

VMS

digital

VMS System Dump Analyzer Utility Manual

Order Number AA-LA87A-TE

VMS System Dump Analyzer Utility Manual

Order Number: AA-LA87A-TE

April 1988

This manual explains how to use the System Dump Analyzer (SDA) to investigate system failures and examine a running system.

Revision/Update Information: This book supersedes the *VAX/VMS System Dump Analyzer Reference Manual* for VAX/VMS Version 4.4, published April 1986.

Operating System and Version: VMS Version 5.0

**digital equipment corporation
maynard, massachusetts**

April 1988

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Copyright ©1988 by Digital Equipment Corporation

All Rights Reserved.
Printed in U.S.A.

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem-10	PDP	VT
DECSYSTEM-20	PDT	
DECUS	RSTS	
DECwriter	RSX	digital ™

ZK4556

**HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS**

USA & PUERTO RICO*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire
03061

CANADA

Digital Equipment
of Canada Ltd.
100 Herzberg Road
Kanata, Ontario K2K 2A6
Attn: Direct Order Desk

INTERNATIONAL

Digital Equipment Corporation
PSG Business Manager
c/o Digital's local subsidiary
or approved distributor

In Continental USA and Puerto Rico call 800-258-1710.

In New Hampshire, Alaska, and Hawaii call 603-884-6660.

In Canada call 800-267-6215.

* Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575). Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminister, Massachusetts 01473.

Production Note

This book was produced with the VAX DOCUMENT electronic publishing system, a software tool developed and sold by DIGITAL. In this system, writers use an ASCII text editor to create source files containing text and English-like code; this code labels the structural elements of the document, such as chapters, paragraphs, and tables. The VAX DOCUMENT software, which runs on the VMS operating system, interprets the code to format the text, generate a table of contents and index, and paginate the entire document. Writers can print the document on the terminal or line printer, or they can use DIGITAL-supported devices, such as the LN03 laser printer and PostScript[™] printers (PrintServer 40 or LN03R ScriptPrinter), to produce a typeset-quality copy containing integrated graphics.

[™] PostScript is a trademark of Adobe Systems, Inc.

Contents

PREFACE	ix
NEW AND CHANGED FEATURES	xiii
SDA Description	SDA-1
1	SYSTEM MANAGEMENT AND SDA SDA-2
1.1	The System Dump File SDA-2
1.1.1	Dump File Style • SDA-4
1.2	Saving System Dumps SDA-4
1.3	Invoking SDA in the Site-Specific Startup Command Procedure SDA-5
2	ANALYZING A SYSTEM DUMP SDA-6
2.1	Requirements SDA-6
2.2	Invoking SDA SDA-6
2.3	Mapping the Contents of the Dump File SDA-7
2.4	Building the SDA Symbol Table SDA-7
2.5	Executing the SDA Initialization File (SDA\$INIT) SDA-8
3	ANALYZING A RUNNING SYSTEM SDA-8
4	SDA CONTEXT SDA-9
5	SDA COMMAND FORMAT SDA-10
5.1	General Command Format SDA-11
5.2	Expressions SDA-11
5.2.1	Radix Operators • SDA-12
5.2.2	Arithmetic and Logical Operators • SDA-12
5.2.3	Precedence Operators • SDA-13
5.2.4	Symbols • SDA-13
6	INVESTIGATING SYSTEM FAILURES SDA-15
6.1	General Procedure for Analyzing System Failures SDA-15
6.2	Fatal Bugcheck Conditions SDA-16
6.2.1	Fatal Exceptions • SDA-16
6.2.2	Illegal Page Faults • SDA-19

Contents

7	A SAMPLE SYSTEM FAILURE	SDA-21
7.1	Identifying the Bugcheck _____	SDA-21
7.2	Identifying the Exception _____	SDA-21
7.3	Locating the Source of the Exception _____	SDA-23
7.3.1	Finding the Driver by Using the Program Counter • SDA-23	
7.3.2	Calculating the Offset into the Driver's Program Section • SDA-24	
7.4	Finding the Problem Within the Routine _____	SDA-24
7.4.1	Examining the Routine • SDA-25	
7.4.2	Checking the Values of Key Variables • SDA-26	
7.4.3	Identifying and Fixing the Defective Code • SDA-27	
<hr/>		
8	INDUCING A SYSTEM FAILURE	SDA-28
8.1	Meeting Crash Dump Requirements _____	SDA-28
8.2	Examples of How to Cause System Failures _____	SDA-29
<hr/>		
SDA Usage Summary		SDA-32
<hr/>		
SDA Qualifiers		SDA-34
	/CRASH_DUMP	SDA-35
	/RELEASE	SDA-36
	/SYMBOL	SDA-37
	/SYSTEM	SDA-38
<hr/>		
SDA Commands		SDA-39
	@ (EXECUTE PROCEDURE)	SDA-40
	ATTACH	SDA-41
	COPY	SDA-42
	DEFINE	SDA-43
	EVALUATE	SDA-48
	EXAMINE	SDA-51
	EXIT	SDA-55
	FORMAT	SDA-56
	HELP	SDA-58
	READ	SDA-59
	REPEAT	SDA-64
	SEARCH	SDA-66
	SET CPU	SDA-68
	SET LOG	SDA-71
	SET OUTPUT	SDA-72
	SET PROCESS	SDA-73
	SET RMS	SDA-76
	SHOW CALL_FRAME	SDA-79
	SHOW CLUSTER	SDA-82

SHOW CONNECTIONS	SDA-87
SHOW CPU	SDA-89
SHOW CRASH	SDA-93
SHOW DEVICE	SDA-98
SHOW EXECUTIVE	SDA-104
SHOW HEADER	SDA-106
SHOW LOCK	SDA-108
SHOW PAGE_TABLE	SDA-111
SHOW PFN_DATA	SDA-115
SHOW POOL	SDA-118
SHOW PORTS	SDA-123
SHOW PROCESS	SDA-126
SHOW RESOURCE	SDA-143
SHOW RMS	SDA-147
SHOW RSPID	SDA-148
SHOW SPINLOCKS	SDA-150
SHOW STACK	SDA-157
SHOW SUMMARY	SDA-159
SHOW SYMBOL	SDA-161
SPAWN	SDA-162
VALIDATE QUEUE	SDA-164

INDEX

FIGURES

SDA-1	First Argument List on the Stack _____	SDA-17
SDA-2	Mechanism Array _____	SDA-17
SDA-3	Signal Array _____	SDA-18
SDA-4	Stack Following an Illegal Page-Fault Error _____	SDA-20
SDA-5	Call Frame _____	SDA-80

TABLES

SDA-1	Comparison of Full and Subset Dump Files _____	SDA-4
SDA-2	SDA Operators _____	SDA-12
SDA-3	SDA Symbols _____	SDA-13
SDA-4	Modules Containing Global Symbols Used by SDA _____	SDA-60
SDA-5	Modules Defining Global Locations Within the Executive Image _____	SDA-60
SDA-6	SET RMS Command Keywords for Displaying Process RMS Information _____	SDA-76

Contents

SDA-7	Contents of the SHOW LOCK and SHOW PROCESS/LOCKS Displays _____	SDA-108
SDA-8	Virtual Page Information in the SHOW PAGE_TABLE Display _____	SDA-112
SDA-9	Physical Page Information in the SHOW PAGE_TABLE Display _____	SDA-113
SDA-10	Page Frame Number Information in the SHOW PFN_DATA Display _____	SDA-116
SDA-11	Process Section Table Entry Information in the SHOW PROCESS Display _____	SDA-130
SDA-12	Process I/O Channel Information in the SHOW PROCESS Display _____	SDA-131
SDA-13	Resource Information in the SHOW RESOURCE Display _	SDA-143
SDA-14	Static Spin Locks _____	SDA-151
SDA-15	Process Information in the SHOW SUMMARY Display ____	SDA-159

Preface

Intended Audience

The *VMS System Dump Analyzer Utility Manual* is primarily intended for the system programmer who must investigate the causes of system failures and debug kernel mode code, such as a device driver. This programmer should possess some knowledge of VMS data structures to properly interpret the results of System Dump Analyzer (SDA) commands.

The *VMS System Dump Analyzer Utility Manual* also includes information required by the system manager in order to maintain the system resources necessary to capture and store system crash dumps. Those who need to determine the cause of a hung process or improve system performance may refer to this manual for instructions for using SDA to analyze a running system.

Document Structure

The *VMS System Dump Analyzer Utility Manual* includes the following four sections:

- The first section provides an introduction to the functions of the System Dump Analyzer (SDA), a description of its features, a discussion of key concepts, and an illustration of its use. It includes instructions for maintaining the optimal environment for the analysis of system failures, and notes the requirements for processes invoking SDA.
- The second section outlines the following aspects of SDA usage:
 - Invoking SDA
 - Exiting from SDA
 - Recording the output of an SDA session
 - Required privileges
- The third section describes those qualifiers to the ANALYZE command that govern the behavior of SDA.
- The last section describes the function, format, and parameters of each SDA command. It also provides examples of situations in which specific commands are useful.

