CONTROL DATA

60469720

Diagnostic Virtual System for NOS/VE

Usage

Diagnostic Virtual System (DVS) for NOS/VE

Usage

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features and parameters.

Publication Number 60469720

Manual History

Technical changes and additions are indicated by change bars and are correlated with the revision of the page on which they occur. Other changes, such as editorial and pagination, are not identified by change bars, but may be included as part of a revision.

Revision	System Version	PSR Level	Date
A	NOS/VE 1.1.1 DVS 1.1	613	June 1984
B	NOS/VE 1.1.2 DVS 1.2	630	March 1985
C	NOS/VE 1.1.3 DVS 1.3	644	September 1985
D	NOS/VE 1.1.4 DVS 1.4	649	December 1985
E	NOS/VE 1.2.1 DVS 2.1	664	September 1986
F	NOS/VE 1.2.2 DVS 2.2	678	April 1987
G	NOS/VE 1.2.3 DVS 2.3	688	September 1987
H	NOS/VE 1.3.1 DVS 3.1	700	April 1988
J	NOS/VE 1.4.1 DVS 4.1	716	December 1988
K	NOS/VE 1.5.1 DVS 5.1	741	June 1990
L	NOS/VE 1.5.2 DVS 5.2	757	September 1990
M	NOS/VE 1.6.1 DVS 6.1	780	August 1991

Revision M documents DVS version 6.1 under NOS/VE version 1.6.1 at PSR level 780.

Revision letters I, O, Q, S, X, and Z are not used.

Contents

About This Manual

This manual describes the CONTROL DATA® Diagnostic Virtual System (DVS) Utility. DVS exercises a defined set of diagnostic programs on line under CDC® Network Operating System/Virtual Environment (NOS/VE).

Availability

DVS and NOS/VE are available on the following systems:

CYBER 170 Models 815, 825, 835, and 855.
CYBER 180 Models 810, 830, 835, 840, 845, 850, 855, and 860.
CYBER 930, 932, 960, 962, 990, 990E, 992, 994, 995E, and 2000.

Audience

This manual is a reference for customer engineers and for site personnel, authorized to run online maintenance software. It assumes that you understand NOS/VE and System Command Language (SCL) concepts as presented in the SCL Language Definition and SCL System Interface manuals.

Organization

The Diagnostic Virtual System manual is divided into two parts. Part I, Tutorial, describes the basics for running DVS. Part II, Usage, contains more detailed information for the more experienced user.

- Part I contains four chapters that give an overview of DVS. They include essential information about the access and use of the utility.
- Part II contains 15 chapters. Chapters 5 through 19 provide detailed command and parameter information intended for use primarily by engineering and technical support personnel. These chapters briefly describe the maintenance software tests. For more detailed test information, refer to the MSL15X Model Independent Tests Maintenance Software Reference Manual listed in the Related Manuals section of this manual.
- Appendix A provides a glossary of terms used in this manual.
- Appendix B provides a list of related manuals.
- Appendix C provides error message templates.

Conventions

All numbers are assumed to be decimal unless otherwise noted.

Technical changes and additions are indicated by change bars and are correlated with the revision of the page on which they occur.

Related Manuals

The following manuals are either referenced in this manual or used for background information. Refer to the SCL for NOS/VE Language Definition manual for a complete list of NOS/VE manuals. Refer to the CYBER 170/180 Systems Virtual State Hardware Reference Manual, Volume 2, for more details about the hardware.

Manual Title	Publication Number
CYBER 170/180 Systems Virtual State Hardware Reference, Volume 2	60458890
SCL for NOS/VE Language Definition Usage	60464013
SCL for NOS/VE Quick Reference	60464018
SCL for NOS/VE System Interface Usage	60464014
Diagnostic Messages for NOS/VE Usage	60464013
Diagnostic Virtual System for NOS/VE Usage	60469720
MSL15X Model Independent Tests Maintenance Software Reference	60469390
NOS/VE Analysis Usage	60463915
NOS/VE Operations Usage	60463914

Online Information Manual

If your site has installed the online manuals, you can find an abstract for each NOS/VE manual in the online System Information Manual. To access this manual, enter the following command.

```
explain
```

Submitting Comments

Use the comment sheet at the back of this manual to give us your opinion of the manual's usability, to suggest specific improvements, and to report errors. In the absence of a comment sheet, mail your comments to:

Control Data
Technical Publications ARH219
4201 Lexington Avenue N.
St. Paul, MN 55126-6198

Please indicate whether you would like a written response.

If you have access to SOLVER, the Control Data facility for reporting problems, you can use it to submit comments about the manual. When entering your comments, use NV0 (zero) as the product identifier. Include the name and publication number of the manual.

Ordering Manuals

To order a Control Data manual, send an order form to:

Control Data
Literature and Distribution Services ARHLDS
4201 Lexington Avenue N.
St. Paul, MN 55126-6198

To obtain an order form or to get more information about ordering Control Data manuals, write to the above address or call (612) 482-3800 or 482-3801, or FAX your inquiry to (612) 482-3813. If you are a Control Data employee, use the Controlnet number 235-3800, 235-3801, or 235-3813.

Part I: Tutorial

Introduction 1

Introduction 1

This chapter introduces the DVS Command Utility and subcatalog Hardware_Maintenance.

DVS Command Utility

DVS is a NOS/VE command utility that controls online diagnostic test execution. Programs used by DVS are enhanced versions of offline diagnostics. They employ the same testing strategy but contain unique code for online interfacing.

DVS commands execute diagnostics in two modes: batch and interactive. Execution of all or a subset of diagnostics can be selected in both modes. Diagnostics can be sequenced one at a time, executed concurrently as multiple tasks, or run as multiple jobs. Each selection gives a different level of confidence and stress test capability.

DVS is most frequently run in batch mode. You can start test execution by using one of three commands: RUN_DVS, RUN_CPU_DIAGNOSTICS, and RUN_SYSTEM_DIAGNOSTICS. Batch mode is convenient for the purposes of background, confidence, and system stress testing.

When you require specific test control, you can run all tests interactively. To start the utility from an interactive session, use the DVS_UTILITY command. Executing in an interactive session gives access to all DVS commands described in the manual. These commands are categorized as follows:

Invalid Diagnostic Selections

CYBER 810, 815, 825, and 830 Systems

Selection of BYTE, EDIT, VINT, VCMP, VGTH, and VGSI are invalid.

CYBER 835 Systems

Selection of BIMM, NUMR, RFST, DBUG, BYTE, EDIT, VINT, VCMP, VGTH, and VGSI are invalid.

CYBER 840, 845, 850, 855, and 860 Systems

Selection of the DBUG, VINT, VCMP, VGTH, and VGSI are invalid.

CYBER 93X Systems

Selection of VINT, VCMP, VGTH, and VGSI are invalid.

CYBER 96X Systems

Selection of DBUG, NUMR, VINT, VCMP, VGTH, and VGSI are invalid.

CYBER 99X Systems

Selection of VGSI is invalid.

CYBER 2000 Systems

Selection of FCT3, and NUMR are invalid.

Subcatalog Hardware_Maintenance

DVS is packaged as a group of library files residing in subcatalog DVS of Hardware_Maintenance in family $SYSTEM, user $SYSTEM. Three files are used for command processing and diagnostic test interfacing. The online manual resides in file DVS_USAGE. You can access it as follows:

```
EXPLAIN m=$system.hardware_maintenance.dvs.dvs_usage
```

A history summary for batch execution is contained in file HISTORY. The remaining files are the diagnostic test program libraries.

Access to the Hardware_Maintenance subcatalog is available to maintenance users and the system operator from the console. If you require remote access, you must obtain permission from the site administrator.

DVS files on Hardware_Maintenance.DVS are listed below by name and usage.

File Name	Use	Access Mode
DVS_COMMAND_LIBRARY	Contains utility command processors	Read, Execute
INTLIB	Interface program between utility and CPU diagnostics	Read, Execute
HSTINT	Interface program between utility and system diagnostics	Read, Execute
DVS_USAGE	Online DVS Usage manual	Read
RCT1LIB	Random Command Test 1	Read, Execute
RCT2LIB	Random Command Test 2	Read, Execute
FCT3LIB	Fixed Command Test 3	Read, Execute
SNGLLIB	Single-Precision Floating-Point Test	Read, Execute
DOBLLIB	Double-Precision Floating-Point Test	Read, Execute
BRCHLIB	Floating-Point Branch Test	Read, Execute
FINTLIB	Full-Word Integer Test	Read, Execute
HINTLIB	Half-Word Integer Test	Read, Execute
FIMMLIB	Full-Word Immediate Test	Read, Execute
HIMMLIB	Half-Word Immediate Test	Read, Execute
BIMMLIB	BDP Immediate Test	Read, Execute
NUMRLIB	BDP Numeric Test	Read, Execute
BYTELIB	BDP Byte End-Case Test	Read, Execute

File Name	Use	Access Mode
EDITLIB	BDP Edit End-Case Test	Read, Execute
VINTLIB	Vector Integer Test	Read, Execute
VCMPLIB	Vector Compare Test	Read, Execute
VGTHLIB	Vector Gather/Scatter Test	Read, Execute
VGSILIB	Vector Gather/Scatter Indexed Text	Read, Execute
RFSTLIB	Random Fast/Slow Test	Read, Execute
DBUGLIB	Debug Test	Read, Execute
DBUG_LIBRARY	Debug Manager	Read, Execute
CMEMLIB	Central Memory Test	Read, Execute
DISKLIB	Disk Test	Read, Execute
TAPELIB	Tape Test	Read, Execute
HISTORY	Batch mode history summary	Read, Shorten, Append, Modify

NOTE

DVS files are distributed with the NOS/VE BASE_2 products tape. The Hardware_Maintenance.DVS subcatalog and DVS files are installed without user permission. To use DVS from a remote terminal, obtain access by creating the appropriate file access permission.

For example, to create access permission for registered user XYZ on family NVE, you would use the following CREATE_CATALOG_PERMIT command:

```
CRECP catalog=$system.hardware_maintenance group=user
      family_name=nve user=xyz
```

To create read, shorten, append, and modify access permission for the HISTORY file, you would use the following CREATE_FILE_PERMIT command:

```
CREFP file=$system.hardware_maintenance.dvs.history
      group=user family_name=nve user=xyz access_mode=all
```

System Access 2

System Access 2

This chapter describes how to gain access to DVS for interactive and batch jobs.

User Validation

You can use DVS from the system console or by remote user login. Validation is necessary for remote login to NOS/VE. That is, you must have been assigned a user name within a NOS/VE family.

To run DVS, you need permit access to the system's Hardware_Maintenance subcatalog. DVS commands and test library files are maintained on the DVS subcatalog within Hardware_Maintenance.

DVS commands are generally available to maintenance users only. To access these commands, obtain permission from the system site administrator.

Initiating a Session from the System Console

You must obtain access to the DVS_COMMAND_LIBRARY before initiating DVS from the system console. To include DVS commands in the system command list, use the CREATE_COMMAND_LIST_ENTRY command as follows:

```
CRECLE e =$system.hardware_maintenance.dvs.dvs_command_library
```

Initiating a Remote Interactive Session

To log in to NOS/VE from an interactive terminal, you must be validated to access the dual-state NOS system. Your family name, user name, and password must be the same for both systems. The initial login dialog for an interactive user is with the NOS Network Access Method (NAM). All terminal input is received first by NAM and then passed to NOS/VE. DVS is written to support a CDC 721 terminal.

When initiating a NOS login sequence, you see lines similar to these displayed on your terminal.

```
WELCOME TO THE NOS SOFTWARE SYSTEM.
COPYRIGHT CONTROL DATA 1978, 1987.

87/09/02. 08.59.38 T4040
(04) CY180-855 SN002 NVE 1ST CLSH.          NOS 52G2 VE 15271.
FAMILY:
```

Enter your family name (or a comma for the default family), user name, password, and the NOS/VE application VEIAF.

Example:

```
FAMILY:  family1,user123,pass456,veiaf
```

Each NAM validation defines a default application that is selected if the required information is omitted during login. Once you are validated by NAM, NOS/VE verifies validation for access.

Before executing DVS, you must obtain access to the DVS_COMMAND_LIBRARY. This library is maintained on family $SYSTEM, user $SYSTEM, and subcatalog DVS within subcatalog Hardware_Maintenance. DVS commands, classed as maintenance commands, are not provided to all users. To include DVS in your command list, use the CREATE_COMMAND_LIST_ENTRY command as follows:

```
CRECLE e =$system.hardware_maintenance.dvs.dvs_command_library
```

DVS commands are automatically made available to you when you include the previous CREATE_COMMAND_LIST_ENTRY command in your PROLOG file. A prolog is an SCL statement list, executed each time you log in to the system.

Initiating a Batch Session

NOS/VE batch jobs are submitted from other batch, interactive, or NOS dual-state jobs. DVS provides the RUN_DVS, RUN_CPU_DIAGNOSTICS, and RUN_SYSTEM_DIAGNOSTICS commands, which automatically submit a DVS test sequence as a batch job. You can create your own test sequence by submitting a batch job with the appropriate DVS commands.

Initiating a Batch Session from CYBER 170 State

A DVS batch job can also be submitted from the CYBER 170 state of a dual-state system. Create a set of virtual-state commands to execute DVS in the desired fashion. A file containing these commands is routed to the NOS/VE batch input queue.

Example:

```
LOGIN,user123,pass456,family1,dvsjob
CRECLE e =$system.hardware_maintenance.dvs.dvs_command_library
RUN_DVS all h=1 of=$system.hardware_maintenance.dvs.history
LOGOUT
```

The above commands, submitted as a batch job, run DVS with all tests for an hour and place results in the HISTORY file. A file containing these commands can be created interactively from NOS. The NOS ROUTE command that follows is then used to route the file to a queue for NOS/VE jobs. The LOGIN command identifies the user name, password, family name, and optional job name. It must be the first command in a batch job.

Example:

```
ROUTE,<filename>,dc=in,st=nve
```

The ST=NVE parameter informs NOS that the file is to be routed to the NOS/VE batch input queue.

How to Use the DVS Utility 3

How to Use the DVS Utility 3

DVS allows execution of selected diagnostics on line. This chapter explains DVS at its simplest, when running in either batch or interactive mode.

Running DVS in Batch Mode

RUN_DVS, RUN_CPU_DIAGNOSTICS, and RUN_SYSTEM_DIAGNOSTICS are vehicles for stress and system confidence testing, but they provide limited test control and display capability.

NOTE

The User's task validation limit must be set to a minimum of 11.

RUN_DVS

The RUN_DVS command runs diagnostics from a batch job. It starts DVS and begins execution of selected tests. To use the RUN_DVS command, enter:

```
RUN_DVS test_name=all
```

This command submits a batch job that runs ALL diagnostics. To run selected tests for a specified period of time, alter the above command to include either or both of the following parameters:

```
HOURS (H) = 0...24
MINUTES (M) = 0...59
```

Depending upon the number of tests selected, the RUN_DVS command submits one, two, or three batch jobs. If all tests are selected (currently up to 24 tests, depending on the machine type), up to three jobs are submitted. A selection value of less than 10 submits only one job.

To determine the status of a DVS job, you can use the DISPLAY_JOB_STATUS command. The job status display message line indicates the current DVS running status. The display message line appears as one of the following:

```
$DVS_NO_ERROR_FOUND(x,y,z)
  or
$DVS_ERROR_FOUND(x,y,z)
```

where:

x = hours argument for $AWAIT_TEST_COMPLETION function

y = minutes argument for $AWAIT_TEST_COMPLETION function

z = abort_on_error argument for $AWAIT_TEST_COMPLETION function

If the display message line indicates $DVS_ERROR_FOUND, you have the option of terminating the job and examining the status output.

Keywords for RUN_DVS TEST_NAME parameter are as follows:

RCT1 (Random Command Test 1)
RCT2 (Random Command Test 2)
FCT3 (Fixed Command Test 3)
SNGL (Single-Precision Floating-Point Test)
DOBL (Double-Precision Floating-Point Test)
BRCH (Floating-Point Branch Test)
FINT (Full-Word Integer Test)
HINT (Half-Word Integer Test)
FIMM (Full-Word Immediate Test)
HIMM (Half-Word Immediate Test)
BIMM (BDP Immediate Test)
NUMR (BDP Numeric Test)
BYTE (BDP Byte End-Case Test)
EDIT (BDP Edit End-Case Test)
VINT (Vector Integer Test)
VCMP (Vector Compare Test)
VGTH (Vector Gather/Scatter Test)
VGSI (Vector Gather/Scatter Indexed Test)
RFST (Random Fast/Slow Test)
DBUG (Debug Test)
CMEM (Central Memory Test)
DISK (Disk Test)
TAPE (Tape Test)

NOTE

Before the RUN_DVS command can execute correctly, the command list must include the DVS_Command_Library. For more information, see chapter 2, System Access.

Running DVS in batch mode with all diagnostics results in a significant system load. Use the REDUCE_TASK_PRIORITY parameter option whenever possible. To use reduced priority, enter:

```
RUN_DVS test_name=all reduce_task_priority=true
```

You can run selected diagnostics for a specified period of time with minimal effort. The RUN_DVS command also has a larger parameter set, which can be used for more specific test runs. These parameters are found in chapter 5 of this manual, Running DVS in Batch Mode. In the following examples, RUN_DVS is used in its simplest form.

Examples:

Execution of this command results in running RCT1, RCT2, FCT3, and SNGL for a total of 1-1/2 hours.

```
/run_dvs tn=(rct1,rct2,fct3,sngl) h=1 m=30
```

This command causes FCT3 to execute for a total of 10 hours.

```
/run_dvs tn=fct3 h=10
```

This command causes RCT1 and RCT2 to run for a total of 15 minutes.

```
/run_dvs tn=(rct1,rct2) m=15
```

This command results in the execution of SNGL until test completion.

```
/run_dvs tn=sngl
```

NOTE

Both the HOURS and MINUTES parameters are optional for the RUN_DVS command. If neither is specified, the test(s) run until completion.

RUN_CPU_DIAGNOSTICS (RUNCD)
RUN_SYSTEM_DIAGNOSTICS (RUNSD)

These commands are used to run CPU or system diagnostics in batch mode. They create appropriate command line images for the RUN_DVS procedure and submit them for processing. Both commands update a permanent summary file for each job submitted. You can have this file displayed by using either the DISPLAY_HISTORY or the DISPLAY_FAILURES command.

NOTE

To issue the RUN_CPU_DIAGNOSTICS or RUN_SYSTEM_DIAGNOSTICS command, DVS must be previously activated with the DVS_UTILITY command.

To use these commands, enter the following after the DVS input prompt:

```
DVS/run_cpu_diagnostics test_name=all

   or

DVS/run_system_diagnostics test_name=all
```

The above command lines submit all CPU or system diagnostics for batch processing.

NOTE

If DVS is currently running on a CYBER 990, the three vector tests (VINT, VCMP, and VGTH) are also submitted for execution.

Example:

Assume that DVS is executing on a CYBER 990.

```
DVS/run_cpu_diagnostics tn=all
RUN_DVS Input String:
RUN_DVS TN=(RCT1, RCT2, RFST, FCT3, SNGL, DOBL, BRCH,
  FINT, HINT, FIMM, HIMM, BIMM, NUMR, BYTE, EDIT) RCT1RC=10000
  RCT2RC=5000 RFSTRC=1 FCT3RC=1 MIGDSRC=1
  OF=:$SYSTEM.$SYSTEM.HARDWARE_MAINTENANCE.DVS.HISTORY
BATCH JOBS:  DVSJ_1A AND DVSJ_1B STARTED EXECUTION.
```

Running DVS in Interactive Mode

DVS can be run in interactive mode to provide greater test control and to obtain additional information from a failing test.

DVS_UTILITY

To run diagnostics in interactive mode, enter DVS by using the DVS_UTILITY command. DVS is available when you receive the DVS command menu and input prompt. At this point, tests can be run interactively. To do this, enter:

```
EXECUTE_TEST test_name = [Keyword]
```

The keyword value refers to any one of the following valid diagnostic test names:

RCT1 (Random Command Test 1)
RCT2 (Random Command Test 2)
FCT3 (Fixed Command Test 3)
SNGL (Single-Precision Floating-Point Test)
DOBL (Double-Precision Floating-Point Test)
BRCH (Floating-Point Branch Test)
FINT (Full-Word Integer Test)
HINT (Half-Word Integer Test)
FIMM (Full-Word Immediate Test)
HIMM (Half-Word Immediate Test)
BIMM (BDP Immediate Test)
NUMR (BDP Numeric Test)
BYTE (BDP Byte End-Case Test)
EDIT (BDP Edit End-Case Test)
VINT (Vector Integer Test)
VCMP (Vector Compare Test)
VGTH (Vector Gather/Scatter Test)
VGSI (Vector Gather/Scatter Indexed Test)
RFST (Random Fast/Slow Test)
DBUG (Debug Test)
CMEM (Central Memory Test)
DISK (Disk Test)
TAPE (Tape Test)
COMCT (Communications Confidence Test)

Interactively, only three commands are required to run and monitor tests. They are the following:

Command	Action
EXECUTE_TEST	Begins the test program specified by the required TEST_NAME parameter.
DROP_TEST	Terminates the test currently in control.
DISPLAY_STATUS	Displays the current running status of the controlling test.

The EXECUTE_TEST command requires that the TEST_NAME parameter be specified from the list shown on the previous page. DROP_TEST and DISPLAY_STATUS do not have required parameters.

The following examples show how to run and monitor tests using only EXECUTE_TEST, DROP_TEST, and DISPLAY_STATUS.

The three basic commands are used to execute RCT1, to display the running status of the test, and to terminate it after running 18 passes.

```
DVS/execute_test tn=rct1
RCT1_1/display_status
TEST ID           : RCT1_1 IN EXECUTION
PROCESSOR ID      : 1
TIME              : 10:31:25.177
STATUS            : NORMAL
TRAPS TAKEN       : EXPECTED=0 UNEXPECTED=0
PARAMS            : OL, LE, SE
ERROR COUNT       : 0
PASS COUNT        : 7
RCT1 RU PC=000011 PADS=00 PURGE=NONE TE=0000
PIT - VALU=ALL ONES+INTRUPT=00000000 OCCURNCS=000 LOST=00000
RCT1_1/drop_test
TERMINATING_TEST: RCT1_1, TOTAL ERRORS=0, PASSES COMPLETED=18
```

The following two commands result in the execution of 71 passes of RCT2_1 without any errors. A DISPLAY_STATUS command could have been used throughout the RCT2_1 run to display the running status at different sections.

```
DVS/exet rct2
Type MENU for RCT2_1 Commands; or HELP for DVS Commands.
RCT2_1/drot
TERMINATING_TEST: RCT2_1, TOTAL ERRORS=0, PASSES COMPLETED=71
```

NOTE

Use the QUIT command to exit from DVS. For more information, refer to chapter 8 of this manual.

Error Logging 4

Error Logging 4

DVS logs all errors in your job log. This version does not create an incident report or place information in the engineering log.

All tests execute with the LOG_ERRORS parameter set to TRUE. You can alter this bit to a FALSE setting with the SET_PARAMETER_WORD_ZERO command. The only pieces of information placed in the log are the total errors recorded and passes completed.

The following command initiates FCT3. If an error is recorded by the test, an entry is made in the job log. A representative entry is shown below the command.

```
DVS/execute_test fct3

00:50:54.188.PR.TEST ID=FCT3_1FCT3 RU PC000000 S...LP2=240
00:50:54.195 PR.TEST ID=FCT3_1EC1= EC2          TE=00001
00:50:54.197.PR.TEST ID=FCT3_1
00:50:54.199.PR.TEST ID=FCT3_1
00:50:54.201.PR.TEST ID=FCT3_1TRANB OP = EB BDP...REF=088
00:50:54.203 PR.TEST ID=FCT3_1BYTE TRANSLATE...PER(AI) D
00:50:54.205.PR.TEST ID=FCT3_1SECTION REVISED 80/05/31...0
00:50:54.211.PR.TEST ID=FCT3_1
00:50:54.214.PR.TEST ID=FCT3_1
00:50:54.216.PR.TEST ID=FCT3_1
00:50:54.217.PR.TEST ID=FCT3_1
00:50:54.219.PR.TEST ID=FCT3_1
00:50:54.257.PR.TEST ID=FCT3_1
00:50:54.263.PR.TEST ID=FCT3_1
00:50:54.265.PR.TEST ID=FCT3_1
00:50:54.267.PR.TEST ID=FCT3_1RECEIVED DEST 001328... 020...
00:50:54.269.PR.TEST ID=FCT3_1FAILING BYTE LIST = EC00 .. 00
00:50:54.270.PR.TEST ID=FCT3_1(ONE BIT PER BYTE) 00000000...
00:50:54.272.PR.TEST ID=FCT3_1DATA ERROR
00:50:54.274 PR.TEST ID=FCT3_1
```

NOTE

Page width limitations of this document prevent an exact reproduction of an entry in the job log.

When either the RUN_CPU_DIAGNOSTICS or RUN_SYSTEM_DIAGNOSTICS command is used to submit batch jobs, each output is recorded in a history file kept on the Hardware_Maintenance.DVS subcatalog.

DVS provides the DISPLAY_HISTORY command to display the history file contents and the DISPLAY_FAILURES command to display the failure information contained in the file.

NOTE

All CPU diagnostics record only 25 errors and are then terminated.

Part II: Usage

Running DVS in Batch Mode 5

Running DVS in Batch Mode 5

This chapter describes the DVS commands used to run tests in batch mode and to display corresponding batch results.

DVS Batch Mode Commands

RUN_DVS, RUN_CPU_DIAGNOSTICS, and RUN_SYSTEM_DIAGNOSTICS are vehicles for stress and system confidence testing, but they provide limited control and display capability. The interactive mode provides greater test control and should be used to obtain additional information from a failing test.

RUN_DVS

Purpose Executes selected DVS tests in batch mode.

Format

RUN_DVS
TEST_NAME=list of keywords or ALL
SELECTED_UTILITY_PROCESSOR=0...1
SELECTED_TEST_PROCESSOR=0...1
JOB_NAME=name
JOB_CLASS=name
REDUCE_TASK_PRIORITY=boolean
TEST_COPIES=1...100
JOB_COPIES=1...10
ACTIVE_TASKS=1...9
HOURS=0...24
MINUTES=0...59
ABORT_ON_ERROR=boolean
USER=boolean
LOAD_MAP=boolean
OUTPUT_FILE=file reference
MULTIPROCESSING_OPTION=boolean
DISCARD_STANDARD_OUTPUT=boolean
RCT1_REPEAT_COUNT=0...$max_integer
RCT1_PIT_VALUE=1000...7fffffff$_{16}$
RCT1_PAD_COUNT=0...10
RCT1_INTERMITTENT_COUNT=0...255
RCT1_SKIP_PASS_COUNT=0...$max_integer
RCT2_REPEAT_COUNT=0...$max_integer
RCT2_PIT_VALUE=1000...7fffffff$_{16}$
RCT2_SKIP_PASS_COUNT=0...$max_integer
RFST_REPEAT_COUNT=0...$max_integer
RFST_PIT_VALUE=1000...7fffffff$_{16}$
RFST_PAD_COUNT=0...16
RFST_SKIP_PASS_COUNT=0...$max_integer
RFST_DELETE_BDP=boolean
FCT3_REPEAT_COUNT=0...$max_integer
FCT3_DELETE_BDP=boolean
MIGDS_REPEAT_COUNT=0...$max_integer
MIGDS_PIT_VALUE=1000...7fffffff$_{16}$
MIGDS_SKIP_PASS_COUNT=0...$max_integer
MIGDS_VECTOR_QUICK_LOOK=boolean
CMEM_REPEAT_COUNT=0...$max_integer
CMEM_SEG_FILE_SIZE=100...$max_integer
CMEM_ELEMENT_NAME=name
DBUG_REPEAT_COUNT=0...$max_integer
DBUG_PIT_VALUE=1000...7fffffff$_{16}$
DBUG_SKIP_PASS_COUNT=0...$max_integer
DBUG_DELETE_BDP=boolean
DISK_REPEAT_COUNT=0...$max_integer
DISK_RECORD_COUNT=1...2147483647
DISK_RECORD_SIZE=2048...4398046511103
DISK_ELEMENT_NAME=name
TAPE_REPEAT_COUNT=0...$max_integer
TAPE_RECORD_COUNT=1...2147483647
TAPE_RECORD_SIZE=2048...4398046511103
TAPE_DENSITY=800...6250

STATUS=status variable

NOTE

The term MIGDS refers to all tests in the MIGDS category (SNGL...VGTH).

Parameters

TEST_NAME (TN)

Identifies tests to be run in batch mode by DVS. You can specify a list of test names or the keyword ALL. A maximum of 9 tests can execute concurrently. DVS runs between 1 and 9 tests in one batch job; between 10 and 18 tests are split into two jobs; and 19 or more tests are split into three jobs. By default, the job names are DVSJ_nA (9 or fewer tests); DVSJ_nA and DVSJ_nB (between 10 and 18 tests); and DVSJ_nA, DVSJ_nB, and DVSJ_nC (19 or more tests). A, B, and C identify the job partitions, and n is the current job number. This parameter is required.

RCT1

Random Command Test 1

RCT2

Random Command Test 2

FCT3

Fixed Command Test 3

SNGL

Single-Precision Floating-Point Test

DOBL

Double-Precision Floating-Point Test

BRCH

Floating-Point Branch Test

FINT

Full-Word Integer Test

HINT

Half-Word Integer Test

FIMM

Full-Word Immediate Test

HIMM

Half-Word Immediate Test

BIMM

BDP Immediate Test

NUMR

BDP Numeric Test

BYTE
BDP Byte End-Case Test

EDIT
BDP Edit End-Case Test

VINT
Vector Integer Test

VCMP
Vector Compare Test

VGTH
Vector Gather/Scatter Test

VGSI
Vector Gather/Scatter Indexed Test

RFST
Random Fast/Slow Test

DBUG
Debug Test

CMEM
Central Memory Test

DISK
Disk Test

TAPE
Tape Test

NOTE

The TAPE test requires operator intervention and is not executed when the ALL keyword is selected. The vector tests (VINT, VCMP, and VGTH) execute only on the CYBER 990 or CYBER 2000. The vector test VGSI executes only on the CYBER 2000.

SELECTED_UTILITY_PROCESSOR (SUP)

Specifies the logical identifier of the processor in which the utility is to run. It is intended for use with dual-processor systems. If SELECTED_UTILITY_PROCESSOR is omitted, the system determines the processor to be used.

SELECTED_TEST_PROCESSOR (STP)

Specifies the logical identifier of the processor in which the test(s) is to run. It is intended for use with dual-processor systems. A test can run in either processor, independent of the utility. If SELECTED_TEST_PROCESSOR is omitted, the test inherits the processor identifier of the utility.

JOB_NAME (JN)

Specifies the user-supplied name by which the submitted job is to be known.

JOB_CLASS (JBC)

Specifies the class for this job. NOS/VE uses the job class as a scheduling parameter to control the number and mix of jobs to execute concurrently. If JOB_CLASS is omitted, BATCH is used.

REDUCE_TASK_PRIORITY (RTP)

Indicates whether the CPU dispatching priority for all tests is to be reduced to the lowest priority of any task within the same job class.

TRUE

CPU dispatching priority is reduced.

FALSE

CPU dispatching priority is not reduced.

If REDUCE_TASK_PRIORITY is omitted, FALSE is used.

NOTE

Running DVS with multiple diagnostics results in a significant system load. Use the REDUCE_TASK_PRIORITY option whenever possible.

TEST_COPIES (TC)

Specifies the number of test copies executed. It allows repeated execution of tests in a round-robin fashion. If TEST_COPIES is omitted, one copy of each test is submitted.

JOB_COPIES (JC)

Specifies the number of job copies executed. If JOB_COPIES is omitted, one copy of each job is submitted.

ACTIVE_TASKS (AT)

Specifies the number of concurrent tests executed. If the number of tests selected exceeds the number of active tasks specified, DVS submits tests up to the maximum allowed. The remaining selected tests are submitted one at a time for each test that terminates. If ACTIVE_TASKS is omitted, only nine tests can execute concurrently.

HOURS (H)

Specifies the number of hours of continued execution. The time is calculated based on wall clock hours in execution. The job terminates if all tests complete before the time is exhausted. The HOURS and MINUTES parameters are added together to determine completion time. A value of 0 hours and 0 minutes runs all tests until completion. If HOURS is omitted, 0 is used.

MINUTES (M)

Specifies the number of minutes of continued execution. The time is calculated based on wall clock minutes in execution. The job terminates if all tests complete before the time is exhausted. The HOURS and MINUTES parameters are added together to determine completion time. A value of 0 hours and 0 minutes runs all tests until completion. If MINUTES is omitted, 0 is used.

ABORT_ON_ERROR (AOE)

Indicates whether the execution of all tests should be aborted when an error is detected.

TRUE

Aborts execution when an error is detected.

FALSE

Continues execution when an error is detected.

If ABORT_ON_ERROR is omitted, FALSE is used.

USER (U)

Specifies the family and user from which files are accessed.

TRUE

Files are accessed from the user's master catalog and Hardware_Maintenance.DVS subcatalog.

FALSE

Files are accessed from the $SYSTEM master catalog and Hardware_Maintenance.DVS subcatalog.

If user is omitted, FALSE is used.

LOAD_MAP (LM)

Indicates whether the load map created for each test is to be saved as a permanent file. The file name is a concatenation of the test identifier with "LOADMAP" and the map count value. Example: RCT1_LOADMAP_1

TRUE

Saves the test load map as a permanent file.

FALSE

Does not save the test load map as a permanent file.

If LOAD_MAP is omitted, FALSE is used.

OUTPUT_FILE (OF)

Specifies the disposition of the output file. Test errors, status, and running information are written to this file. If OUTPUT_FILE is omitted, $OUTPUT is used.

MULTIPROCESSING_OPTION (MO)

Enables or disables the multiprocessing capability of the NOS/VE operating system.

TRUE

Enables multiprocessing capability if applicable.

FALSE

Disables multiprocessing capability.

If MULTIPROCESSING_OPTION is omitted, FALSE is used.

DISCARD_STANDARD_OUTPUT (DSO)

Enables or disables batch listings.

TRUE

Disables batch listings.

FALSE

Enables batch listings.

If DISCARD_STANDARD_OUTPUT is omitted, TRUE is used.

