2	1
Illustrations, examples (useful)	5	4	3	2	1
Overall ease of use	5	4	3	2	1
Page Layout (easy to find information)	5	4	3	2	1
Print Quality (easy to read)	5	4	3	2	1

What things did you like *most* about this manual? ________

____

____

What things did you like *least* about this manual? ________

____

____

Please list and describe any errors you found in the manual.

Page	Description/Location of Error
____	________
____	________
____	________
____	________

Additional comments or suggestions for improving this manual: ________

____

____

Name ________	Job Title ________
Street ________	Company ________
City ________	Department ________
State/Country ________	Telephone Number ________
Postal (ZIP) Code ________	Date ________

Fold Here and Tape

Affix
Stamp
Here

DIGITAL EQUIPMENT CORPORATION
CORPORATE USER PUBLICATIONS
200 FOREST STREET MRO1-3/L12
MARLBOROUGH, MA 01752-9101

Fold Here

VAX/DS
Diagnostic Supervisor
User's Guide
AA-FK66A-TE

READER'S COMMENTS

Your comments and suggestions help us to improve the quality of our publications.

For which tasks did you use this manual? (Circle your responses.)

(a) Installation	(c) Maintenance	(e) Training
(b) Operation/use	(d) Programming	(f) Other (Please specify.) ______

Did the manual meet your needs? Yes ☐ No ☐ Why? ______

______

Please rate the manual in the following categories. (Circle your responses.)

	Excellent	Good	Fair	Poor	Unacceptable
Accuracy (product works as described)	5	4	3	2	1
Clarity (easy to understand)	5	4	3	2	1
Completeness (enough information)	5	4	3	2	1
Organization (structure of subject matter)	5	4	3	2	1
Table of Contents, Index (ability to find topic)	5	4	3	2	1
Illustrations, examples (useful)	5	4	3	2	1
Overall ease of use	5	4	3	2	1
Page Layout (easy to find information)	5	4	3	2	1
Print Quality (easy to read)	5	4	3	2	1

What things did you like *most* about this manual? ______

______

______

What things did you like *least* about this manual? ______

______

______

Please list and describe any errors you found in the manual.

Page	Description/Location of Error
______	______
______	______
______	______
______	______

Additional comments or suggestions for improving this manual: ______

______

______

Name ______	Job Title ______
Street ______	Company ______
City ______	Department ______
State/Country ______	Telephone Number ______
Postal (ZIP) Code ______	Date ______

Fold Here and Tape

Affix
Stamp
Here

DIGITAL EQUIPMENT CORPORATION
CORPORATE USER PUBLICATIONS
200 FOREST STREET MRO1-3/L12
MARLBOROUGH, MA 01752-9101

Fold Here