Associated Documents

The *VMS System Dump Analyzer Utility Manual* presumes an understanding of the material discussed in the following documents:

- VMS naming conventions as described in the *Guide to Creating VMS Modular Procedures*
- VMS operational concepts as described in the *Introduction to VMS System Management*, *Guide to Setting Up a VMS System*, *Guide to Maintaining a VMS System*, and *Guide to VMS Performance Management*
- VMS data structures and concepts as described in the *VAX/VMS Internals and Data Structures* and the *VMS Device Support Manual*

Investigators of VAXcluster failures will find the discussions in the *VMS VAXcluster Manual* and the *VMS Show Cluster Utility Manual* helpful in understanding the output of several SDA commands.

Conventions

Typographical conventions used in this book include the following:

- The term “quotation marks” refers to double quotation marks (“”). The term “apostrophe” refers to a single quotation mark (’).
- Terms that serve as parameters to commands and qualifiers appear in boldface in the text of the manual. For example:

The value **csid** is the cluster system identification number (CSID) of the node to be displayed.

- Terms that serve as variables in a mathematical expression or a file specification appear in italic print:

The default file specification is as follows:

`SYS$DISK:[default-dir]SYSDUMP.DMP`

`SYS$DISK` and `[default-dir]` represent the disk and directory specified in your last SET DEFAULT command.

- In format descriptions, brackets indicate that the enclosed item is optional. (Brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.)

FORMAT[/*qualifier*] *location*

- In format descriptions, stacked lists of items are enclosed in either braces or brackets. In either case, the parameters or qualifiers in the stack are optional in the syntax of the command.

When *braces* enclose the list, you can include *only one* of the items from the list in the command. For example:

EVALUATE $\left\{ \begin{array}{l} /CONDITION_VALUE \\ /PSL \\ /PTE \\ /SYMBOLS \end{array} \right\}$ *expression*

When *brackets* enclose the list, you can generally include *more than one* item from the stack in the command. Incompatible qualifiers and parameters are given special mention in the syntax descriptions.

SHOW POOL $\left[\begin{array}{l} /FREE \\ /HEADER \\ /SUMMARY \\ /TYPE=block-type \end{array} \right] \left[\begin{array}{l} range \\ /ALL \\ /IRP \\ /LRP \\ /NONPAGED \\ /PAGED \\ /SRP \end{array} \right]$

- In format descriptions, a horizontal ellipsis indicates that additional parameters, values, or information can be entered. For example:

SET RMS = (option[,...])

In examples, a horizontal ellipsis also indicates that columns of information have been omitted from the display. For example:

```
SDA> SHOW HEADER
Dump file header
-----
7FF03944 . . . 00000000 . . . . . N...D9.. 00000000
00000000 . . . 80185200 .R. . . . . 00000020
00000000 . . . 00000000 . . . . . 00000040
00020000 . . . 00000000 . . . . . 00000060
```

- In format descriptions, parentheses indicate that you should enclose the choices you select in parentheses if you select more than one. For example:

SET RMS = (option[,...])

- In interactive examples, all output lines or prompting characters that the system prints or displays appear in black letters. All user-entered commands are shown in red letters. For example:

```
$ ANALYZE/CRASH_DUMP SYS$SYSTEM:SYSDUMP.DMP
$ ANAL/CRASH SYS$SYSTEM
```

- In interactive examples, a symbol with a 1- to 6-character abbreviation indicates that you press a key on the terminal, for example, **RET**.
- In interactive examples, the symbol **CTRL/x** indicates that you must press the key labeled CTRL while you simultaneously press another key: for instance, **CTRL/C**.
- In interactive examples, a vertical ellipsis means either that not all the data that the system would display in response to the particular command is shown or that not all the data a user would enter is shown. For example:

```
SDA> READ SYS$SYSTEM:SYSDEF.STB
SDA> FORMAT 800B81F0
800B81F0   UCB$L_FQFL           80000F10
           UCB$L_RQFL
           .
           .
           .
           UCB$L_FIRST           8002CA00
```

New and Changed Features

This manual applies to Version 5.0 of the VMS operating system.

The following list summarizes the major changes to the previous edition of the manual.

- The context of certain SDA commands now depends on the definition of both the SDA current process and the SDA current CPU. The descriptions of the following commands have been altered to reflect the behavior of SDA in analyzing a Version 5.0 multiprocessor system failure:
 - SET PROCESS
 - SHOW CRASH
 - SHOW PROCESS
 - SHOW STACK

The following new commands have been added to the manual:

SET CPU	Selects a processor to become the SDA current CPU
SHOW CPU	Displays information about the state of a processor at the time of the system failure
SHOW SPINLOCKS	Displays the data structures that provide system synchronization in a VAX multiprocessing system

Section 4 discusses the implications of changing SDA CPU and process context.

- Version 5.0 changes in the composition of the VMS executive and its corresponding loaded images may be evident in the names of certain symbols displayed in the output of various commands (such as SHOW STACK) in an SDA session. To aid in identifying locations within the executive image, SDA provides the following features:
 - A new command, SHOW EXECUTIVE, lists the location and size of each loadable image that is part of the Version 5.0 VMS executive.
 - The /EXECUTIVE qualifier has been added to the READ command to facilitate the loading of symbols into the SDA symbol table to further identify global locations within the VMS executive.

The description of the SHOW EXECUTIVE command includes a discussion of the components of the Version 5.0 VMS executive.

- Several options have been added to the SET RMS and SHOW PROCESS/RMS commands, allowing the display of the following RMS structures: recovery unit block (RUB), recovery unit stream block (RUSB), recovery unit file block (RUFB), shared file synchronization block (SFSB), global buffer synchronization block (GFSB), and network work area (NWA). These commands also allow the display of the structures associated with process I/O by means of the [NO]PIO option.
- When invoked, SDA reads into its symbol table a subset of SYS\$SYSTEM:SYSDEF.STB, called SYS\$SYSTEM:REQSYSDEF.STB, that it requires to identify certain locations in memory. SDA reads REQSYSDEF.STB in addition to reading the system symbol table

New and Changed Features

(SYS\$SYSTEM:SYS.STB). See Section 2.4 for a description of this procedure.

- The symbol MP and the symbol table file (MP.STB) are no longer available to SDA in Version 5.0.
- Instructions on inducing a system failure on newer VAX processors have been added to Section 8.
- A new system parameter, DUMPSTYLE, when set, allows a system whose memory capacity exceeds the disk space available for storing crash dumps to preserve a subset of memory contents for analysis by SDA. (See the discussion in Section 1.1.1 for a discussion of this feature.)
- Various minor revisions, as well as some reorganization of material, may be apparent throughout the manual. In addition, some examples have been added and others have been corrected to better reflect the operation of SDA under VMS Version 5.0.

SDA Description

When a fatal error causes the system to fail, the VMS operating system copies the contents of memory to a system dump file, recording the hardware context of each processor in the system as well. The System Dump Analyzer (SDA) provides a means of interpreting the contents of this file, thus enabling you to examine the status of each processor at the time of the failure and investigate the probable causes of the crash. To do so, you invoke SDA by means of the DCL command ANALYZE/CRASH_DUMP.

You can use SDA commands to perform the following operations:

- Direct (or echo) the output of an SDA session to a file or device (SET OUTPUT or SET LOG)
- Display the condition of the operating system and the hardware context of each processor in the system at the time of the crash (SHOW CRASH)
- Select a specific processor in a multiprocessing system as the subject of analysis (SET CPU).
- Display the contents of a specific process stack or the interrupt stack of a specific processor (SHOW STACK)
- Format a call frame from a stack location (SHOW CALL_FRAME)
- Read a set of global symbols into the SDA symbol table (READ)
- Define symbols to represent values or locations in memory and add them to the SDA symbol table (DEFINE)
- Evaluate an expression in hexadecimal and decimal, interpreting its value as a symbol, a condition value, a page-table entry (PTE), or a processor status longword (PSL) (EVALUATE)
- Examine the contents of memory locations, optionally interpreting them as VAX MACRO instructions, a PTE, or a PSL (EXAMINE)
- Display device status as reflected in system data structures (SHOW DEVICE)
- Format system data structures (FORMAT)
- Validate the integrity of the links in a queue (VALIDATE)
- Display a summary of all processes on the system (SHOW SUMMARY)
- Examine the memory of any process (SHOW PROCESS)
- Display the RMS data structures of a process (SHOW PROCESS/RMS)
- Display memory management data structures (SHOW POOL, SHOW PFN_DATA, SHOW PAGE_TABLE)
- Display lock management data structures (SHOW RESOURCE, SHOW LOCK)
- Display VAXcluster management data structures (SHOW CLUSTER, SHOW CONNECTIONS, SHOW RSPID, SHOW PORTS)
- Display multiprocessor synchronization information (SHOW SPINLOCKS)

SDA Description

- Display the layout of the loadable executive images (SHOW EXECUTIVE)
- Copy the system dump file (COPY)
- Define keys to invoke SDA commands (DEFINE/KEY)
- Search memory for a given value (SEARCH)

Although SDA provides a great deal of information, it does not analyze all the control blocks and data contained in memory. For this reason, in the event of system failure it is extremely important that you send DIGITAL a Software Performance Report (SPR) and a copy of the system dump file written at the time of the failure.