RCT1_REPEAT_COUNT (RCT1RC)

Specifies the repeat count value for test RCT1. If RCT1_REPEAT_COUNT is omitted, $max_integer is used.

RCT1_PIT_VALUE (RCT1PV)

Specifies the PIT value for test RCT1. If RCT1_PIT_VALUE is omitted, the maximum positive PIT value, $7fffffff_{16}$ is used.

RCT1_PAD_COUNT (RCT1PC)

Specifies the pad count value and enables padding for test RCT1. If RCT1_PAD_COUNT is omitted, 0 is used and pads are disabled.

RCT1_INTERMITTENT_COUNT (RCT1IC)

Specifies the intermittent count value for test RCT1. If RCT1_INTERMITTENT_COUNT is omitted, 0 is used and intermittent checking is disabled.

RCT1_SKIP_PASS_COUNT (RCT1SPC)

Specifies the skip pass count value for test RCT1. If RCT1_SKIP_PASS_COUNT is omitted, 0 is used and the test starts execution at the first pass.

RCT2_REPEAT_COUNT (RCT2RC)

Specifies the repeat count value for test RCT2. If RCT2_REPEAT_COUNT is omitted, $max_integer is used.

RCT2_PIT_VALUE (RCT2PV)

Specifies the PIT value for test RCT2. If RCT2_PIT_VALUE is omitted, the maximum positive PIT value, $7fffffff_{16}$ is used.

RCT2_SKIP_PASS_COUNT (RCT2SPC)

Specifies the skip pass count value for test RCT2. If RCT2_SKIP_PASS_COUNT is omitted, 0 is used and the test starts execution at the first pass.

RFST_REPEAT_COUNT (RFSTRC)

Specifies the repeat count value for test RFST. If RFST_REPEAT_COUNT is omitted, $max_integer is used.

RFST_PIT_VALUE (RFSTPV)

Specifies the PIT value for test RFST. If RFST_PIT_VALUE is omitted, the maximum positive PIT value, $7fffffff_{16}$ is used.

RFST_PAD_COUNT (RFSTPC)

Specifies the pad count value and enables padding for test RFST. If RFST_PAD_COUNT is omitted, 0 is used. This inserts no-op instructions incrementally from 0 to 16 no-ops per instruction.

RFST_SKIP_PASS_COUNT (RFSTSPC)

Specifies the skip pass count value for test RFST. If RFST_SKIP_PASS_COUNT is omitted, 0 is used and the test starts execution at the first pass.

RFST_DELETE_BDP (RFSTDB)

Indicates whether BDP opcodes are to be deleted from the instruction list.

TRUE

BDP opcodes are deleted from the instruction list.

FALSE

BDP opcodes are included in the instruction list.

If RFST_DELETE_BDP is omitted, TRUE is used when executing on a CYBER 840, 855, or 860; otherwise FALSE is used.

FCT3_REPEAT_COUNT (FCT3RC)

Specifies the repeat count value for test FCT3. If FCT3_REPEAT_COUNT is omitted, $max_integer is used.

FCT3_DELETE_BDP (FCT3DB)

Indicates whether BDP opcodes are to be deleted from the instruction list.

TRUE

BDP opcodes are deleted from the instruction list.

FALSE

BDP opcodes are included in the instruction list.

If FCT3_DELETE_BDP is omitted, FALSE is used.

MIGDS_REPEAT_COUNT (MIGDSRC)

Specifies the repeat count value for the test category of MIGDS tests. If MIGDS_REPEAT_COUNT is omitted, $max_integer is used.

MIGDS_PIT_VALUE (MIGDSPV)

Specifies the PIT value for the test category of MIGDS tests. If MIGDS_PIT_VALUE is omitted, the maximum positive PIT value, $7fffffff_{16}$ is used.

MIGDS_SKIP_PASS_COUNT (MIGDSSPC)

Specifies the skip pass count value for the test category of MIGDS tests. If MIGDS_SKIP_PASS_COUNT is omitted, 0 is used and each of the selected MIGDS tests starts execution at the first pass.

MIGDS_VECTOR_QUICK_LOOK (MIGDSVQL)

Indicates whether the vector tests (VINT, VCMP, VGSI, and VGTH) run with fewer test cases. The fewer the test cases, the faster it executes.

TRUE

The vector tests run with fewer test cases.

FALSE

The vector tests run with all test cases.

If MIGDS_VECTOR_QUICK_LOOK is omitted, FALSE is used.

CMEM_REPEAT_COUNT (CMEMRC)

Specifies the repeat count value for test CMEM. If CMEM_REPEAT_COUNT is omitted, $max_integer is used.

CMEM_SEG_FILE_SIZE (CMEMSFS)

Specifies the segment file size of the local segment access file used by CMEM for disk testing. If CMEM_SEG_FILE_SIZE is omitted, 1 048 576 bytes is used.

CMEM_ELEMENT_NAME (CMEMEN)

Specifies the name of the element on which the segment access file is to be written. If CMEM_ELEMENT_NAME is omitted, the system selects the element.

NOTE

Element selection is only permitted for SYSTEM or MAINTENANCE jobs.

DBUG_REPEAT_COUNT (DBUGRC)

Specifies the repeat count value for test DBUG. If DBUG_REPEAT_COUNT is omitted, $max_integer is used.

DBUG_PIT_VALUE (DBUGPV)

Specifies the PIT value for the test DBUG. If DBUG_PIT_VALUE is omitted, the maximum positive PIT value, $7fffffff_{16}$ is used.

DBUG_SKIP_PASS_COUNT (DBUGSPC)

Specifies the skip pass count value for test DBUG. If DBUG_SKIP_PASS_COUNT is omitted, 0 is used and the test starts execution at the first pass.

DBUG_DELETE_BDP (DBUGDB)

Indicates whether BDP opcodes are to be deleted from the instruction list.

TRUE

BDP opcodes are deleted from the instruction list.

FALSE

BDP opcodes are included in the instruction list.

If DBUG_DELETE_BDP is omitted, FALSE is used.

DISK_REPEAT_COUNT (DISKRC)

Specifies the repeat count value for test DISK. If DISK_REPEAT_COUNT is omitted, $max_integer is used.

DISK_RECORD_COUNT (DISKREC)

Specifies the record count value for test DISK. If DISK_RECORD_COUNT is omitted, 512 is used.

DISK_RECORD_SIZE (DISKRS)

Specifies the record size value for test DISK. If DISK_RECORD_SIZE is omitted, 2048 is used.

NOTE

The record size must be a multiple of 2048 bytes.

DISK_ELEMENT_NAME (DISKEN)

Specifies the name of the element on which the disk file is to be written. If DISK_ELEMENT_NAME is omitted, the system selects the element.

NOTE

Element selection is only permitted for SYSTEM or MAINTENANCE jobs.

TAPE_REPEAT_COUNT (TAPERC)

Specifies the repeat count value for test TAPE. If TAPE_REPEAT_COUNT is omitted, 1 is used.

TAPE_RECORD_COUNT (TAPEREC)

Specifies the record count value for test TAPE. If TAPE_RECORD_COUNT is omitted, 512 is used.

TAPE_RECORD_SIZE (TAPERS)

Specifies the record size value for test TAPE. If TAPE_RECORD_SIZE is omitted, 2048 is used.

NOTE

The record size must be a multiple of 2048 bytes.

TAPE_DENSITY (TAPED)

Specifies the density value for test TAPE. If TAPE_DENSITY is omitted, 1600 is used.

NOTE

The only valid densities are 800, 1600, and 6250 cpi.

STATUS (S)

Specifies the termination condition of the command.

Remarks

To determine the status of a DVS job, use the DISPLAY_JOB_STATUS command. The job's status display line indicates the current DVS running status and appears as one of the following:

```
$DVS_NO_ERROR_FOUND(x,y,z)

$DVS_ERROR_FOUND(x,y,z)
```

NOTE

x=hours argument for $AWAIT_TEST_COMPLETION function.

y=minutes argument for $AWAIT_TEST_COMPLETION function.

z=abort_on_error argument for $AWAIT_TEST_COMPLETION function.

Examples The following RUN_DVS command initiates the utility and executes the selected tests for one hour.

```
/run_dvs (rct1,rct2,sngl) stp=0 h=1
```

In the prceeding example, only three tests are selected. Of the three, one test fails with an unexpected trap. The RUN_DVS procedure detects an error and performs a DISPLAY_STATUS command. The following is a representative listing.

```
TEST ID              : SNGL_1 EXECUTION COMPLETED
PROCESSOR ID         : 0
TIME                 : 15:19:06.021
STATUS               : --FATAL-- exponent overflow at P=0B 2E 27A2
TRAPS TAKEN          : EXPECTED=31682 UNEXPECTED=1
UNEXPECTED TRAP, DETECTED BY TEST TRAP HANDLER
MONITOR CONDITIONS:      USER CONDITIONS:
  EXCHANGE REQUEST         EXPONENT OVERFLOW
P=0000B02F000027A2
INSTRUCTION AT P: 309B
A0=0000B02F00000760    A1=00FFB02F00000760
A2=FFFFB02F00000700    A3=FFFCB03000000000
A4=0020B02F00000768    A5=0400B032000019A8
A6=0000B03200000080    A7=0000B03200002DD0
A8=0000B03200012810    A9=0000B03200000000
AA=77A4B032000019A8    AB=7517B032000019A8
AC=0000B02E000027A0    AD=A2C0B032000019B0
AE=0000B02F00000778    AF=0064B00F000003F4
X0=FFFFFFFFFFFFFCCC    X1=FFFFFFFFFFFFFFFF
X2=4FFF800000000000    X3=0000000000000000
X4=D000800000000000    X5=FFFFFFFFFFFFFFFF
X6=0000000000004FFF    X7=0000000000004FFF
X8=0000800000000000    X9=0000000000000001
XA=0000000000000002    XB=0000000000000001
XC=0000000000000000    XD=0000000000000020
XE=0000000000000020    XF=D000800000000000
PARAMS               : BM, OL, LE
ERROR COUNT          : 0
PASS COUNT           : 1
SINGLE PRECISION FLOATING-POINT ALL PAGES SNGL100_DVS 10/25/82
RUNNING ALL OPERATIONS
PC00001 TE=1 UM=FE00 ADDF 001,001 PV=47483667
NUMBER OF PASSES SELECTED TO RUN       9999
TERMINATING_TEST: SNGL_1, TOTAL ERRORS=1, PASSES COMPLETED=1
TERMINATING_TEST: RCT1_1, TOTAL ERRORS=0, PASSES COMPLETED=17705
TERMINATING_TEST: RCT2_1, TOTAL ERRORS=0, PASSES COMPLETED=1354
```

RUN_CPU_DIAGNOSTICS (RUNCD)

Purpose Submits CPU diagnostics for batch processing.

NOTE

To issue the RUN_CPU_DIAGNOSTICS command, you must activate the utility with the DVS_UTILITY command.

A permanent summary file is updated for each job. You can display this file by using the DISPLAY_HISTORY or DISPLAY_FAILURES command.

Format

RUN_CPU_DIAGNOSTICS
TEST_NAME=list of keywords or ALL
RUN_OPTIONS=keyword
SELECTED_UTILITY_PROCESSOR=0...1
SELECTED_TEST_PROCESSOR=0...1
JOB_NAME=name
JOB_CLASS=name
REDUCE_TASK_PRIORITY=boolean
TEST_COPIES=1...100
JOB_COPIES=1...10
ACTIVE_TASKS=1...9
HOURS=0...24
MINUTES=0...59
ABORT_ON_ERROR=boolean
USER=boolean
LOAD_MAP=boolean
MULTIPROCESSING_OPTION=boolean
DISCARD_STANDARD_OUTPUT=boolean
RCT1_REPEAT_COUNT=0...$max_integer
RCT1_PIT_VALUE=1000...7fffffff$_{16}$
RCT1_PAD_COUNT=0...10
RCT1_INTERMITTENT_COUNT=0...255
RCT1_SKIP_PASS_COUNT=0...$max_integer
RCT2_REPEAT_COUNT=0...$max_integer
RCT2_PIT_VALUE=1000...7fffffff$_{16}$
RCT2_SKIP_PASS_COUNT=0...$max_integer
RFST_REPEAT_COUNT=0...$max_integer
RFST_PIT_VALUE=100...7fffffff$_{16}$
RFST_PAD_COUNT=0...16
RFST_SKIP_PASS_COUNT=0...$max_integer
FCT3_REPEAT_COUNT=0...$max_integer
MIGDS_REPEAT_COUNT=0...$max_integer
MIGDS_PIT_VALUE=1000...7fffffff$_{16}$
MIGDS_SKIP_PASS_COUNT=0...$max_integer
DBUG_REPEAT_COUNT=0...$max_integer
DBUG_PIT_VALUE=1000...7fffffff$_{16}$
DBUG_SKIP_PASS_COUNT=0...$max_integer
STATUS=status variable

NOTE

The term MIGDS refers to all tests in the MIGDS category (SNGL...VGSI).

Parameters

TEST_NAME (TN)

Identifies CPU tests to be run in batch mode by DVS. You can specify a list of test names or the keyword ALL. A maximum of 9 tests can execute concurrently. DVS runs between 1 and 9 tests in one batch job; between 10 and 18 tests are split into two jobs; and 19 or more tests are split into three jobs. By default, the job names are DVSJ_nA (9 or fewer tests); DVSJ_nA and DVSJ_nB (between 10 and 18 tests); and DVSJ_nA, DVSJ_nB, and DVSJ_nC (19 or more tests). A, B, and C identify the job partitions, and n is the current job number. This parameter is required.

RCT1
Random Command Test 1

RCT2
Random Command Test 2

FCT3
Fixed Command Test 3

SNGL
Single-Precision Floating-Point Test

DOBL
Double-Precision Floating-Point Test

BRCH
Floating-Point Branch Test

FINT
Full-Word Integer Test

HINT
Half-Word Integer Test

FIMM
Full-Word Immediate Test

HIMM
Half-Word Immediate Test

BIMM
BDP Immediate Test

NUMR
BDP Numeric Test

BYTE
BDP Byte End-Case Test

EDIT
BDP Edit End-Case Test

VINT

Vector Integer Test

VCMP

Vector Compare Test

VGTH

Vector Gather/Scatter Test

VGSI

Vector Gather/Scatter Indexed Test

RFST

Random Fast/Slow Test

DBUG

Debug Test

NOTE

The vector tests (VINT, VCMP, and VGTH) execute only on a CYBER 990 or a CYBER 2000. The vector test VGSI executes only on a CYBER 2000.

RUN_OPTIONS (RO)

Specifies the running mode of the utility.

ONE_PASS (OP)

Runs all selected tests for one pass.

ONE_HOUR (OH)

Runs all selected tests for one hour.

CONTINUOUSLY (C)

Runs all selected tests continuously.

TIMED (T)

Runs all selected tests for a timed period. If selected, the HOURS and/or MINUTES parameters must be provided.

If RUN_OPTIONS is omitted, ONE_PASS is used.

NOTE

The RUN_CPU_DIAGNOSTICS command adjusts the RCT1 and RCT2 repeat count value to 10 000 and 5 000 passes respectively if ONE_PASS is selected for the RUN_OPTIONS parameter.

SELECTED_UTILITY_PROCESSOR (SUP)

Specifies the logical identifier of the processor in which the utility is to run. It is intended for use with dual-processor systems. If SELECTED_UTILITY_PROCESSOR is omitted, the system determines the processor to be used.

SELECTED_TEST_PROCESSOR (STP)

Specifies the logical identifier of the processor in which the tests are to run. It is intended for use with dual-processor systems. A test can run in either processor, independent of the utility. If SELECTED_TEST_PROCESSOR is omitted, the test inherits the processor identifier of the utility.

JOB_NAME (JN)

Specifies the user-supplied name by which the submitted job is to be known.

JOB_CLASS (JBC)

Specifies the class for this job. NOS/VE uses the job class as a scheduling parameter to control the number and mix of jobs to execute concurrently. If JOB_CLASS is omitted, MAINTENANCE is used.

REDUCE_TASK_PRIORITY (RTP)

Indicates whether the CPU dispatching priority for all tests is to be reduced to the lowest priority of any task within the same job class.

TRUE

CPU dispatching priority is reduced.

FALSE

CPU dispatching priority is not reduced.

If REDUCE_TASK_PRIORITY is omitted, FALSE is used.

NOTE

Running DVS with multiple diagnostics results in a signficant system load. Use the REDUCE_TASK_PRIORITY option whenever possible.

TEST_COPIES (TC)

Specifies the number of test copies executed. It allows repeated execution of tests in a round-robin fashion. If TEST_COPIES is omitted, one copy of each test is submitted.

JOB_COPIES (JC)

Specifies the number of job copies executed. If JOB_COPIES is omitted, one copy of each job is submitted.

ACTIVE_TASKS (AT)

Specifies the number of concurrent tests executed. If the number of tests selected exceeds the number of active tasks specified, DVS submits tests up to the maximum allowed. The remaining selected tests are submitted one at a time for each test that terminates. If ACTIVE_TASKS is omitted, only nine tests can execute concurrently.

HOURS (H)

Specifies the number of hours of continued execution. The time is calculated based on wall clock hours in execution. The job terminates if all tests complete before the time is exhausted. The HOURS and MINUTES parameters are added together to determine completion time. A value of 0 hours and 0 minutes runs all tests until completion. If HOURS is omitted, 0 is used.

MINUTES (M)

Specifies the number of minutes of continued execution. The time is calculated based on wall clock minutes in execution. The job terminates if all tests complete before the time is exhausted. The HOURS and MINUTES parameters are added together to determine completion time. A value of 0 hours and 0 minutes runs all tests until completion. If MINUTES is omitted, 0 is used.

ABORT_ON_ERROR (AOE)

Indicates whether the execution of all tests should be aborted when an error is detected.

TRUE

Aborts execution when an error is detected.

FALSE

Continues execution when an error is detected.

If ABORT_ON_ERROR is omitted, FALSE is used.

USER (U)

Specifies the family and user from which files are accessed.

TRUE

Files are accessed from the user's master catalog and Hardware_Maintenance.DVS subcatalog.

FALSE

Files are accessed from the $SYSTEM master catalog and Hardware_Maintenance.DVS subcatalog.

If USER is omitted, FALSE is used.

LOAD_MAP (LM)

Indicates whether the load map created for each test is to be saved as a permanent file. The file name is a concatenation of the test identifier with "LOADMAP" and the map count value. Example: FCT3_LOADMAP_1

TRUE

Saves the test load map as a permanent file.

FALSE

Does not save the test load map as a permanent file.

If LOAD_MAP is omitted, FALSE is used.

MULTIPROCESSING_OPTION (MO)

Enables or disables the multiprocessing capability of the NOS/VE operating system.

TRUE

Enables multiprocessing capability if applicable.

FALSE

Disables multiprocessing capability.

If MULTIPROCESSING_OPTION is omitted, FALSE is used.

DISCARD_STANDARD_OUTPUT (DSO)

Enables or disables batch listings.

TRUE

Disables batch listings.

FALSE

Enables batch listings.

If DISCARD_STANDARD_OUTPUT is omitted, TRUE is used.

RCT1_REPEAT_COUNT (RCT1RC)

Specifies the repeat count value for test RCT1. If RCT1_REPEAT_COUNT is omitted, $max_integer is used.

RCT1_PIT_VALUE (RCT1PV)

Specifies the PIT value for test RCT1. If RCT1_PIT_VALUE is omitted, the maximum positive PIT value, $7fffffff_{16}$ is used.

RCT1_PAD_COUNT (RCT1PC)

Specifies the pad count value and enables padding for test RCT1. If RCT1_PAD_COUNT is omitted, 0 is used and pads are disabled.

RCT1_INTERMITTENT_COUNT (RCT1IC)

Specifies the intermittent count value for test RCT1. If RCT1_INTERMITTENT_COUNT is omitted, 0 is used and intermittent checking is disabled.

RCT1_SKIP_PASS_COUNT (RCT1SPC)

Specifies the skip pass count value for test RCT1. If RCT1_SKIP_PASS_COUNT is omitted, 0 is used and the test starts execution at the first pass.

RCT2_REPEAT_COUNT (RCT2RC)

Specifies the repeat count value for test RCT2. If RCT2_REPEAT_COUNT is omitted, $max_integer is used.

RCT2_PIT_VALUE (RCT2PV)

Specifies the PIT value for test RCT2. If RCT2_PIT_VALUE is omitted, the maximum positive PIT value, $7fffffff_{16}$ is used.

RCT2_SKIP_PASS_COUNT (RCT2SPC)

Specifies the skip pass count value for test RCT2. If RCT2_SKIP_PASS_COUNT is omitted, 0 is used and the test starts execution at the first pass.

RFST_REPEAT_COUNT (RFSTRC)

Specifies the repeat count value for test RFST. If RFST_REPEAT_COUNT is omitted, $max_integer is used.

RFST_PIT_VALUE (RFSTPV)

Specifies the PIT value for test RFST. If RFST_PIT_VALUE is omitted, the maximum positive PIT value, $7fffffff_{16}$ is used.

RFST_PAD_COUNT (RFSTPC)

Specifies the pad count value and enables padding for test RFST. If RFST_PAD_COUNT is omitted, 0 is used. This inserts no-op instructions incrementally from 0 to 16 no-ops per instruction.

RFST_SKIP_PASS_COUNT (RFSTSPC)

Specifies the skip pass count value for test RFST. If RFST_SKIP_PASS_COUNT is omitted, 0 is used and the test starts execution at the first pass.

FCT3_REPEAT_COUNT (FCT3RC)

Specifies the repeat count value for test FCT3. If FCT3_REPEAT_COUNT is omitted, $max_integer is used.

MIGDS_REPEAT_COUNT (MIGDSRC)

Specifies the repeat count value for the test category of MIGDS tests. If MIGDS_REPEAT_COUNT is omitted, $max_integer is used.

MIGDS_PIT_VALUE (MIGDSPV)

Specifies the PIT value for the test category of MIGDS tests. If MIGDS_PIT_VALUE is omitted, the maximum positive PIT value, $7fffffff_{16}$ is used.

MIGDS_SKIP_PASS_COUNT (MIGDSSPC)

Specifies the skip pass count value for the test category of MIGDS tests. If MIGDS_SKIP_PASS_COUNT is omitted, 0 is used and each of the selected MIGDS tests starts execution at the first pass.

DBUG_REPEAT_COUNT (DBUGRC)

Specifies the repeat count value for test DBUG. If DBUG_REPEAT_COUNT is omitted, $max_integer is used.

DBUG_PIT_VALUE (DBUGPV)

Specifies the PIT value for test DBUG. If DBUG_PIT_VALUE is omitted, the maximum positive PIT value, $7fffffff_{16}$ is used.

DBUG_SKIP_PASS_COUNT (DBUGSPC)

Specifies the skip pass count value for test DBUG. If DBUG_SKIP_PASS_COUNT is omitted, 0 is used and the test starts execution at the first pass.

STATUS (S)

Specifies the termination condition of the command.

Remarks To determine the status of a DVS job, use the DISPLAY_JOB_STATUS command. The job's status display line indicates the current DVS running status and appears as one of the following:

```
$DVS_NO_ERROR_FOUND(x,y,z)

   or

$DVS_ERROR_FOUND(x,y,z)
```

NOTE

x=hours argument for $AWAIT_TEST_COMPLETION function.

y=minutes argument for $AWAIT_TEST_COMPLETION function.

z=abort_on_error argument for $AWAIT_TEST_COMPLETION function.

Examples The following RUN_CPU_DIAGNOSTICS command submits a batch job to run all CPU diagnostics for a timed event of 5 minutes. Assume that DVS is currently executing on a CYBER 990.

```
DVS/runcd tn=all ro=t m=5
RUN_DVS Input String:
RUN_DVS TN=(RCT1,RCT2,RFST,FCT3,SNGL,DOBL,
  BRCH,FINT,HINT,FIMM,HIMM,BIMM,NUMR,BYTE,EDIT,
  VINT,VCMP,VGTH) M=5 OF=:$SYSTEM.$SYSTEM.
  HARDWARE_MAINTENANCE.DVS.HISTORY
BATCH JOBS: DVSJ_1A AND DVSJ_1B STARTED
  EXECUTION.
```

RUN_SYSTEM_DIAGNOSTICS (RUNSD)

Purpose Submits system diagnostics for batch processing.

NOTE

To issue the RUN_SYSTEM_DIAGNOSTICS command, you must activate the utility with the DVS_UTILITY command.

A permanent summary file is updated for each job. You can display this file by using the DISPLAY_HISTORY or DISPLAY_FAILURES command.

Format

RUN_SYSTEM_DIAGNOSTICS
TEST_NAME=list of keywords or ALL
RUN_OPTIONS=keyword
SELECTED_UTILITY_PROCESSOR=0...1
SELECTED_TEST_PROCESSOR=0...1
JOB_NAME=name
JOB_CLASS=name
REDUCE_TASK_PRIORITY=boolean
TEST_COPIES=1...100
JOB_COPIES=1...10
ACTIVE_TASKS=1...9
HOURS=0...24
MINUTES=0...59
ABORT_ON_ERROR=boolean
USER=boolean
LOAD_MAP=boolean
MULTIPROCESSING_OPTION=boolean
DISCARD_STANDARD_OUTPUT= boolean
CMEM_REPEAT_COUNT=0...$max_integer
CMEM_SEG_FILE_SIZE=100...$max_integer
CMEM_ELEMENT_NAME=name
DISK_REPEAT_COUNT=0...$max_integer
DISK_RECORD_COUNT=1...2147483647
DISK_RECORD_SIZE=2048...4398046511103
DISK_ELEMENT_NAME=name
TAPE_REPEAT_COUNT=0...$max_integer
TAPE_RECORD_COUNT=1...2147483647
TAPE_RECORD_SIZE=2048...4398046511103
TAPE_DENSITY=800...6250
STATUS=status variable

Parameters

TEST_NAME (TN)

Identifies system tests to be run in batch mode by DVS. You can specify a list of test names or the keyword ALL. A maximum of nine tests can execute concurrently. DVS runs between one and nine tests in one batch job. By default, the job name is DVSJ_nA (nine or fewer tests). A identifies the job partition, and n is the current job number. This parameter is required.

CMEM

Central Memory Test

DISK

Disk Test

TAPE

Tape Test

RUN_OPTIONS (RO)

Specifies the running mode of the utility.

ONE_PASS (OP)

Runs all selected tests for one pass.

ONE_HOUR (OH)

Runs all selected tests for 1 hour.

CONTINUOUSLY (C)

Runs all selected tests continuously.

TIMED (T)

Runs all selected tests for a timed period. If selected, the HOURS and/or MINUTES parameters must be provided.

If RUN_OPTIONS is omitted, ONE_PASS is used.

SELECTED_UTILITY_PROCESSOR (SUP)

Specifies the logical identifier of the processor in which the utility is to run. It is intended for use with dual-processor systems. If SELECTED_UTILITY_PROCESSOR is omitted, the system determines the processor to be used.

SELECTED_TEST_PROCESSOR (STP)

Specifies the logical identifier of the processor in which the tests are to run. It is intended for use with dual-processor systems. A test can run in either processor, independent of the utility. If SELECTED_TEST_PROCESSOR is omitted, the test inherits the processor of the utility.

JOB_NAME (JN)

Specifies the user-supplied name by which the submitted job is to be known.

JOB_CLASS (JBC)

Specifies the class for this job. NOS/VE uses the job class as a scheduling parameter to control the number and mix of jobs to execute concurrently. If JOB_CLASS is omitted, MAINTENANCE is used.

REDUCE_TASK_PRIORITY (RTP)

Indicates whether the CPU dispatching priority is to be reduced to the lowest priority of any task within the same job class.

TRUE

CPU dispatching priority is reduced.

FALSE

CPU dispatching priority is not reduced.

If REDUCE_TASK_PRIORITY is omitted, FALSE is used.

NOTE

Running DVS with multiple diagnostics results in a significant system load. Use the REDUCE_TASK_PRIORITY option whenever possible.

TEST_COPIES (TC)

Specifies the number of test copies executed. It allows repeated execution of tests in a round-robin fashion. If TEST_COPIES is omitted, one copy of each test is submitted.

JOB_COPIES (JC)

Specifies the number of job copies executed. If JOB_COPIES is omitted, one copy of each job is submitted.

ACTIVE_TASKS (AT)

Specifies the number of concurrent tests executed. If the number of tests selected exceeds the number of active tasks specified, DVS submits tests up to the maximum allowed. The remaining selected tests are submitted one at a time for each test that terminates. If ACTIVE_TASKS is omitted, only nine tests can execute concurrently.

HOURS (H)

Specifies the number of hours of continued execution. The time is calculated based on wall clock hours in execution. The job terminates if all tests complete before the time is exhausted. The HOURS and MINUTES parameters are added together to determine completion time. A value of 0 hours and 0 minutes runs all tests until completion. If HOURS is omitted, 0 is used.

MINUTES (M)

Specifies the number of minutes of continued execution. The time is calculated based on wall clock minutes in execution. The job terminates if all tests complete before the time is exhausted. The HOURS and MINUTES parameters are added together to determine completion time. A value of 0 hours and 0 minutes runs all tests until completion. If MINUTES is omitted, 0 is used.

ABORT_ON_ERROR (AOE)

Indicates whether the execution of all tests should be aborted when an error is detected.

TRUE

Aborts execution when an error is detected.

FALSE

Continues execution when an error is detected.

If ABORT_ON_ERROR is omitted, FALSE is used.

USER (U)

Specifies the family and user from which files are accessed.

TRUE

Files are accessed from the user's master catalog and Hardware_Maintenance.DVS subcatalog.

FALSE

Files are accessed from the $SYSTEM master catalog and Hardware_Maintenance.DVS subcatalog.

If USER is omitted, FALSE is used.

LOAD_MAP (LM)

Indicates whether the load map created for each test is to be saved as a permanent file. The file name is a concatenation of the test identifier with "LOADMAP" and the map count value. Example: CMEM_LOADMAP_2.

TRUE

Saves the test load map as a permanent file.

FALSE

Does not save the test load map as a permanent file.

If LOAD_MAP is omitted, FALSE is used.

MULTIPROCESSING_OPTION (MO)

Enables or disables the multiprocessing capability of the NOS/VE operating system.

TRUE

Enables multiprocessing capability if applicable.

FALSE

Disables multiprocessing capability.

If MULTIPROCESSING_OPTION is omitted, FALSE is used.

DISCARD_STANDARD_OUTPUT (DSO)

Enables or disables batch listings.

TRUE

Disables batch listings.

FALSE

Enables batch listings.

If DISCARD_STANDARD_OUTPUT is omitted, TRUE is used.

CMEM_REPEAT_COUNT (CMEMRC)

Specifies the repeat count value for test CMEM. If CMEM_REPEAT_COUNT is omitted, $max_integer is used.

CMEM_SEG_FILE_SIZE (CMEMSFS)

Specifies the segment file size of the local segment access file used by CMEM for disk testing. If CMEM_SEG_FILE_SIZE is omitted, 1 048 576 bytes is used.

CMEM_ELEMENT_NAME (CMEMEN)

Specifies the name of the element on which the segment access file is to be written. If CMEM_ELEMENT_NAME is omitted, the system selects the element.

NOTE

ENGINEERING OPERATIONS capability is required for element selection..

DISK_REPEAT_COUNT (DISKRC)

Specifies the repeat count value for test DISK. If DISK_REPEAT_COUNT is omitted, $max_integer is used.

DISK_RECORD_COUNT (DISKREC)

Specifies the record count value for test DISK. If DISK_RECORD_COUNT is omitted, 512 is used.

DISK_RECORD_SIZE (DISKRS)

Specifies the record size value for test DISK. If DISK_RECORD_SIZE is omitted, 2048 is used.

NOTE

The record size must be a multiple of 2048 bytes.

DISK_ELEMENT_NAME (DISKEN)

Specifies the name of the element on which the disk file is to be written. If DISK_ELEMENT_NAME is omitted, the system selects the element.

NOTE

ENGINEERING OPERATIONS capability is required for element selection.

TAPE_REPEAT_COUNT (TAPERC)

Specifies the repeat count value for test TAPE. If TAPE_REPEAT_COUNT is omitted, 1 is used.

TAPE_RECORD_COUNT (TAPEREC)

Specifies the record count value for test TAPE. If TAPE_RECORD_COUNT is omitted, 512 is used.

TAPE_RECORD_SIZE (TAPERS)

Specifies the record size value for test TAPE. If TAPE_RECORD_SIZE is omitted, 2048 is used.

NOTE

The record size must be a multiple of 2048 bytes.

TAPE_DENSITY (TAPED)

Specifies the density value for test TAPE. If TAPE_DENSITY is omitted, 1600 is used.

NOTE

The only valid densities are 800, 1600, and 6250 cpi.

STATUS (S)

Specifies the termination condition of the command.

Remarks

To determine the status of a DVS job, use the DISPLAY_JOB_STATUS command. The job's status display line indicates the current DVS running status and appears as one of the following:

```
$DVS_NO_ERROR_FOUND(x,y,z)

   or

$DVS_ERROR_FOUND(x,y,z)
```

NOTE

x=hours argument for $AWAIT_TEST_COMPLETION function.

y=minutes argument for $AWAIT_TEST_COMPLETION function.

z=abort_on_error argument for $AWAIT_TEST_COMPLETION function.

Examples

The following RUN_SYSTEM_DIAGNOSTICS command submits five batch jobs to run all system diagnostics for a timed event of 3 minutes.

```
DVS/runsd tn=all jc=5 m=3 ro=t
RUN_DVS Input String:
RUN_DVS TN=(CMEM,DISK,TAPE,COMCT) JC=5 M=3
  OF=:$SYSTEM.$SYSTEM.HARDWARE_MAINTENANCE.DVS.HISTORY
BATCH JOBS: DVSJ_1A, DVSJ_2A, DVSJ_3A, DVSJ_4A,
  AND DVSJ_5A STARTED EXECUTION.
```

DVS History File Commands

DISPLAY_HISTORY and DISPLAY_FAILURES are DVS commands used to display DVS batch job results. You can clear the history file at any time by using the RESET_HISTORY_FILE command.

DISPLAY_HISTORY (DISH)

Purpose Displays the history file produced when you submit batch jobs with the RUN_CPU_DIAGNOSTICS or RUN_SYSTEM_DIAGNOSTICS commands.

NOTE

To issue the DISPLAY_HISTORY command, you must activate the utility with the DVS_UTILITY command.