You can also invoke SDA to analyze a running system, using the DCL command ANALYZE/SYSTEM. Most SDA commands generate useful output in this mode of operation. Although the analysis of a running system may be instructive, you should undertake such an operation with the caution that system context, process context, and a processor's hardware context remain fluid during any given display. A user in a multiprocessing environment should especially note that it is highly possible that a process running SDA could be rescheduled to a different processor frequently during analysis. It is thus advisable to avoid the examination of the hardware context of processors in a running system.

1 System Management and SDA

The system manager must ensure that the system writes a dump file whenever the system fails. The manager must also see that the dump file is large enough to contain all the information to be saved, and that the dump file is saved for analysis. The following sections describe these tasks.

1.1 The System Dump File

The VMS operating system will attempt to write information into the system dump file only if the system parameter DUMPBUG is set.¹ If DUMPBUG is set and the operating system fails, VMS writes the contents of the error-log buffers, processor registers, and physical memory into the system dump file, overwriting its previous contents.

If the system dump file is too small, VMS cannot copy all of memory to the file when a system failure occurs. For most systems, this means that the system's page table (SPT) is not included in the dump. SDA cannot analyze a dump unless the SPT is included in the dump in its entirety.

The file SYS\$SYSTEM:SYSDUMP.DMP is furnished as an empty file in the VMS software distribution kit. In order to successfully store a crash dump, you must make SYS\$SYSTEM:SYSDUMP.DMP large enough to hold all the information to be written when the system fails. If this is not possible, you can have VMS write dumps into the system paging file, SYS\$SYSTEM:PAGEFILE.SYS. You can enlarge or adjust the size of either of these files by using the CREATE command of the System Generation Utility (SYSGEN), as described in the *VMS System Generation Utility Manual*.

¹ The DUMPBUG parameter is set by default. To examine and/or change its value, consult the *VMS System Generation Utility Manual*.

SDA Description

To calculate the correct size for SYS\$SYSTEM:SYSDUMP.DMP, use the following formula:

```
size-in-blocks(SYS$SYSTEM:SYSDUMP.DMP)
= size-in-pages(physical-memory)
+ number-of-error-log-buffers
+ 1
```

You can use the DCL command SHOW MEMORY to determine the total size of physical memory on your system. In addition, you must account for any MA780 multiport memory installed on your system. There are a variable number of error log buffers in any given VAX system, depending upon the setting of the ERRORLOGBUFFERS system parameter. (See the *VMS System Generation Utility Manual* for additional information about this parameter.)

If SYS\$SYSTEM:SYSDUMP.DMP does not exist, the VMS operating system writes the dump of physical memory into SYS\$SYSTEM:PAGEFILE.SYS, the system's paging file, overwriting the contents of that file. If the SAVEDUMP system parameter is set, the dump file is retained in PAGEFILE.SYS when the system is booted. If it is clear, VMS uses the entire paging file for paging and any dump written to the paging file is lost.²

To calculate the minimum size for SYS\$SYSTEM:PAGEFILE.SYS, use the following formula:

```
size-in-blocks(SYS$SYSTEM:PAGEFILE.SYS)
= size-in-pages(physical-memory)
+ number-of-error-log-buffers
+ 1
+ 1000
```

Note that this formula calculates the minimum size requirement for saving a dump in the system's paging file. Generally, the paging file must be larger than this for most systems to avoid hanging the system (see the *Guide to Setting Up a VMS System*). Use of SYS\$SYSTEM:PAGEFILE.SYS to take system crash dumps presumes that you will later free the space occupied by the dump for use by the pager. Generally, you include SDA commands in the site-specific startup command procedure (SYS\$MANAGER:SYSTARTUP.COM) that do this. Otherwise, your system may hang during the startup procedure.

A common method for doing this is to copy the dump from SYS\$SYSTEM:PAGEFILE.SYS to another file, using the SDA COPY command. Although the DCL COPY command can also be used to copy a dump file, only the SDA COPY command causes the pages occupied by the dump to be freed from the system's paging file.

Occasionally, you may want to free the pages in the paging file that are taken up by the dump without having to copy the dump elsewhere. When you issue the ANALYZE/CRASH_DUMP/RELEASE command, SDA immediately releases the pages to be used for system paging, effectively deleting the dump. Note that this command does *not* allow you to analyze the dump before deleting it.

² The SAVEDUMP parameter is clear by default. To examine and/or change its value, consult the *VMS System Generation Utility Manual*.

SDA Description

1.1.1 Dump File Style

In certain VAX system configurations, it may be impossible to preserve the entire contents of memory in a disk file. For instance, a large memory system or a system with small disk capacity may not be able to supply enough disk space for a full memory dump. In normal circumstances, if the system dump file cannot accommodate all of memory, SDA cannot analyze the dump.

To preserve those portions of memory that contain information most useful in determining the causes of system failures, a system manager sets the static system parameter DUMPSTYLE to 1. When the DUMPSTYLE parameter is set, AUTOGEN attempts to create a dump file large enough to contain ample information for SDA to analyze a failure. When the DUMPSTYLE parameter is clear, the default case, AUTOGEN attempts to create a dump file large enough to contain all of physical memory.

A comparison of full and subset style dump files appears in Table SDA-1.

Table SDA-1 Comparison of Full and Subset Dump Files

	Full	Subset
Available Information	Complete contents of physical memory in use, stored in order of increasing physical address (for instance, system and global page tables are stored last).	System page table, global page table, system space memory, and process and control regions (plus global pages) for all saved processes.
Unavailable Information	Contents of paged-out memory at the time of the crash.	Contents of paged-out memory at the time of the crash, process and control regions of unsaved processes, and memory not mapped by a page table (such as the free and modified lists).
SDA Command Limitations	None.	The following commands are not useful for unsaved processes: SHOW PROCESS /CHANNELS, SHOW PROCESS/RMS, SHOW STACK, and SHOW SUMMARY/IMAGE.

1.2 Saving System Dumps

Every time the operating system writes information to the system dump file, it writes over whatever was previously stored in the file. For this reason, the system manager should save the contents of the file after a system failure has occurred.

The system manager can use the SDA COPY command or the DCL COPY command. Either command can be used in your site-specific startup procedure, but the SDA COPY command is preferred because it marks the dump file as copied. As mentioned earlier, this is particularly important if the dump was written into the paging file, SYS\$SYSTEM:PAGEFILE.SYS, because it releases those pages occupied by the dump to the pager.

Because system dump files are set to NOBACKUP, the Backup Utility (BACKUP) does not copy them to tape unless you use the qualifier /IGNORE=NOBACKUP when invoking BACKUP. When you use the SDA COPY command to copy the system dump file to another file, VMS does not set the new file to NOBACKUP.

As included in the VMS distribution kit, SYS\$SYSTEM:SYSDUMP.DMP is protected against world access. Because a dump file can contain privileged information, it is a good idea for the system manager to continue to protect dump files from universal read access.

1.3 Invoking SDA in the Site-Specific Startup Command Procedure

Because a listing of the SDA output is an important source of information in determining the cause of a system failure, it is a good idea to have SDA produce such a listing after every failure. The system manager can ensure the creation of a listing by modifying the site-specific startup command procedure SYS\$MANAGER:SYSTARTUP.COM so that it invokes SDA when the system is booted.

When invoked in the site-specific startup procedure, SDA executes the specified commands only if the system is booting immediately after a system failure. SDA examines a flag in the dump file's header that indicates whether it has already processed the file. If the flag is set, SDA merely exits. If the flag is clear, SDA executes the specified commands and sets the flag. This flag is clear when the operating system initially writes a crash dump, except for those resulting from an operator-requested shutdown (for instance, SYS\$SYSTEM:OPCCRASH.COM or SYS\$SYSTEM:SHUTDOWN.COM).

The following example shows typical commands that might be added to your site-specific startup command procedure to produce an SDA listing after each failure.

```
$ !
$ !      Print dump listing if system just failed
$ !
$ ANALYZE/CRASH_DUMP SYS$SYSTEM:SYSDUMP.DMP
  COPY SYS$SYSTEM:SAVEDUMP.DMP      ! Save dump file
  SET OUTPUT LPAO:SYSDUMP.LIS      ! Create listing file
  SHOW CRASH                        ! Display crash
                                   ! information
  SHOW STACK                        ! Show current stack
  SHOW SUMMARY                      ! List all active
                                   ! processes
  SHOW PROCESS/PCB/PHD/REG          ! Display current process
  SHOW SYMBOL/ALL                   ! Print system symbol
                                   ! table
EXIT
```

The COPY command in the preceding example saves the contents of the file SYS\$SYSTEM:SYSDUMP.DMP. If your system's startup command file does not save a copy of the contents of this file, this crash dump information will be lost in the next system failure, when the system saves the information on the new failure, overwriting the contents of SYS\$SYSTEM:SYSDUMP.DMP.

If you are using the SYS\$SYSTEM:PAGEFILE.SYS as the crash dump file, you must include SDA commands in SYS\$MANAGER:SYSTARTUP.COM that free the space occupied by the dump so that the pager can use it. For instance:

```
$ ANALYZE/CRASH_DUMP SYS$SYSTEM:PAGEFILE.SYS
.
.
.
COPY dump_filespec
EXIT
```

SDA Description

2 Analyzing a System Dump

SDA performs certain tasks prior to bringing a dump into memory, presenting its initial displays, and accepting command input. This section describes those tasks, which include

- Verifying that the process invoking it is suitably privileged to read the dump file
- Mapping the contents of the specified dump file
- Reading the system symbol tables (SYS\$SYSTEM:SYS.STB and SYS\$SYSTEM:REQSYSDEF.STB)
- Executing the commands in the SDA initialization file

For detailed information on the investigation of a system failure, see Section 6.

2.1 Requirements

In order to be able to analyze a dump file, your process must have

- *Read access* both to the file that contains the dump and to copies of the symbol tables SYS\$SYSTEM:SYS.STB (the system symbol table) and SYS\$SYSTEM:REQSYSDEF.STB (the required subset of the symbols in the file SYSDEF.STB). SDA reads these tables by default. As included in the VMS distribution kit, SYS\$SYSTEM:SYSDUMP.DMP, SYS\$SYSTEM:SYS.STB, and SYS\$SYSTEM:REQSYSDEF.STB are protected against world access. Either a system UIC or SYSPRV privilege thus is needed for a process to read the dump file.
- *Sufficient virtual address space* for SDA to map the entire dump and any required symbol tables, plus space to be used for the stacks. To ensure that SDA has the correct amount of virtual address space, the value of the system parameter VIRTUALPAGECNT must be larger than the size of the system's dump file by approximately 3000 pages. Further increases in the parameter may be required if your particular installation places extra heavy demands upon the virtual address space of the process.

2.2 Invoking SDA

If your process satisfies these conditions, you can issue the DCL command ANALYZE/CRASH_DUMP to invoke SDA. If you do not specify the name of a dump file in the command, SDA prompts you for the name of the file, as follows:

```
$ ANALYZE/CRASH_DUMP
_Dump File:
```

The default file specification is as follows:

```
SYS$DISK:[default-dir]SYSDUMP.DMP
```

SYS\$DISK and [default-dir] represent the disk and directory specified in your last SET DEFAULT command.

2.3 Mapping the Contents of the Dump File

SDA first attempts to map the contents of physical memory as stored in the specified dump file. To do this, it must first locate the system page table (SPT) among its contents. The SPT contains one entry for each page of system virtual address space.

The SPT appears at the largest physical addresses in a typical VMS configuration. As a result, if a dump file is too small, the SPT cannot be written to it in the event of system failure.

If SDA cannot find the SPT in the dump file, it displays either of the following messages:

`%SDA-E-SPTNOTFND`, system page table not found in dump file

`%SDA-E-SHORTDUMP`, the dump only contains m out of n pages of physical memory

If either of these error messages is displayed, you cannot analyze the crash dump, but must take steps to ensure that any subsequent dump can be preserved. To do this, you must increase the size of the dump file as indicated in Section 1.1 or adjust the system `DUMP_STYLE` parameter as discussed in Section 1.1.1.

Under certain conditions, some memory locations might not be saved in the system dump file. For instance, during halt/restart bugchecks, the contents of general registers are not preserved. If such a bugcheck occurs, SDA indicates in the `SHOW CRASH` display that the contents of the registers were destroyed. Additionally, if a bugcheck occurs during system initialization, the contents of the register display may be unreliable. The symptom of such a bugcheck is a `SHOW SUMMARY` display that shows no processes or only the swapper process.

Also, you should note that if you use an SDA command to access a virtual address that has no corresponding physical address, SDA generates the following error message:

`%SDA-E-NOTINPHYS`, 'location' not in physical memory

When analyzing a subset dump file, if you use an SDA command to access a virtual address that has a corresponding physical address but was not saved in the dump file, SDA generates the following error message:

`%SDA-E-MEMNOTSVD`, memory not saved in the dump file

2.4 Building the SDA Symbol Table

After locating and reading the system dump file, SDA attempts to read the system symbol table file into the SDA symbol table. This file, named `SYS$SYSTEM:SYS.STB` by default, contains most of the global symbols used by the VMS operating system. If SDA cannot find the system symbol table file—or is given a file that is not a system symbol table in the `/SYMBOL` qualifier to the `ANALYZE` command—it halts with a fatal error. SDA also reads into its symbol table a subset of `SYS$SYSTEM:SYSDEF.STB`, called `SYS$SYSTEM:REQSYSDEF.STB`, that it requires to identify locations in memory.

SDA Description

When SDA finishes building its symbol table, it displays a message identifying itself and the immediate cause of the crash. In the following example, the cause of the crash was an illegal exception occurring at an IPL above IPL\$_ASTDEL or while using the interrupt stack.

```
VAX/VMS System dump analyzer
```

```
Dump taken on 28-Jan-1989 18:10:09.79
```

```
INVEXCEPTN, Exception while above ASTDEL or on interrupt stack
```

2.5 Executing the SDA Initialization File (SDA\$INIT)

After displaying the crash summary, SDA executes the commands in the SDA initialization file, if you have established one. SDA refers to its initialization file by using the logical name SDA\$INIT. If SDA cannot find the file defined as SDA\$INIT, it searches for the file SYS\$LOGIN:SDA.INIT.

This initialization file can contain SDA commands that read symbols into SDA's symbol table, define keys, establish a log of SDA commands and output, or perform other tasks. For instance, you may want to use an SDA initialization file to augment SDA's symbol table with definitions helpful in locating system code. If you issue the following command, SDA includes those symbols that define many of the system's data structures, including those in the I/O database.

```
READ SYS$SYSTEM:SYSDEF.STB
```

You may also find it very helpful to define those symbols that identify the modules in the images that make up the VMS executive. You can do this by issuing the following command:

```
READ/EXECUTIVE SYS$LOADABLE_IMAGES
```

After SDA has executed the commands in the initialization file, it displays its prompt, as follows:

```
SDA>
```

The SDA> prompt indicates that you can use SDA interactively and enter SDA commands.

3 Analyzing a Running System

Occasionally VMS encounters an internal problem that hinders system performance without causing a system failure. By allowing you to examine the running system, SDA provides the means to search for the solution to the problem without disturbing the operating system. For example, you can use SDA to examine the stack and memory of a process that is stalled in a scheduler state, such as a miscellaneous wait (MWAIT) or a suspended (SUSP) state (see the *Guide to VMS Performance Management*).

If your process has change-mode-to-kernel (CMKRNL) privilege, you can invoke SDA to examine the system. Use the following DCL command:

```
$ ANALYZE/SYSTEM
```

SDA attempts to load the system symbol table (SYS\$SYSTEM:SYS.STB) and symbol table SYS\$SYSTEM:REQSYSDEF.STB. It then executes the contents of any existing SDA initialization file, as it does when invoked to analyze a crash dump (see Sections 2.4 and 2.5, respectively). SDA subsequently displays its identification message and prompt, as follows:

```
VAX/VMS System analyzer
```

```
SDA>
```

The *SDA>* prompt indicates that you can use SDA interactively and enter SDA commands. When analyzing a running system, SDA sets its process context to that of the process running SDA.

If you are undertaking an analysis of a running system, you should take into account the following considerations:

- When used in this mode, SDA does not map the entire system, but instead retrieves only the information it needs to process each individual command. To update any given display, you must reissue the previous command. When using SDA to analyze a running system, you should thus use caution in interpreting its displays. Because system states change frequently, it is possible that the information SDA displays may be inconsistent with the actual, volatile state of the system at any moment.
- Certain SDA commands are illegal in this mode, such as SHOW CPU and SET CPU. Use of these commands results in the error message

```
%SDA-E-CMDNOTVLD, command not valid on the running system
```
- The SHOW CRASH command, although valid, does not display the contents of any of the processor's set of hardware registers. Also, the "Time of system crash" information refers to the time at which the ANALYZE/SYSTEM command was given.

4 SDA Context

When invoked to analyze either a crash dump or a running system, SDA establishes a default context for itself from which it interprets certain commands.