Format

DISPLAY_HISTORY
OUTPUT=file reference
STATUS=status variable

Parameters

OUTPUT (O)

Specifies the file to which information is displayed. If OUTPUT is omitted, $OUTPUT is used.

STATUS (S)

Specifies the termination condition of the command.

Examples The following DISPLAY_HISTORY command displays the history file to $OUTPUT.

```
DVS/display_history
DVS HISTORY FILE:
DIAGNOSTIC VIRTUAL SYSTEM UTILITY (DVS) - VERSION 3.1 L700 APRIL 18, -
1988
SINGLE PRECISION FLOATING-POINT ALL PAGES SNGL100_DVS 10/25/82
RUNNING ALL OPERATIONS
PC00001 TE=1 UM=FE00 ADDF 001, 001 PV=47483667
NUMBER OF PASSES SELECTED TO RUN      9999
TERMINATING_TEST: SNGL_1, TOTAL ERRORS=0, PASSES COMPLETED=1
TERMINATING_TEST: RCT1_1, TOTAL ERRORS=0, PASSES COMPLETED=17705
TERMINATING_TEST: RCT2_1, TOTAL ERRORS=0, PASSES COMPLETED=1354
```

DISPLAY_FAILURES (DISF)

Purpose Displays the failure entries in the history file produced when you submit batch jobs with the RUN_CPU_DIAGNOSTICS or RUN_SYSTEM_DIAGNOSTICS command.

NOTE

To issue the DISPLAY_FAILURES command, you must activate the utility with the DVS_UTILITY command.

Format

DISPLAY_FAILURES
OUTPUT=file reference
STATUS=status variable

Parameters

OUTPUT (O)

Specifies the file to which information is displayed. If OUTPUT is omitted, $OUTPUT is used.

STATUS (S)

Specifies the termination condition of the command.

Examples The following DISPLAY_FAILURES command displays the failure entries in the history file to $OUTPUT.

```
DVS/display_failures
DVS FAILURES:
RCT1_1 TOTAL ERRORS=1 PASSES COMPLETED=17705
```

RESET_HISTORY_FILE (RESHF)

Purpose Resets the DVS history file to beginning-of-information. All history information in the file is discarded.

NOTE

To issue the RESET_HISTORY_FILE command, you must activate the utility with the DVS_UTILITY command.

Format

RESET_HISTORY_FILE
STATUS=status variable

Parameters

STATUS (S)

Specifies the termination condition of the command.

Examples In the following example, the DVS history file is reset to beginning-of-information, discarding all history information.

```
DVS/reset_history_file
```

DVS Menu Facility 6

DVS Menu Facility 6

This chapter describes the menu facility and related commands available for use with DVS.

DVS Menu Structure

There are three levels in the DVS menu structure.

- Command menu
- Parameter menu
- Descriptive text

Command Menu

The first menu level contains test commands. The DVS command menu is automatically displayed when the DVS_UTILITY command is executed and the MENU_MODE parameter is set to TRUE. Test command menus are invoked after the specific test is executed and the MENU command is initiated.

Examples:

```
DVS_UTILITY

DIAGNOSTIC VIRTUAL SYSTEM UTILITY (DVS) - VERSION X.X.X LXXX September 7, 1989
DVS COMMAND SELECTIONS                                    ( PAGE 1 OF 1 )

                                BATCH COMMANDS

 1.RUN_CPU_DIAGNOSTICS        RUNCD     2.RUN_SYSTEM_DIAGNOSTICS      RUNSD
 3.DISPLAY_HISTORY            DISH      4.DISPLAY_FAILURES            DISF
 5.RESET_HISTORY_FILE         RESHF

                             INTERACTIVE COMMANDS

 6.EXECUTE_TEST               EXET      7.SWITCH_TEST                 SWIT
 8.DISPLAY_STATUS             DISS      9.DROP_TEST                   DROT
10.SET_PARAMETER_WORD_ZERO    SETPWZ   11.SET_STATUS_DISPLAY          SETSD
12.RESTART_TEST               REST     13.CONTINUE_TEST               CONT
14.STOP_TEST                  STOT     15.DISPLAY_EXECUTING_TESTS     DISET

TYPE PARAMETERS 1...15 FOR COMMAND PARAMETERS; QUIT TO TERMINATE DVS
     EXPLAIN 1...15 FOR COMMAND DESCRIPTIVE TEXT.
```

Parameter Menu

The second level of the three-tiered hierarchical menu structure contains the command parameters. The following example produces the parameter selection list for the RUN_CPU_DIAGNOSTICS command.

Examples:

```
DVS/ MENU 1

DVS RUN_CPU_DIAGNOSTICS COMMAND, PARAMETER SELECTIONS     ( PAGE 1 OF 4 )

 1.TEST_NAME                    TN       REQUIRED          KEYWORD VALUE
 2.RUN_OPTIONS                  RO       OPTIONAL/DEFAULT  KEYWORD VALUE
 3.SELECTED_UTILITY_PROCESSOR   SUP      OPTIONAL          INTEGER VALUE
 4.SELECTED_TEST_PROCESSOR      STP      OPTIONAL          INTEGER VALUE
 5.JOB_NAME                     JN       OPTIONAL          NAME VALUE
 6.JOB_CLASS                    JBC      OPTIONAL          NAME VALUE
 7.REDUCE_TASK_PRIORITY         RTP      OPTIONAL          BOOLEAN VALUE
 8.TEST_COPIES                  TC       OPTIONAL          INTEGER VALUE
 9.JOB_COPIES                   JC       OPTIONAL          INTEGER VALUE
10.ACTIVE_TASKS                 AT       OPTIONAL          INTEGER VALUE

TYPE PARAMETERS 1...15 FOR COMMAND PARAMETERS; QUIT TO TERMINATE DVS
     EXPLAIN 1...10 FOR RUN_CPU_DIAGNOSTICS COMMAND, PARAMETER DESCRIPTIVE TEXT.
     NEXP FOR NEXT PAGE OF TEXT; PREP FOR PREVIOUS PAGE OF TEXT
```

NOTE

Some of the displays are too lengthy to be contained on one page. In these cases use the commands NEXT_PAGE (NEXP) and PREVIOUS_PAGE (PREP) to traverse the pages of text.

```
DVS/ NEXP

DVS RUN_CPU_DIAGNOSTICS COMMAND, PARAMETER SELECTIONS     ( PAGE 2 OF 4 )

 1.HOURS                        H        OPTIONAL          INTEGER VALUE
 2.MINUTES                      M        OPTIONAL          INTEGER VALUE
 3.ABORT_ON_ERROR               AOE      OPTIONAL          BOOLEAN VALUE
 4.USER                         U        OPTIONAL          BOOLEAN VALUE
 5.LOAD_MAP                     LM       OPTIONAL          BOOLEAN VALUE
 6.MULTIPROCESSING_OPTION       MO       OPTIONAL          BOOLEAN VALUE
 7.DISCARD_STANDARD_OUTPUT      DSO      OPTIONAL          BOOLEAN VALUE
 8.RCT1_REPEAT_COUNT            RCT1RC   OPTIONAL          INTEGER VALUE
 9.RCT1_PIT_VALUE               RCT1PV   OPTIONAL          INTEGER VALUE
10.RCT1_PAD_COUNT               RCT1PC   OPTIONAL          INTEGER VALUE

TYPE PARAMETERS 1...15 FOR COMMAND PARAMETERS; QUIT TO TERMINATE DVS
     EXPLAIN 1...10 FOR RUN_CPU_DIAGNOSTICS COMMAND, PARAMETER DESCRIPTIVE TEXT.
     NEXP FOR NEXT PAGE OF TEXT; PREP FOR PREVIOUS PAGE OF TEXT
```

```
DVS/ PREP

DVS RUN_CPU_DIAGNOSTICS COMMAND, PARAMETER SELECTIONS     ( PAGE 1 OF 4 )

 1.TEST_NAME                      TN      REQUIRED          KEYWORD VALUE
 2.RUN_OPTIONS                    RO      OPTIONAL/DEFAULT  KEYWORD VALUE
 3.SELECTED_UTILITY_PROCESSOR     SUP     OPTIONAL          INTEGER VALUE
 4.SELECTED_TEST_PROCESSOR        STP     OPTIONAL          INTEGER VALUE
 5.JOB_NAME                       JN      OPTIONAL          NAME VALUE
 6.JOB_CLASS                      JBC     OPTIONAL          NAME VALUE
 7.REDUCE_TASK_PRIORITY           RTP     OPTIONAL          BOOLEAN VALUE
 8.TEST_COPIES                    TC      OPTIONAL          INTEGER VALUE
 9.JOB_COPIES                     JC      OPTIONAL          INTEGER VALUE
10.ACTIVE_TASKS                   AT      OPTIONAL          INTEGER VALUE

TYPE PARAMETERS 1...15 FOR COMMAND PARAMETERS; QUIT TO TERMINATE DVS
     EXPLAIN 1...10 FOR RUN_CPU_DIAGNOSTICS COMMAND, PARAMETER DESCRIPTIVE TEXT.
     NEXP OR NEXT PAGE OF TEXT; PREP FOR PREVIOUS PAGE OF TEXT
```

Descriptive Text

The third and final level of the menu structure contains descriptive text for the selected parameters. Where applicable, keyword values are also displayed. The following example displays the parameter descriptive text for the TEST_NAME parameter of the RUN_CPU_DIAGNOSTICS command.

```
DVS/ explain 1

DVS RUN_CPU_DIAGNOSTICS COMMAND, TEST_NAME PARAMETER DESCRIPTIONS
The TEST_NAME parameter identifies a keyword name for the test to be loaded and e-
xecuted.
You may specify one of the following keywords:

KEYWORD VALUE: ALL, RCT1, RCT2, RFST, FCT3, SNGL, DOBL, BRCH, FINT, HINT, FIMM,
               HIMM, BIMM, NUMR, BYTE, EDIT, DBUG, VINT, VCMP, VGSI, VGTH

TYPE PARAMETERS 1...15 FOR COMMAND PARAMETERS; QUIT TO TERMINATE DVS
     EXPLAIN 1..10 FOR RUN_CPU_DIAGNOSTICS COMMAND, PARAMETER DESCRIPTIVE TEXT.
```

The following example returns the user to the first level of the most current test menu, or in this case to the DVS command menu.

```
DVS/ MENU
```

To present the information in this chapter in a structured format, this page has been left blank.

DVS Menu Commands

DVS provides a number of menu commands used to travel through the menu structure, obtain test commands and command parameters, and display descriptive text.

PARAMETERS (PARAMS, MENU, MEN)

Purpose Displays a command or parameter menu.

NOTE

Entering the PARAMETERS command without a selection value produces a command menu corresponding to the current input prompt. When you enter the PARAMETERS command and a valid selection value, a parameter menu for the appropriate command is displayed.

Format

PARAMETERS
SELECTION_VALUE=integer
AUTO=boolean
STATUS=status variable

Parameters

SELECTION_VALUE (SV)

Specifies the menu item to be selected from the current command menu. If SELECTION_VALUE is omitted, a command menu corresponding to the current input prompt is displayed.

AUTO (A)

Indicates whether the DVS command menu is automatically displayed when control is returned to the utility.

TRUE

The DVS command menu is automatically displayed when control is returned to the utility.

FALSE

The DVS command menu is not automatically displayed when control is returned to the utility.

If AUTO is omitted, TRUE is used.

STATUS (S)

Specifies the termination condition of the command.

Examples The following examples make references to the DVS menu shown below.

```
DVS_UTILITY

DIAGNOSTIC VIRTUAL SYSTEM UTILITY (DVS) - VERSION X.X.X LXXX Sept. 7, 1989
DVS COMMAND SELECTIONS                                    ( PAGE 1 OF 1 )

                              BATCH COMMANDS

 1.RUN_CPU_DIAGNOSTICS       RUNCD    2.RUN_SYSTEM_DIAGNOSTICS     RUNSD
 3.DISPLAY_HISTORY           DISH     4.DISPLAY_FAILURES           DISF
 5.RESET_HISTORY_FILE        RESHF

                           INTERACTIVE COMMANDS

 6.EXECUTE_TEST              EXET     7.SWITCH_TEST                SWIT
 8.DISPLAY_STATUS            DISS     9.DROP_TEST                  DROT
10.SET_PARAMETER_WORD_ZERO   SETPWZ   11.SET_STATUS_DISPLAY        SETSD
12.RESTART_TEST              REST     13.CONTINUE_TEST             CONT
14.STOP_TEST                 STOT     15.DISPLAY_EXECUTING_TESTS   DISET

TYPE MENU 1...15 FOR COMMAND PARAMETERS; QUIT TO TERMINATE DVS
     EXPLAIN 1...15 FOR COMMAND DESCRIPTIVE TEXT.
```

The following MENU command produces the parameter menu for the DISPLAY_HISTORY command.

```
DVS/Parameters 3
```

The following MENU command produces the parameter menu for the RESET_HISTORY command.

```
DVS/Parameters 5
```

In the following example, the MENU command enables the automatic display of the DVS command menu when control is returned to the utility.

```
DVS/Parameters auto=yes
```

The following MENU command produces the DVS command menu.

```
DVS/Parameters
```

NEXT_PAGE (NEXP)

Purpose Displays the next page of a menu or descriptive text.

NOTE

Entering the NEXT_PAGE command when the last page in a series is displayed produces the first display page.

Format

NEXT_PAGE
STATUS=status variable

Parameters *STATUS (S)*

Specifies the termination condition of the command.

PREVIOUS_PAGE (PREP)

Purpose Displays the previous page of a menu or descriptive text.

NOTE

Entering the PREVIOUS_PAGE command when the first page in a series is displayed produces the same display page.

Format

PREVIOUS_PAGE
STATUS=status variable

Parameters *STATUS (S)*

Specifies the termination condition of the command.

HELP (HEL)

Purpose Displays the DVS command menu. It can be initiated from within a test or from DVS itself.

Format

HELP
STATUS=status variable

Parameters *STATUS (S)*

Specifies the termination condition of the command.

EXPLAIN (EXP)

Purpose Displays descriptive text about DVS or about a test, command, or parameter.

NOTE

There are two forms of the EXPLAIN command. You can obtain information directly by specifying the TEST_NAME, COMMAND_NAME, or PARAMETER_NAME parameter as required. You can also use EXPLAIN when moving through the menu structure. In this case, use the SELECTION_VALUE parameter.

Format

EXPLAIN
SELECTION_VALUE=integer
TEST_NAME=keyword
COMMAND_NAME=name
PARAMETER_NAME=name
STATUS=status variable

Parameters

SELECTION_VALUE (SV)

Specifies the menu item to be selected from the current menu.

NOTE

If you specify SELECTION_VALUE, the TEST_NAME, COMMAND_NAME, and PARAMETER_NAME parameters cannot be used.

TEST_NAME (TN)

Specifies the test name of the descriptive text required.

RCT1
Random Command Test 1

RCT2
Random Command Test 2

FCT3
Fixed Command Test 3

SNGL
Single-Precision Floating-Point Test

DOBL
Double-Precision Floating-Point Test

BRCH
Floating-Point Branch Test

FINT
Full-Word Integer Test

HINT
Half-Word Integer Test

FIMM
Full-Word Immediate Test

HIMM
Half-Word Immediate Test

BIMM
BDP Immediate Test

NUMR
BDP Numeric Test

BYTE
BDP Byte End-Case Test

EDIT
BDP Edit End-Case Test

VINT
Vector Integer Test

VCMP
Vector Compare Test

VGTH
Vector Gather/Scatter Test

VGSI
Vector Gather/Scatter Indexed Test

RFST
Random Fast/Slow Test

DBUG
Debug Test

CMEM
Central Memory Test

DISK
Disk Test

TAPE
Tape Test

COMMAND_NAME (CN)

Specifies the command name of the descriptive text required.

NOTE

If you omit TEST_NAME the controlling test is used in reference to the specified COMMAND_NAME.

PARAMETER_NAME (PN)

Specifies the parameter name of the descriptive text required.

NOTE

If PARAMETER_NAME is specified, the COMMAND_NAME of the command under which the parameter resides is required.

If TEST_NAME is omitted, the controlling test is used in reference to the specified COMMAND_NAME.

STATUS (S)

Specifies the termination condition of the command.

NOTE

If you omit the SELECTION_VALUE, TEST_NAME, COMMAND_NAME, and PARAMETER_NAME parameters, descriptive text corresponding to the current input prompt is displayed.

Examples The following example makes use of the SELECTION_VALUE parameter and the FCT3 menu that follows.

```
FCT3 MENU LIST, COMMAND SELECTIONS   (PAGE 1 OF 2)

1. DISPLAY_MESSAGES          DISM     2. DISPLAY_MEMORY_DATA   DISMD
3. PLUS                      PLU      4. MINUS                 MIN
5. DISPLAY_PARAMETERS        DISP     6. SET_REPEAT_COUNT      SETRC
7. SET_SECTION_SELECTIONS    SETSS

TYPE, MENU 1...7 FOR PARAMETER SELECTION LIST; QUIT TO TERMINATE DVS
      EXPLAIN 1...7 FOR COMMAND DESCRIPTIVE TEXT.
```

The following EXPLAIN command produces descriptive text about the FCT3 test.

```
FCT3_1/explain
```

The following EXPLAIN command produces descriptive text for the FCT3 DISPLAY_MEMORY_DATA command.

```
FCT3_1/explain 2
```

The following example makes use of the SELECTION_VALUE parameter and the FCT3 PLUS parameter menu.

```
FCT3 PLUS COMMAND, PARAMETER SELECTIONS     (PAGE 1 OF 1)

1. SEGMENT_OFFSET  SO    OPTIONAL    INTEGER VALUE
2. OUTPUT          O     OPTIONAL    FILE VALUE
3. STATUS          S     OPTIONAL    VARIABLE REFERENCE

TYPE, MENU 1...4 FOR PARAMETER SELECTION LIST; QUIT TO TERMINATE DVS
      EXPLAIN 1...3 FOR PLUS COMMAND, PARAMETER DESCRIPTIVE TEXT.
```

The following EXPLAIN command produces descriptive text for the FCT3 test.

```
FCT3_1/explain
```

The following EXPLAIN command produces descriptive text for the STATUS parameter of the FCT3 PLUS command.

```
FCT3_1/explain 3
```

The following example makes use of the SELECTION_VALUE parameter and the STATUS parameter description for the FCT3 PLUS command that follows.

```
FCT3 PLUS COMMAND, STATUS PARAMETER DESCRIPTIONS

This parameter allows you to test for error conditions
detected by the command.  If you use this parameter, you
must declare a status variable.

TYPE, MENU 1...4 FOR PARAMETER SELECTION LIST; QUIT TO TERMINATE DVS
      EXPLAIN 1...3 FOR PLUS COMMAND, PARAMETER DESCRIPTIVE TEXT.
```

The following EXPLAIN command produces descriptive text for the FCT3 test.

```
FCT3_1/explain
```

The following EXPLAIN command produces descriptive text for the OUTPUT parameter of the FCT3 PLUS command.

```
FCT3_1/explain 2
```

The following example makes use of the TEST_NAME, COMMAND_NAME, and PARAMETER_NAME parameters of the EXPLAIN command.

The EXPLAIN command that follows displays descriptive text for the UCR_MASK_SELECTION parameter of the RCT1 SET_MODE command.

```
DVS/explain test_name=rct1 command_name=set_
mode parameter_name=ucr_mask_selection
```

The following EXPLAIN command displays descriptive text for the UCR_MASK_SELECTION parameter of the RCT1 SET_MODE command.

```
RCT1_1/exp cn=setm pn=ums
```

The following EXPLAIN command displays descriptive text for the SNGL test.

```
DVS/explain tn=sngl
```

The following EXPLAIN command displays descriptive text for the RCT2 MINUS command.

```
DVS/exp tn=rct2 cn=minus
```

The following EXPLAIN command is used to display descriptive text for the SEGMENT_OFFSET parameter of the FCT3 DISPLAY_MEMORY_DATA command.

```
FCT3_2/explain cn=display_memory_data pn=so
```

DISPLAY_EXECUTING_TESTS (DISET)

Purpose Displays a list of all executing tests including the test identifier, total errors, and passes completed.

Format **DISPLAY_EXECUTING_TESTS**
STATUS=status variable

Parameters *STATUS (S)*

Specifies the termination condition of the command.

Examples In the following example DISPLAY_EXECUTING_TESTS displays a list of two tests that are currently executing.

```
RCT1_1/DISPLAY_EXECUTING_TESTS
 RCT2_1, TOTAL.ERRORS = 0, PASSES COMPLETED = 100
 RCT1_1, TOTAL.ERRORS = 0, PASSES COMPLETED = 20
```

The DVS Utility 7

The DVS Utility 7

This chapter describes the DVS_UTILITY command, its format, and its parameters.

Starting the DVS Utility

You call DVS by using the DVS_UTILITY command. Parameters are optional and conform to SCL order dependency. You can specify them as keyword or positional parameters.

DVS_UTILITY

Purpose Initiates DVS.

Format

DVS_UTILITY
INPUT=file reference
OUTPUT=file reference
SELECT_PROCESSOR=0...1
USER=boolean
SUBCATALOG=name
MENU_MODE=boolean
STATUS=status variable

Parameters

INPUT (I)

Specifies the name of the input file. If INPUT is omitted, $COMMAND is used.

OUTPUT (O)

Specifies the file to which information is displayed. If OUTPUT is omitted, $OUTPUT is used.

SELECT_PROCESSOR (SP)

Specifies the logical identifier of the processor in which the utility runs. This parameter is intended for use with dual-processor systems. If SELECT_PROCESSOR is omitted, the system determines the processor used.

USER (U)

Specifies the family and user from which files are accessed.

TRUE

Files are accessed from the user's master catalog and Hardware_Maintenance.DVS subcatalog.

FALSE

Files are accessed from the $SYSTEM master catalog and Hardware_Maintenance.DVS subcatalog.

If USER is omitted, FALSE is used.

SUBCATALOG (SC)

Specifies the subcatalog name under which the DVS subcatalog resides. If SUBCATALOG is omitted, Hardware_Maintenance is used.

MENU_MODE (MM)

Turns the DVS menu displays on or off.

TRUE

Turns the DVS menu display on.

FALSE

Turns the DVS menu display off.

If MENU_MODE is omitted, TRUE is used.

STATUS (S)

Specifies the termination condition of the command.

File Names on DVS Commands

A file reference identifies a local or permanent file. It consists of a path (1- to 31-character file name), a cycle reference (for permanent files), and a file position indicator. The form of the file reference is:

```
path.cycle reference.file position
```

The cycle reference and file position indicator are optional. If the cycle reference is omitted, $HIGH is assumed for permanent files. A cycle reference is not valid for local files.

The file position indicator specifies file positioning before it is used. The file position indicators are:

$BOI

Positions the file at the beginning-of-information (BOI).

$EOI

Positions the file at the end-of-information (EOI).

If the file position indicator is omitted, the file is positioned at the beginning-of-information, with the following exceptions:

OUTPUT

Positioned at $EOI.

$OUTPUT

Positioned at $EOI.

NOTE

$OUTPUT is normally positioned at $BOI. Because it is connected to OUTPUT, which is positioned at EOI, it too is positioned at EOI as long as the connection remains.

DVS Test Control 8

DVS Test Control 8

This chapter provides a list of DVS commands along with the format for each.

Test Execution and Monitoring Commands

DVS provides commands that execute tests, monitor their progress, and switch control between concurrently executing diagnostics.

EXECUTE_TEST (EXET)

Purpose Initiates a test program as an asynchronous task.

Format

EXECUTE_TEST
TEST_NAME=keyword
TEST_IDENTIFIER=name
SELECT_PROCESSOR=0...1
SWITCH_TEST=boolean
REDUCE_TASK_PRIORITY=boolean
PARAMETER_STOP=boolean
LOAD_MAP=file reference
LOAD_MAP_OPTIONS=keyword
TERMINATION_ERROR_LEVEL=keyword
PRESET_VALUE=keyword
STACK_SIZE=integer
DEBUG_INPUT=file reference
DEBUG_OUTPUT=file reference
ABORT_FILE=file reference
DEBUG_MODE=keyword
STATUS=status variable

Parameters **TEST_NAME (TN)**

Specifies a keyword name for a diagnostic test to be loaded and executed. This parameter is required.

RCT1

Random Command Test 1

RCT2

Random Command Test 2

FCT3

Fixed Command Test 3

SNGL

Single-Precision Floating-Point Test

DOBL

Double-Precision Floating-Point Test

BRCH

Floating-Point Branch Test

FINT

Full-Word Integer Test

HINT

Half-Word Integer Test

FIMM

Full-Word Immediate Test

HIMM
Half-Word Immediate Test

BIMM
BDP Immediate Test

NUMR
BDP Numeric Test

BYTE
BDP Byte End-Case Test

EDIT
BDP Edit End-Case Test

VINT
Vector Integer Test

VCMP
Vector Compare Test

VGTH
Vector Gather/Scatter Test

VGSI
Vector Gather/Scatter Indexed Test

RFST
Random Fast/Slow Test

DBUG
Debug Test

CMEM
Central Memory Test

DISK
Disk Test

TAPE
Tape Test

TEST_IDENTIFIER (TI)

Identifies a diagnostic by a unique name. This identifier becomes the interactive input prompt, which is a positive visual indication of the current controlling test. If TEST_IDENTIFIER is omitted, the utility selects the identifier by appending the test name to the copy number. For example, RCT1_1.

SELECT_PROCESSOR (SP)

Specifies the logical identifier of the processor in which the test is to run. It is intended for use with dual-processor systems. A test can run in either processor, independent of the utility. If SELECT_PROCESSOR is omitted, the test inherits the processor identifier of the utility.

SWITCH_TEST (ST)

Specifies whether the command selection list for the current diagnostic is enabled.

TRUE

Begins test execution with the test command list.

FALSE

Begins test execution without the test command list. Utility commands are recognized; test commands are not.

If SWITCH_TEST is omitted, TRUE is used.

NOTE

When you execute the utility from an SCL procedure with DVS, a FALSE setting is required.

REDUCE_TASK_PRIORITY (RTP)

Indicates whether the CPU dispatching priority for all tests is to be reduced to the lowest priority of any task within the same job class.

TRUE

CPU dispatching priority is reduced.

FALSE

CPU dispatching priority is not reduced.

If REDUCE_TASK_PRIORITY is omitted, FALSE is used.

NOTE

Running DVS with multiple diagnostics results in a significant system load. Use the REDUCE_TASK_PRIORITY option whenever possible. This option, however, lowers the priority of the entire job to the lowest priority valid for the job class. At any time, the priority of the job can be changed by use of the NOS/VE CHANGE_JOB_ATTRIBUTES command.

PARAMETER_STOP (PS)

Specifies whether a diagnostic testing sequence is to be delayed for parameter viewing and alteration.

TRUE

The test stops and awaits parameter input.

NOTE

The CONTINUE_TEST command continues test execution.

FALSE

The test executes using default parameters.

If PARAMETER_STOP is omitted, FALSE is used.

NOTE

If PARAMETER_STOP is set to TRUE when a MIGDS test (SNGL...VGTH) is executed, a question-answer dialog is invoked, asking for parameter values.

LOAD_MAP (LM)

Specifies the name of the output file that is to receive the load map. If LOAD_MAP is omitted, the job default value is used. The initial default value is $LOCAL.LOADMAP.$BOI.

LOAD_MAP_OPTIONS (LMO)

Specifies the amount of information included in the load map.

NONE (N)

No load map is written.

SEGMENT (S)

Segment map.

BLOCK (B)

Block map.

ENTRY_POINT (EP)

Entry point map.

CROSS_REFERENCE (CR)

Entry point cross-reference map.

ALL (A)

Segment map, block map, entry point map, and cross-reference map.

If LOAD_MAP_OPTIONS is omitted, the job default load map option is used. The initial load map option is NONE.

TERMINATION_ERROR_LEVEL (TEL)

Specifies the error level that terminates program loading.

WARNING (W)

Warning, error, or fatal error.

ERROR (E)

Error or fatal error only.

FATAL (F)

Fatal error only.

If TERMINATION_ERROR_LEVEL is omitted, the job default termination error level is used. The initial termination error level is ERROR.

PRESET_VALUE (PV)

Specifies the value stored in unused data words.

ZERO (Z)

All zeros.

FLOATING_POINT_INDEFINITE (FPI)

Floating-point indefinite value.

INFINITY (I)

Floating-point infinite value.

ALTERNATE_ONES (AO)

Alternate 0 and 1 bits, the leftmost (highest-order) bit is 1.

If PRESET_VALUE is omitted, the job default preset value is used. The initial preset value is ZERO.

STACK_SIZE (SS)

Specifies, in bytes, the upper limit of the run-time stack used for procedure call linkages and local variables. If STACK_SIZE is omitted, the system default value is used.

DEBUG_INPUT (DI)

Specifies the file from which DEBUG reads commands. If DEBUG_INPUT is omitted, the default DEBUG input file for the job is used. The initial default DEBUG input file is COMMAND.

DEBUG_OUTPUT (DO)

Specifies the file to which DEBUG writes its output. If DEBUG_OUTPUT is omitted, the default DEBUG output file for the job is used. The initial default DEBUG output file is $OUTPUT.

ABORT_FILE (AF)

Specifies the file containing DEBUG commands to be processed if the program aborts. If ABORT_FILE is omitted, the job default abort file is used. The initial default abort file is $NULL, indicating that no DEBUG commands are executed if the program aborts.

DEBUG_MODE (DM)

Indicates whether the program is to run under the control of DEBUG.

INTERACTIVE (I)

Use interactive debug.

TRAP_STACK_FRAME (TSF)

Create trap stack frames.

FETCH_STACK_FRAME (FSF)

Create instruction fetch stack frames for test RFST only.

NONE (N)

Disable debug mode.

If DEBUG_MODE is omitted, NONE is used.

STATUS (S)

Specifies the termination condition of the command.

Remarks The utility imposes a limit of 10 asynchronous tasks per job.

Examples The following EXECUTE_TEST command executes RCT1 with its default parameters. The utility supplies a test identifier of the form RCT1_n. n is the current copy of the test.

```
DVS/execute_test rct1
```

Following, the EXECUTE_TEST command causes FCT3 to load but not to execute until instructed to do so, because PARAMETER_STOP has been set to TRUE.

```
DVS/exet fct3 ps=yes
```

The following EXECUTE_TEST command is used to execute RCT2 in background mode. Test reference is made using the test identifier RCT2_10.

```
DVS/exet tn=rct2 ti=rct2_10 rtp=true
```

In the following example, FCT3 executes and continues to run, overlooking any FATAL load errors encountered.

```
DVS/execute_test fct3 tel=fatal
```

The following EXECUTE_TEST command loads but does not execute the DISK test until instructed to do so.

```
DVS/exet tn=disk ps=true
```

8

SWITCH_TEST (SWIT)

Purpose Allows processing of commands for another test.

NOTE

When using the SWITCH_TEST command, the current command list is switched for the new command list. For a switch to take place, both tests must be executing concurrently.

Format

SWITCH_TEST
TEST_IDENTIFIER=name
STATUS=status variable

Parameters

TEST_IDENTIFER (TI)

Specifies the test identifier of the diagnostic to which control is to be switched. The input prompt is changed to the new test identifier value. Subcommands for the test associated with this identifier are now accepted by the utility. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SWITCH_TEST command causes control to be passed to FCT3_1. RCT1_1 is no longer the controlling test.

```
RCT1_1/switch_test fct3_1
```

Execution of the following SWITCH_TEST command switches test control from FCT3_2 to RCT2_1.

```
FCT3_2/swit ti=rct2_1
```

DISPLAY_STATUS (DISS)

Purpose Displays the current running status of a test.

Format

DISPLAY_STATUS
TEST_IDENTIFIER=list of names or ALL
DISPLAY_OPTION=keyword
OUTPUT=file reference
STATUS=status variable

Parameters

TEST_IDENTIFER (TI)

Specifies the test identifier of the diagnostic for which a status display is to be produced. If TEST_IDENTIFIER is omitted, the current test status is displayed.

DISPLAY_OPTION (DO)

Specifies the amount of status information displayed. Selection overrides the SET_STATUS_DISPLAY default selections.

FULL (F)

BRIEF (B)

DEFAULT (D)

If DISPLAY_OPTION is omitted, DEFAULT is used.

OUTPUT (O)

Specifies the file to which information is displayed. If OUTPUT is omitted, $OUTPUT is used.

STATUS (S)

Specifies the termination condition of the command.

Remarks The information displayed is tailored for each test. The defaults can be changed to add or delete display information, using the SET_STATUS_DISPLAY command.

Examples In the following examples, the status and running displays are shown.

The pass counts in the running and status displays may not correspond because the values are obtained at different times. The running display begins on the line containing the RU indicator found immediately after the test identifier.

Below is the default display for RCT1. It appears when a DISPLAY_STATUS command is executed for RCT1 and no status display parameters have been altered.

```
RCT1_2/diss
TEST ID              : RCT1_2 IN EXECUTION
PROCESSOR ID         : 1
TIME                 : 10:31:25.177
STATUS               : NORMAL
TRAPS TAKEN          : EXPECTED=0 UNEXPECTED=0
PARAMS               : OL,LE,SE
ERROR COUNT          : 0
PASS COUNT           : 7
RCT1 RU PC=000011 PADS=00 PURGE=NONE TE=0000
PIT - VALU=ALL ONES+INTRUPT=00000000 OCCURNCS=000 LOST=00000
```

In the following example, the complete status display containing all parameter values for RCT1_2 is shown.

```
RCT1_2/display_status rct1_2 do=full
TEST ID              : RCT1_2 IN EXECUTION
PROCESSOR ID         : 1
TIME                 : 10:32:12.028
STATUS               : NORMAL
TRAPS TAKEN          : EXPECTED=0 UNEXPECTED=0
PARAMS               : OL,LE,SE
SECTION SELECTED     : NONE
ERROR COUNT          : 0
MESSAGES QUEUED      : 0
PASS COUNT           : 1325
SECTION              : 0
SUB-SECTION          : 0
CONDITION            : 0
RCT1 RU PC=001329 PADS=00 PURGE=NONE TE=0000
PIT - VALU=ALL ONES+INTRUPT=00000000 OCCURNCS=000 LOST=00000
```

The following status display is the brief version for RCT1_2.

```
RCT1_2/diss do=b
TEST ID              : RCT1_2 IN EXECUTION
PROCESSOR ID         : 0
TIME                 : 10:35:31.266
STATUS               : NORMAL
TRAPS TAKEN          : EXPECTED=0 UNEXPECTED=0
ERROR COUNT          : 0
```

The following status display can also be obtained by entering the DISPLAY_STATUS command without any parameters.

```
RCT1_2/diss display_option=default
TEST ID              : RCT1_2 IN EXECUTION
PROCESSOR ID         : 0
TIME                 : 10:39:56.206
STATUS               : NORMAL
TRAPS TAKEN          : EXPECTED=0 UNEXPECTED=0
PARAMS               : OL,LE,SE
ERROR COUNT          : 0
PASS COUNT           : 14889
RCT1_2 RU PC=014890 PADS=00 PURGE=NONE TE=0000
PIT - VALU=ALL ONES+INTRUPT=00000000 OCCURNCS=000 LOST=00000
```

8

SET_STATUS_DISPLAY (SETSD)

Purpose Defines the default content of the status display.