When the subject of analysis is a VMS uniprocessor system, SDA's context is solely *process context*. That is, SDA can interpret its process-specific commands in the context of either the process current on the uniprocessor or some other process in some other scheduling state. When you initially invoke SDA to analyze a crash dump, its process context defaults to that of the process that was current at the time of the crash. When you invoke SDA to analyze a running system, its process context defaults to that of the current process; that is, the one executing SDA. You can change SDA's process context by issuing any of the following commands:

```
SET PROCESS/INDEX=nn  
SET PROCESS name  
SHOW PROCESS/INDEX=nn
```

SDA Description

When you invoke SDA to analyze a crash dump from a VMS multiprocessing system with more than one active CPU, SDA maintains a second dimension of context—its *CPU context*—that allows it to display certain processor-specific information, such as the reason for the bugcheck exception, the currently executing process, the current IPL, the contents of processor-specific registers, the interrupt stack pointer (ISP), and the spin locks owned by the processor. When you invoke SDA to analyze a multiprocessor's crash dump, its CPU context defaults to that of the processor that induced the system failure.³

You can change the SDA CPU context by using any of the following commands:

```
SET CPU cpu-id  
SHOW CPU cpu-id  
SHOW CRASH
```

Changing CPU context involves an implicit change in process context in either of the following ways:

- If there is a current process on the CPU made current, SDA process context is changed to that of that CPU's current process.
- If there is no current process on the CPU made current, SDA process context is undefined and no process-specific information is available until SDA process context is set to that of a specific process.

Likewise, changing process context can involve a switch of CPU context as well. For instance, if you issue a SET PROCESS command for a process that is current on another CPU, SDA will automatically change its CPU context to that of the CPU on which that process is current. The following commands can have this effect if the **name** or index number (**nn**) refers to a current process.

```
SET PROCESS name  
SET PROCESS/INDEX=nn  
SHOW PROCESS name  
SHOW PROCESS/INDEX=nn
```

5

SDA Command Format

The following sections describe the format of SDA commands and the expressions you can use with SDA commands.

³ When you are analyzing a running system, CPU context is not accessible to SDA. Therefore, the SET CPU and SHOW CPU commands are not permitted.

5.1 General Command Format

SDA uses a command format similar to that used by the DCL interpreter. You issue commands in this general format:

```
command-name[/qualifier...] [parameter][[/qualifier...]] [!comment]
```

The **command-name** is an SDA command. Each command tells the utility to perform a function. Commands can consist of one or more words, and can be abbreviated to the number of characters that make the command unique. For example, SH stands for SHOW, and SE stands for SET.

The **parameter** is the target of the command. For example, SHOW PROCESS RUSKIN tells SDA to display the context of the process RUSKIN. The command EXAMINE 80104CD0;40 displays the contents of 40 bytes of memory, beginning with location 80104CD0.

When you supply part of a file specification as a parameter, SDA assumes the following default values for the omitted portions of the specification. The default device is *SYSDISK*, the device specified in your most recent SET DEFAULT command. Likewise, the default directory is the directory specified in the most recent SET DEFAULT command. See the *VMS DCL Dictionary* for a description of the DCL command SET DEFAULT.

The **qualifier** modifies the action of an SDA command. A qualifier is always preceded by a slash (/). Several qualifiers can follow a single parameter or command name, but each must be preceded by a slash. Qualifiers can be abbreviated to the shortest string of characters that uniquely identifies the qualifier.

The **comment** consists of text that describes the command, but is not actually part of the command. Comments are useful for documenting SDA command procedures. When executing a command, SDA ignores the exclamation point and all characters that follow it on the same line.

5.2 Expressions

You can use expressions as parameters for some SDA commands, such as SEARCH and EXAMINE. To create expressions, you can use any of the following elements:

- Numerals
- Radix operators
- Arithmetic and logical operators
- Precedence operators
- Symbols

As mentioned, numerals are one possible component of an expression. The following sections describe the use of the other components.

SDA Description

5.2.1 Radix Operators

Radix operators determine which numeric base SDA uses to evaluate expressions. You can use one of the three radix operators to specify the radix of the numeric expression that follows the operator:

- ^X (hexadecimal)
- ^O (octal)
- ^D (decimal)

The default radix is hexadecimal. SDA displays hexadecimal numbers with leading zeros and decimal numbers with leading spaces.

5.2.2 Arithmetic and Logical Operators

There are two types of arithmetic and logical operators, both of which are listed in Table SDA–2.

- *Unary operators* affect the value of the expression that follows them.
- *Binary operators* combine the operands that precede and follow them.

In evaluating expressions containing binary operators, SDA performs logical AND, OR, and XOR operations, and multiplication, division, and arithmetic shifting before addition and subtraction. Note that the SDA arithmetic operators perform integer arithmetic on 32-bit operands.

Table SDA–2 SDA Operators

Operator	Action
Unary Operators	
#	Performs a logical NOT of the expression
+	Makes the value of the expression positive
–	Makes the value of the expression negative
@	Evaluates the following expression as a virtual address, then uses the contents of that address as value
G	Adds 80000000_{16} to the value of the expression ¹
H	Adds $7FFE0000_{16}$ to the value of the expression ²
Binary Operators	
+	Addition
–	Subtraction
*	Multiplication
&	Logical AND
	Logical OR

¹The unary operator G corresponds to the first virtual address in system space. For example, the expression GD40 can be used to represent the address $8000D40_{16}$.

²The unary operator H corresponds to a convenient base address in the control region of a process ($7FFE0000_{16}$). You can therefore refer to an address such as $7FFE2A64_{16}$ as H2A64.

Table SDA–2 (Cont.) SDA Operators

Operator	Action
Binary Operators	
\	Logical XOR
/	Division ³
@	Arithmetic shifting

³In division, SDA truncates the quotient to an integer, if necessary, and does not retain a remainder.

5.2.3 Precedence Operators

SDA uses parentheses as *precedence operators*. Expressions enclosed in parentheses are evaluated first. SDA evaluates nested parenthetical expressions from the innermost to the outermost pairs of parentheses.

5.2.4 Symbols

Names of symbols can contain from 1 to 31 alphanumeric characters and can include the dollar sign (\$) and underscore (_) characters. Symbols can take values from $-7FFFFFFF_{16}$ to $7FFFFFFF_{16}$.

By default, SDA copies symbols into its symbol table from SYS\$SYSTEM:SYS.STB and SYS\$SYSTEM:REQSYSDEF.STB. Additional symbols can be taken from other symbol tables or object modules and added to the SDA symbol table with the READ command. You can also use the DEFINE command to create symbols and add them to the symbol table.

In addition, SDA provides the symbols described in Table SDA–3:

Table SDA–3 SDA Symbols

Symbol	Meaning
.	Current location
AP	Argument pointer ¹
CLUSTERLOA	Base address of loadable VAXcluster code
<i>nn</i> DRIVER	Base address of a driver prologue table (DPT); such a symbol exists for each loaded device driver in the system ²
ESP	Executive stack pointer ¹
FP	Frame pointer ¹
FPEMUL	Base address of the code that emulates floating-point instructions

¹The value of those symbols representing the current SDA process context changes whenever you issue a command that changes this context (see Section 4). These symbols include the general purpose registers (R0 through R11, AP, FP, PC, and SP); the perprocess stack pointers (USP, SSP, KSP); the page table base and length registers (POBR, POLR, P1BR, and P1LR); and the processor status longword (PSL).

²The notation *nn* within the symbol *nn*DRIVER represents a 2-letter, generic device /controller name (for example, LPDRIVER).

SDA Description

Table SDA–3 (Cont.) SDA Symbols

Symbol	Meaning
G	80000000 ₁₆ , the base address of system space
H	7FFE0000 ₁₆
KSP	Kernel stack pointer ¹
MCHK	Address within loadable CPU-specific routines
MSCP	Address of loadable MSCP server code
POBR	Base register for the program region (PO) ¹
POLR	Length register for the program region (PO) ¹
P1BR	Base register for the control region (P1) ¹
P1LR	Length register for the control region (P1) ¹
PC	Program counter ¹
PSL	Processor status longword ¹
R0 through R11	General registers ¹
RMS	Base address of the RMS image
SCSLOA	Base address of loadable common SCS services
SP	Current stack pointer of a process ¹
SSP	Supervisor stack pointer ¹
SYSLOA	Base address of loadable processor-specific system code
USP	User stack pointer ¹

¹The value of those symbols representing the current SDA process context changes whenever you issue a command that changes this context (see Section 4). These symbols include the general purpose registers (R0 through R11, AP, FP, PC, and SP); the perprocess stack pointers (USP, SSP, KSP); the page table base and length registers (POBR, POLR, P1BR, and P1LR); and the processor status longword (PSL).

When SDA displays an address, it displays that address both in hexadecimal and as a symbol, if possible. If the address is within FFF₁₆ of the value of a symbol, SDA displays the symbol plus the offset from the value of that symbol to the address. If more than one symbol's value is within FFF₁₆ of the address, SDA displays the symbol whose value is the closest. If no symbols have values within FFF₁₆ of the address, SDA displays no symbol. (For an example, see the description of the SHOW STACK command.)

6 Investigating System Failures

This section discusses how the VMS operating system handles internal errors, and suggests procedures that can aid you in determining the causes of these errors. To conclude, it illustrates, through detailed analysis of a sample system failure, how SDA helps you find the causes of operating system problems.

For a complete description of the commands discussed in the sections that follow, refer to the last part of this document, where all the SDA commands are discussed in alphabetical order.

6.1 General Procedure for Analyzing System Failures

When the VMS operating system detects an internal error so severe that normal operation cannot continue, it signals a condition known as a fatal bugcheck and shuts itself down. A specific bugcheck code describes each such error.

To resolve the problem, you must find the reason for the bugcheck. Most failures are caused by errors in user-written device drivers or other privileged code not supplied by DIGITAL. To identify and correct these errors, you need a listing of the code in question.

Occasionally a system failure is the result of a hardware failure or an error in code supplied by DIGITAL. A hardware failure requires the attention of DIGITAL Field Service. To diagnose an error in code supplied by DIGITAL, you need listings of that code, which is available from DIGITAL on microfiche.

Start the search for the error by locating the line of code that signaled the bugcheck. Invoke SDA and use the SHOW CRASH command to display the content of the program counter (PC). The content of the PC is the address of the next instruction after the instruction that signaled the bugcheck.

The PC often contains an address in the exception handler, which signaled the bugcheck but did not cause it. In this case, the address of the instruction that caused the bugcheck is located on the stack. Use the SHOW STACK command to display the contents of the stack. See Section 6.2 for information on how to proceed for several types of bugchecks.

Once you have found the address of the instruction that caused the bugcheck, you need to find the module in which the failing instruction resides. Use the SHOW DEVICE command to determine whether the instruction is part of a device driver.

If it is not part of a driver, examine the linker's map of the module or modules you are debugging to determine whether the instruction that caused the bugcheck is in your programs.

If it is not within a driver or other code not supplied by DIGITAL, perform the following steps:

- 1 Issue the SDA command

```
SDA> SHOW EXECUTIVE
```

This command shows the location and size of each of the loadable images that make up the VMS executive.

SDA Description

- 2 Compare the suspected address with the addresses of these system images.
- 3 If the address is in fact within one of these images, issue the command

```
SDA> READ/EXECUTIVE SYS$LOADABLE_IMAGES:
```

This command loads the symbols that define locations within the loadable portion of the VMS executive.
- 4 Examine the failing address by issuing the command

```
SDA> EXAMINE @PC
```

SDA then displays the address in the PC as an offset from the nearest global symbol. This symbol may be the module's starting address, although it is possible that the code you are examining may not be in the module whose name is displayed.

Now, to determine the general cause of the system failure, examine the code that signaled the bugcheck.

6.2 Fatal Bugcheck Conditions

There are several conditions that cause VMS to issue a bugcheck. Normally, these occasions are rare. When they do occur, it is likely that they are in the nature of a fatal exception or an illegal page fault occurring within privileged code. This section describes the symptoms of these bugchecks. A discussion of other exceptions and VMS condition handling in general appears in the *VMS System Services Volume*.

6.2.1 Fatal Exceptions

An exception is fatal when it occurs while the following conditions exist:

- The process is using the interrupt stack.
- The process is executing above IPL 2 (IPL\$_ASTDEL).
- The process is executing in a privileged (kernel or executive) processor access mode and has not declared a condition handler to deal with the exception.

When the system fails, VMS reports the approximate cause of the failure on the console terminal. SDA displays a similar message when you issue a SHOW CRASH command. For instance, for a fatal exception, SDA can display one of these messages:

```
FATALEXCPT, Fatal executive or kernel mode exception
```

```
INVEXCEPTN, Exception while above ASTDEL or on interrupt stack
```

```
SSRVEXCEPT, Unexpected system service exception
```

Although there are several possible exception conditions, access violations are most common. When the hardware detects an access violation, information useful in finding the cause of the violation is pushed onto either the kernel stack or the interrupt stack. If the access violation occurred when it was using the interrupt stack, VMS places this information on the interrupt stack.

The INVEXCEPTN and SSRVEXCEPT bugchecks place three argument lists, or arrays, on the stack:

The first argument list appears near the top of the stack (see Figure SDA-1) and contains the number 2 in its first longword, indicating that the following two longwords complete the array.⁴ These two longwords contain the addresses on the stack of the *signal array* and *mechanism array*.

Figure SDA-1 First Argument List on the Stack

00000002
signal array address
mechanism array address

ZK-1920-84

The *mechanism array* (see Figure SDA-2) appears lower on the stack, at the address specified in the first argument list. Its first longword contains a 4, indicating that the four subsequent longwords complete the array. These four longwords are used by the VMS procedures that search for a condition handler and report exceptions.

Figure SDA-2 Mechanism Array

00000004
frame
depth
R0
R1

ZK-1921-84

⁴ This array sometimes does not appear on the stack. The mechanism and signal arrays, however, may still be present.

SDA Description

The values contained in the mechanism array are defined as follows:

Value	Meaning
00000004	Number of longwords that follow. In a mechanism array, this value is always 4.
Frame	Address of the FP (frame pointer) of the establisher's call frame.
Depth	Depth of the VMS search for a condition handler.
R0	Contents of R0 at the time of the exception.
R1	Contents of R1 at the time of the exception.

The *signal array* (see Figure SDA-3) appears somewhat further down the stack, at the address specified in the first argument list. A signal array contains the exception code, zero or more exception parameters, the PC, and the PSL. The size of a signal array can thus vary from exception to exception.

Figure SDA-3 Signal Array

00000005
0000000C
reason mask
virtual address
PC
PSL

ZK-1922-84

For access violations, the signal array is set up as follows:

Value	Meaning
00000005	Number of longwords that follow. For access violations, this value is always 5.
0000000C	Exception code. The value $0C_{16}$ represents an access violation. You can identify the exception code by using the SDA command EVALUATE/CONDITION.
Reason mask	Longword mask. If bit 0 of this longword is set, the failing instruction (at the PC saved below) caused a length violation. If bit 1 is set, it referred to a location whose page table entry is in a "no access" page. Bit 2 indicates the type of access used by the failing instruction: it is set for write and modify operations and clear for read operations.
Virtual address	Virtual address that the failing instruction tried to reference.
PC	PC whose execution resulted in the exception.
PSL	PSL at the time of the exception.

If VMS encounters a fatal exception, you can find the code that signaled it by examining the PC in the signal array. Use the SHOW STACK command to display the stack in use when the failure occurred, then locate the mechanism and signal arrays. Once you obtain the PC, which points to the instruction that signaled the exception, you can identify the module where the instruction is located by following the instructions in Section 7.3.

6.2.2 Illegal Page Faults

VMS signals a PGFIPLHI bugcheck when a page fault occurs while the interrupt priority level (IPL) is greater than 2 (IPL\$_ASTDEL). When VMS fails because of an illegal page fault, it prints the following message on the console terminal:

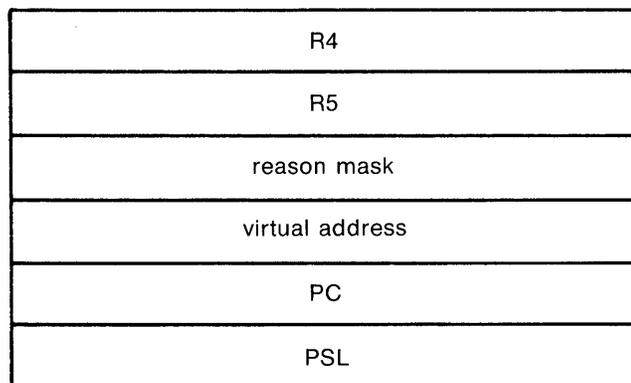
```
PGFIPLHI, Page fault with IPL too high
```

SDA Description

When an illegal page fault occurs, the stack appears as pictured in Figure SDA-4. Six longwords describe the exception:

Longword	Contents
R4	Contents of R4 at the time of the bugcheck.
R5	Contents of R5 at the time of the bugcheck.
Reason mask	Longword mask. If bit 0 of this longword is set, the failing instruction (at the PC saved below) caused a length violation. If bit 1 is set, it referred to a location whose page table entry is in an "access" page. Bit 2 indicates the type of access used by the failing instruction: it is set for write and modify operations and clear for read operations.
Virtual address	Virtual address being referenced by the instruction that caused the page fault.
PC	PC containing the address of the instruction that caused the page fault.
PSL	PSL at the time of the page fault.