NOTE

The attribute settings altered by this command remain until the test is dropped or until another SET_STATUS_DISPLAY command is used.

Format

SET_STATUS_DISPLAY
TEST_IDENTIFIER=name
ATTRIBUTE_OPTION=keyword
STATUS_VARIABLE=boolean
PARAMETERS=boolean
SECTION_SELECTIONS=boolean
ERROR_COUNT=boolean
MESSAGES_QUEUED=boolean
RUN_DISPLAY=boolean
PASS_COUNT=boolean
SECTION_NO=boolean
SUBSECTION_NO=boolean
CONDITION_NO=boolean
STATUS=status variable

Parameters

TEST_IDENTIFIER (TI)

Specifies the test identifier of the diagnostic for which the status display defaults are to be changed. If TEST_IDENTIFIER is omitted, the current test is used.

ATTRIBUTE_OPTION (AO)

Sets or clears all status attribute defaults.

SET_ALL (SA)

Sets all status attribute defaults.

CLEAR_ALL (CA)

Clears all status attribute defaults.

STATUS_VARIABLE (SV)

Indicates whether the status variable is to be part of the status display.

TRUE

Enables display of the status variable.

FALSE

Disables display of the status variable.

If STATUS_VARIABLE is omitted, TRUE is used.

FALSE

Disables display of the parameter bit selections.

If PARAMETERS is omitted, TRUE is used.

SECTION_SELECTIONS (SS)

Indicates whether the sections selected are to be part of the status display.

TRUE

Enables display of the sections selected for testing.

FALSE

Disables display of the sections selected for testing.

If SECTIONS_SELECTED is omitted for test RCT2, FCT3, CMEM, or TRUE is used; for all other tests, FALSE is used.

ERROR_COUNT (EC)

Indicates whether the error count is to be part of the status display.

TRUE

Enables display of the error count.

FALSE

Disables display of the error count.

If ERROR_COUNT is omitted, TRUE is used.

MESSAGES_QUEUED (MQ)

Indicates whether the messages queued count is to be part of the status display. This value is useful when the running display is disabled.

TRUE

Enables display of the messages queued counter.

FALSE

Disables display of the messages queued counter.

If MESSAGES_QUEUED is omitted, FALSE is used.

RUN_DISPLAY (RD)

Indicates whether the test's running display is to be part of the status display. The running display can also be obtained with the DISPLAY_MESSAGES command.

TRUE

Enables the test's running display.

FALSE

Disables the test's running display.

If RUN_DISPLAY is omitted, TRUE is used.

PASS_COUNT (PC)

Indicates whether the test's pass count value is to be part of the status display.

TRUE

Enables display of the test's pass count value.

FALSE

Disables display of the test's pass count value.

If PASS_COUNT is omitted, TRUE is used.

SECTION_NO (SN)

Indicates whether the test's section number is to be part of the status display.

TRUE

Enables display of the test's section number.

FALSE

Disables display of the test's section number.

If SECTION_NO is omitted for test RCT2, FCT3, or CMEM, TRUE is used; for all other tests, FALSE is used.

SUBSECTION_NO (SBN)

Indicates whether the test's subsection number is to be part of the status display.

TRUE

Enables display of the test's subsection number.

FALSE

Disables display of the test's subsection number.

If SUBSECTION_NO is omitted for test FCT3, CMEM, or TRUE is used; for all other tests, FALSE is used.

CONDITION_NO (CN)

Indicates whether the test's condition number is to be part of the status display.

TRUE

Enables display of the test's condition number.

FALSE

Disables display of the test's condition number.

If CONDITON_NO is omitted for test FCT3 or CMEM, TRUE is used; for all other tests, FALSE is used.

STATUS (S)

Specifies the termination condition of the command.

Remarks Following is a list of all of the display status attributes available and the tests in which they are valid.

NOTE

MIGDS refers to all tests in the MIGDS category (SNGL...VGTH).

DISPLAY_STATUS attributes:

STATUS_VARIABLE (SV)

RCT1, RCT2, FCT3, MIGDS, RFST, CMEM, DBUG, DISK, TAPE

PARAMETERS (P)

RCT1, RCT2, FCT3, MIGDS, RFST, CMEM, DBUG, DISK, TAPE

SECTION_SELECTIONS (SS)

RCT2, FCT3, CMEM

ERROR_COUNT (EC)

RCT1, RCT2, FCT3, MIGDS, RFST, CMEM, DBUG, DISK, TAPE

MESSAGES_QUEUED (MQ)

RCT1, RCT2, FCT3, MIGDS, RFST, CMEM, DBUG, DISK, TAPE

RUN_DISPLAY (RD)

RCT1, RCT2, FCT3, MIGDS, RFST, CMEM, DBUG, DISK, TAPE

PASS_COUNT (PC)

RCT1, RCT2, FCT3, MIGDS, RFST, CMEM, DBUG, DISK, TAPE

SECTION_NO (SN)

RCT2, FCT3, CMEM

SUBSECTION_NO (SBN)

FCT3, CMEM

CONDITION_NO (CN)

FCT3, CMEM

Each DISPLAY_STATUS attribute can be set for any test even if it is not valid. The status display shows such an attribute as being 0 or NONE if not used by the specified test.

Examples In the following example, the SET_STATUS_DISPLAY command enables all of the DISPLAY_STATUS attributes. These attributes become part of the status display.

```
RCT2_1/set_status_display rct2_1 ao=set_all
```

Following, the SET_STATUS_DISPLAY command enables display of the number of currently queued messages and includes it as part of the status display.

```
FCT3_2/setsd mq=true
```

The following SET_STATUS_DISPLAY command adds the SECTION_SELECTIONS attribute to the status display for RCT1_1. The SECTION_SELECTIONS attribute is given a value of NONE since this attribute does not apply to RCT1.

```
RCT1_1/set_status_display section_selections=true
```

Test Control Commands

DVS provides commands that alter test control. These commands alter flags in the test's parameter word block, word zero. They also issue keyboard requests by placing an ASCII command code in the test's keyboard buffer.

SET_PARAMETER_WORD_ZERO (SETPWZ)

Purpose Sets or clears single or multiple flags in the parameter block of a specific test. It allows direct control of the test's parameter control flags.

NOTE

You can have the parameter control flag settings displayed at any time by using the DISPLAY_STATUS command.

Format

SET_PARAMETER_WORD_ZERO
TEST_IDENTIFIER=name
STOP_END_TEST=boolean
STOP_SECTION=boolean
STOP_SUBSECTION=boolean
STOP_CONDITION=boolean
STOP_ON_ERROR=boolean
QUICK_LOOK=boolean
QUICK_LOOK_OMIT=integer
LOG_ERRORS=boolean
REPEAT_TEST=boolean
REPEAT_SECTION=boolean
REPEAT_SUBSECTION=boolean
REPEAT_CONDITION=boolean
BYPASS_ALL_MESSAGES=boolean
DISPLAY_ERROR_MESSAGES=boolean
STATUS=status variable

NOTE

MIGDS tests do not allow the parameter control flags to be altered through the SET_PARAMETER_WORD_ZERO command. To alter those control flags valid for MIGDS tests, set PARAMETER_STOP=true on execution to have the test prompt you for desired settings.

Parameters

TEST_IDENTIFIER (TI)

Specifies the unique identifier of the test whose parameter block, word zero, is to be altered. If TEST_IDENTIFIER is omitted, the current test's parameter block is used.

STOP_END_TEST (SET)

Specifies that the test is either to stop at the end of execution and await command input or to terminate. The respective flag is set or cleared in the test's parameter block, word zero.

TRUE

The test stops at the end of execution and awaits command input.

FALSE

The test terminates at the end of execution.

If STOP_END_TEST is omitted, TRUE is used for test FCT3 (interactive mode) and FALSE is used for all other tests.

NOTE

The STOP_END_TEST parameter is valid for all tests except for the MIGDS tests.

STOP_SECTION (SS)

Specifies that the test is either to stop at the end of each test section and await command input or to continue execution.

TRUE

The test stops at the end of each test section and awaits command input.

FALSE

The test continues execution at the end of each test section.

If STOP_SECTION is omitted, FALSE is used.

NOTE

The STOP_SECTION parameter is valid only for tests RCT2, FCT3, and CMEM.

STOP_SUBSECTION (SSB)

Specifies that the test is either to stop at the end of each test subsection and await command input or to continue execution.

TRUE

The test stops at the end of each test subsection and awaits command input.

FALSE

The test continues execution at the end of each test subsection.

If STOP_SUBSECTION is omitted, FALSE is used.

NOTE

The STOP_SUBSECTION parameter is valid only for tests FCT3, and CMEM.

STOP_CONDITION (SC)

Specifies that the test is either to stop at the end of each test condition and await command input or to continue execution.

TRUE

The test stops at the end of each test condition and awaits command input.

FALSE

The test continues execution at the end of each test condition.

If STOP_CONDITION is omitted, FALSE is used.

NOTE

The STOP_CONDITION parameter is valid only for tests FCT3, RFST, CMEM, and DBUG.

STOP_ON_ERROR (SOE)

Specifies that the test is either to stop if an error is detected and await command input or to continue execution.

TRUE

The test stops if an error is detected and awaits command input.

FALSE

The test continues execution if an error is detected.

If STOP_ON_ERROR is omitted, TRUE is used in interactive mode and FALSE is used in batch mode.

NOTE

The STOP_ON_ERROR parameter is valid for all tests except the MIGDS tests.

QUICK_LOOK (QL)

Specifies whether FCT3 is to run with fewer test cases.

TRUE

FCT3 runs with fewer test cases.

FALSE

FCT3 runs with all test cases.

If QUICK_LOOK is omitted, TRUE is used.

NOTE

The QUICK_LOOK parameter is valid only for test FCT3.

QUICK_LOOK_OMIT (QLO)

Specifies the quick look omit number to be used by FCT3. It controls the number of test cases to be skipped. The larger the value, the fewer the test cases and the faster the test executes. If QUICK_LOOK_OMIT is omitted, 20_{16} is used.

NOTE

The QUICK_LOOK_OMIT parameter is used only by test FCT3. To use the quick look feature, parameter QUICK_LOOK must be set to TRUE.

LOG_ERRORS (LE)

Indicates whether a detected error is to be logged in the specified output file.

TRUE

A detected error is logged in the output file.

FALSE

A detected error is not logged in the output file.

If LOG_ERRORS is omitted, TRUE is used.

NOTE

The LOG_ERRORS parameter is valid for all tests except the MIGDS tests.

REPEAT_TEST (RT)

Specifies whether the test is to be repeated at the end of execution.

TRUE

The test is repeated at the end of execution.

FALSE

The test is not repeated at the end of execution.

If REPEAT_TEST is omitted, FALSE is used.

NOTE

The REPEAT_TEST parameter is valid for all tests except the MIGDS tests.

REPEAT_SECTION (RS)

Specifies whether a test section is to be repeated.

TRUE

The test section is repeated.

FALSE

The test section is not repeated.

If REPEAT_SECTION is omitted, FALSE is used.

NOTE

The REPEAT_SECTION parameter is valid only for tests RCT2, FCT3 and CMEM.

REPEAT_SUBSECTION (RSB)

Specifies whether a test subsection is to be repeated.

TRUE

The test subsection is repeated.

FALSE

The test subsection is not repeated.

If REPEAT_SUBSECTION is omitted, FALSE is used.

NOTE

The REPEAT_SUBSECTION parameter is valid only for tests FCT3, and CMEM.

REPEAT_CONDITION (RC)

Specifies whether a test condition is to be repeated.

TRUE

The test condition is repeated.

FALSE

The test condition is not repeated.

If REPEAT_CONDITION is omitted, FALSE is used.

NOTE

The REPEAT_CONDITION parameter is valid only for tests FCT3, RFST, CMEM, and DBUG.

BYPASS_ALL_MESSAGES (BAM)

Specifies whether test messages are to be displayed or bypassed.

TRUE

Test messages are bypassed.

FALSE

Test messages are displayed.

If BYPASS_ALL_MESSAGES is omitted, FALSE is used.

NOTE

The BYPASS_ALL_MESSAGES parameter is valid only for tests RCT1, RCT2, FCT3, and CMEM.

DISPLAY_ERROR_MESSAGES (DEM)

Specifies whether only error messages are to be displayed if the test detects an error.

TRUE

Only error messages are displayed.

FALSE

Both error messages and additional test results are displayed.

If DISPLAY_ERROR_MESSAGES is omitted, TRUE is used for test FCT3 and FALSE is used for all other tests.

NOTE

The DISPLAY_ERROR_MESSAGES parameter is valid only for tests RCT1, RCT2, FCT3, and CMEM.

STATUS (S)

Specifies the termination condition of the command.

Examples In the following example, the RCT2 STOP_SECTION control flag is set, causing RCT2 to stop after each section. A CONTINUE_TEST command restarts the test after each section.

```
RCT2_3/set_parameter_word_zero ss=true
```

In the following example, the QUICK_LOOK control flag is enabled and the QUICK_LOOK_OMIT parameter is set to 45 or $2D_{16}$. Its value is larger than its default value of 32 or 20_{16}, resulting in fewer test cases and faster test execution.

```
FCT3_1/setpwz fct3_1 ql=true qlo=45
```

In the following example, the FCT3 DISPLAY_ERROR_MESSAGES control flag is cleared, causing both error messages and additional test results to be displayed. Disabling DISPLAY_ERROR_MESSAGES causes the test to take a longer time to run. FCT3_1 must be running concurrently with RCT1_2 for this command to be valid.

```
RCT1_2/setpwz ti=fct3_1 dem=false
```

RESTART_TEST (REST)

Purpose Causes the selected test to restart from the beginning.

NOTE

Not all tests provide a restart capability, so a restart code is not issued to those tests. The RESTART_TEST command is valid for executing tests only.

Format

RESTART_TEST
TEST_IDENTIFIER=name
PARAMETER_STOP=boolean
STATUS=status variable

Parameters

TEST_IDENTIFIER (TI)

Specifies the test identifier of the diagnostic to be restarted. If TEST_IDENTIFIER is omitted, the current test is used.

PARAMETER_STOP (PS)

Specifies whether a diagnostic testing sequence is to be delayed for parameter viewing and alteration.

TRUE

Test is stopped, awaiting parameter input.

NOTE

The CONTINUE_TEST command continues test execution.

FALSE

Test executes using default parameters.

If PARAMETER_STOP is omitted, FALSE is used.

STATUS (S)

Specifies the termination condition of the command.

Examples In the following example, the RESTART_TEST command causes RCT2_1 to restart from the beginning and to stop for parameter viewing and alteration. If RCT2_1 has terminated, this command is invalid.

```
RCT2_1/restart_test ti=rct2_1 ps=true
```

Following, the RESTART_TEST command causes RCT2_2 to restart from the beginning with its default parameters. For this command to be valid, RCT2_2 must be currently executing.

```
RCT1_1/rest rct2_2
```

CONTINUE_TEST (CONT)

Purpose Causes the selected test to continue execution from a stopped state.

NOTE

The CONTINUE_TEST command is valid for executing tests only.

Format

CONTINUE_TEST
TEST_IDENTIFIER=name
STATUS=status variable

Parameters

TEST_IDENTIFIER (TI)

Specifies the test identifier of the diagnostic to be continued from its current stopped state. If TEST_IDENTIFIER is omitted, the current test is used.

STATUS (S)

Specifies the termination condition of the command.

Examples In the following example, the CONTINUE_TEST command causes FCT3_1 to continue execution if it has stopped but has not terminated.

```
FCT3_1/continue_test
```

Following, the CONTINUE_TEST command causes RCT2_3 to continue execution if it has stopped but has not terminated.

```
RCT1_2/cont ti=rct2_3
```

STOP_TEST (STOT)

Purpose Stops the execution of the selected test.

NOTE

Not all tests support the stop test capability, so the request is not issued to those tests. The STOP_TEST command is valid for executing tests only.

Format

STOP_TEST
TEST_IDENTIFIER=name
STATUS=status variable

Parameters

TEST_IDENTIFIER (TI)

Specifies the test identifier of the diagnostic to be stopped. If TEST_IDENTIFIER is omitted, the current test is used.

STATUS (S)

Specifies the termination condition of the command.

Examples In the following example, the STOP_TEST command stops the execution of RCT1_1 if it has not terminated.

```
RCT1_1/stop_test
```

The following STOP_TEST command stops the execution of RCT1_1 if it has not terminated.

```
RCT2_4/stot ti=rct1_1
```

Test Message and Memory Display Commands

The DVS Utility provides commands that display test messages and memory contents. These commands are available only when a test is executing.

DISPLAY_MESSAGES (DISM)

Purpose Selects a message display and displays queued messages.

NOTE

If the DISPLAY_MESSAGES command is entered without the REQUEST_OPTION parameter, previously queued messages are displayed.

Format RCT1, FCT3, MIGDS, RFST, DBUG, CMEM, DISK, and TAPE:

DISPLAY_MESSAGES
REQUEST_OPTION=keyword
OUTPUT=file reference
STATUS=status variable

RCT2:

DISPLAY_MESSAGES
REQUEST_OPTION=keyword
SET_OPTION=0...9
OUTPUT=file reference
STATUS=status variable

Parameters *REQUEST_OPTION (RO)*

Specifies a keyword for the desired display.

NOTE

REQUEST_OPTION keywords vary with each test.

RCT1

RUN (R)
A_REGISTER_RESULTS (ARR)
X_REGISTER_RESULTS (XRR)
RESULT_LOWER (RL)
RESULT_UPPER (RU)
INITIAL_A_X_REGISTER (IAXR)
INSTRUCTION_LIST (IL)
INITIAL_INPUT_BUFFER (IIB)
INITIAL_OUTPUT_BUFFER (IOB)
PARAMETERS (P)

RCT2

RUN (R)
SECTION_LIST (SL)
RESULTS (RE)
OPERANDS (O)
MACHINE_LIST (ML)
PARAMETERS (P)

FCT3

RUN (R)

MIGDS

RUN (R)
HISTORY (H)

RFST

RUN (R)
A_REGISTER_RESULTS (ARR)
X_REGISTER_RESULTS (XRR)
MEMORY_RESULTS (MR)
INITIAL_A_X_REGISTER (IAXR)
ADDRESS_BUFFER (AB)
PARAMETERS (P)

CMEM

RUN (R)

DBUG

RUN (R)
A_REGISTER_RESULTS (ARR)
X_REGISTER_RESULTS (XRR)
MEMORY_RESULTS (MR)
INITIAL_A_X_REGISTER (IAXR)
ADDRESS_BUFFER (AB)
ERROR_RESULTS (ER)
PARAMETERS (P)

DISK

RUN (R)
PARAMETERS (P)

TAPE

RUN (R)
PARAMETERS (P)

SET_OPTION (SO)

Specifies a display selection set value from 0 to 9. This value determines the portion of data displayed when an error occurs. The RCT2 running display shows valid SET_OPTION ranges that can be selected. The SET_OPTION value should be used with the REQUEST_OPTION parameter to obtain the desired display. If SET_OPTION is omitted, 0 is used.

NOTE

The SET_OPTION parameter is valid for RCT2 only.

OUTPUT (O)

Specifies the file to which information is displayed. If OUTPUT is omitted, $OUTPUT is used.

STATUS (S)

Specifies the termination condition of the command.

Examples The following DISPLAY_MESSAGES command displays all messages currently queued by RCT1_2.

```
RCT1_2/dism
```

Following, the DISPLAY_MESSAGES command displays all operands used in RCT2_3.

```
RCT2_3/dism request_option=o
```

In the following example, the DISPLAY_MESSAGES command displays the running display for FCT3_1.

```
FCT3_1/display_messages ro=run
```

The following DISPLAY_MESSAGES command displays all errors produced by the MIGDS test, SNGL_2. If no errors are recorded, the test continues.

```
SNGL_2/dism ro=history
```

Following, the DISPLAY_MESSAGES command displays the second set of operands used by RCT2_1.

```
RCT2_1/display_messages ro=o so=2
```

The following DISPLAY_MESSAGES command displays the parameter words found in the RCT1_1 parameter block.

```
RCT1_1/dism request_option=p
```

8

DISPLAY_MEMORY_DATA (DISMD)

Purpose Displays memory contents within the test's address space.

NOTE

Select the desired display by supplying a DISPLAY_OPTIONS or SEGMENT_NAME keyword. You can offset the base segment address by supplying a SEGMENT_OFFSET value.

The DISPLAY_MEMORY_DATA command is not valid for tests DISK and TAPE.

Format

hexadecimEMORY_DATA
DISPLAY_OPTIONS=keyword
SEGMENT_NAME=keyword
SEGMENT_OFFSET=integer
OUTPUT=file reference
STATUS=status variable

Parameters *DISPLAY_OPTIONS (DO)*

Specifies a keyword for the desired display. The keywords are actual labels used within the test. A DISPLAY_OPTION keyword displays 256 bytes of memory, starting at the first byte of the selected memory display.

RCT1

MACHINE_INSTRUCTION_LIST (I_LIST)
ACTUAL_PROPORTION_TABLE (T_AIP)
SIMULATOR_INSTRUCTION_LIST (V_SIL)
NORMAL_PROPORTION_TABLE (T_NIP)
HALT_MASK
TRAP_MASK
USER_MASK
INSTRUCTIONS_SELECTED_LIST (PW16)

RCT2

BYTE_BUFFER_A (V_BUFA)
BYTE_BUFFER_B (V_BUFB)
BYTE_BUFFER_C (V_BUFC)
HALT_MASK
TRAP_MASK
USER_MASK

MIGDS

HISTORY_SECTION (HS)
MESSAGE_SECTION(MS)
PARAMETER_SECTION (PS)

RFST

MACHINE_INSTRUCTION_LIST (ELIST)
X_REGISTER_RESULTS (RXREG)
A_REGISTER_RESULTS (RAREG)
MEMORY_RESULTS_NO_PAD (RMEM1)
MEMORY_RESULTS_PAD (RMEM2)
INITIAL_X_REGISTER (IXREG)
INITIAL_A_REGISTER (IAREG)
INITIAL_MEMORY (IMEM)
INITIAL_BDP_DATA (IBDP)
PARAMETER_SECTION (CONT#)
MESSAGE_SECTION (MSGS#)
BDP_DATA_SECTION (BDPS#)
BDP_RESULTS_NO_PAD (RBDP1)
BDP_RESULTS_PAD (RBDP2)
SCAN_TABLE_FOR_EDIT (T_EDIT)
TRANSLATE_TABLE (T_TRANS)
TRANSLATION_TABLE_12_THRU_15
(T_TRTOCM)

DBUG

MACHINE_INSTRUCTION_LIST (ELIST)
X_REGISTER_RESULTS (RXREG)
A_REGISTER_RESULTS (RAREG)
MEMORY_RESULTS_1 (RMEM1)
MEMORY_RESULTS_2 (RMEM2)
INITIAl_X_REGISTER (IXREG)
INITIAL_A_REGISTER (IAREG)
INITIAL_MEMORY (IMEM)
INITIAL_BDP_DATA (IBDP)
PARAMETER_SECTION (CONT#)
MESSAGE_SECTION (MSGS#)
BDP_DATA_SECTION (BDPS#)
UCR_RESULTS (UCRR)
DEBUG_RESULTS_ACTUAL (DEBG)
DEBUG_RESULTS_SIMULATED (DBUGS)

CMEM

PARAMETER_SECTION (ENTER#)
SEGMENT_FILE (SEGPTR)

NOTE

The DISPLAY_OPTIONS parameter is not valid for test FCT3.

SEGMENT_NAME (SN)

Selects the memory display of a particular segment. Starting at the base segment address, 256 bytes of the selected segment are displayed. By using the SEGMENT_OFFSET parameter, you can offset the base address.

CODE (C)
WORKING_STORAGE (WS)
BINDING (B)
START (S)

NOTE

If the DISPLAY_OPTIONS and SEGMENT_NAME parameters are omitted, the base address is taken from the last display produced.

SEGMENT_OFFSET (SO)

Offsets the starting address of a selected memory display. The offset value is added to the base value obtained by the DISPLAY_OPTIONS or SEGMENT_NAME parameters. The resulting Process Virtual Address (PVA) is used to display memory.

NOTE

The SEGMENT_OFFSET parameter must be 0 modulo 8.

OUTPUT (O)

Specifies the file to which information is displayed. If OUTPUT is omitted, $OUTPUT is used.

STATUS (S)

Specifies the termination condition of the command.

Examples

The following DISPLAY_MEMORY_DATA command displays 256 bytes of memory. The first address in this display contains 8 bytes of information, which is the current halt mask. The remaining 248 bytes are the contents of the next 31 memory locations.

```
RCT2_1/display_memory_data display_option=
halt_mask
```

Following, the DISPLAY_MEMORY_DATA command displays 256 bytes of memory. The first 8 bytes are the contents of the test's starting address.

```
RCT2_1/dismd sn=start
```

The following DISPLAY_MEMORY_DATA command displays the current segment, which is set by the DISPLAY_OPTIONS or SEGMENT_NAME parameter and offset by 48 bytes. The new segment address is calculated by adding the segment's base address to the value of the SEGMENT_OFFSET. Assume that the current segment is BYTE_BUFFER_A and its starting address is at memory location 00000710_{16}. Execute the following command to offset this starting address by 48 bytes. The new starting address for the memory display is at memory location 00000740_{16}. This display contains 256 bytes of memory.

```
RCT2_1/dismd so=48
```

PLUS (PLU)

Purpose Increments the address of the last memory display. A 256-byte image, starting at the modified base address, is displayed.

NOTE

The PLUS command requires that a display be previously selected by use of the DISPLAY_MEMORY_DATA command. You can offset the current segment address by supplying a SEGMENT_OFFSET value; otherwise, a default offset of 256 bytes is used.

The PLUS command is not valid for tests DISK and TAPE.

Format

PLUS
SEGMENT_OFFSET=integer
OUTPUT=file reference
STATUS=status variable

Parameters

SEGMENT_OFFSET (SO)

Specifies the offset value of the last memory display. This value is added to the starting address of the previous memory display. The resulting PVA is used to display 256 bytes of memory. If the SEGMENT_OFFSET parameter is omitted, 256 is used.

NOTE

The SEGMENT_OFFSET parameter must be 0 modulo 8.

OUPUT (O)

Specifies the file to which information is displayed. If OUTPUT if omitted, $OUTPUT is used.

STATUS (S)

Specifies the termination condition of the command.

Examples In the following example, the DISPLAY_MEMORY_DATA and PLUS commands are used to produce 256 bytes of memory, offset by 24 bytes from the starting address of working storage.

```
RCT1_1/dismd sn=working_storage
RCT1_1/plus so=24
```

DROP_TEST (DROT)

Purpose Terminates one or more tests and erases all associated running information.

Format

DROP_TEST
TEST_IDENTIFIER=list of names or *ALL*
STATUS=status variable

Parameters

TEST_IDENTIFIER (TI)

Specifies the test identifier(s) of the diagnostic(s) to be dropped. Use the ALL keyword to drop all executing diagnostics. If TEST_IDENTIFIER is omitted, the current diagnostic is used.

STATUS (S)

Specifies the termination condition of the command.

Examples The following DROP_TEST command erases all running information associated with RCT1_1. If RCT1_1 is still executing when this command is entered, the test is terminated.

```
RCT1_2/drot ti=rct1_1
```

Following, the DROP_TEST command terminates FCT3_1 and erases all associated running information. Since a TEST_IDENTIFIER is not specified, the current controlling test, FCT3_1 is dropped.

```
FCT3_1/drop_test
```

Termination Commands

The DVS Utility provides commands that terminate any or all executing tests, as well as the utility itself. All terminating tests record their pass count and error count values in the job log. For interactive sessions, these values are automatically displayed.

8

DROP_TEST (DROT)

Purpose Terminates one or more tests and erases all associated running information.

Format

DROP_TEST
TEST_IDENTIFIER=list of names or *ALL*
STATUS=status variable

Parameters

TEST_IDENTIFIER (TI)

Specifies the test identifier(s) of the diagnostic(s) to be dropped. Use the ALL keyword to drop all executing diagnostics. If TEST_IDENTIFIER is omitted, the current diagnostic is used.

STATUS (S)

Specifies the termination condition of the command.

Examples The following DROP_TEST command erases all running information associated with RCT1_1. If RCT1_1 is still executing when this command is entered, the test is terminated.

```
RCT1_2/drot ti=rct1_1
```

Following, the DROP_TEST command terminates FCT3_1 and erases all associated running information. Since a TEST_IDENTIFIER is not specified, the current controlling test, FCT3_1 is dropped.

```
FCT3_1/drop_test
```

QUIT (QUI)

Purpose Terminates all executing tests and the DVS Utility.

Format **QUIT**
STATUS=status variable

Parameters *STATUS (S)*

Specifies the termination condition of the command.

Remarks

- The QUIT command can be issued while tests are running. QUIT performs the necessary cleanup before leaving the utility. The cleanup process terminates all executing tasks before terminating DVS.
- Essentially, the QUIT command performs the way DROP_TEST does when the TEST_IDENTIFIER parameter is set to ALL. The only difference is that QUIT also terminates DVS.

Examples The following QUIT command terminates all tests, as well as DVS. In this example, QUIT produces the following display:

```
FCT3_1/quit
TERMINATING_TEST: FCT3_1, TOTAL ERRORS=0, PASSES COMPLETED=5
TERMINATING_TEST: DOBL_2, TOTAL ERRORS=0, PASSES COMPLETED=18
TERMINATING_TEST: RCT1_2, TOTAL ERRORS=0, PASSES COMPLETED=194
```

Random Command Test 1 (RCT1) 9

Random Command Test 1 (RCT1) 9

This chapter describes commands used with Random Command Test 1 (RCT1).

Random Command Test 1 Description

Random Command Test 1 tests general and floating-point instructions. It uses random instruction sequences and random operands to isolate a single failing instruction within a failing random sequence of instructions.

SET_DISPLAY_REQUEST (SETDR)

Purpose Sets a default REQUEST_OPTION for the DISPLAY_MESSAGES command.

Format

SET_DISPLAY_REQUEST
REQUEST_OPTION=keyword
STATUS=status variable

Parameters

REQUEST_OPTION (RO)

Specifies a keyword for the desired display and sets it as the default REQUEST_OPTION for the DISPLAY_MESSAGES command.

RUN (R)

A_REGISTER_RESULTS (ARR)

X_REGISTER_RESULTS (XRR)

RESULT_LOWER (RL)

RESULT_UPPER (RU)

INITIAL_A_X_REGISTER (IAXR)

INSTRUCTION_LIST (IL)

INITIAL_INPUT_BUFFER (IIB)

INITIAL_OUTPUT_BUFFER (IOB)

PARAMETERS (P)

NIL (N)

If REQUEST_OPTION is omitted, NIL is used.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_DISPLAY_REQUEST command sets the A_REGISTER_RESULTS display as the default REQUEST_OPTION for the DISPLAY_MESSAGES command. Entering DISPLAY_MESSAGES without any parameters results in the automatic display of all A registers' results. The A_REGISTER_RESULTS display remains the default display for the DISPLAY_MESSAGES command until it is changed.

```
RCT1_2/setdr ro=arr
```

The following SET_DISPLAY_REQUEST command causes the running display to appear whenever the DISPLAY_MESSAGES command is entered without a REQUEST_OPTION parameter.

```
RCT1_2/set_display_request ro=run
```

DISPLAY_PARAMETERS (DISP)

Purpose Displays the test's parameters from its parameter word block.

Format

DISPLAY_PARAMETERS
OUTPUT=file reference
STATUS=status variable

Parameters

OUTPUT (O)

Specifies the file to which information is displayed. If OUTPUT is omitted, $OUTPUT is used.

STATUS (S)

Specifies the termination condition of the command.

Examples The following DISPLAY_PARAMETERS command is used to display the RCT1_1 parameters from its parameter word block. By default, the following parameter display appears. If any parameter values are changed, the display is also changed.

```
RCT1_1/display_parameters
REQUEST OPTION              : NIL
REPEAT COUNT                : 0
PASS COUNT                  : 0
PIT VALUE                   : 7FFFFFFF(16)
INTERMITTENT COUNT          : 0
USER MASK SELECTION         : NO FLOATING POINT
TRIM SELECTION              : NO TRIM
LIST LENGTH                 : 31
PAD COUNT                   : 10
PAD SELECTION               : NORMAL PASS
INSTRUCTION LIST SELECTION  : NORMAL
```

ADD_INSTRUCTIONS (ADDI)

Purpose Adds a user-specified list of instruction opcodes to the test's optional instruction list.

NOTE

The SET_MODE command enables or disables use of this list.

Format

ADD_INSTRUCTIONS
OP_CODE=list of opcodes
STATUS=status variable

Parameters

OP_CODE (OC)

Specifies the opcode or list of opcodes to be added to the test's optional instruction list. Decimal values are assumed unless a radix is specified. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following ADD_INSTRUCTIONS command adds the instruction with opcode 05 to the RCT1_1 optional instruction list.

```
RCT1_1/add_instructions op_code=05(16)
```

In the next example, the ADD_INSTRUCTIONS command adds those instructions having opcodes 01, 05, 07, and 10 to the RCT1_3 optional instruction list.

```
RCT1_3/addi oc=(01(16),05(16),07(16),10(16))
```

DELETE_INSTRUCTIONS (DELI)

Purpose Deletes a user-specified list of instruction opcodes from the test's optional instruction list.

NOTE

The SET_MODE command enables or disables use of this list.