Figure SDA-4 Stack Following an Illegal Page-Fault Error



ZK-1923-84

If the operating system detects a page fault while the IPL is higher than IPL\$_ASTDEL, you can obtain the address of the instruction that caused the fault by examining the PC pushed onto the current operating stack. Follow the steps outlined in Section 7.3 to determine which module issued the instruction.

7 A Sample System Failure

This section steps through the analysis of a system failure using, as an example, a printer driver. Three events lead up to this failure:

- 1 The line printer goes off line for three hours.
- 2 The line printer comes back on line.
- 3 The VMS operating system signals a bugcheck, writes information to the system dump file, and shuts itself down.

The following sections describe the actions you should take when investigating the causes of this system crash.

7.1 Identifying the Bugcheck

First, invoke SDA to analyze the system dump file. The initialization message indicates the type of bugcheck that occurred as follows:

```
VAX/VMS System dump analyzer
Dump taken on 31-JAN-1989 16:34:31.23
INVEXCEPTN, Exception while above ASTDEL or on interrupt stack
SDA>
```

VMS encountered an exception that caused it to signal a bugcheck, and it has created the signal and mechanism arrays on the current operating stack.

7.2 Identifying the Exception

Use the SHOW STACK command to display the current operating stack. In this case, it is the interrupt stack. The following example shows the interrupt stack and the signal and mechanism arrays. See the description of the SHOW STACK command for a complete description of the format of the stack display.

SDA Description

```

CPU 01 Processor stack
-----
Current operating stack (INTERRUPT)
      8006A378      8000844B      ACP$WRITEBLK+OAO
      .
      .
SP => 8006A398      7FFDC340
      8006A39C      8006A3A0
      8006A3A0      80004E7D      EXE$REFLECT+OD4
      8006A3A4      04080009
      8006A3A8      00000004
      8006A3AC      7FFDC368
      8006A3B0      FFFFFFFD
      8006A3B4      8001774E
      8006A3B8      0000074F
      8006A3BC      00000001
      8006A3C0      00000005
      8006A3C4      0000000C
      8006A3C8      00000000
      8006A3CC      80069E00
      8006A3D0      8005D003
      8006A3D4      04080000
      8006A3D8      80009604      EXE$FORKDSPH+O1C
      .
      .

```

The mechanism array begins at address $8006A3A8_{16}$ and ends at address $8006A3B8_{16}$. Its first longword contains 00000004_{16} . The signal array begins at address $8006A3C0_{16}$ and ends at $8006A3D4_{16}$. Its first longword contains 00000005_{16} and its second longword contains $0000000C_{16}$. Examination of the signal array shows that

- The exception code is $0C_{16}$, indicating an access violation.
- The reason mask is zero, indicating that the instruction caused a protection violation (instead of a length violation) when it tried to read the location (rather than write to it).
- The virtual address that the instruction attempted to reference was $80069E00_{16}$.
- The PC of the instruction that referred to the bad virtual address was $8005D003_{16}$.

Issuing the SDA command EVALUATE/PSL 04080000 makes the following information apparent:

- The IPL was 8 at the time of the exception (shown by bits 16 through 20 of the PSL).
- The current operating stack was the interrupt stack (bit 26 of the PSL is set to 1).
- The process was executing in kernel mode at the time of the exception (shown by bits 24 and 25 of the PSL).

Use the SHOW PAGE_TABLE command to display the system page table, as shown in the example following. The page containing location 80069E00₁₆ is not available to any access mode (a null page); thus the virtual address is not valid.

```
SDA> SHOW PAGE_TABLE
System page table
-----
ADDRESS  SVAPTE  PTE      TYPE  PROT  BITS  PAGTYP  LOC STATE TYPE  REFCNT  BAK      SVAPTE  FLINK  BLINK
-----
80068400 80777B08 7C40FFC8 STX   UR    K
80068600 80777B0C 7C40FFC8 STX   UR    K
80068800 80777B10 7C40FFC8 STX   UR    K
80068A00 80777B14 7C40FFC8 STX   UR    K
80068C00 80777B18 7C40FFC8 STX   UR    K
80068E00 80777B1C 7C40FFC8 STX   UR    K
80069000 80777B20 7C40FFC8 STX   UR    K
80069200 80777B24 7C40FFC8 STX   UR    K
80069400 80777B28 7C40FFC8 STX   UR    K
80069600 80777B2C 7C40FFC8 STX   UR    K
80069800 80777B30 7C40FFC8 STX   UR    K
80069A00 80777B34 780016C9 TRANS UR    K SYSTEM FREELST 00 01 0 0040FFC8 80777B34 03AF 0E15
80069C00 80777B38 78000E15 TRANS UR    K SYSTEM FREELST 00 01 0 0040FFC8 80777B38 16C9 2592
-----
40 NULL PAGES
```

7.3 Locating the Source of the Exception

Because the printer went off line and then came back on line, as shown on the console listing, the problem might exist in the driver code. SDA can help you to determine which driver might contain the faulty code.

7.3.1 Finding the Driver by Using the Program Counter

The first step in determining whether the failing instruction is within a driver is to examine the PC in the signal array using the EXAMINE/INSTRUCTION command. This has two results:

- 1 It displays, if possible, the contents of the address as a VAX MACRO instruction.
- 2 It identifies the address as an offset from the symbol, *nnDRIVER*, if the address lies within the first FFF₁₆ bytes of such a symbol. SDA defines a symbol in the form of *nnDRIVER* for each device driver connected to the system. This symbol represents the base of the driver prologue table (DPT). Each DPT is part of the device driver it describes and is immediately followed by that driver's code.

In the following example, the instruction that caused the exception is located within the printer driver.

```
SDA> EXAMINE/INSTRUCTION 8005D003
LPDRIVER+2B3  MOV  (R3)+, (R0)
```

If SDA is unable to find a symbol within FFF₁₆ bytes of the memory location you specify, it displays the location as an absolute address. This often, but not always, means the instruction that caused the exception is not part of a device driver.

SDA Description

To determine whether an instruction is or is not part of a driver, use the SHOW DEVICE command to display the starting addresses and lengths of all the drivers in the system. If the address of the failing instruction falls within the range of addresses shown for a given driver, the failing instruction is a part of that driver. The following example shows a partial list of the drivers in the display generated by the SHOW DEVICE command.

I/O data structures

DDB list					

Address	Controller	ACP	Driver	DPT	DPT size
-----	-----	---	-----	---	-----
80000ECC	HELIUM\$DBA	F11XQP	DBDRIVER	800F7AD0	08FD
80001040	OPA		OPERATOR	80001622	0061
8000126C	MBA		MBDRIVER	800015B0	0578
80001460	NLA		NLDRIVER	800015E9	05A3
801E2800	HELIUM\$DMA	F11XQP	DMDRIVER	800B5C80	0AA0
801E2980	HELIUM\$DLA	F11XQP	DLDRIVER	800B6A50	08D0

7.3.2 Calculating the Offset into the Driver's Program Section

The offsets that SDA displays from *nmDRIVER* are actually offsets from the DPT. As such, these offsets do not exactly correspond to the offsets shown in driver listings, which represent offsets from the beginning of the program section (PSECT) in which a given instruction appears. Because a driver usually contains more than one PSECT, you must use the driver's map to determine the location of the failing instruction within the driver listing.

To calculate the location of the instruction within the driver listing, refer to the "Program Section Synopsis" section of the driver's map. Determine in which PSECT the offset given by SDA occurs and subtract the base of the PSECT from the offset. You can then use the resulting figure as an index into the driver listing.

If SDA does not display the address as an offset from *nmDRIVER*, but the address *is* within the address range of a driver in the SHOW DEVICE display, you must first subtract the address of the DPT from the failing address. Using the result as the offset, you can then follow the steps previously outlined for determining the index of the instruction into a driver listing.