Format

DELETE_INSTRUCTIONS
OP_CODE=list of opcodes
STATUS=status variable

Parameters

OP_CODE (OC)

Specifies the opcode or list of opcodes to be deleted from the test's optional instruction list. Decimal values are assumed unless a radix is specified. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following DELETE_INSTRUCTIONS command deletes those instructions having opcodes 01 and 05 from the RCT1_2 optional instruction list.

```
RCT1_2/del1 op_code=(01(16),05(16))
```

In the following example, the DELETE_INSTRUCTIONS command deletes the instruction with opcode 07 from the RCT1_1 optional instruction list.

```
RCT1_1/delete_instructions oc=(07(16))
```

SET_MODE (SETM)

Purpose Sets the execution modes for the test.

Format

SET_MODE
UCR_MASK_SELECTION=boolean
INSTRUCTION_LIST_SELECTION=keyword
TRIM_SELECTION=boolean
PAD_SELECTION=boolean
STATUS=status variable

Parameters *UCR_MASK_SELECTION (UMS)*

Specifies the user condition register mask selection to be applied.

TRUE

Sets the floating-point user mask bits for overflow, underflow, loss-of-significance, and indefinite.

FALSE

Clears the floating-point user condition mask bits.

If UCR_MASK_SELECTION is omitted, FALSE is used.

INSTRUCTION_LIST_SELECTION (ILS)

Specifies instruction selection or deselection from the optional instruction list. The keyword parameter selections determine the instruction list selection mode.

SELECT (S)

Tests only the instructions in the optional instruction list.

DESELECT (D)

Tests all instructions but those in the optional instruction list.

NORMAL (N)

Tests all instructions.

If INSTRUCTION_LIST_SELECTION is omitted, NORMAL is used.

NOTE

The optional instruction list is formed with the ADD_INSTRUCTIONS and DELETE_INSTRUCTIONS commands. The list length is determined by the SET_LIST_LENGTH command.

TRIM_SELECTION (TS)

Specifies whether instruction isolation is to be performed. Isolation involves trimming execution to a single failing instruction when an error is detected.

TRUE

Instruction isolation is performed.

FALSE

Instruction isolation is not performed. The failing random instruction loop is unaltered when an error is detected.

If TRIM_SELECTION is omitted, FALSE is used.

PAD_SELECTION (PS)

Specifies whether padding of instructions with no-ops is to be performed. Pad selection adds a no-op instruction before each instruction to be tested. The number of pads inserted is increased incrementally from none to the maximum specified.

TRUE

Padding of instructions with no-ops is performed.

FALSE

Padding of instructions with no-ops is not performed.

If PAD_SELECTION is omitted, FALSE is used.

NOTE

The number of pads to be inserted is determined by the SET_PAD_COUNT command. A pad count value of 10 is set by default.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_MODE command sets the RCT1_2 floating-point user mask bits for overflow, underflow, loss-of-significance, and indefinite.

```
RCT1_2/set_mode ums=true
```

The following SET_MODE command selects only the instructions in the RCT1_1 optional instruction list.

```
RCT1_1/setm ils=select
```

The following SET_MODE command selects instruction isolation for RCT1_1.

```
RCT1_1/setm trim_selection=true
```

The following SET_MODE command specifies that test instructions are to be padded with no-ops. PAD_SELECTION adds no-op instructions before each instruction to be tested. The number of pads to be inserted is determined by the SET_PAD_COUNT command.

```
RCT1_1/set_mode pad_selection=true
```

SET_INSTRUCTION_LIST_LENGTH (SETILL)

Purpose Specifies the number of words to be used in the optional instruction list.

Format

SET_INSTRUCTION_LIST_LENGTH
LENGTH=2...31
STATUS=status variable

Parameters

LENGTH (L)

Specifies the number of words to be used in the optional instruction list, commencing with the first word. Each word must have eight opcodes. Duplicate codes can be used to fill out any word. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_INSTRUCTION_LIST_LENGTH command specifies that four words are to be used in the optional instruction list. For this example, 32 opcodes are required in the optional instruction list.

```
RCT1_1/set_instruction_list_length l=4
```

SET_SKIP_PASS_COUNT (SETSPC)

Purpose Sets the starting pass count value for the test.

Format

SET_SKIP_PASS_COUNT
VALUE=integer
STATUS=status variable

Parameters

VALUE (V)

Specifies the pass count for beginning of test execution. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_SKIP_PASS_COUNT command causes the test to skip to pass count 3 prior to execution.

```
RCT1_1/set_skip_pass_count value=3
```

SET_PIT_VALUE (SETPV)

Purpose Sets the Process Interval Timer (PIT) value for the test.

Format

SET_PIT_VALUE
VALUE=integer
STATUS=status variable

Parameters

VALUE (V)

Specifies the PIT value to be loaded prior to the beginning of test execution. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_PIT_VALUE command specifies that a PIT value of 1000_{16} is to be loaded prior to the beginning of test execution.

```
RCT1_1/set_pit_value value=1000(16)
```

SET_PAD_COUNT (SETPC)

Purpose Specifies the number of no-op instructions to be used for padding. Pad selection adds a no-op instruction before each instruction to be tested.

NOTE

Pad selection is enabled with the SET_MODE command.

Format

SET_PAD_COUNT
VALUE=1...10
STATUS=status variable

Parameters

VALUE (V)

Specifies the pad count value. By default, a pad count value of 10 is assumed. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_PAD_COUNT command specifies that the pad count value is 3. That is, the maximum number of no-ops between instructions is three.

```
RCT1_1/setpc v=3
```

SET_REPEAT_COUNT (SETRC)

Purpose Sets the number of test passes to be executed.

NOTE

A repeat count value of 0 executes the test until it is terminated by expiration of a time limit or by a DROP_TEST command.

Format

SET_REPEAT_COUNT
VALUE=integer
STATUS=status variable

Parameters

VALUE (V)

Specifies the repeat count to be loaded prior to the beginning of test execution. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_REPEAT_COUNT command specifies that RCT1_1 is to run through five test passes.

```
RCT1_1/setrc v=5
```

SET_INTERMITTENT_COUNT (SETIC)

Purpose Sets the test's intermittent count value. The value sets the number of times a test pass is repeated to determine whether the failure is intermittent.

Format

SET_INTERMITTENT_COUNT
VALUE=0...255
STATUS=status variable

Parameters

VALUE (V)

Specifies the intermittent count to be loaded prior to the beginning of test execution. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_INTERMITTENT_COUNT command sets the RCT1_1 intermittent count value to 5. This means that a test pass is repeated five times whenever a failure occurs in order to determine whether a failure is intermittent.

```
RCT1_1/setic v=5
```

Random Command Test 2 (RCT2) 10

Random Command Test 2 (RCT2) 10

This chapter describes commands used with Random Command Test 2 (RCT2).

Random Command Test 2 Description

Random Command Test 2 tests BDP instructions, using fixed instruction sequences and random descriptors and operands.

SET_DISPLAY_REQUEST (SETDR)

Purpose Sets a default REQUEST_OPTION for the DISPLAY_MESSAGES command.

Format

SET_DISPLAY_REQUEST
REQUEST_OPTION=keyword
STATUS=status variable

Parameters *REQUEST_OPTION (RO)*

Specifies a keyword for the desired display and sets it as the default REQUEST_OPTION for the DISPLAY_MESSAGES command.

RUN (R)

SECTION_LIST (SL)

RESULTS (RE)

OPERANDS (O)

MACHINE_LIST (ML)

PARAMETERS (P)

NIL (N)

If REQUEST_OPTION is omitted, NIL is used.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_DISPLAY_REQUEST command sets the OPERANDS display as the default REQUEST_OPTION for the DISPLAY_MESSAGES command. Entry of the DISPLAY_MESSAGES command without any parameters results in the automatic display of the operands.

```
RCT2_1/set_display_request ro=operands
```

DISPLAY_PARAMETERS (DISP)

Purpose Displays the test's parameters from its parameter word block.

Format

DISPLAY_PARAMETERS
OUTPUT=file reference
STATUS=status variable

Parameters

OUTPUT (O)

Specifies the file to which information is displayed. If OUTPUT is omitted, $OUTPUT is used.

STATUS (S)

Specifies the termination condition of the command.

Examples The following DISPLAY_PARAMETERS command is used to display the RCT2_1 parameters from its parameter word block. By default, the following parameter display appears. If any parameter values are changed, the display is also changed.

```
RCT2_1/display_parameters
REQUEST OPTION     : NIL
REPEAT COUNT       : 0
PASS COUNT         : 0
PIT VALUE          : 7FFFFFFF(16)
SECTION SELECTED   : 0...15
```

SET_SKIP_PASS_COUNT (SETSPC)

Purpose Sets the starting pass count value for the test.

Format

SET_SKIP_PASS_COUNT
VALUE=integer
STATUS=status variable

Parameters

VALUE (V)

Specifies the pass count for the beginning of test execution. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_SKIP_PASS_COUNT command causes the test to skip to pass count 9 prior to execution.

```
RCT2_1/setspc value=9
```

SET_PIT_VALUE (SETPV)

Purpose Sets the Process Interval Timer (PIT) value for the test.

Format

SET_PIT_VALUE
VALUE=integer
STATUS=status variable

Parameters

VALUE (V)

Specifies the PIT value to be loaded prior to the beginning of test execution. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_PIT_VALUE command specifies that a PIT value of 4096 is to be loaded prior to the beginning of test execution.

```
RCT2_1/set_pit_value v=4096
```

SET_REPEAT_COUNT (SETRC)

Purpose Sets the number of test passes to be executed.

NOTE

A repeat count value of 0 executes the test until it is terminated by expiration of a time limit or by a DROP_TEST command.

Format

SET_REPEAT_COUNT
VALUE=integer
STATUS=status variable

Parameters

VALUE (V)

Specifies the repeat count to be loaded prior to the beginning of test execution. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples In the following example, the SET_REPEAT_COUNT command specifies that RCT2_3 is to run through four test passes.

```
RCT2_3/set_repeat_count value=4
```

SET_SECTION_SELECTIONS (SETSS)

Purpose Selects or deselects test sections to be executed.

Format

SET_SECTION_SELECTIONS
ADD=list of 0...16
DELETE=list of 0...16
STATUS=status variable

Parameters

ADD (A)

Specifies one or more sections to be selected for execution by the test. Sections can be specified as a list or range of sections.

DELETE (D)

Specifies one or more sections to be deselected for execution by the test. Sections can be specified as a list or range of sections.

STATUS (S)

Specifies the termination condition of the command.

Examples Below, the SET_SECTION_SELECTIONS command selects sections 0 through 5, 7, and 13 through 16 to be executed by RCT2_1.

```
RCT2_1/set_section_selections add=(0...5,7,13...16)
```

In the following SET_SECTION_SELECTIONS command, sections 3, 7, and 14 through 16 are deselected from the current selection list. These sections are not executed by the test.

```
RCT2_3/setss d=(3,7,14...16)
```

Fixed Command Test 3 (FCT3) 11

Fixed Command Test 3 (FCT3) 11

This chapter describes commands used with Fixed Command Test 3 (FCT3).

Fixed Command Test 3 Description

Fixed Command Test 3 used on line is a subset of the offline version. Only the sections dealing with the testing of general and BDP instructions are incorporated. FCT3 uses fixed instruction sequences and fixed operands to perform instruction testing.

DISPLAY_PARAMETERS (DISP)

Purpose Displays the test's parameters from its parameter word block.

Format

DISPLAY_PARAMETERS
OUTPUT=file reference
STATUS=status variable

Parameters

OUTPUT (O)

Specifies the file to which information is displayed. If OUTPUT is omitted, $OUTPUT is used.

STATUS (S)

Specifies the termination condition of the command.

Examples In the following example, the DISPLAY_PARAMETERS command sends the FCT3_1 parameter display to the output file, DISPLAY_OUTPUT.

```
FCT3_1/disp o=$local.display_output
```

SET_REPEAT_COUNT (SETRC)

Purpose Sets the number of test passes to be executed.

NOTE

A repeat count value of 0 executes the test until it is terminated by expiration of a time limit or by a DROP_TEST command.

Format

SET_REPEAT_COUNT
VALUE=integer
STATUS=status variable

Parameters

VALUE (V)

Specifies the repeat count to be loaded prior to the beginning of test execution. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_REPEAT_COUNT command specifies that FCT3_1 is to run through six test passes.

```
FCT3_1/setrc v=6
```

SET_SECTION_SELECTIONS (SETSS)

Purpose Selects or deselects test sections to be executed.

Format

SET_SECTION_SELECTIONS
ADD=list of 0...176
DELETE=list of 0...176
STATUS=status variable

Parameters

ADD (A)

Specifies one or more sections to be selected for execution by the test. Sections can be specified as a list or range of sections.

DELETE (D)

Specifies one or more sections to be deselected for execution by the test. Sections can be specified as a list or range of sections.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_SECTION_SELECTIONS command selects sections 0 through 36, 45, and 65 through 171 for execution by FCT3_1.

```
FCT3_1/set_section_selections a=(0...36,45,65...171)
```

The following SET_SECTION_SELECTIONS command deselects section 41 from the current FCT3_1 section execution list.

```
FCT3_1/setss d=41
```

MIDGS Tests 12

MIGDS Tests 12

This chapter describes commands used with the MIGDS tests.

MIGDS Tests Description

The MIGDS tests are used for testing end case interrupts. The tests currently available on line cover full- and half-word integer instructions, floating-point instructions, BDP numeric and BDP immediate instructions, and vector instructions. These tests use the same basic flow; they differ only according to the instructions tested. The purpose of the MIGDS tests in the online environment is to place heavy emphasis on end case and stress testing.

The online MIGDS tests are a subset of the same tests that run off line. Tests dealing with system instructions are not incorporated. The MIGDS tests incorporated on line are the following:

Single-Precision Floating-Point Test	SNGL
Double-Precision Floating-Point Test	DOBL
Floating-Point Branch Test	BRCH
Full-Word Integer Test	FINT
Half-Word Integer Test	HINT
Full-Word Immediate Test	FIMM
Half-Word Immediate Test	HIMM
BDP Immediate Test	BIMM
BDP Numeric Test	NUMR
BDP Byte End-Case Test	BYTE
BDP Edit End-Case Test	EDIT
Vector Integer Test	VINT
Vector Compare Test	VCMP
Vector Gather/Scatter Test	VGTH
Vector Gather/Scatter Indexed Test	VGSI

NOTE

The vector tests (VINT, VCMP, and VGTH) execute only on the CYBER 990 or CYBER 2000. The vector test VGSI executes only on the CYBER 2000.

The tests begin by establishing the test environment. This is done by default or by a request for input in the form of yes-or-no answers to questions dealing with the environment desired.

Each test includes a set of canned operands for testing all end cases of an instruction. In addition, a set of random operands is generated based on the test pass count. Each test pass runs through both sets of operands.

When all operations for a test are selected, each instruction included is tested by running each operand against all other operands, including itself. The online version runs with traps enabled and alternates the user mask between clear and set.

Each test has the ability to focus on one operation. In this mode, all operands are run against each other for the operation selected. An option is available for running a selected operand against itself or against another operand for a selected operation.

Error results are saved in a result table and are displayed on request by use of the DISPLAY_MESSAGES command. The table is indexed, and operands can be selected based on the index number in the table.

Both the command interface for setting parameters and the manual sequencing of test execution for MIGDS tests are different from all other tests. The MIGDS tests interact with you by asking questions about parameter settings and test operation selection. You can answer these questions with either YES or NO responses or with data input, using the ENTER_KEYBOARD_BUFFER command.

NOTE

When running MIGDS tests, monitor the display periodically, using the DISPLAY_STATUS or DISPLAY_MESSAGES command to determine whether the test is requesting input.

Examples:

The MIGDS test SNGL executes with defaults.

```
DVS/execute_test sngl
```

The MIGDS test DOBL runs without defaults. The test initiates a question-and-answer session to obtain the information required for DOBL to run.

```
DVS/exet dobl ps=true
```

ENTER_KEYBOARD_BUFFER (ENTKB)

Purpose Enters a string of data into the test's keyboard buffer in response to a test question.

Format

ENTER_KEYBOARD_BUFFER
VALUE=string
STATUS=status variable

Parameters

VALUE (V)

Specifies a string value to be transferred to the test's keyboard buffer. The string must be enclosed in quotes. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples In the following example, the ENTER_KEYBOARD_BUFFER command transfers the string value '10' into the SNGL_1 keyboard buffer.

```
SNG_1/enter_keyboard_buffer v='10'
```

The following ENTER_KEYBOARD_BUFFER command transfers the string value 'repeat' into the DOBL_1 keyboard buffer.

```
DOBL_1/entkb v='repeat'
```

YES (Y)

Purpose Enters a YES command response in the test's keyboard buffer.

Format **YES**
STATUS=status variable

Parameters *STATUS (S)*

Specifies the termination condition of the command.

Examples In the following example, the YES command is used instead of the ENTER_KEYBOARD_BUFFER command to respond to a test question. It places a YES command in the HIMM_1 keyboard buffer.

```
HIMM_1/yes
```

NO (N)

Purpose Enters a NO command response in the test's keyboard buffer.

Format **NO**
STATUS=status variable

Parameters *STATUS (S)*

Specifies the termination condition of the command.

Examples Following, the NO command is used instead of the ENTER_KEYBOARD_BUFFER command to respond to a test question. It places a NO command in the SNGL_1 keyboard buffer.

```
SNGL_1/no
```

MIGDS Test Examples

The following examples are typical question-and-answer sessions used by MIGDS tests. To initiate the first question, enter the DISPLAY_MESSAGES command.

This example illustrates the complete execution of all of the SNGL tests and operations running with multiple passes.

```
DVS/exet sngl ps=true
TEST ID: SNGL_1, IN THE PROCESS OF LOADING.
SNGL_1, WAITING FOR PARAMETERS:TYPE DISPLAY_MESSAGES TO SEE QUESTIONS.
  ANSWER TEST QUESTIONS WITH ENTER_KEYBOARD_BUFFER COMMAND (EXCEPT Y/N).
Type MENU for SNGL_1 Commands; or HELP for DVS Commands.
SNGL_1/dism
IMMEDIATE STOP ON ERROR DISABLED (Y/N)
SNGL_1/N
RUN WITH SMALL PIT (Y/N)
SNGL_1/Y
ENTER PIT VALUE
SNGL_1/entkb v='4096'
ENTER STARTING PASS COUNT
SNGL_1/entkb '3'
RUN ALL TESTS (Y/N)
SNGL_1/Y
RUN ALL OPERATIONS (OP CODES) (Y/N)
SNGL_1/Y
RUN RANDOM OPERANDS ONLY (Y/N)
SNGL_1/Y
ENTER NUMBER OF PASSES TO RUN
SNGL_1/entkb v='15'
PC000003 TE=1 UM=FE00  ADDF 045,045 PV=00004096
NUMBER OF PASSES SELECTED TO RUN=       15
SNGL_1/dism
ANOTHER TEXT (Y/N)
SNGL_1/no
ON-LINE TEST COMPLETED
SNGL_1/drop_test
TERMINATING_TEST: SNGL_1, TOTAL ERRORS=0, PASSES COMPLETED=1
```

This example illustrates the flexibility available for configuration of an MIGDS test. In this case, only certain tests are executed. These tests are selected by the op codes and operands chosen.

```
DVS/execute_test dobl ps=yes
TEST ID: DOBL_1, IN THE PROCESS of LOADING.
DOBL_1, WAITING FOR PARAMETERS:  TYPE, DISPLAY_MESSAGES TO SEE QUESTIONS.
  ANSWER TEST QUESTIONS WITH ENTER_KEYBOARD_BUFFER COMMAND (EXCEPT Y/N)
Type MENU for DOBL_1 Commands; or HELP for DVS Commands.
DOBL_1/display_messages
IMMEDIATE STOP ON ERROR DISABLED (Y/N)
DOBL_1/ yes
LOG ERRORS DISABLED (Y/N)
DOBL_1/Y
RUN WITH SMALL PIT (Y/N)
DOBL_1/n
ENTER STARTING PASS COUNT
DOBL_1/entkb v='9'
RUN ALL TESTS (Y/N)
DOBL_1/no
MNEMONIC (A)DDD (S)UBD (M)ULD (D)IVD
OP CODE    34     35     36     37
ENTER OPERATION TO BE TESTED
DOBL_1/entkb 's'
USER MASK SET (Y/N)
DOBL_1/n
RUN TEST WITH ALL OPERANDS - FULL TEST (Y/N)
DOBL_1/no
INDEXING OPERAND TABLE (Y/N)
DOBL_1/n
ENTER OPERANDS (XK,XK+1,XJ,XJ+1(HEX))
DOBL_1/entkb '4f,5abc,6789abc,aa'
    NO MIS-MATCHES    SUBD   UM CLEAR  TRAPS  ENABLED
                         UPPER            LOWER              UCR P
     IN      XK,K+1 4F00000000000000   5ABC000000000000
             XJ,J+1 6789ABC000000000   AA00000000000000
    OUT ACT  XJ,J+1 6789ABC000000000   AA00000000000000      0020
             XK,K+1 D000000000000000   D000000000000000
    OUT EXP  XK,K+1 D000000000000000   D000000000000000      0020
    OUT DIF  XK,K+1 0000000000000000   0000000000000000
35    SUBD  UM=0 TE=1  TRAP EXP=0 ACT=0

ENTER (R)EPEAT, (N)EW TEST, OR (T)ERMINATE
DOBL_1/entkb v='T'
ON-LINE TEST COMPLETED
DOBL_1/drot
TERMINATING_TEST: DOBL_1, TOTAL ERRORS=0, PASSES COMPLETED=0
```

This example illustrates the indexing of the operand table to obtain the operands for the chosen instruction.

```
DVS/exet dobl ps=true
TEST ID: DOBL_1 IN THE PROCESS OF LOADING.
DOBL_1, WAITING FOR PARAMETERS: TYPE, DISPLAY_MESSAGES TO SEE QUESTIONS.
  ANSWER TEST QUESTIONS WITH ENTER KEYBOARD_BUFFER_COMMAND (EXCEPT Y/N).
Type MENU for DOBL_1 Commands; or HELP for DVS Commands.
DOBL_1/dism
IMMEDIATE STOP ON ERROR DISABLED (Y/N)
DOBL_1/yes
LOG ERRORS DISABLED (Y/N)
DOBL_1/y
RUN WITH SMALL PIT (Y/N)
DOBL_1/n
ENTER STARTING PASS COUNT
DOBL_1/entkb v='9'
RUN ALL TESTS (Y/N)
DOBL_1/n
MNEMONIC (A)DDD (S)UBD (M)ULD (D)IVD
OP CODE    34     35     36     37
ENTER OPERATION TO BE TESTED
DOBL_1/entkb '37'
USER MASK SET (Y/N)
DOBL_1/no
RUN TEST WITH ALL OPERANDS (Y/N)
DOBL_1/n
INDEXING OPERAND TABLE (Y/N)
DOBL_1/y
DISPLAY TABLE (Y/N)
DOBL_1/YES
                        OPERAND TABLE
  1 4001800000000000     8 CFFF800000000000    15 0FFF800000000000
    0000000000000000       0000000000000000       0000000000000000
  2 4002800000000000     9 B001040000000000    16 0FFF000000000000
    0000000000000000       0000000000000000       0000000000000000
  3 4FFF800000000000    10 C000FFFFFFFFFFFF    17 8FFF800000000000
    0000000000000000       FFFFFFFFFFFFFFFF       0000000000000000
  4 3001040000000000    11 0000000000000000    18 8FF0000000000000
    0000000000000000       0000000000000000       0000000000000000
  5 4000FFFFFFFFFFFF    12 0000123456789ABC    19 2FFF800000000000
    FFFFFFFFFFFFFFFF       0000000000000000       0000000000000000
  6 C001800000000000    13 8000000000000000    20 1000000000000000
    0000000000000000       0000000000000000       0000000000000000
  7 C002800000000000    14 8000CBA987654321    21 AFFF800000000000
    0000000000000000       0000000000000000       0000000000000000
```

```
ENTER (N)EXT, (E)ND DISPLAY OR DECIMAL INDEX
DOBL_1/entkb '27'
                          OPERAND TABLE
 27 C001000000000000    34 EFFFCBA987654321    41 4000000000000000
    0000000000000000       0000000000000000       0000000000000000
 28 CFFF000000000000    35 7000000000000000    42 3000FFFFFFFFFFFF
    0000000000000000       0000000000000000       3000FFFFFFFFFFFF
 29 C080000000000000    36 7FFF123456789ABC    43 4FFF000000000000
    0000000000000000       0000000000000000       4FFF000000000000
 30 B001000000000000    37 F000000000000000    44 4001FFFFFFFFFFFF
    0000000000000000       0000000000000000       4001FFFFFFFFFFFF
 31 5000000000000000    38 FFFFCBA987654321    45 4000000000000000
    0000000000000000       0000000000000000       4000800000000000
 32 6FFF123456789ABC    39 3000800000000000    46 4000000000000000
    0000000000000000       0000000000000000       4000000000000001
 33 D000000000000000    40 4FFFC00000000000    47 C000000000000000
    0000000000000000       0000000000000000       0000000000000001

ENTER (N)EXT, (E)ND DISPLAY OR DECIMAL INDEX

DOBL_1/entkb 'e'
ENTER INDICES (XJ,XK)
DOBL_1/entkb '12,14'
    NO MIS-MATCHES      DIVD UM CLEAR     TRAPS  ENABLED
                            UPPER             LOWER          UCR P
    IN       XK,K+1   0000123456789ABC   0000000000000000
             XJ,J+1   8000CBA987654321   0000000000000000
    OUT ACT  XJ,J+1   8000CBA987654321   0000000000000000  0100
             XK,K+1   0000123456789ABC   0000000000000000
    OUT EXP  XK,K+1   0000123456789ABC   0000000000000000  0100
    OUT DIF  XK,K+1   0000000000000000   0000000000000000
37    DIVD  UM=0   TE=1  TRAP EXP=0  ACT=0

ENTER (R)EPEAT, (N)EW TEST, OR (T)ERMINATE
DOBL_1/entkb v='T'
ON-LINE TEST COMPLETED
DOBL_1/drot
TERMINATING_TEST: DOBL_1, TOTAL ERRORS=0, PASSES COMPLETED=0
```

Random Fast/Slow Test (RFST) 13

Random Fast/Slow Test (RFST) 13

This chapter describes commands used with the Random Fast/Slow Test (RFST).

Random Fast/Slow Test Description

RFST generates a random set of instructions and operands during each pass. The random set of instructions is executed; the results are saved. The instruction stream is executed again; results are saved (zero pads). Results are compared, and any difference is flagged as an error. Next, one pad instruction is inserted between all instructions. The set of instructions with pads is executed; the results are saved. The results are compared with the original nonpadded results, and any difference is flagged as an error.

The pad count is incremented by one, and the set of instructions with pads is executed again, with results saved. Results are compared with the original nonpadded results. This sequence is repeated, with the pad count incremented to 16. This constitutes a test pass. Each succeeding pass generates a new set of random instructions and operands.

NOTE

Running with the first pad count set to 0 is essentially repeating the fast loop. This is done to verify that no errors occur when re-executing the instruction stream.

The basic steps of each pass are the following:

1. Display the running message.
2. Process any keyboard input.
3. Generate random instructions, A/X register values, and random data memory buffers.
4. Execute the set of instructions using the A/X registers and memory buffers. Save A/X register values and memory data.
5. Insert zero pad instructons between all instructions in the instruction set being used. Re-initialize A/X memory with initial random values. Execute the padded instruction set, and save A/X memory values.
6. Compare the first execution results with the padded execution results. Any differences are considered an error and are reported.
7. If a miscompare occurs, and if TRIM mode is selected, RFST reduces the failing set of instructions to the minimum number needed to cause the failure. This is done by removing each instruction, one at a time, and re-executing to see whether the error still occurs.
8. Increase the pad count by one, repeat padded execution, and compare (steps 5 and 6) for a pad count up to 16.

SET_DISPLAY_REQUEST (SETDR)

Purpose Sets a default REQUEST_OPTION for the DISPLAY_MESSAGES command.

Format

SET_DISPLAY_REQUEST
REQUEST_OPTION=keyword
STATUS=status variable

Parameters *REQUEST_OPTION (RO)*

Specifies a keyword for the desired display and sets it as the default REQUEST_OPTION for the DISPLAY_MESSAGES command.

RUN (R)

A_REGISTER_RESULTS (ARR)

X_REGISTER_RESULTS (XRR)

MEMORY_RESULTS (MR)

INITIAL_A_AND_X_REGISTER (IAXR)

ADDRESS_BUFFER (AB)

PARAMETERS (P)

NIL (N)

If REQUEST_OPTION is omitted, NIL is used.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_DISPLAY_REQUEST command sets the A_REGISTER_RESULTS display as the default REQUEST_OPTION for the DISPLAY_MESSAGES command. Entering the DISPLAY_MESSAGES command without any parameters results in the automatic display of all A registers' results.

```
RFST_2/setdr ro=arr
```

The following SET_DISPLAY_REQUEST command causes the running display to appear whenever the DISPLAY_MESSAGES command is entered without a REQUEST_OPTION.

```
RFST_1/set_display_request ro=run
```

DISPLAY_PARAMETERS (DISP)

Purpose Displays the test's parameters from its parameter word block.

Format

DISPLAY_PARAMETERS
OUTPUT=file reference
STATUS=status variable

Parameters

OUTPUT (O)

Specifies the file to which information is displayed. If OUTPUT is omitted, $OUTPUT is used

STATUS (S)

Specifies the termination condition of the command.

Examples The following DISPLAY_PARAMETERS command is used to display the RFST_1 parameters from its parameter word block. By default, the following parameter display appears. If any parameter values are changed, the display is also changed.

```
RFST_1/display_parameters
REQUEST OPTION              : NIL
REPEAT COUNT                : 0
PASS COUNT                  : 1
PIT VALUE                   : 7FFFFFFF(16)
USER MASK SELECTION         : SET (FF7F)
TRIM SELECTION              : NO TRIM
LIST LENGTH                 : 500
PAD COUNT                   : 0
INSTRUCTION LIST SELECTION  : NORMAL
```

ADD_INSTRUCTONS (ADDI)

Purpose Adds a user-specified list of instruction opcodes to the test's optional instruction list.

NOTE

The SET_MODE command enables or disables use of this list.

Format

ADD_INSTRUCTIONS
OP_CODE=list of opcodes
VECTOR_OP_CODES=boolean
BDP_OP_CODES=boolean
STATUS=status variable

Parameters

OP_CODE (OC)

Specifies the opcode or list of opcodes to be added to the test's optional instruction list. Decimal values are assumed unless a radix is specified.

VECTOR_OP_CODES (VOC)

Specifies whether all vector instructions are to be added to the optional instrucion list.

TRUE

All vector instructions are added to the optional instruction list.

FALSE

Vector instructions are not added to the optional instruction list.

If VECTOR_OP_CODES is omitted, FALSE is used.

BDP_OP_CODES (BOC)

Specifies whether all BDP instructions are to be added to the optional instruction list.

TRUE

All BDP instructions are added to the optional instruction list.

FALSE

BDP instructions are not added to the optional instruction list.

If BDP_OP_CODES is omitted, FALSE is used.

STATUS (S)

Specifies the termination condition of the command.

Examples The following ADD_INSTRUCTIONS command adds all BDP instruction opcodes to the RFST_2 optional instruction list.

```
RFST_2/add_instructions bdp_op_codes=true
```

The following ADD_INSTRUCTIONS command adds the instructions having opcodes 01, 05, 07, and 10 to the RFST_1 optional instruction list.

```
RFST_1/addi oc=(01(16),05(16),07(16),10(16))
```

DELETE_INSTRUCTIONS (DELI)

Purpose Deletes a user-specified list of instruction opcodes from the test's optional instruction list.

NOTE

The SET_MODE command enables or disables use of this list.

Format

DELETE_INSTRUCTIONS
OP_CODE=list of opcodes
STATUS=status variable

Parameters

OP_CODE (OC)

Specifies the opcode or list of opcodes to be deleted from the test's optional instruction list. Decimal values are assumed unless a radix is specified. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following DELETE_INSTRUCTIONS command deletes the instructions having opcodes 01 and 05 from the RFST_2 optional instruction list.

```
RFST_2/deli op_code=(01(16),05(16))
```

In the following example, the DELETE_INSTRUCTIONS command deletes the instruction with opcode 07 from the RFST_1 optional instruction list.

```
RFST_1/delete_instrucions oc=07(16)
```

SET_MODE (SETM)

Purpose Sets the execution modes for the test.

Format

SET_MODE
UCR_MASK_SELECTION=boolean
INSTRUCTION_LIST_SELECTION=keyword
TRIM_SELECTION=boolean
STATUS=status variable

Parameters *UCR_MASK_SELECTION (UMS)*

Specifies the user condition register mask selection to be applied. The mask is changed only at the beginning of a test, at the end of a test pass, or at a restart. This avoids miscompare errors due to trap/no trap differences.

TRUE

Sets the user mask to $FF7F_{16}$. All bits are set except debug.

FALSE

Clears the user mask to $FF00_{16}$. Arithmetic, floating-point, and BDP user mask bits are cleared.

If UCR_MASK_SELECTION is omitted, TRUE is used.

NOTE

When running with user mask clear, BDP invalid data results are undefined and, on some machines, are not repeatable. On these machines, if the user mask is cleared, all BDP opcodes should be disabled with the following commands:

ADD_INSTRUCTIONS bdp_op_codes=true

SET_MODE instruction_list_selection=deselect

INSTRUCTION_LIST_SELECTION (ILS)

Specifies instruction selection or deselection from the optional instruction list. The keyword parameter selections determine the instruction list selection mode.

SELECT (S)

Tests only the instructions in the optional instruction list.

DESELECT (D)

Tests all instructions but those in the optional instruction list.

NORMAL (N)

Tests all instructions.

If INSTRUCTION_LIST_SELECTION is omitted, NORMAL is used.

NOTE

The optional instruction list is formed with the ADD_INSTRUCTIONS and DELETE_INSTRUCTIONS commands. The list length is determined by the SET_LIST_LENGTH command.

TRIM_SELECTION (TS)

Specifies whether instruction isolation is to be performed. Isolation involves trimming execution to a single failing instruction when an error is detected.

TRUE

Instruction isolation is performed.

FALSE

Instruction isolation is not performed. The failing random instruction loop is unaltered when an error is detected.

If TRIM_SELECTION is omitted, FALSE is used.

STATUS (S)

Specifies the termination condition of the command.

Examples In the following example, the SET_MODE command selects only the instructions in the RFST_1 optional instruction list.

```
RFST_1/setm ils=select
```

The following SET_MODE command specifies that instruction isolation for RFST_1 is to be performed.

```
RFST_1/setm trim_selection=true
```

SET_INSTRUCTION_LIST_LENGTH (SETILL)

Purpose Specifies the number of random instructions to be placed in the optional instruction list.

Format

SET_INSTRUCTION_LIST_LENGTH
LENGTH=1...500
STATUS=status variable

Parameters **LENGTH (L)**

Specifies the number of random instructions to be placed in the instruction list. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_INSTRUCTION_LIST_LENGTH command causes the test to generate an instruction list containing 40 random instructions.

```
RFST_1/set_instruction_list_length l=40
```

SET_SKIP_PASS_COUNT (SETSPC)

Purpose Sets the starting pass count value for the test.

Format

SET_SKIP_PASS_COUNT
VALUE=integer
STATUS=status variable

Parameters **VALUE (V)**

Specifies the pass count for beginning of test execution. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_SKIP_PASS_COUNT command causes the test to skip to pass count 3 prior to execution.

```
RFST_1/setspc value=3
```

SET_PIT_VALUE (SETPV)

Purpose Sets the Process Interval Timer (PIT) value for the test.

Format

SET_PIT_VALUE
VALUE=integer
STATUS=status variable

Parameters

VALUE (V)

Specifies the PIT value to be loaded prior to the beginning of test execution. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_PIT_VALUE command specifies that a PIT value of 1000_{16} is to be loaded prior to the beginning of test execution.

```
RFST_1/set_pit_value v=1000(16)
```

SET_PAD_COUNT (SETPC)

Purpose Specifies the number of no-op instructions to be used for padding. Pad selection adds a no-op instruction before each instruction to be tested.

Format

SET_PAD_COUNT
VALUE=0...16
STATUS=status variable

Parameters

VALUE (V)

Specifies the pad count value. By default, a pad count value of 0 is used. This inserts no-op instructions incrementally from 0 to 16 no-ops per instruction. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_PAD_COUNT command specifies that the pad count value is 4. That is, the test inserts no-op instructions incrementally from zero to four no-ops per instruction.

```
RFST_1/set_pad_count value=4
```

SET_REPEAT_COUNT (SETRC)

Purpose Set the number of test passes to be executed.

NOTE

A repeat count value of 0 executes the test until it is terminated by expiration of a time limit or by a DROP_TEST command.

Format

SET_REPEAT_COUNT
VALUE=integer
STATUS=status variable

Parameters

VALUE (V)

Specifies the repeat count to be loaded prior to the beginning of test execution. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples In the following example, the SET_REPEAT_COUNT command sets RFST_1 to run through 75 test passes.

```
RFST_1/set_repeat_count value=75
```

Test Messages

Messages are displayed using the DISPLAY_MESSAGES command. DISPLAY_MESSAGES selects a message display by using the REQUEST_OPTION parameter. The valid request options for RFST are the following:

PARAMETERS	P
RUN	R
A_REGISTER_RESULTS	ARR
X_REGISTER_RESULTS	XRR
MEMORY_RESULTS	MR
INITIAL_A_X_REGISTER	IAXR
ADDRESS_BUFFER	AB

Parameter Display

The Parameter display contains the test's parameter settings.

```
RANDOM FAST/SLOW TEST  CPU X    XXXX PAGES  DVS  MM/DD/YY
RFST   CC   PC=XXXXXXXX   EC=XXXX   TE=XXXX PADS=XX   PA=XXXXXXXX
(COMMENT FIELD)                    PV=XXXXXXXX

   PW0  - XXXX XXXX XXXX XXXX  CONTROL BITS
   PW1  - XXXX XXXX XXXX XXXX  REPEAT COUNTER
   PW6  - XXXX XXXX XXXX XXXX  DISPLAY SELECT
   PW7  - XXXX XXXX XXXX XXXX  STARTING PASS COUNT
   PW8  - XXXX XXXX XXXX XXXX  PIT VALUE
   PW9  - XXXX XXXX XXXX XXXX  SIT VALUE
   PW12 - XXXX XXXX XXXX XXXX  MONITOR CONTROL
   PW13 - XXXX XXXX XXXX XXXX  SCOPE LOOP CONTROL
   PW16 - XXXX XXXX XXXX XXXX  TRIM CONTROL
   PW17 - XXXX XXXX XXXX XXXX  STARTING PAD COUNT
   PW18 - XXXX XXXX XXXX XXXX  INSTR SELECT CONTROL
   PW19 - XXXX XXXX XXXX XXXX  PIT CONTROL
   PW20 - XXXX XXXX XXXX XXXX  SIT CONTROL
   PW21 - XXXX XXXX XXXX XXXX  USER MASK CONTROL
   PW22 - XXXX XXXX XXXX XXXX  INSTR LIST LENGTH
```

Where:

CPU X	Current CPU.
XXXX PAGES	Number of pages in use if option selected.
CC	Display identifier. RU Running. SP Stopped for parameters. SE Stopped on error. ST Stopped at end of test pass. RT Repeat test pass. RC Repeat inst/operands/pad count.
PC	Pass count (decimal).
PADS	Pad count (decimal).
EC	Error code (hexadecimal). 0001 X register error. 0002 A register error. 0004 Memory error. 0008 UCR error. 0010 Trim mode. 0030 Lost error during trim.
TE	Total number of errors (decimal).
PA	Address of parameter words (hex-byte address).
PV	PIT value.
Comment Field	Keyboard input error message. If running with a small PIT value, line 3 displays the comment field as shown by the RUN display.

Run Display

The Run display contains the test's current pass count and error count values.

```
RANDOM FAST/SLOW TEST  CPU X    XXXX PAGES  DVS  MM/DD/YY
RFST   CC   PC=XXXXXXXX   EC=XXXX   TE=XXXX PADS=XX   PA=XXXXXXXX
(   Comment Field          )    PV=XXXXXXXX
```

(Comment Field) PL=XXXX GZ=XXXX

Where:

PL Number of PIT losses.

GZ Number of PIT interrupts with PIT value greater than 0.

PV PIT value.

A_Register_Results Display

The A_Register_Results display contains all A registers and their respective values upon completion of a test pass.

```
              NO PADS          PADS          DIFFERENCE

REG-A0        X------X         X------X        X------X
REG-A1        X------X         X------X        X------X
   .             .                .              .
   .             .                .              .
   .             .                .              .
REG-AF        X------X         X------X        X------X
```

X_Register_Results Display

The X_Register_Results display contains all X registers and their respective values upon completion of a test pass.

```
              NO PADS          PADS          DIFFERENCE

REG-X0        X------X         X------X        X------X
REG-X1        X------X         X------X        X------X
   .             .                .              .
   .             .                .              .
   .             .                .              .
REG-XF        X------X         X------X        X------X
```

Memory_Results Display

The Memory_Results display contains the first 15 memory errors detected at the address displayed. The address displayed for each error is the byte offset relative to the starting buffer addresses. The NO PADS and PADS starting buffer addresses are PVA byte addresses.

```
STARTING BUFFER ADRS   NO PADS=PVA PADS=PVA

ADDRS          NO PADS           PADS          DIFFERENCE

XXXXX          X------X          X------X        X------X
XXXXX          X------X          X------X        X------X
  .              .                 .              .
  .              .                 .              .
  .              .                 .              .
XXXXX          X------X          X------X        X------X
```

Initial_A_X_Register Display

The Initial_A_X_Register display contains all A and X registers and their initial values prior to execution of a test pass.

```
            A0-AF   INITIAL A AND X REGISTERS   X0-XF

REG-00  XXXXXXXXXXXX                    XXXXXXXXXXXXXXXX
REG-01  XXXXXXXXXXXX                    XXXXXXXXXXXXXXXX
    .       .                               .
    .       .                               .
    .       .                               .
REG-0F  XXXXXXXXXXXX                    XXXXXXXXXXXXXXXX
```

Address_Buffer Display

The Address_Buffer display contains the tag names of all buffers used by the test.

```
RANDOM FAST/SLOW TEST  CPU X    XXXX PAGES  DVS  MM/DD/YY
RFST   CC   PC=XXXXXXXX   EC=XXXX   TE=XXXX PADS=XX   PA=XXXXXXXX
                                PV=XXXXXXXX
     TAG NAMES FOR DISPLAYING BUFFERS
        INSTRUCTION STREAM  -  ELIST
        X REGISTER RESULTS  -  RXREG
        A REGISTER RESULTS  -  RAREG
        MEMORY RESULTS (1)  -  RMEM1
        MEMORY RESULTS (2)  -  RMEM2
        INITIAL X REGISTERS -  IXREG
        INITIAL A REGISTERS -  IAREG
        INITIAL MEMORY      -  IMEM
        INITIAL BDP MEMORY  -  IBDP
        UCR RESULTS         -  UCRR
```

NOTE

Buffers are displayed by use of the DISPLAY_MEMORY_DATA command, with the DISPLAY_OPTIONS parameter specifying the desired tag name.

Error Messages

RFST displays one of the following messages when an RFST command is entered incorrectly. A valid command clears the error message.

INVALID ENTRY ERROR
INST LIST FULL ERROR
CANNOT TEST XX ERROR

Invalid Entry Error

The invalid entry error occurs when RFST does not recognize the command entered or when the command parameters are not acceptable.

Inst List Full Error

The inst list full error occurs when the instructions added by the ADD_INSTRUCTIONS command exceed the instruction buffer.

Cannot Test XX Error

The cannot test XX error occurs when RFST cannot execute the instruction opcode XX. The following opcodes are not tested.

HALT	(00)
SYNC	(01) Used at the end of the instruction list for scoping.
EXCHANGE	(02) Used but not tested.
INTERRUPT	(03)
RETURN	(04) Tested in conjunction with a CALL instruction.
POP	(06) Not tested during online execution.
CPYXS	(0F)
KEYPOINT	(B1)
EXECUTE ALGORITHM	(C0-C7)

When running on line, RFST runs as an unprivileged job. Because of this, LPAGE (17) is not tested and for PURGE (05), only K values of 03 through 07 are used. When running off line, RFST tests the LPAGE instruction and all K values for the PURGE instruction. For both online and offline testing, BRCR (9F) is tested only with K values of 2, 3, 5, 6, 7, A, B, D, E, and F.

Debug Test (DBUG) 14

Debug Test (DBUG) 14

This chapter describes commands used with the Debug Test (DBUG).

Debug Test Description

DBUG generates a random set of instructions and operands during each pass. A simulated debug buffer is created based on the debug code. It saves the debug index and P-counter for each instruction causing a debug trap. The random instruction stream is executed with debug disabled; all results are saved. Debug is then enabled, and the random instructions are executed again. The results are saved and compared to the run with debug disabled. The actual debug buffer (created by the trap handler) is compared to the simulated buffer to ensure that debug occurred correctly. Each individual debug code is used with a combination of all codes (READ, WRITE, FETCH, BRANCH, CALL). After executing with each code and then with the combination, one test pass is complete. Each succeeding pass generates a new set of random instructions and operands.

The basic steps of each pass are the following:

1. Display the running message.
2. Process any keyboard input.
3. Generate random instructions, A/X register values, and random data memory buffers. Generate the simulated debug buffer.
4. Execute the set of instructions using the A/X registers and memory buffers with debug disabled. Save the A/X register values and memory data.
5. Pick up the debug code and enable debug. Re-initialize A/X memory with initial random values. Execute the instruction stream with debug enabled. Save A/X memory values.
6. Compare first execution results with debug execution results. Any differences are considered an error and are reported.
7. Use the next debug code and repeat debug enabled execution and compare (steps 5 and 6). Repeat for all debug codes.

SET_DISPLAY_REQUEST (SETDR)

Purpose Sets a default REQUEST_OPTION for the DISPLAY_MESSAGES command.

Format

SET_DISPLAY_REQUEST
REQUEST_OPTION=keyword
STATUS=status variable

Parameters *REQUEST_OPTION (RO)*

Specifies a keyword for the desired display and sets it as the default REQUEST_OPTION for the DISPLAY_MESSAGES command.

RUN (R)

A_REGISTER_RESULTS (ARR)

X_REGISTER_RESULTS (XRR)

MEMORY_RESULTS (MR)

INITIAL_A_AND_X_REGISTER (IAXR)

ADDRESS_BUFFER (AB)

ERROR_RESULTS (ER)

PARAMETERS (P)

NIL (N)

If REQUEST_OPTION is omitted, NIL is used.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_DISPLAY_REQUEST command sets the A_REGISTER_RESULTS display as the default REQUEST_OPTION for the DISPLAY_MESSAGES command. Entering the DISPLAY_MESSAGES command without any parameters results in the automatic display of all A registers' results.

```
DBUG_1/set_display_request ro=arr
```

The following SET_DISPLAY_REQUEST command causes the running display to appear whenever the DISPLAY_MESSAGES command is entered without a REQUEST_OPTION.

```
DBUG_2/setdr ro=run
```

DISPLAY_PARAMETERS (DISP)

Purpose Displays the test's parameters from its parameter word block.

Format

DISPLAY_PARAMETERS
OUTPUT=file reference
STATUS=status variable

Parameters

OUTPUT (O)

Specifies the file to which information is displayed. If OUTPUT is omitted, $OUTPUT is used.

STATUS (S)

Specifies the termination condition of the command.

Examples The following DISPLAY_PARAMETERS command is used to display the DBUG_1 parameters from its parameter word block. By default, the following parameter display appears. If any parameter values are changed, the display is also changed.

```
DBUG_1/display_parameters
REQUEST OPTION              : NIL
REPEAT COUNT                : 0
PASS COUNT                  : 1
PIT VALUE                   : 7FFFFFFF(16)
USER MASK SELECTION         : SET (FF7F)
DEBUG CODE SELECTION        : NOT SET
LIST LENGTH                 : 500
INSTRUCTION LIST SELECTION  : DESELECT
DEBUG LIST POINTER          : PVA
```

ADD_INSTRUCTIONS (ADDI)

Purpose Adds a user-specified list of instruction opcodes to the test's optional instruction list.

NOTE

The SET_MODE command enables or disables use of this list.

Format

ADD_INSTRUCTIONS
OP_CODE=list of opcodes
VECTOR_OP_CODES=boolean
BDP_OP_CODES=boolean
STATUS=status variable

Parameters

OP_CODE (OC)

Specifies the opcode or list of opcodes to be added to the test's optional instruction list. Decimal values are assumed unless a radix is specified.

VECTOR_OP_CODES (VOC)

Specifies whether all vector instructions are to be added to the optional instruction list.

TRUE

All vector instructions are added to the optional instruction list.

FALSE

Vector instructions are not added to the optional instruction list.

If VECTOR_OP_CODES is omitted, FALSE is used.

BDP_OP_CODES (BOC)

Specifies whether all BDP instructions are to be added to the optional instruction list.

TRUE

All BDP instructions are added to the optional instruction list.

FALSE

BDP instructions are not added to the optional instructions list.

If BDP_OP_CODES is omitted, FALSE is used.

STATUS (S)

Specifies the termination condition of the command.

Examples The following ADD_INSTRUCTIONS command adds all BDP instruction opcodes to the DBUG_1 optional instruction list.

```
DBUG_1/add_instructions boc=true
```

The following ADD_INSTRUCTIONS command adds the instructions having opcodes 01, 05, 07, and 10 to the DBUG_1 optional instruction list.

```
DBUG_1/addi oc=(01(16),05(16),07(16),10(16))
```

DELETE_INSTRUCTIONS (DELI)

Purpose Deletes a user-specified list of instruction opcodes from the test's optional instruction list.

NOTE

The SET_MODE command enables or disables use of this list.

Format

DELETE_INSTRUCTIONS
OP_CODE=list of opcodes
STATUS=status variable

Parameters

OP_CODE (OC)

Specifies the opcode or list of opcodes to be deleted from the test's optional instruction list. Decimal values are assumed unless a radix is specified. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following DELETE_INSTRUCTIONS command deletes the instructions having opcodes 01 and 05 from the DBUG_2 optional instruction list.

```
DBUG_2/deli op_code=(01(16),05(16))
```

In the following example, the DELETE_INSTRUCTIONS command deletes the instruction with opcode 10 from the DBUG_1 optional instruction list.

```
DBUG_1/delete_instructions oc=10(16)
```

SET_MODE (SETM)

Purpose Sets the execution modes for the test.

Format

SET_MODE
UCR_MASK_SELECTION=boolean
INSTRUCTION_LIST_SELECTION=keyword
DEBUG_CODE_SELECTION=list of keywords
STATUS=status variable

Parameters *UCR_MASK_SELECTION (UMS)*

Specifies the user condition register mask selection to be applied. The mask is changed only at the beginning of a test, at the end of a test pass, or at a restart. This avoids miscompare errors due to trap/no trap differences.

TRUE

Sets the user mask to $FF7F_{16}$. All bits are set except debug.

FALSE

Clears the user mask to $FF00_{16}$. Arithmetic, floating-point, and BDP user mask bits are cleared.

If UCR_MASK_SELECTION is omitted, TRUE is used.

NOTE

When running with user mask clear, BDP invalid data results are undefined and, on some machines, are not repeatable. On these machines, if the user mask is cleared, all BDP opcodes should be disabled with the following commands:

ADD_INSTRUCTIONS bdp_op_codes=true

ADD_INSTRUCTIONS op_code=37_{16}

SET_MODE instruction_list_selection=deselect

On all machines, if user mask is cleared, single-precision opcodes are disabled.

Setting the UCR_MASK_SELECTION parameter to FALSE via the SET_MODE command results in the following single-precision floating point opcodes being added to the optional instruction list.

30_{16}	ADDF	$3c_{16}$	CMPF
31_{16}	SUBF	98_{16}	BRFEQ
32_{16}	MULF	99_{16}	BRFNQ
33_{16}	DIVF	$9a_{16}$	BRFGT
$3a_{16}$	CNIF	$9b_{16}$	BRFGE
$3b_{16}$	CNFI	$9e_{16}$	BR...

The instruction list selection is also deselected.

INSTRUCTION_LIST_SELECTION (ILS)

Specifies instruction selection or deselection from the optional instruction list. The keyword parameter selections determine the instruction list selection made.

SELECT (S)

Tests only the instructions in the optional instruction list.

DESELECT (D)

Tests all instructions but those in the optional instruction list.

NORMAL (N)

Tests all instructions.

If INSTRUCTION_LIST_SELECTION is omitted, NORMAL is used.

NOTE

The optional instruction list is formed with the ADD_INSTRUCTIONS and DELETE_INSTRUCTIONS commands. The list length is determined by the SET_LIST_LENGTH command.

DEBUG_CODE_SELECTION (DCS)

Specifies the debug code setting. The test runs only with the specified debug code. A list of keywords can be specified to run the test with more than one debug operation.

ALL (A)

READ (R)

WRITE (W)

FETCH (F)

BRANCH (B)

CALL (C)

If DEBUG_CODE_SELECTION is omitted, ALL is used.

STATUS (S)

Specifies the termination condition of the command.

Examples In the following example, the SET_MODE command selects only the instructions in the DBUG_1 optional instruction list.

```
DBUG_1/set_mode ils=select
```

SET_INSTRUCTION_LIST_LENGTH (SETILL)

Purpose Specifies the number of random instructions to be placed in the optional instruction list.

Format

```
SET_INSTRUCTION_LIST_LENGTH
   LENGTH=1...500
   STATUS=status variable
```

Parameters

LENGTH (L)

Specifies the number of random instructions to be placed in the instruction list. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_INSTRUCTION_LIST_LENGTH command specifies that the test generate an instruction list containing 100 random instructions.

```
DBUG_1/set_instruction_list_length length=100
```

SET_SKIP_PASS_COUNT (SETSPC)

Purpose Sets the starting pass count value for the test.

Format

```
SET_SKIP_PASS_COUNT
   VALUE=integer
   STATUS=status variable
```

Parameters

VALUE (V)

Specifies the pass count for beginning of test execution. This parameter is required.

STATUS (S)

Specifies the termination condition of the command

Examples The following SET_SKIP_PASS_COUNT command caues the test to skip to pass count 57 prior to execution.

```
DBUG_1/set_skip_pass_count value=57
```

SET_PIT_VALUE (SETPV)

Purpose Sets the Process Interval Timer (PIT) value for the test.

Format

SET_PIT_VALUE
VALUE=integer
STATUS=status variable

Parameters

VALUE (V)

Specifies the PIT value to be loaded prior to the beginning of test execution. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_PIT_VALUE command specifies that a PIT value of 4096 is to be loaded prior to the beginning of test execution.

```
DBUG_1/set_pit_value value=4096
```

SET_REPEAT_COUNT (SETRC)

Purpose Sets the number of test passes to be executed.

NOTE

A repeat count value of 0 executes the test until it is terminated by expiration of a time limit or by a DROP_TEST command.

Format

SET_REPEAT_COUNT
VALUE=integer
STATUS=status variable

Parameters

VALUE (V)

Specifies the repeat count to be loaded prior to the beginning of test execution. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_REPEAT_COUNT command sets DBUG_1 to run through 100 test passes.

```
DBUG_1/set_repeat_count v=100
```

Test Messages

Messages are displayed using the DISPLAY_MESSAGES command. DISPLAY_MESSAGES selects a message display by using the REQUEST_OPTION parameter. The valid request options for DBUG are the following:

PARAMETERS	P
RUN	R
A_REGISTER_RESULTS	ARR
X_REGISTER_RESULTS	XRR
MEMORY_RESULTS	MR
INITIAL_A_X_REGISTER	IAXR
ADDRESS_BUFFER	AB
ERROR_RESULTS	ER

Parameter Display

The Parameter display contains the test's parameter settings.

```
DEBUG TEST    CPU X        XXXX PAGES     DVS       MM/DD/YY
DBUG   CC   PC=XXXXXXXX   EC=XXXX   TE=XXXX DC=XX   PA=XXXXXXXX
(COMMENT FIELD)                     PV=XXXXXXXX

   PW0  - XXXX XXXX XXXX XXXX  CONTROL BITS
   PW1  - XXXX XXXX XXXX XXXX  REPEAT COUNTER
   PW6  - XXXX XXXX XXXX XXXX  DISPLAY SELECT
   PW7  - XXXX XXXX XXXX XXXX  STARTING PASS COUNT
   PW8  - XXXX XXXX XXXX XXXX  PIT VALUE
   PW9  - XXXX XXXX XXXX XXXX  SIT VALUE
   PW12 - XXXX XXXX XXXX XXXX  MONITOR CONTROL
   PW13 - XXXX XXXX XXXX XXXX  SCOPE LOOP CONTROL
   PW17 - XXXX XXXX XXXX XXXX  SELECTED DEBUG CODE
   PW18 - XXXX XXXX XXXX XXXX  INSTR SELECT CONTROL
   PW19 - XXXX XXXX XXXX XXXX  PIT CONTROL
   PW20 - XXXX XXXX XXXX XXXX  SIT CONTROL
   PW21 - XXXX XXXX XXXX XXXX  USER MASK CONTROL
   PW22 - XXXX XXXX XXXX XXXX  INSTR LIST LENGTH
```

Where:

XXXX PAGES	Number of pages in use if option selected.
X.X	Current version of DVS monitor.
CC	Display identifer. RU Running. SP Stopped for parameters. SE Stopped on error. ST Stopped at end of test pass. RT Repeat test pass. RC Repeat inst/operands/pad count.
PC	Pass count (decimal).
PADS	Pad count (decimal).
EC	Error code (hexadecimal). 0001 X Register Error. 0002 A Register Error. 0004 Memory Error. 0008 Debug Error.
TE	Total number of errors (decimal).
PA	Address of parameter words (hex-byte address).
PV	PIT value.
Comment Field	Keyboard input error message. If running with a small PIT value, line 3 displays the comment field as shown by the RUN display.

Run Display

The Run display contains the test's current pass count and error count values.

```
DEBUG TEST    CPU X        XXXX PAGES     DVS       MM/DD/YY
DBUG   CC   PC=XXXXXXXX   EC=XXXX   TE=XXXX DC=XX   PA=XXXXXXXX
(   Comment Field          )    PV=XXXXXXXX
```

(Comment Field) PL=XXXX GZ=XXXX

Where:

PL Number of PIT losses.

GZ Number of PIT interrupts with PIT value greater than zero.

PV PIT value.

A_Register_Results Display

The A_Register_Results display contains all A registers and their respective values upon completion of a test pass.

```
          NO DEBUG          DEBUG          DIFFERENCE

REG-A0  X------X         X------X         X------X
REG-A1  X------X         X------X         X------X
     .     .              .               .
     .     .              .               .
     .     .              .               .
REG-AF  X------X         X------X         X------X
```

X_Register_Results Display

The X_Register_Results display contains all X registers and their respective values upon completion of a test pass.

```
          NO DEBUG          DEBUG          DIFFERENCE

REG-X0  X------X         X------X         X------X
REG-X1  X------X         X------X         X------X
     .     .              .               .
     .     .              .               .
     .     .              .               .
REG-XF  X------X         X------X         X------X
```

Memory_Results Display

The Memory_Results display contains the first 15 memory errors detected at the address displayed. The address displayed for each error is the byte offset relative to the starting buffer addresses. The NO DEBUG and DEBUG starting buffer addresses are PVA byte addresses.

```
STARTING BUFFER ADRS  NO DEBUG=PVA    DEBUG=PVA

   ADDRS      NO DEBUG         DEBUG         DIFFERENCE

   XXXXX      X------X         X------X        X------X
   XXXXX      X------X         X------X        X------X
     .          .                .              .
     .          .                .              .
     .          .                .              .
   XXXXX      X------X         X------X        X------X
```

Initial_A_X_Register Display

The Initial_A_X_Register display contains all A and X registers and their initial values prior to execution of a test pass.

```
          A0-AF   INITIAL A AND X REGISTERS   X0-XF

REG-00  XXXXXXXXXXXX                  XXXXXXXXXXXXXXXX
REG-01  XXXXXXXXXXXX                  XXXXXXXXXXXXXXXX
    .       .                             .
    .       .                             .
    .       .                             .
REG-0F  XXXXXXXXXXXX                  XXXXXXXXXXXXXXXX
```

Address_Buffer Display

The Address_Buffer display contains the tag names of all buffers used by the test.

```
DEBUG TEST    CPU X        XXXX PAGES     DVS       MM/DD/YY
DBUG   CC   PC=XXXXXXXX   EC=XXXX   TE=XXXX DC=XX   PA=XXXXXXXX
                                     PV=XXXXXXXX

     TAG NAMES FOR DISPLAYING BUFFERS

        INSTRUCTION STREAM  -  ELIST
        X REGISTER RESULTS  -  RXREG
        A REGISTER RESULTS  -  RAREG
        MEMORY RESULTS (1)  -  RMEM1
        MEMORY RESULTS (2)  -  RMEM2
        INITIAL X REGISTERS -  IXREG
        INITIAL A REGISTERS -  IAREG
        INITIAL MEMORY      -  IMEM
        INITIAL BDP MEMORY  -  IBDP
        UCR RESULTS         -  UCRR
        DEBUG RESULTS-ACT   -  DEBG
        DEBUG RESULTS-SIM   -  DBUGS
```

NOTE

Buffers are displayed by use of the DISPLAY_MEMORY_DATA command, with the DISPLAY_ OPTIONS parameter specifying the desired tag name.

Error_Results Display

The Error_Results display contains the necessary information if an error is detected in the debug buffers.

```
EXPECTED DEBUG TRAP ON AAAAAA          EXP BUFFER AT XXXXXXXXXXXX
DEBUG MASK = XX                        ACT BUFFER AT XXXXXXXXXXXX
DEBUG LIST ENTRY = XXXXXXXXXXXXXXXX
                   XXXXXXXXXXXXXXXX
INSTRUCTION AT P = IIII IIII IIII IIII
                  EXP                                  ACT
    YYYYYY  X--------------X           YYYYYY  X--------------X
            X--------------X                   X--------------X
            X--------------X                   X--------------X
     MASK = FF00FFFFFFFFFFFF
```

- AAAAAA indicates the type of debug trap expected (READ, WRITE, FETCH, CALL, or BRANCH).
- The address of the buffer is the starting address of each buffer (PVA if on line).
- The debug list entry that applies to the expected debug trap is displayed.
- II--II displays 8 bytes of instructions, starting at the P of the expected debug trap.
- The EXP and ACT displays are the entry preceding the miscompare, the entry that miscompares, and the next entry in the EXPECTED and ACTUAL debug buffers. The mask shows which bits are used in the compare.

Error Messages

DBUG displays one of the following messages when a DBUG command is entered incorrectly. A valid command clears the error message.

INVALID ENTRY ERROR
INST LIST FULL ERROR
CANNOT TEST XX ERROR

Invalid Entry Error

The invalid entry error occurs when DBUG does not recognize the command entered or when the command parameters are not acceptable.

Inst List Full Error

The inst list full error occurs when the instructions added by the ADD_INSTRUCTIONS command exceed the instruction buffer.

Cannot Test XX Error

The cannot test XX error occurs when DBUG cannot execute the opcode XX. The following opcodes are not tested.

HALT	(00)
SYNC	(01) Used at the end of the instruction list for scoping.
EXCHANGE	(02) Used but not tested.
INTERRUPT	(03)
RETURN	(04) Tested in conjunction with a CALL instruction.
POP	(06) Not tested during online execution.
CPYXS	(0F)
KEYPOINT	(B1)
EXECUTE ALGORITHM	(C0-C7)

When running on line, DBUG runs as an unprivileged job. Because of this, LPAGE (17) is not tested and for PURGE (05), only K values of 03 through 07 are used. When running off line, DBUG tests the LPAGE instruction and all K values for the PURGE instruction. For both online and offline testing, BRCR (9F) is tested only with K values.

Central Memory Test (CMEM) 15

Central Memory Test (CMEM) 15

This chapter describes commands used with the Central Memory Test (CMEM).

Central Memory Test Description

CMEM is used to perform disk paging and memory testing of the working set of pages assigned to the test. CMEM tests disk paging associated with a segment access local file opened by the test. The default file size set by DVS is one megabyte, but it can be altered easily with the SET_FILE_SIZE command.

CMEM is organized into 11 sections. Following is a brief description of the testing performed by each section.

1. Tests segment access disk file with an all-zeros pattern.
2. Tests segment access disk file with an all-ones pattern.
3. Tests segment access disk file with a 5_{16} pattern.
4. Tests segment access disk file with an A_{16} pattern.
5. Tests segment access disk file with a DF_{16} pattern.
6. Tests segment access disk file with a matching-ones pattern.
7. Tests segment access disk file with an address pattern in which the word address contains the address as data.
8. Same as section 7 but uses the complement of the address as data.
9. Tests segment access disk file with a DF_{16} pattern, using a Load/Store Byte with a length of 5 and a following increment.
10. Tests segment access disk file with a DF_{16} pattern using Load/Store Multiple.
11. Tests segment access disk file with a random data pattern.

15

SET_DISPLAY_REQUEST (SETDR)

Purpose Selects a default REQUEST_OPTION for the DISPLAY_MESSAGES command.

Format

SET_DISPLAY_REQUEST
REQUEST_OPTION=keyword
STATUS=status variable

Parameters *REQUEST_OPTION (RO)*

Specifies a keyword for the desired display and sets it as the default REQUEST_OPTION for the DISPLAY_MESSAGES command.

RUN (R)

NIL (N)

If REQUEST_OPTION is omitted, NIL is used.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_DISPLAY_REQUEST command selects the running display as the default REQUEST_OPTION for the DISPLAY_MESSAGES command. Entering the DISPLAY_MESSAGES command without any parameters results in the automatic display of the test's running messages.

```
CMEM_1/set_display_request ro=run
```

DISPLAY_PARAMETERS (DISP)

Purpose Displays the test's parameters from its parameter word block.

Format

DISPLAY_PARAMETERS
OUTPUT=file reference
STATUS=status variable

Parameters *OUTPUT (O)*

Specifies the file to which information is displayed. If OUTPUT is omitted, $OUTPUT is used.

STATUS (S)

Specifies the termination condition of the command.

Examples The following DISPLAY_PARAMETERS command is used to display the CMEM_1 parameters from its parameter word block. By default, the following parameter display appears. If any parameter values are changed, the display is also changed.

```
CMEM_1/display_parameters
REQUEST OPTION      : NIL
REPEAT COUNT        : 0
SEGMENT FILE SIZE   : 1048576
SECTION SELECTED    : 1...11
```

SET_REPEAT_COUNT (SETRC)

Purpose Sets the number of test passes to be executed.

NOTE

A repeat count value of 0 executes the test until it is terminated by expiration of a time limit or by a DROP_TEST command.

Format

SET_REPEAT_COUNT
VALUE=integer
STATUS=status variable

Parameters

VALUE (V)

Specifies the repeat count to be loaded prior to the beginning of test execution. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_REPEAT_COUNT command sets CMEM_3 to run through five test passes.

```
CMEM_3/set_repeat_count v=5
```

SET_SECTION_SELECTIONS (SETSS)

Purpose Selects or deselects test sections to be executed.

Format

SET_SECTION_SELECTIONS
ADD=list of 1...11
DELETE=list of 1...11
STATUS = status variable

Parameters

ADD (A)

Specifies one or more sections to be selected for execution by the test. Sections can be specified as a list or range of sections.

DELETE (D)

Specifies one or more sections to be deselected for execution by the test. Sections can be specified as a list or range of sections.

STATUS (S)

Specifies the termination condition of the command

Examples The following SET_SECTION_SELECTIONS command selects sections 1 through 5, 7, and 9 through 11 for execution by CMEM_1.

```
CMEM_1/set_section_selections add=(1...5,7,9...11)
```

The following SET_SECTION_SELECTIONS command deselects sections 3, 7, and 9 through 11 from the current CMEM_2 section execution list. These sections are not executed by the test.

```
CMEM_2/setss d=(3,7,9...11)
```

15

SET_FILE_SIZE (SETFS)

Purpose Sets the segment file size o.f the local segment access file.

Format

SET_FILE_SIZE
FILE_SIZE=100...amc$file_byte_limit
STATUS=status variable

Parameters

FILE_SIZE (FS)

Specifies the size of the file to be opened for read/write testing. The maximum file size is determined by the file byte limit for a segment access file. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_FILE_SIZE command sets the size of the local segment access file to 100000 bytes.

```
CMEM_1/set_file_size fs=100000
```

SET_ELEMENT_NAME (SETEN)

Purpose Selects the element on which the segment access file is to be written.

NOTE

The SET_ELEMENT_NAME command is only permitted in a SYSTEM or MAINTENANCE job.

Format

SET_ELEMENT_NAME
ELEMENT_NAME=name
STATUS=status variable

Parameters

ELEMENT_NAME (EN)

Specifies the name of the element on which the segment access file is to be written. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_ELEMENT_NAME command selects DISK_UN_40 as the element on which the segment access file is to be written.

```
CMEM_1/seten en=disk_un_40
```

Test Messages

Messages are displayed using the DISPLAY_MESSAGES command. DISPLAY_MESSAGES selects a message display by using the REQUEST_OPTION parameter. The valid request option for CMEM is the following:

RUN R

Run Display

The Run display contains the test's current pass count and error count values.

```
CMEM  CC  PCXXXX  SXXXX  SBXXXX  CXXXX
SEG FILE  RING=X  SEG=XX  BYTE OFFSET=XXXXXXXX
```

Where:

CC Display identifier.

RU	Running.
SP	Stopped for parameters.
SE	Stopped on error.
ST	Stopped at end of test pass.
SS	Stopped at end of section.
SB	Stopped at end of subsection.
SC	Stopped at end of condition.

Error Messages

Data errors detected by CMEM result in the display of the following standard error message:

```
CMEM  SE  PCXXXX  SXXXX  SBXXXX CXXXX
EC1 = XXXX  EC2 = XXXX  TE = XXXX

  ADRS            EXP              RCV
XXXXXXXXX     XXXXXXXXXXXX     DOUBLE BIT ERROR
```

Where:

SE	Stopped on error.
TE	Total errors.
ADRS	Address of word in error 16.
EXP	Expected result.
RCV	Double bit error.

Error data can be displayed by using the DISPLAY_MEMORY_DATA command. The DISPLAY_OPTION parameter should specify SEGPTR, which is the starting PVA address of the segment access local file used by CMEM. The PLUS and MINUS commands can be used to scan the file.

Disk Test (DISK) 16

Disk Test (DISK) 16

This chapter describes commands used with the Disk Test (DISK).

Disk Test Description

The DISK test creates a disk file (via sequential access) according to record count, record size, and data pattern specifications. By default, the file contains 512 records, each of which consists of four 64-word (512-byte) blocks. The default data pattern is defined by the test. The test fills each of the blocks with a different pattern in the following order:

1. All ones
2. All zeros
3. All fives
4. All a's
5. 55AA
6. Shifting ones
7. Shifting zeros
8. Random

Each record is an exact duplicate of all others within the file. The first block of a default record is filled with all ones, the second with all zeros, the third with all fives, and the fourth with all a's.

If the record size is increased from 2048 bytes to 6144 bytes, each record then contains twelve 64-word blocks. Each of the twelve blocks contains one of the valid data patterns: all ones through random. These patterns are stored in order, one per block, starting with all ones in block one, all zeros in block two, and continuing through random in block eight, all ones in block nine, all zeros in block ten, all fives in block eleven, and all a's in block twelve.

After the disk file is created, it is verified. Each record is verified twice. The first time, the test reads the records sequentially and verifies the data as well as the file attributes and parameters. The second time, the test reads the records directly, beginning at record one, then reads record n (where n is the maximum record), back to record two, then to record n-1, and so on until all the records are verified. If an error is found, it is recorded and test execution continues.

DISPLAY_PARAMETERS (DISP)

Purpose Displays the test's parameters from its parameter word block.

Format

DISPLAY_PARAMETERS
OUTPUT=file reference
STATUS=status variable

Parameters

OUTPUT (O)

Specifies the file to which information is displayed. If OUTPUT is omitted, $OUTPUT is used.

STATUS (S)

Specifies the termination condition of the command.

Examples The following DISPLAY_PARAMETERS command is used to display the DISK_1 parameters from its parameter word block. By default, the following parameter display appears. If any parameter values are changed, the display is also changed.

```
DISK_1/display_parameters
REQUEST OPTION : NIL
REPEAT COUNT   : 0
DATA PATTERN   : TEST DEFINED
RECORD COUNT   : 512
RECORD SIZE    : 2048
ELEMENT NAME   : UNKNOWN
```

SET_DATA_PATTERN (SETDP)

Purpose Selects the data pattern stored in each word of the disk file. Variation of the patterns tests data integrity.

Format

SET_DATA_PATTERN
ALL_ONES=boolean
ALL_ZEROS=boolean
ALL_FIVES=boolean
ALL_A=boolean
FIVE_FIVE_A_A=boolean
SHIFTING_ONES=boolean
SHIFTING_ZEROS=boolean
RANDOM=boolean
USER_DEFINED=boolean
USER_PATTERN=string variable
TEST_DEFINED=boolean
STATUS=status variable

Parameters

ALL_ONES (AO)

Specifies whether the data pattern stored in each word of the record block is set to all ones. The binary representation of one byte within a word is 11111111.

TRUE

The data pattern is set to all ones.

FALSE

The data pattern is not set to all ones.

If ALL_ONES is omitted, FALSE is used.

ALL_ZEROS (AZ)

Specifies whether the data pattern stored in each word of the record block is set to all zeros. The binary representation of one byte within a word is 00000000.

TRUE

The data pattern is set to all zeros.

FALSE

The data pattern is not set to all zeros.

If ALL_ZEROS is omitted, FALSE is used.

ALL_FIVES (AF)

Specifies whether the data pattern stored in each word of the record block is set to all fives. The binary representation of one byte within a word is 01010101.

TRUE

The data pattern is set to all fives.

FALSE

The data pattern is not set to all fives.

If ALL_FIVES is omitted, FALSE is used.

ALL_A (AA)

Specifies whether the data pattern stored in each word of the record block is set to all a's. The binary representation of one byte within a word is 10101010.

TRUE

The data pattern is set to all a's.

FALSE

The data pattern is not set to all a's.

If ALL_A is omitted, FALSE is used.

FIVE_FIVE_A_A (FFAA)

Specifies whether the data pattern stored in each word of the record block is set to five, five, a, a, five, five, a, a, and so on. The binary representation of two bytes within a word is 0101010110101010.

TRUE

The data pattern is set to five, five, a, a.

FALSE

The data pattern is not set to five, five, a, a.

If FIVE_FIVE_A_A is omitted, FALSE is used.

SHIFTING_ONES (SO)

Specifies whether the data pattern stored in each word of the record block is set to all zeros, with a single bit enabled. This single bit is shifted from right to left within the record block.

TRUE

The data pattern is set to shifting ones.

FALSE

The data pattern is not set to shifting ones.

If SHIFTING_ONES is omitted, FALSE is used.

SHIFTING_ZEROS (SZ)

Specifies whether the data pattern stored in each word of the record block is set to all ones, with a single bit disabled. This single bit is shifted from right to left within the record block.

TRUE

The data pattern is set to shifting zeros.

FALSE

The data pattern is not set to shifting zeros.

If SHIFTING_ZEROS is omitted, FALSE is used.

RANDOM (R)

Specifies whether the data pattern stored in each word of the record block is set to a random pattern.

TRUE

The data pattern is set to a random pattern.

FALSE

The data pattern is not set to a random pattern.

If RANDOM is omitted, FALSE is used.

USER_DEFINED (UD)

Specifies whether the data pattern stored in each word of the record block is to contain the 16 digits supplied by the USER_PATTERN parameter.

TRUE

The data pattern is defined by the user.

FALSE

The data pattern is not defined by the user.

If USER_DEFINED is omitted, FALSE is used.

NOTE

The USER_PATTERN parameter is required when the USER_DEFINED parameter is set to TRUE.

USER_PATTERN (UP)

Specifies a string of 16 characters (0...9, A...F) to be stored in each word of the record block. Any combination of these characters can be used.

TEST_DEFINED (TD)

Specifies whether the data pattern is to be defined by the test. The test sets each block within a record to contain one of the eight valid data patterns (all ones...random). If the record size is 4096 bytes or greater, all patterns are stored at least once, depending on the size of the record. If the record size is 2048 bytes, however, only four of the data patterns (all ones...all a's) are stored since each record contains only four blocks.

TRUE

The data pattern is defined by the test.

FALSE

The data pattern is not defined by the test.

If TEST_DEFINED is omitted, FALSE is used.

STATUS (S)

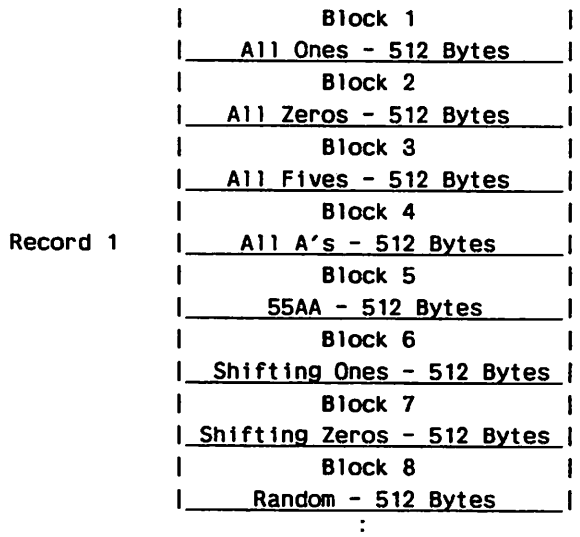
Specifies the termination condition of the command.

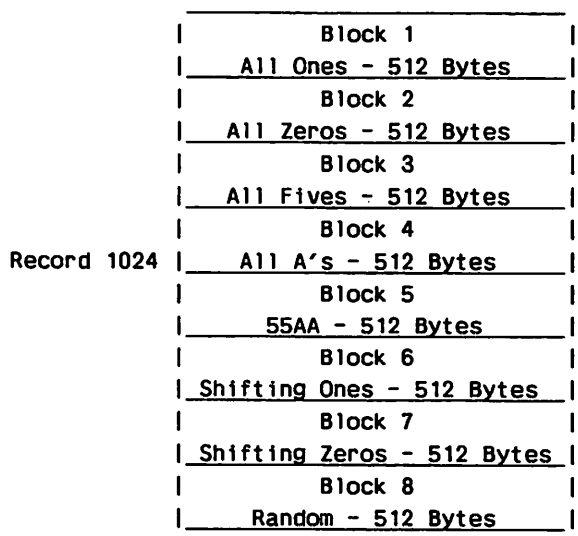
NOTE

By default, the data pattern is TEST_DEFINED.

Remarks Following is the basic structure of the disk file. Assume that the file is created using the following values:

- Record Count = 1024
- Record Size = 4096
- Data Pattern = Test Defined





Examples In the following example, the SET_DATA_PATTERN command sets the data pattern stored in each word of the record block to contain all a's.

```
DISK_1/set_data_pattern aa=true
```

The following SET_DATA_PATTERN command sets the data pattern stored in each word of the record block to contain FEDCBA9876543210.

```
DISK_4/setdp user_defined=true up='FEDCBA9876543210'
```

SET_ELEMENT_NAME (SETEN)

Purpose Selects the element on which the disk file is to be written.

NOTE

The SET_ELEMENT_NAME command is only permitted in a SYSTEM or MAINTENANCE job.

Format

SET_ELEMENT_NAME
ELEMENT_NAME=name
STATUS=status variable

Parameters **ELEMENT_NAME (EN)**

Specifies the name of the element on which the disk file is to be written. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_ELEMENT_NAME command selects DISK_YELLOW_81 as the element on which the disk file is to be written.

```
DISK_1/set_element_name en=disk_yellow_81
```

SET_RECORD_COUNT (SETREC)

Purpose Selects the number of records contained in the disk file.

Format

SET_RECORD_COUNT
RECORD_COUNT=1...2147483647
STATUS=status variable

Parameters **RECORD_COUNT (RC)**

Specifies the number of records to be contained in the disk file. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

NOTE

By default, the file contains 512 records.

Examples In the following example, SET_RECORD_COUNT sets the number of records stored in the disk file to 2500.

```
DISK_1/set_record_count record_count=2500
```

SET_RECORD_SIZE (SETRS)

Purpose Selects the size of each record contained in the disk file.

Format

SET_RECORD_SIZE
RECORD_SIZE=2048...4398046511103
STATUS=status variable

Parameters

RECORD_SIZE (RS)

Specifies the size of each record in the disk file. This parameter is required.

NOTE

The record size must be a multiple of 2048 bytes.

STATUS (S)

Specifies the termination condition of the command.

NOTE

By default, the record size is 2048 bytes.

Examples The following SET_RECORD_SIZE command sets each record in the disk file to contain 6144 bytes.

```
DISK_5/setrs rs=6144
```

SET_REPEAT_COUNT (SETRC)

Purpose Sets the number of test passes to be executed.

NOTE

A repeat count value of 0 executes the test until it is terminated by expiration of a time limit or by a DROP_TEST command.

Format

SET_REPEAT_COUNT
VALUE=integer
STATUS=status variable

Parameters

VALUE (V)

Specifies the repeat count to be loaded prior to the beginning of test execution. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

Examples The following SET_REPEAT_COUNT command sets the repeat count value to 10.

```
DISK_3/set_repeat_count v=10
```

Test Messages

Messages are displayed using the DISPLAY_MESSAGES command. DISPLAY_MESSAGES selects a message display by using the REQUEST_OPTION parameter. The valid request options for DISK are:

PARAMETERS	P
RUN	R

Parameter Display

The Parameter display contains the test's parameter settings.

```
REPEAT COUNT             = 0
DATA PATTERN SELECTION   = TEST DEFINED
USER DATA PATTERN        = NOT INITIALIZED
RECORD COUNT             = 512
RECORD SIZE              = 2048
ELEMENT NAME             = UNKNOWN
```

Run Display

The Run display contains the test's current pass count and error count values.

```
DISK CC PC=X ERRORS=X NO. OF RECORDS=X  RECORD SIZE=X  PATTERN=X
ELEMENT NAME=X MM CURRENT RECORD=X
```

Where:

CC Display identifier.

RU	Running.
SP	Stopped for parameters.
SE	Stopped on error.
ST	Stopped at end of test.

PC Pass count.

MM Mode.

- Stopped.
- Writing sequentially.
- Reading direct.
- Reading sequentially.

Error Messages

When the DISK test detects an error, it displays one of the following error messages, which are added to the running display.

FILE POSITION ERROR
TRANSFER COUNT ERROR
FILE DIRECTORY ERROR
RECORD NUMBER MISMATCH ERROR
RECORD SIZE MISMATCH ERROR
DATA PATTERN CODE MISMATCH ERROR
DIRECTORY ADDRESS ERROR
DATA ERROR

File Position Error

The file position error occurs when the file is not positioned at the END-OF-RECORD as is expected after a record is read from the file.

```
FILE POSITION ERROR
  EXPECTED = END-OF-RECORD
  ACTUAL = X
```

Where:

X Current file position.

Transfer Count Error

The transfer count error occurs when the size of the record read differs from the size of the record written.

```
TRANSFER COUNT ERROR
  EXPECTED = X
  ACTUAL = Y
```

Where:

X Size of the record written.
Y Size of the record read.

File Directory Error

The file directory error occurs when a record address in the directory of the file read differs from a record address in the directory of the file written.

```
FILE DIRECTORY ERROR
  EXPECTED = X
  ACTUAL = Y
```

Where:

X Address of record written.
Y Address of record read.

Record Number Mismatch Error

The record number mismatch error occurs when the file directory contains a value for the number of records written that is different from that for the number of records read.

```
RECORD NUMBER MISMATCH ERROR
  EXPECTED = X
  ACTUAL = Y
```

Where:

X Number of records written to the file directory.
Y Number of records read from the file directory.

Record Size Mismatch Error

The record size mismatch error occurs when the file directory contains a value for the size of the records written that is different from that for the size of the records read.

```
RECORD SIZE MISMATCH ERROR
  EXPECTED = X
  ACTUAL = Y
```

Where:

X Size of the records written to the file directory.
Y Size of the records read from the file directory.

Data Pattern Code Mismatch Error

The data pattern code mismatch error occurs when the file directory contains a value for the data pattern code written that is different from that for the data pattern code read.

```
DATA PATTERN CODE MISMATCH ERROR
  EXPECTED = X
  ACTUAL = Y
```

Where:

X Data pattern code written to the file directory.
Y Data pattern code read from the file directory.

Directory Address Error

The directory address error occurs when the address of the file directory written differs from the address of the file directory read.

```
DIRECTORY ADDRESS ERROR
  EXPECTED = X
  ACTUAL = Y
```

Where:

X Address of the file directory written.
Y Address of the file directory read.

Data Error

The data error occurs when a word within a record written to the file differs from the same word within the same record read from the file.

```
                          DATA ERROR
  WORD      EXPECTED            ACTUAL           DIFFERENCE
   W    XXXXXXXXXXXXXXXX   YYYYYYYYYYYYYYYY   ZZZZZZZZZZZZZZZZ
```

Where:

W	Word containing erroneous data.
X	Word written to the record.
Y	Word read from the record.
Z	Difference between word written and word read.

Tape Test (TAPE) 17

Tape Test (TAPE) 17

This chapter describes commands used with the Tape Test (TAPE).

Tape Test Description

The TAPE test creates a disk file (via sequential access) according to record count, record size, and data pattern specifications. By default, the file contains 512 records, each of which consists of four 64-word (512-byte) blocks. The default data pattern is defined by the test. The test fills each of the blocks with a different pattern in the following order:

1. All ones
2. All zeros
3. All fives
4. All a's
5. 55AA
6. Shifting ones
7. Shifting zeros
8. Random

Each record is an exact duplicate of all others within the file. The first block of a default record is filled with all ones, the second with all zeros, the third with all fives, and the fourth with all a's.

If the record size is increased from 2048 bytes to 6144 bytes, each record then contains twelve 64-word blocks. Each of the twelve blocks contains one of the valid data patterns: all ones through random. The patterns are stored in order, one per block, starting with all ones in block one, all zeros in block two, and continuing through random in block eight, all ones in block nine, all zeros in block ten, all fives in block eleven, and all a's in block twelve.

After the disk file is created, it is verified. Each record is verified twice. The first time, the test reads the records sequentially and verifies the data as well as the file attributes and parameters. The second time, the test reads the records directly, beginning at record one, then reads record n (where n is the maximum record), back to record two, then to record n-1, and so on until all the records are verified. If the disk file contains an error, the error is reported and the test is terminated. However, if the disk file is created without error, it is backed up to tape, using the BACKUP_PERMANENT_FILE Utility. A tape request is issued by the test and appears on the system console. The backup utility waits for the tape (external vsn=DVS) to be mounted onto the specified tape drive. Note that a write ring is required on the tape. When the backup process is complete, the file is deleted by the test and restored via the RESTORE_PERMANENT_FILE Utility. Upon completion of the restore, the file is once again verified, using both sequential and direct accesses.

DISPLAY_PARAMETERS (DISP)

Purpose Displays the test's parameters from its parameter word block.

Format

DISPLAY_PARAMETERS
OUTPUT=file reference
STATUS=status variable

Parameters

OUTPUT (O)

Specifies the file to which information is displayed. If OUTPUT is omitted, $OUTPUT is used.

STATUS (S)

Specifies the termination condition of the command.

Examples The following DISPLAY_PARAMETERS command displays the TAPE_1 parameters from its parameter word block. By default, the following parameter display appears. If any parameter values are changed, the display is also changed.

```
TAPE_1/display_parameters
REQUEST OPTION : NIL
REPEAT COUNT   : 1
DATA PATTERN   : TEST DEFINED
RECORD COUNT   : 512
RECORD SIZE    : 2048
DENSITY        : 1600
```

SET_DATA_PATTERN (SETDP)

Purpose Selects the data pattern stored in each word of the disk file. Variation of the patterns tests data integrity.

Format

SET_DATA_PATTERN
ALL_ONES=boolean
ALL_ZEROS=boolean
ALL_FIVES=boolean
ALL_A=boolean
FIVE_FIVE_A_A=boolean
SHIFTING_ONES=boolean
SHIFTING_ZEROS=boolean
RANDOM=boolean
USER_DEFINED=boolean
USER_PATTERN=string variable
TEST_DEFINED=boolean
STATUS=status variable

Parameters *ALL_ONES (AO)*

Specifies whether the data pattern stored in each word of the record block is set to all ones. The binary representation of one byte within a word is 11111111.

TRUE

The data pattern is set to all ones.

FALSE

The data pattern is not set to all ones.

If ALL_ONES is omitted, FALSE is used.

ALL_ZEROS (AZ)

Specifies whether the data pattern stored in each word of the record block is set to all zeros. The binary representation of one byte within a word is 00000000.

TRUE

The data pattern is set to all zeros.

FALSE

The data pattern is not set to all zeros.

IF ALL_ZEROS is omitted, FALSE is used.

ALL_FIVES (AF)

Specifies whether the data pattern stored in each word of the record block is set to all fives. The binary representation of one byte within a word is 01010101.

TRUE

The data pattern is set to all fives.

FALSE

The data pattern is not set to all fives.

If ALL_FIVES is omitted, FALSE is used.

ALL_A (AA)

Specifies whether the data pattern stored in each word of the record block is set to all a's. The binary representation of one byte within a word is 10101010.

TRUE

The data pattern is set to all a's.

FALSE

The data pattern is not set to all a's.

If ALL_A is omitted, FALSE is used.

FIVE_FIVE_A_A (FFAA)

Specifies whether the data pattern stored in each word of the record block is set to five, five, a, a, five, five, a, a, and so on. The binary representation of two bytes within a word is 0101010110101010.

TRUE

The data pattern is set to five, five, a, a.

FALSE

The data pattern is not set to five, five, a, a.

If FIVE_FIVE_A_A is omitted, FALSE is used.

SHIFTING_ONES (SO)

Specifies whether the data pattern stored in each word of the record block is set to all zeros, with a single bit enabled. This single bit is shifted from right to left within the record block.

TRUE

The data pattern is set to shifting ones.

FALSE

The data pattern is not set to shifting ones.

If SHIFTING_ONES is omitted, FALSE is used.

SHIFTING_ZEROS (SZ)

Specifies whether the data pattern stored in each word of the record block is set to all ones, with a single bit disabled. This single bit is shifted from right to left within the record block.

TRUE

The data pattern is set to shifting zeros.

FALSE

The data pattern is not set to shifting zeros.

If SHIFTING_ZEROS is omitted, FALSE is used.

RANDOM (R)

Specifies whether the data pattern stored in each word of the record block is set to a random pattern.

TRUE

The data pattern is set to a random pattern.

FALSE

The data pattern is not set to random pattern.

If RANDOM is omitted, FALSE is used.

USER_DEFINED (UD)

Specifies whether the data pattern stored in each word of the record block is to contain the 16 digits supplied by the USER_PATTERN parameter.

TRUE

The data pattern is defined by the user.

FALSE

The data pattern is not defined by the user.

If USER_DEFINED is omitted, FALSE is used.

NOTE

The USER_PATTERN parameter is required when the USER_DEFINED parameter is set to TRUE.

USER_PATTERN (UP)

Specifies a string of 16 characters (0...9, A...F) to be stored in each word of the record block. Any combination of these characters can be used.

TEST_DEFINED (TD)

Specifies whether the data pattern is to be defined by the test. The test sets each block within a record to contain one of the eight valid data patterns (all ones...random). If the record size is 4096 bytes or greater, all patterns are stored at least once, depending on the size of the record. If the record size is 2048 bytes, however, only four of the data patterns (all ones...all a's) are stored since each record contains only four blocks.

TRUE

The data pattern is defined by the test.

FALSE

The data pattern is not defined by the test.

If TEST_DEFINED is omitted, FALSE is used.

STATUS (S)

Specifies the termination condition of the command.

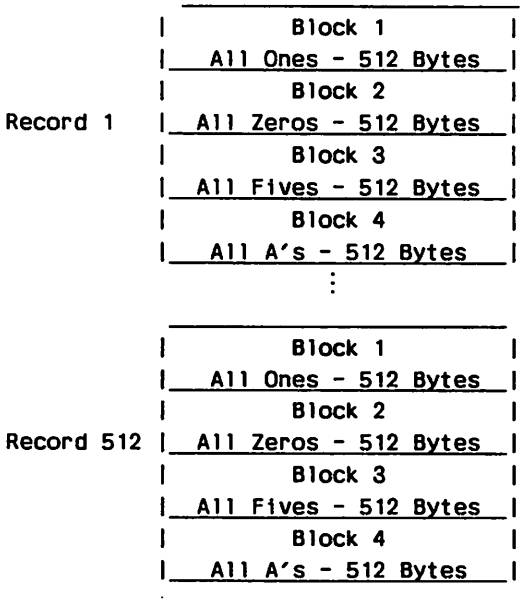
NOTE

By default, the data pattern is TEST_DEFINED.

Remarks

Following is the basic structure of the disk file, which is to be backed up to and restored from tape. Assume that the file is created using the following defaults:

- Record Count = 512 records
- Record Size = 2048 bytes
- Data Pattern = Test Defined



Examples In the following example, the SET_DATA_PATTERN command sets the data pattern stored in each word of the record block to contain all fives.

```
TAPE_1/set_data_pattern af=true
```

In the following example, the SET_DATA_PATTERN command sets the data pattern stored in each word of the record block to contain A0F3B7DEB1AA4178.

```
TAPE_3/set_data_pattern ud=true up='A0F3B7DEB1AA4178'
```

SET_DENSITY (SETD)

Purpose Selects the density of the magnetic tape used to back up and restore the created disk file.

Format

SET_DENSITY
DENSITY=800...6250
STATUS=status variable

Parameters

DENSITY (D)

Specifies the density of the magnetic tape. This parameter is required.

NOTE

The only valid densities are 800, 1600, and 6250 cpi.

STATUS (S)

Specifies the termination condition of the command.

NOTE

By default, the density is 1600 cpi.

Examples In the following example, the SET_DENSITY command sets the tape density to 6250 cpi.

```
TAPE_1/set_density density=6250
```

SET_RECORD_COUNT (SETREC)

Purpose Selects the number of records contained in the disk file. This file is backed up to and restored from tape.

Format

SET_RECORD_COUNT
RECORD_COUNT=1...2147483647
STATUS=status variable

Parameters

RECORD_COUNT (RC)

Specifies the number of records to be contained in the disk file. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

NOTE

By default, the file contains 512 records.

Examples In the following example, SET_RECORD_COUNT sets the number of records stored in the disk file to 1000.

```
TAPE_3/set_record_count rc=1000
```

SET_RECORD_SIZE (SETRS)

Purpose Selects the size of each record contained in the disk file. The file is backed up to and restored from tape.

Format

SET_RECORD_SIZE
RECORD_SIZE=2048...4398046511103
STATUS=status variable

Parameters

RECORD_SIZE (RS)

Specifies the size of each record in the disk file. This parameter is required.

NOTE

The record size must be a multiple of 2048 bytes.

STATUS (S)

Specifies the termination condition of the command.

NOTE

By default, the record size is 2048 bytes.

Examples The following SET_RECORD_SIZE command is used to set each record in the disk file to contain 4096 bytes.

```
TAPE_1/setrs record_size=4096
```

SET_REPEAT_COUNT (SETRC)

Purpose Sets the number of test passes to be executed.

NOTE

A repeat count value of 0 executes the test until it is terminated by expiration of a time limit or by a DROP_TEST command.

Format

SET_REPEAT_COUNT
VALUE=integer
STATUS=status variable

Parameters

VALUE (V)

Specifies the repeat count to be loaded prior to the beginning of test execution. This parameter is required.

STATUS (S)

Specifies the termination condition of the command.

NOTE

By default, the repeat count is set to 1.

Examples In the following example, the SET_REPEAT_COUNT command sets the repeat count value to 5.

```
TAPE_2/set_repeat_count v=5
```

Test Messages

Messages are displayed using the DISPLAY_MESSAGES command. DISPLAY_MESSAGES selects a message display by using the REQUEST_OPTION parameter. The valid request options for TAPE are the following:

PARAMETERS	P
RUN	R

Parameter Display

The Parameter display contains the test's parameter settings.

```
REPEAT COUNT            = 1
DATA PATTERN SELECTION  = TEST DEFINED
USER DATA PATTERN       = NOT INITIALIZED
RECORD COUNT            = 512
RECORD SIZE             = 2048
DENSITY                 = 1600
```


Run Display

The Run display contains the test's current pass count and error count values.

```
TAPE CC PC=X ERRORS=X NO. OF RECORDS=X  RECORDS SIZE=X  PATTERN=X
ELEMENT NAME=X MM CURRENT RECORD=X
```

Where:

CC Display identifier.

RU	Running.
SP	Stopped for parameters.
SE	Stopped on error.
ST	Stopped at end of test.

PC Pass count

MM Mode

- Stopped.
- Writing sequentially.
- Reading direct.
- Reading sequentially.

Error Messages

When the TAPE test detects an error, it displays one of the following error messages, which are added to the running display.

FILE POSITION ERROR
TRANSFER COUNT ERROR
FILE DIRECTORY ERROR
RECORD NUMBER MISMATCH ERROR
RECORD SIZE MISMATCH ERROR
DATA PATTERN CODE MISMATCH ERROR
DIRECTORY ADDRESS ERROR
DATA ERROR

File Position Error

The file position error occurs when the file is not positioned at the END-OF-RECORD as is expected after a record is read from the file.

```
FILE POSITION ERROR
  EXPECTED = END-OF-RECORD
  ACTUAL = X
```

Where:

X Current file position.

Transfer Count Error

The transfer count error occurs when the size of the record read differs from the size of the record written.

```
TRANSFER COUNT ERROR
  EXPECTED = X
  ACTUAL = Y
```

Where:

X Size of the record written.
Y Size of the record read.

File Directory Error

The file directory error occurs when a record address in the directory of the file read differs from the record address in the directory of the file written.

```
FILE DIRECTORY ERROR
  EXPECTED = X
  ACTUAL = Y
```

Where:

X Address of the record written.
Y Address of the record read.

Record Number Mismatch Error

The record number mismatch error occurs when the file directory contains a value for the number of records written that is different from that of the number of records read.

```
RECORD NUMBER MISMATCH ERROR
  EXPECTED = X
  ACTUAL = Y
```

Where:

X Number of records written to the file directory.
Y Number of records read from the file directory.

Record Size Mismatch Error

The record size mismatch error occurs when the file directory contains a value for the size of the records written that is different from that of the size of the records read.

```
RECORD SIZE MISMATCH ERROR
  EXPECTED = X
  ACTUAL = Y
```

Where:

X Record size written to the file directory.
Y Record size read from the file directory.

Data Pattern Code Mismatch Error

The data pattern code mismatch error occurs when the file directory contains a value for the data pattern code written that is different from that of the data pattern code read.

```
DATA PATTERN CODE MISMATCH ERROR
  EXPECTED = X
  ACTUAL = Y
```

Where:

X Data pattern code written to the file directory.
Y Data pattern code read from the file directory.

Directory Address Error

The directory address error occurs when the address of the file directory written differs from that of the file directory read.

```
DIRECTORY ADDRESS ERROR
  EXPECTED = X
  ACTUAL = Y
```

Where:

X Address of the file directory written.
Y Address of the file directory read.

Data Error

The data error occurs when a word within a record written to the file differs from the same word within the same record read from the file.

```
                          DATA ERROR
  WORD       EXPECTED          ACTUAL          DIFFERENCE
   W     XXXXXXXXXXXXXXXX  YYYYYYYYYYYYYYYY  ZZZZZZZZZZZZZZZZ
```

Where:

W	Word containing erroneous data.
X	Word written to the record.
Y	Word read from the record.
Z	Difference between word written and word read.

Appendixes

Glossary A

A

Abort

The immediate abnormal termination of a task.

Absolute Path

Identifies a file or catalog via a system path, family path, user path, or local path. See also Relative Path.

Alphabetic Character

One of the following letters:

A to Z

a to z

See also Character and Alphanumeric Character.

Alphanumeric Character

An alphabetic character or a digit. See also Character, Alphabetic Character, and Digit.

Application Value

The kind of value whose syntax and semantics are defined by the application program that interprets the value.

ASCII

American National Standard Code for Information Interchange. A 7-bit code representing a prescribed set of characters. The 7-bit ASCII code character is stored right-justified in an 8-bit byte.

Assignment Statement

A statement that assigns a value to a variable.

B

Batch Mode

A mode of execution in which a job is submitted and processed as a unit with no intervention from the user. Contrast with Interactive Mode.

Beginning-of-Information (BOI)

The file boundary that marks the beginning of the file.

Bit

A binary digit. A bit has the value 0 or 1. See also Byte.

BOI

See also Beginning-of-Information.

Boolean

A value that can be either TRUE or FALSE.

A

Boolean Constant

A constant that represents a boolean (logical) value of TRUE or FALSE. One of the following names can be used to specify a boolean constant:

TRUE	FALSE
YES	NO
ON	OFF

Byte

A group of bits. For NOS/VE, a byte is 8 bits. An ASCII character code uses the rightmost 7 bits of one byte.

Byte-Addressable File Organization

A file organization in which records are accessed by their byte address.

C

Catalog

A directory of files and catalogs maintained by the system for a user. The catalog $LOCAL contains only file entries.

Also, the part of a path that identifies a particular catalog in a catalog hierarchy. The format is as follows:

```
name.name. ... .name
```

where each name is a catalog. See also Catalog Name and Path.

Catalog Name

The name of a catalog in a catalog hierarchy (path). By convention, the name of the user's master catalog is the same as the user's user name.

Character

A letter, digit, space, or symbol that is represented by a code in one or more of the standard character sets.

It is also referred to as a byte when used as a unit of measure to specify block length, record length, and so forth.

A character can be a graphic character or a control character. A graphic character is printable; a control character is nonprintable and is used to control an input or output operation.

Command

A statement that initiates a specific operation on NOS/VE. A command name is recognized by the SCL interpreter if it appears as an entry in the command list.

Command List

One or more entries that define the commands currently available. The command list consists of object libraries, catalogs, or the special entry $SYSTEM.

Command Utility

A NOS/VE processor that adds its command list (referred to as its subcommands) to the beginning of the SCL command list. The subcommands are removed from the command list when the processor terminates.

Comment

Any character or sequence of characters, except the quotation mark ("), that is preceded by a quotation mark and terminated by another quotation mark or an end of line. A comment is treated exactly as a space.

Condition Code

The six-digit code that uniquely identifies a NOS/VE diagnostic. The condition code is returned as part of the status record when an abnormal status occurs.

Condition Handler

A WHEN statement to which control is transferred when a condition occurs. The WHEN statement is executed only if it has been established as the condition handler for the specified condition and the condition occurs in its scope.

Control Statement

A statement used to structure and control the flow of a job.

Cycle

A numbered version of a file that can be registered with the same file name and in the same catalog as the other versions of the file. Each permanent file can have from 1 through 999 cycles. See also Cycle Number and Cycle Reference.

Cycle Number

An unsigned integer from 1 through 999 that identifies a specific version of a permanent file.

Cycle Reference

The cycle of a permanent file to be accessed. A cycle reference can be either an unsigned integer or one of the following designators:

$HIGH

$LOW

$NEXT

$NEXT_LOW

D

Default

The assumed value for a parameter when the parameter is not specified by the user.

Delimiter

An indicator that separates and organizes data.

A

Digit

One of the following characters:

0 1 2 3 4 5 6 7 8 9

Display Code

A 64-character subset of the ASCII code, which consists of alphabetic letters, symbols, and numerals.

Dual State

State in which two operating systems execute simultaneously on the same mainframe. Currently, NOS/VE and NOS Version 2 are such partners.

E

Ellipsis

1. Two or more consecutive periods at the end of a physical line to indicate command line continuation. The ellipsis can be optionally preceded and/or followed by a space.
2. Two consecutive periods separating two values to indicate a range of values in a parameter list. See also Value Element.

End-of-Information (EOI)

The point at which data in the file ends.

EOI

See End-of-Information.

Epilog

The SCL statement list that is executed at the end of a job.

Execution Ring

The level of hardware protection assigned to a procedure while it is executing.

Expression

Notation that represents a value. A constant or variable appearing alone, or combinations of constants, variables, and operators.

F

Family

A logical grouping of NOS/VE users that determines the location of their permanent files. A family can be subdivided into accounts and projects.

Family Administrator

The family member who creates, deletes, and otherwise manages the other members of the family. A system analyst assigns a family administrator at the time the family is created.

Family Name

A name that identifies a NOS/VE family. See also Family.

Family Path

Identifies a file via a family name and a user path as follows:

:family name.user path

or

$FAMILY.user path

Field

A subdivision of a record that is referenced by name. For example, the field NORMAL in a record named OLD_STATUS is referenced as follows:

OLD_STATUS.NORMAL

File

1. A collection of information referenced by a name.
2. An SCL element specifying a temporary or permanent file, including its path and, optionally, a cycle reference (for permanent files). A file is identified by specifying a path and, optionally, a cycle reference (for permanent files) as follows:

```
path.cycle reference
```

File Attribute

A characteristic of a file. Each file has a set of attributes that completely define file structure and processing limitations.

File Name

The name of a NOS/VE file. It is used in a file reference to identify the file. See also Name.

File Position

The location in the file at which the next read or write operation is to begin. The file position designators are:

$ASIS Leave the file in its current position.

$BOI Position the file at the beginning-of-information.

$EOI Position the file at the end-of-information.

File Reference

An SCL element that identifies a file and, optionally, the file position to be established prior to use. The format of a file reference is as follows:

```
file.file position
```

See also File and File Position.

I

Integer

A value representing one of the numbers 0, +1, -1, +2, -2,...

Integer Constant

A constant that represents an integer value. A sequence of one or more digits and, for hexadecimal integer constants, the following characters:

A B C D E F a b c d e f

A hexadecimal integer constant must begin with a digit. A preceding sign and subsequent radix are optional.

Interactive Mode

A mode of execution in which a user enters commands at a terminal and each command elicits a response from the computer. Contrast with Batch Mode.

J

Job

A set of commands executed for a user name. NOS/VE accepts interactive and batch jobs.

Job Attribute

A characteristic of a job.

Job Class

An attribute of a job. The job classes are batch, interactive, and maintenance.

Job File

A file that contains the statements and input data for the job and the output produced by the job. The job files are identified by the following names:

COMMAND

INPUT

OUTPUT

$JOB_LOG

$NULL

K

Keyword

A parameter value that has special meaning in the context of a particular parameter. For example, a parameter called COUNT might normally expect integer values but could be given the keyword ALL.

Family Path

Identifies a file via a family name and a user path as follows:

:family name.user path

or

$FAMILY.user path

Field

A subdivision of a record that is referenced by name. For example, the field NORMAL in a record named OLD_STATUS is referenced as follows:

OLD_STATUS.NORMAL

File

1. A collection of information referenced by a name.
2. An SCL element specifying a temporary or permanent file, including its path and, optionally, a cycle reference (for permanent files). A file is identified by specifying a path and, optionally, a cycle reference (for permanent files) as follows:

```
path.cycle reference
```

File Attribute

A characteristic of a file. Each file has a set of attributes that completely define file structure and processing limitations.

File Name

The name of a NOS/VE file. It is used in a file reference to identify the file. See also Name.

File Position

The location in the file at which the next read or write operation is to begin. The file position designators are:

$ASIS Leave the file in its current position.

$BOI Position the file at the beginning-of-information.

$EOI Position the file at the end-of-information.

File Reference

An SCL element that identifies a file and, optionally, the file position to be established prior to use. The format of a file reference is as follows:

```
file.file position
```

See also File and File Position.

I

Integer

A value representing one of the numbers 0, +1, –1, +2, -2,...

Integer Constant

A constant that represents an integer value. A sequence of one or more digits and, for hexadecimal integer constants, the following characters:

A B C D E F a b c d e f

A hexadecimal integer constant must begin with a digit. A preceding sign and subsequent radix are optional.

Interactive Mode

A mode of execution in which a user enters commands at a terminal and each command elicits a response from the computer. Contrast with Batch Mode.

J

Job

A set of commands executed for a user name. NOS/VE accepts interactive and batch jobs.

Job Attribute

A characteristic of a job.

Job Class

An attribute of a job. The job classes are batch, interactive, and maintenance.

Job File

A file that contains the statements and input data for the job and the output produced by the job. The job files are identified by the following names:

COMMAND

INPUT

OUTPUT

$JOB_LOG

$NULL

K

Keyword

A parameter value that has special meaning in the context of a particular parameter. For example, a parameter called COUNT might normally expect integer values but could be given the keyword ALL.

L

Label

A name used to reference a structured statement.

List

A command format notation specifying that a parameter can be given several values. See also Value List.

Load Module

An object module reformatted for code sharing and efficient loading. When the user generates an object library, each object module in the module list is reformatted and written as a load module on the object library.

Local File

A file that is accessed via the local catalog named $LOCAL. See also File, Path, and Local Path.

Local Path

Identifies a local file as follows:

$LOCAL.file name

M

Master Catalog

The catalog the system creates for each new user name. The master catalog contains entries for all permanent files and catalogs a user creates. By convention, the name of the master catalog is the same as the user name.

Module

A unit of text accepted as input by the loader, linker, or object library generator. See also Object Module and Load Module.

N

Name

A combination of from 1 through 31 characters chosen from the following set:

Alphabetic characters (A through Z and a through z)

Digits (0 through 9)

Special characters (#, @, $, or _)

The first character of a name cannot be a digit.

O

Object Code

Executable code produced by a compiler.

Object File

A file containing one or more object modules.

A

Object Library

A file containing one or more load modules and a directory to each module.

Object Module

A compiler-generated unit containing object code and instructions for loading the object code. It is accepted as input by the loader and the object library generator.

Operand

An entity to which an operation is applied.

Operator

The symbol that represents the action to be performed in an operation.

P

Parameter

A value list optionally preceded by and equated to a parameter name. For example:

```
parameter name = value list

    or

value list
```

Parameter Definition

Specifies the parameter names, value specification, and default specification for an SCL procedure parameter.

Parameter Expression

An expression that, when evaluated, results in a parameter.

Parameter List

A series of parameters separated by spaces or commas.

Parameter Name

A name that uniquely identifies a parameter.

Parameter Value

See also Value.

Path

Identifies a file. It may include the family name, user name, subcatalog name or names, and file name.

Permanent Catalog

A catalog of permanent files.

Permanent File

A file preserved by NOS/VE across job executions and system deadstarts. A permanent file has an entry in a permanent catalog. See also File.

Position-Dependent Parameter

A parameter that must appear in a specified location, relative to other parameters. Contrast with Position-Independent Parameter.

Position-Independent Parameter

A parameter that consists of a parameter name followed by a value list. Contrast with Position-Dependent Parameter.

Processor Attribute

A characteristic of the hardware processor on which the system is running.

Program Attribute

A characteristic of a program as defined in the program description or by a job default value.

Program Description

Information that defines a program, including the modules that constitute the program and the options to be used when it is executed.

Prolog

The SCL statement list that is executed at the beginning of each job.

R

Radix

Specifies the base of a number. NOS/VE recognizes binary, octal, decimal, and hexadecimal number bases. A radix enclosed in parentheses must follow a nondecimal number. The following numbers can be used to represent the radix:

2	Binary number base
8	Octal number base
10	Decimal number base
16	Hexadecimal number base

See also Integer Constant.

Range

A value represented as two values separated by an ellipsis. The element is associated with the values from the first value through the second value. The first value must be less than or equal to the second value. For example:

```
value..value
```

Relative Path

Identifies a file via defaults established with the current working catalog or an absolute path. A relative path is used in a family path, user path, and local path. SCL supplies any omitted values necessary for creating an absolute path.

Ring

The level of hardware protection given a file or segment. A file is protected from unauthorized access by tasks executing in higher rings.

See also Execution Ring.

A

Ring Attribute

A file attribute whose value consists of three ring numbers, referred to as r1, r2, and r3. The ring numbers define the four ring brackets for the file as follows:

Read bracket is 1 through r2.

Write bracket is 1 through r1.

Execute bracket is r1 through r2.

Call bracket is r2+1 through r3.

S

Sign

Indicates whether a number is positive or negative. It is one of the following characters:

+	Positive number
-	Negative number
space	Positive number

See also Integer Constant.

Standard File

A file that provides a default file for use by job files and other files. The standard files are identified by the following names:

$COMMAND	$COMMAND_OF_CALLER
$ECHO	$ERRORS
$INPUT	$LIST
$OUTPUT	$RESPONSE

Statement

The basic unit of input that is interpreted by SCL. The following are the SCL statement types:

Command

Assignment statement

Control statement

Statement List

One or more statements separated by delimiters.

Status Variable

A variable record of kind status that holds the completion status of a command.

String

A value that represents a sequence of characters.

String Constant

A constant that represents a string value. A sequence of characters delimited by apostrophes ('). An apostrophe can be included in the string by specifying two consecutive apostrophes.

Subcatalog

A catalog registered in the master catalog or another subcatalog. See also Catalog.

Subcommand

A command that has been added to the command list by a command utility.

System File

See Job File and Standard File.

System Operator

A user with special privileges. These privileges are available only at the system console.

System Path

Identifies a file or catalog in the system catalog. Its format is as follows:

$SYSTEM.relative path

T

Task

The instance of execution of a program.

U

User Name

A name that identifies a NOS/VE user and the location of a user's permanent files in the user's family.

User Path

Identifies a file or catalog via a user name and, optionally, a relative path as follows:

.user name.relative path

or

$USER.relative path

Utility

A NOS/VE processor consisting of routines that perform a specific operation. See also Command Utility.

V

Value

An expression or application value specified in a parameter list. Each value must match the defined kind of value for the parameter. Keywords, constants, and variable references are all values.

A

Value Count

An integer expression indicating the number of value elements supplied for a parameter.

Value Element

A single value or a range of values represented by two values separated by an ellipsis. For example:

value

or

value..value

See also Value, Value List, and Value Set.

Value List

A series of value sets separated by spaces or commas and enclosed in parentheses. If only one value set is given in the list, the parentheses can be omitted. For example:

(value set,value set,value set)

or

value set

See also Value, Value Element, and Value Set.

Value Set

A series of value elements separated by spaces or commas and enclosed in parentheses. If only one value element is given in the set, the parentheses can be omitted. For example:

(value element,value element,value element)

or

value element

See also Value, Value Element, and Value List.

Value Set Count

An integer expression indicating the number of value sets supplied for a parameter.

Variable

Represents a data value.

SCL defines the following kinds of variables:

string

integer

boolean

status

Variable Attribute

A characteristic of a variable.

Variable Name

A name that identifies a variable.

Variable Reference

An integer, string, or boolean variable reference identifying an integer, string, or boolean variable by its name and optional subscript. For example:

```
variable name(subscript)
```

A status variable reference identifies a status variable by its name and optional subscript and/or field. For example:

```
variable name(subscript).field
```

W

Working Catalog

The catalog prefixed to a file reference that begins with a name (that is, not a period, colon, or a system name beginning with a dollar sign).

Error Message Templates B

Error Message Templates B

This chapter describes the error messages produced by DVS and applicable tests.

DVS Error Messages

DVS uses the status record to report the results of all command and system requests to NOS/VE. The system message generator takes the status record supplied by the utility and uses the utility-supplied message templates to generate an error message in a standard format. The messages are listed alphabetically.

The error message templates shown in this section are the error messages generated by the utility. Error messages generated by the system as a result of incorrect command input are not shown.

Message Severity Levels

Each utility message includes a severity level indicating the severity of the message. DVS detects errors at three severity levels: WARNING, ERROR, and FATAL.

--WARNING--

These messages report conditions encountered during command processing that may have caused incorrect or incomplete operation of a command. Execution of tests is not affected.

--ERROR--

These messages report that the last command was not completed correctly. Execution of tests may be affected, so you should consult the status display.

--FATAL--

These messages report that the last command was not completed correctly. A majority of fatal errors are a result of internal errors detected by the utility. If you encounter a fatal warning, you may continue executing utility commands, but you should do so with caution.

DVS Utility

--ERROR--Duplicate entries in halt/trap masks.

Description: The test's halt and trap masks have duplicate entries. The utility is unable to process the reset trap mask requests.

--ERROR--DVS display request error; cause: blank insertion error.

Description: The test blank insertion request has caused the line length value to be exceeded.

--ERROR--DVS display request error; cause: invalid character count value.

Description: The test character count value exceeds the maximum number of displayable characters.

--ERROR--DVS display request error; cause: line index or size error.

Description: The test line index or line size is exceeded when a display request is attempted.

--ERROR--DVS display request error; cause: line number invalid.

Description: The test line number is invalid when a display request is attempted.

--ERROR--DVS queue error; cause: message queue filled.

Description: The utility's message queue is full. The queue should be emptied by using one of the display commands.

--ERROR--DVS request error; cause: pit value out of range.

Description: The test issues a set pit request in which the PIT value is either too small or too large.

--ERROR--DVS test interlock error; cause: interlock time-out.

Description: The utility detects an interlock time-out between test communications.

--ERROR--DVS test interlock error; cause: lock already set.

Description: The utility detects that an interlock is already locked when it is not supposed to be.

--ERROR--DVS test interlock error; cause: lock not locked.

Description: The utility detects that an interlock is not locked although it is supposed to be.

--ERROR--Invalid end-of-list code.

Description: The debug list does not contain an end-of-list code.

--ERROR--Message type invalid, keyboard buffer empty.

Description: The message type is invalid because the keyboard buffer did not contain a valid command.

--ERROR--Named_task not found, {test_identifier}.

Description: The utility is unable to locate the named_task entry in the HEAP that corresponds to the TEST_IDENTIFIER.

--ERROR--Table overflow, too many tasks: named task.

Description: The utility is unable to allocate space in the HEAP for the current named task.

--ERROR--Table overflow, too many tasks: task_message_ptr.

Description: The utility is unable to allocate space in the HEAP for task messages.

--ERROR--Table overflow, too many tasks: task shared variables.

Description: The utility is unable to allocate space in the HEAP for the task shared variables.

--ERROR--Table overflow, too many tasks: trap history.

Description: The utility is unable to allocate space in the HEAP for the trap history.

--ERROR--Test_id: {test_identifier}, already in use.

Description: The utility detects that the requested TEST_IDENTIFIER is already in use for another test.

--ERROR--Test name, {test name}, invalid for CYBER 930 and 932 mainframes.

Description: The selected test cannot currently run on a CYBER 930 and 932.

--ERROR--Test name, {test name} invalid for CYBER 810, 815, 825, and 830 mainframes.

Description: The selected test cannot currently run on CYBER 810, 815, 825 and 830 mainframes.

--ERROR--Test name, {test name} invalid for CYBER 835 mainframes.

Description: The selected test cannot currently run on a CYBER 835.

--ERROR--Test name, {test name} invalid for CYBER 840, 855, and 860 mainframes.

Description: The selected test cannot currently run on CYBER 840, 855, and 860 mainframes.

--ERROR--Test name, {test name} invalid for CYBER 990 mainframes.

Description: The selected test cannot currently run on a CYBER 990.

--ERROR--Test parameter error; cause: selected processor value out of range.

Description: The selected processor value is not one of the available processors.

--ERROR--Unexpected halt, detected by DVS.

Description: An unexpected halt occurred as specified by the test's halt mask.

--ERROR--Unexpected trap, detected by DVS.

Description: An unexpected trap occurred as specified by the test's trap mask.

--FATAL--DVS internal error; cause: activity index too large.

Description: The utility detects that the number of tasks that it can capably monitor exceeds an internal activity limit.

--FATAL--DVS internal error; cause: FCT3 incorrect entry to DVS.

Description: FCT3 called DVS at an invalid entry point. This entry point is used only for offline systems.

--FATAL--DVS internal error; cause: forward link ptr invalid in named_task link list.

Description: The utility detected invalid pointers in the HEAP used for the named_task.

--FATAL--DVS internal error; cause: heap ptr NIL.

Description: The utility is unable to generate a shared segment access file for task communications.

--FATAL--DVS internal error; cause: incorrect condition type.

Description: A condition type other than system condition is processed.

--FATAL--DVS internal error; cause: send error call.

Description: The test issued a "send error call" to the utility. The utility does not support this request.

--FATAL--DVS internal error; cause: unimplemented call.

Description: The test issued to the utility a request that is not implemented.

--FATAL--DVS internal error; cause: upper bound of PVA table reached.

Description: The utility is unable to initialize the PVA table of memory addresses.

RUN_CPU_DIAGNOSTICS and RUN_SYSTEM_DIAGNOSTICS

--ERROR--ALL must appear by itself, TEST_NAME.

Description: ALL is used with other keywords for the TEST_NAME parameter.

--ERROR--Test parameter error; cause: HOURS and/or MINUTES must be given when RUN_OPTIONS = TIMED.

Description: Selection of the TIMED keyword requires the specification of the HOURS and/or MINUTES parameters.

--ERROR--Test parameter error; cause: HOURS cannot be specified when RUN_OPTIONS = ONE_HOUR.

Description: Selection of the ONE_HOUR keyword prevents specification of the HOUR parameter.

--ERROR--Test parameter error; cause: MINUTES cannot be specified when RUN_OPTIONS = ONE_HOUR.

Description: Selection of the ONE_HOUR keyword prevents specification of the MINUTES parameter.

DISPLAY_HISTORY

--ERROR--Invalid DVS HISTORY file header.

Description: The DVS history file header is destroyed, indicating that the file may be corrupted.

DISPLAY_FAILURES

--ERROR--Invalid DVS HISTORY file header.

Description: The DVS history file header is destroyed, indicating that the file may be corrupted.

Menu Facility

--ERROR--A COMMAND_NAME must be specified along with a PARAMETER_NAME.

Description: In order to search for explanatory text, the EXPLAIN command requires both the command and parameter names.

--ERROR--DVS range error; cause: function out of range.

Description: The utility detects that the function value specified for the MENU or EXPLAIN command exceeds the range allowed for the current menu.

--ERROR--Invalid COMMAND_NAME.
--ERROR--Invalid PARAMETER_NAME.
--ERROR--Invalid TEST_NAME.

Description: The test, command, or parameter name specified with the EXPLAIN command is invalid. In a search through the appropriate tables, the specified name has not been found.

--ERROR--Invalid command name; command not implemented.

Description: The specified command has not yet been implemented for use.

--ERROR--Invalid command search; index out of range.

Description: The specified index of the MENU or EXPLAIN command exceeds the range allowed for the current command menu.

--ERROR--Negative index; cause: parameter selection.

Description: The upper bound of the parameter selection index for the current menu page is set to a negative value.

EXECUTE _TEST

--ERROR--ALL must appear by itself, LOAD_MAP_OPTIONS.

Description: ALL is used with other keywords for the LOAD_MAP_OPTIONS parameter.

--ERROR--Invalid KEYWORD value for the LOAD_MAP_OPTIONS parameter.
--ERROR--Invalid KEYWORD value for the PRESET_VALUE parameter.
--ERROR--Invalid KEYWORD value for the TERMINATION_ERROR_LEVEL parameter.

Description: The utility detects an invalid keyword specification.

--ERROR--No test identifier, cause: Execute_Test.

Description: The utility detects that the TEST_IDENTIFIER has not been assigned.

--ERROR--NONE must appear by itself, LOAD_MAP_OPTIONS.

Description: NONE is used with other keywords for the LOAD_MAP_OPTIONS parameter.

--ERROR--Redundancy in selections, BLOCK.
--ERROR--Redundancy in selections, CROSS_REFERENCE.
--ERROR--Redundancy in selections, ENTRY_POINT.
--ERROR--Redundancy in selections, SEGMENT.

Description: A redundancy was detected between the program attributes from the LOAD_MAP_OPTIONS parameter.

--ERROR--Test_id: {test_identifier}, already in use.

Description: The TEST_IDENTIFIER supplied with the EXECUTE_TEST subcommand is already in use.

--ERROR--Test parameter error; cause: selected processor value out of range.

Description: The selected processor value is not one of the available processors.

--ERROR--Test name invalid, {test name}.

Description: The selected test is not one of the available DVS tests.

--FATAL--DVS initialization error; cause: load not complete with REDUCE_TASK_PRIORITY.

Description: The selected test has not successfully loaded for background operations.

--FATAL--DVS internal error; cause: attached local file table full.

Description: Maximum number of tests are in execution.

DISPLAY_STATUS

--ERROR--DVS test selection error; cause: test not selected.

Description: The utility is unable to display status because the test has not been selected.

--ERROR--Invalid KEYWORD value used for DISPLAY_OPTIONS parameter.

Description: The utility detects an invalid keyword specification.

--ERROR--No test identifier; cause: no active tests.

Description: The utility is unable to display status because the test is no longer executing.

SET_STATUS DISPLAY

--ERROR--Invalid KEYWORD value for the ATTRIBUTE_OPTION parameter.

Description: The utility detects an invalid keyword specification.

--ERROR--No test identifier, cause: test_id not specified or test not selected.

Description: The utility is unable to set the status display because either the test has not been selected or the identifier is missing.

DROP_TEST

--ERROR--Incorrect backward link ptr: {test_identifier}.

Description: The backward link pointer for the indicated test is invalid. The HEAP containing these pointers may have been corrupted.

--ERROR--Incorrect forward link ptr: {test_identifier}.

Description: The forward link pointer for the indicated test is invalid. The HEAP containing these pointers may have been corrupted.

--ERROR--No test identifier; cause: test not selected.

Description: The utility is unable to process the command because the test has not been selected.

--FATAL--DVS internal error; cause: link selection error.

Description: The utility detects a discrepancy in the forward and backward link selections.

DISPLAY_MESSAGES and DISPLAY_MEMORY_DATA

--ERROR--Invalid KEYWORD value for the DISPLAY_OPTION parameter.
--ERROR--Invalid KEYWORD value for the REQUEST_OPTION parameter.
--ERROR--Invalid KEYWORD value for the SEGMENT_NAME parameter.

Description: The utility detects an invalid keyword specification.

--FATAL--DVS internal error; cause: incorrect display request.

Description: The utility detects a display request that has not been issued for the current named task.

--FATAL--DVS internal error; cause: msg_ptr = NIL.

Description: The utility is unable to allocate space in the HEAP for requested messages.

--WARNING--Segment offset must be a multiple of 8.

Description: The utility requires that the segment offset value be 0 modulo 8.

--WARNING--Test completed execution, DISPLAY_MEMORY_DATA command invalid.

Description: The utility cannot process the command because the test has already completed execution.

--WARNING--Test execution for {test_identifier}, completed.

Description: The utility detects that the test completed execution; the command can no longer be performed.

--WARNING--Test parameter error; cause: memory PVA not initialized.

Description: The PVA for the segment to be displayed must have been initialized with the DISPLAY_MEMORY_DATA command.

--WARNING--Test {test_identifier}, busy; request queued.

Description: The test has not responded with the requested display within the utility's time-out period. The display is queued instead.

--WARNING--Test {test_identifier} in process of loading; check status display.

Description: The utility cannot process the command because the test has not completed loading. Do not enter test commands until loading is complete.

PLUS and MINUS

--ERROR--Test parameter error; cause: memory PVA not initialized.

Description: The PVA for the segment to be displayed must be initialized with the DISPLAY_MEMORY_DATA command.

--WARNING--Segment offset must be a multiple of 8.

Description: The utility requires that the segment offset value be 0 modulo 8.

--WARNING--Test completed execution, MINUS command invalid.
--WARNING--Test completed execution, PLUS command invalid.

Description: The utility cannot process the command because the test has already completed execution.

--WARNING--Test {test_identifier} in process of loading; check status display.

Description: The utility cannot process the command because the test has not completed loading. Do not enter test commands until loading is complete.

SET_PARAMETER_WORD_ZERO

--ERROR--Illegal command; cause: BYPASS_ALL_MESSAGES invalid.
--ERROR--Illegal command; cause: DISPLAY_ERROR_MESSAGES invalid.
--ERROR--Illegal command; cause: LOG_ERRORS invalid.
--ERROR--Illegal command; cause: QUICK_LOOK invalid.
--ERROR--Illegal command; cause: QUICK_LOOK_OMIT invalid.
--ERROR--Illegal command; cause: REPEAT_CONDITION invalid.
--ERROR--Illegal command; cause: REPEAT_SECTION invalid.
--ERROR--Illegal command; cause: REPEAT_SUBSECTION invalid.
--ERROR--Illegal command; cause: REPEAT_TEST invalid.
--ERROR--Illegal command; cause: STOP_CONDITION invalid.
--ERROR--Illegal command; cause: STOP_END_TEST invalid.
--ERROR--Illegal command; cause: STOP_ON_ERROR invalid.
--ERROR--Illegal command; cause: STOP_SECTION invalid.
--ERROR--Illegal command; cause: STOP_SUBSECTION invalid.

Description: The utility cannot process the command because the test does not support the requested operation.

--ERROR--Illegal command; cause: test_identifier missing.

Description: The utility cannot process the command because the TEST_IDENTIFIER is missing.

--WARNING--Test completed execution, SET_PARAMETER_WORD_ZERO command invalid.

Description: The utility cannot process the command because the test has already completed execution.

--WARNING--Test {test_identifier} in process of loading; check status display.

Description: The utility cannot process the command because the test has not completed loading. Do not enter test commands until loading is complete.

RESTART_TEST

--ERROR--Illegal command; cause: RESTART_TEST invalid.

Description: The command has been specified for a test that does not support the restart function.

--ERROR--No test response from: {test_identifier}.

Description: The test has not responded after the RESTART_TEST command within the time-out period.

--WARNING--Test completed execution, RESTART_TEST command invalid.

Description: The utility cannot process the command because the test has already completed execution.

--WARNING--Test {test_identifier} in process of loading; check status display.

Description: The utility cannot process the command because the test has not completed loading. Do not enter test commands until loading is complete.

CONTINUE_TEST

--ERROR--Illegal command; cause: CONTINUE_TEST invalid.

Description: The command has been specified for a test that does not support the continue function.

--ERROR--No test response from: {test_identifier}.

Description: The test has not responded after the CONTINUE_TEST command within the time-out period.

--WARNING--Test completed execution, CONTINUE_TEST command invalid.

Description: The utility cannot process the command because the test has already completed execution.

--WARNING--Test {test_identifier} in process of loading; check status display.

Description: The utility cannot process the command because the test has not completed loading. Do not enter test commands until loading is complete.

STOP_TEST

--ERROR--Illegal command; cause: STOP_TEST invalid.

Description: The command has been specified for a test that does not support the stop function.

--ERROR--No test response from: {test_identifier}.

Description: The test has not responded after the STOP_TEST command within the time-out period.

--WARNING--Test completed execution, STOP_TEST command invalid.

Description: The utility cannot process the command because the test has already completed execution.

--WARNING--Test {test_identifier} in process of loading; check status display.

Description: The utility cannot process the command because the test has not completed loading. Do not enter test commands until loading is complete.

SET_SKIP_PASS_COUNT

--ERROR--Illegal command; cause: SET_SKIP_PASS_COUNT invalid.

Description: The command has been specified for a test that does not support the skip pass count function.

--WARNING--Test completed execution, SET_SKIP_PASS_COUNT command invalid.

Description: The utility cannot process the command because the test has already completed execution.

--WARNING--Test {test_identifier} in process of loading; check status display.

Description: The utility cannot process the command because the test has not completed loading. Do not enter test commands until loading is complete.

SET_PIT_VALUE

--ERROR--Illegal command; cause: SET_PIT_VALUE invalid.

Description: The command has been specified for a test that does not support the set pit value function.

--WARNING--Test completed execution, SET_PIT_VALUE command invalid.

Description: The utility cannot process the command because the test has already completed execution.

--WARNING--Test {test_identifier} in process of loading; check status display.

Description: The utility cannot process the command because the test has not completed loading. Do not enter test commands until loading is complete.

SET_REPEAT_COUNT

--ERROR--Illegal command; cause: SET_REPEAT_COUNT invalid.

Description: The command has been specified for a test that does not support the repeat count function.

--WARNING--Test completed execution, SET_REPEAT_COUNT command invalid.

Description: The utility cannot process the command because the test has already completed execution.

--WARNING--Test {test_identifier} in process of loading; check status display.

Description: The utility cannot process the command because the test has not completed loading. Do not enter test commands until loading is complete.

Test Commands

--ERROR--DATA PATTERN to be defined by user was omitted.

Description: The USER_DEFINED parameter is set to TRUE, but the USER_PATTERN parameter has not been specified.

--ERROR--Illegal command; cause: {command name} invalid.

Description: The command has been specified for a test that does not support the requested function.

--ERROR--Invalid character in user data pattern; characters must be 0..9 or A..F.

Description: The USER_PATTERN string contains an invalid character; it was not one of 0..9 or A..F.

--ERROR--Invalid density; valid values are 800, 1600, and 6250.

Description: Density value specified is not one of 800, 1600, or 6250 cpi.

--ERROR--Invalid hex character found in pw10 of the test's pw block.

Description: The stored user data pattern contains a digit that is not in the range 0..9 or A..F.

--ERROR--Invalid KEYWORD value for the INSTRUCTION_LIST_SELECTION parameter.

Description: The utility detects an invalid keyword specification.

--ERROR--Invalid record size; must be a multiple of 2048.

Description: The test requires that the record size value be 0 modulo 2048.

--ERROR--No test response: {test identifier}.

Description: The test has not responded to the command within a time-out period.

--ERROR--Test parameter error; cause: only one type of data pattern may be specified.

Description: Only one data pattern may be selected for a SET_DATA_PATTERN command request.

--ERROR--Test parameter error; cause: optional list too long for {command name} command.

Description: The command cannot add the requested number of op codes to the optional instruction list.

--ERROR--Test parameter error; cause: two-digit op code required.

Description: The command expects a two-digit op code to be supplied.

--WARNING--Test completed execution, {command name} command invalid.

Description: The utility cannot process the command because the test has already completed execution.

--WARNING--Test {test_identifier} in process of loading; check status display.

Description: The utility cannot process the command because the test has not completed loading. Do not enter commands until loading is complete.

DISK Test

--ERROR--Invalid data pattern code.

Description: The data pattern code found in the test's parameter block is not a valid data pattern selection.

--ERROR--Invalid display code found in pw6 of parameter block.

Description: The display code found in parameter word six of the test's parameter block is not a valid display selection.

--ERROR--Invalid hex character found in pw10 of the DISK's pw block.

Description: The stored user data pattern contains a digit that is not in the range 0..9 or A..F.

--FATAL--DVS internal error; cause: data item not selected for format buffer.

Description: One of the displayable data items contains an unknown value type that cannot be formatted for output.

TAPE Test

--ERROR--Invalid data pattern code.

Description: The data pattern code found in the test's parameter block is not a valid data pattern selection.

--ERROR--Invalid display code found in pw6 of parameter block.

Description: The display code found in parameter word six of the test's parameter block is not a valid display selection.

--ERROR--Invalid hex character found in pw10 of the TAPE's pw block.

Description: The stored user data pattern contains a digit that is not in the range 0..9 or A..F.

--FATAL--DVS internal error; cause: data item not selected for format buffer.

Description: One of the displayable data items contains an unknown value type that cannot be formatted for output.

Comments (continued from other side)

Please fold on dotted line;
seal edges with tape only.

FOLD

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
First-Class Mail Permit No. 4760 St. Paul, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CONTROL DATA
Technical Publications ARH219
4201 Lexington Avenue N.
St. Paul, MN 55126-9983

Diagnostic Virtual System (DVS) for NOS/VE 60469720 M

We would like your comments on this manual to help us improve it. Please take a few minutes to fill out this form.

Who are you?

- ☐ Manager
- ☐ Systems analyst or programmer
- ☐ Applications programmer
- ☐ Operator
- ☐ Other _____

How do you use this manual?

- ☐ As an overview
- ☐ To learn the product or system
- ☐ For comprehensive reference
- ☐ For quick look-up
- ☐ Other _____

What programming languages do you use? _____

_____

How do you like this manual? Answer the questions that apply.

Yes	Somewhat	No	
☐	☐	☐	Does it tell you what you need to know about the topic?
☐	☐	☐	Is the technical information accurate?
☐	☐	☐	Is it easy to understand?
☐	☐	☐	Is the order of topics logical?
☐	☐	☐	Can you easily find what you want?
☐	☐	☐	Are there enough examples?
☐	☐	☐	Are the examples helpful? (☐ Too simple? ☐ Too complex?)
☐	☐	☐	Do the illustrations help you?
☐	☐	☐	Is the manual easy to read (print size, page layout, and so on)?
☐	☐	☐	Do you use this manual frequently?

Comments? If applicable, note page and paragraph. Use other side if needed.

Check here if you want a reply: ☐

Name _____ Company _____

Address _____ Date _____

_____ Phone _____

_____

Please send program listing and output if applicable to your comment.