7.4 Finding the Problem Within the Routine

To find the problem within the routine, examine the printer's driver code. In the system failure discussed in this example, the instruction that caused the exception is `MOVB (R3)+,(R0)`. To check the contents of R3, use the EXAMINE command as follows:

```
SDA> EXAMINE R3
R3: 80069E00 "...."
```

The invalid virtual address, as recorded in the signal array, is stored in R3. In the following driver code excerpt, the instruction in question appears at line 599. It is likely that the contents of R3 have been incremented too many times.

```

581 STARTIO:
582     MOVL   UCB$L_IRP(R5),R3      ;Retrieve address of I/O packet
583     MOVW   IRP$L_MEDIA+2(R3),-
584     UCB$W_BOFF(R5)              ;Set number of characters to print
585     MOVL   UCB$L_SVAPTE(R5),R3   ;Get address of system buffer
586     MOVAB  12(R3),R3             ;Get address of data area
587     MOVL   UCB$L_CRE(R5),R4      ;Get address of CRB
588     MOVL   @CRB$L_INTD+VEC$L_IDB(R4),R4 ;Get device CSR address
589 ;
590 ; START NEXT OUTPUT SEQUENCE
591 ;
592
593 10$: ADDL3  #LP_DBR,R4,R0        ;Calculate address of data buffer register
594     MOVZWL UCB$W_BOFF(R5),R1    ;Get number of characters remaining
595     MOVW   #^X8080,R2           ;Get control register test mask
596     BRB   25$                  ;Start output
597 20$: BITW  R2,(R4) ❶           ;Printer ready or have paper problem?
598     BLEQ  30$                  ;If LEQ not ready or paper problem
599     MOVB  (R3)+,(R0) ❷        ;Output next character
600     ASHL  #1,G^EXE$GL_UBDELAY,-(SP) ;Delay 3*2 u-seconds
601 24$: SOBGEQ (SP),24$          ;Delay loop calibrated to machine speed
602     ADDL  #4,SP                ;Pop extra longword off stack
603 25$: SOBGEQ R1,20$ ❸        ;Any more characters to output?
604     BRW  70$                  ;All done, BRW to set return status

```

7.4.1 Examining the Routine

The MOVW instruction is part of a routine that reads characters from a buffer and writes them to the printer. The routine contains the loop of instructions that starts at the label 20\$ and ends at 25\$. This loop executes once for each character in the buffer, performing these five steps:

- ❶ The driver checks the printer's status register to see if the printer is ready.
- ❷ If the printer is ready, the driver gets a character from the buffer and moves it to the printer's data register, to which R0 points
- ❸ It then decrements R1, which contains the count of characters left to print. If R1 contains a number greater than zero, control is passed back to the instruction at 20\$, and the loop begins again.

Steps 1 and 2 are repeated until the contents of R1 are 0 or the printer signals that it is not ready.

If the printer signals that it is not ready, the driver transfers control to 30\$ (line 598), the beginning of a routine that waits for an interrupt from the printer. When the printer becomes ready, it interrupts the driver, and execution of the loop resumes.

Examine the code to determine which variables control the loop.

The byte count (BCNT) is the number of characters in the buffer. Note that BCNT is set by a function decision table (FDT) routine and that this routine sets the value of BCNT to the number of characters in the buffer. In line 586, the starting address of a buffer that is BCNT bytes in size is moved into R3.

Note also that the number of characters left to be printed is represented by the byte offset (BOFF), the offset into the buffer at which the driver finds the next character to be printed. This value controls the number of times the loop is executed.

SDA Description

Because the exception is an access violation, either R3 or R0 must contain an incorrect value. You can determine that R0 is probably valid by the following logic:

- The instruction at 10\$ (ADDL3 #LP_DBR,R4,R0) places an address in R0 and R0 is not modified again until the failing instruction (line 599).
- The value in R4 at the time that the instruction at 10\$ is executed was derived from the addresses of the device's unit control block (UCB) (line 587) and CRB (line 599). Although it is possible that these data structures may contain wrong information, it is unlikely.

Thus, the contents of R3 seem to be the cause of the failure.

The most likely reason that the contents of R3 are wrong is that the MOV B instruction at line 599 executes too many times. You can check this by comparing the contents of UCB\$W_BOFF and UCB\$W_BCNT. If UCB\$W_BOFF contains a larger value than that in UCB\$W_BCNT, then R3 contains a value that is too large, indicating that the MOV B instruction has incremented the contents of R3 too many times.

7.4.2 Checking the Values of Key Variables

Because the start-I/O routine requires that R5 contain the address of the printer's UCB, and because several other instructions reference R5 without error before any instruction in the loop does, you can assume that R5 contains the address of the right UCB. To compare BOFF and BCNT, use the command `FORMAT @R5` to display the contents of the UCB, as shown in the following session:

```
SDA> READ SYS$SYSTEM:SYSDEF.STB
SDA> FORMAT @R5

8005D160  UCB$L_FQFL      800039A8
          UCB$L_RQFL
          UCB$W_MB_SEED
          UCB$W_UNIT_SEED
8005D164  UCB$L_FQBL      800039A8
          UCB$L_RQBL
8005D168  UCB$W_SIZE      0122
8005D16A  UCB$B_TYPE      10
8005D16B  UCB$B_FIPL      34
          UCB$B_FLCK
          .
          .
          .
8005D1C8  UCB$L_SVAPTE    80062720
8005D1CC  UCB$W_BOFF      0795
8005D1CE  UCB$W_BCNT      006D
8005D1D0  UCB$B_ERTCNT    00
8005D1D1  UCB$B_ERTMAX    00
8005D1D2  UCB$W_ERRCNT    0000
          .
          .
          .
SDA>
```

If you have only one printer in your system configuration, you need not use the `FORMAT` command. Instead, you can use the command `SHOW DEVICE LP`. Because only one printer is connected to the VAX processor, only one UCB is associated with a printer for SDA to display.

The output produced by the FORMAT @R5 command shows that UCB\$W_BOFF contains a value greater than that in UCB\$W_BCNT; it should be smaller. Therefore, the value stored in BOFF is incorrect.

Thus, the value of BOFF is not the number of characters that remain in the buffer. This value is used in calculating an address that is referenced at an elevated IPL. When this address is within a null page (unreadable in all access modes), an attempt to reference it causes the system to fail.

7.4.3 Identifying and Fixing the Defective Code

Examine the printer driver code to locate all instructions that modify UCB\$W_BOFF. The value changes in two circumstances:

- Immediately after the driver detects that the printer is not ready and that the problem is not a paper problem (line 609).
- When the wait-for-interrupt routine's timeout count of 12 seconds is exhausted (lines 616 and 630). At this time, the contents of R1, plus one, are stored in UCB\$W_BOFF (line 631).

When the printer times out, the driver should not modify UCB\$W_BOFF. It does so, however, in line 631. The driver should modify the contents of UCB\$W_BOFF only when it is certain that the printer printed the character. When the printer times out, this is not the case. Furthermore, the wait-for-interrupt routine preserves only registers R3, R4, and R5, so only these registers can be used unmodified after the execution of the wait-for-interrupt routine. Thus the use of R1 in line 631 is an error.

To correct the problem, change the WFIKPCH argument (line 616) so that, when the printer times out, the WFIKPCH macro transfers control to 50\$ rather than to 40\$.

```

607
608 30$: BNEQ    40$                ;If NEQ paper problem
609      ADDW3   #1,R1,UCB$W_BOFF(R5) ;Save number of characters remaining
610      DEVICELOCK -
611          LOCKADDR=UCB$L_DLCK(R5),- ;Lock device interrupts
612          SAVIPL=- (SP)            ;Save current IPL
613      BITW    #^X80,LP_CSR(R4)    ;Is it ready now?
614      BNEQ    35$                ;If NEQ, yes, it's ready
615      BISB    #^X40,LP_CSR(R4)    ;Set interrupt enable
616      WFIKPCH 40$,#12            ;Wait for ready interrupt
617      IOFORK                      ;Create a fork process
618      BRB     10$                ; ...and start next output
619
620 35$:
621      DEVICEUNLOCK -
622          LOCKADDR=UCB$L_DLCK(R5),- ;Unlock device interrupts
623          NEWIPL=(SP)+             ;Restore IPL
624      CLRW    LP_CSR(R4)          ;Disable device interrupts
625      BRB     10$                ;Go transfer more characters

```

SDA Description

```
626 ;
627 ; PRINTER HAS PAPER PROBLEM
628 ;
629
630 40$: CLRL    UCB$L_LP_OFLCNT(R5) ;Clear offline counter
631    ADDW3    #1,R1,UCB$W_BOFF(R5) ;Save number of characters remaining
632 50$: CLRW    LP_CSR(R4)          ;Disable printer interrupt
633    IOFORK                    ;Lower to fork level
634    BBS      #UCB$V_CANCEL,UCB$W_STS(R5),80$ ;If set, cancel I/O operation
635    TSTW    LP_CSR(R4)          ;Printer still have paper problem?
636    BLSS    55$                 ;If LSS yes
637    MOVL    #15,UCB$L_LP_TIMEOUT(R5) ;Set timeout value
638    BRB     10$                 ; ...and start next output
```

8 Inducing a System Failure

If the operating system is not performing well and you want to create a dump you can examine, you must induce a system failure. Occasionally a device driver or other user-written, kernel-mode code can cause the system to execute a loop of code at a high priority, interfering with normal system operation. This can occur even though you have set a breakpoint in the code if the loop is encountered before the breakpoint. To gain control of the system in such circumstances, you must cause the system to fail and then reboot it.

If the system has suspended all noticeable activity (if it is “hung”), see the examples of causing system failures at the end of this section.

If you are generating a system crash in response to a system hang, be sure to record the PC at the time of the system halt as well as the contents of the general registers. Submit this information to DIGITAL, along with the Software Performance Report (SPR), and a copy of the generated system dump file.

8.1 Meeting Crash Dump Requirements

The following requirements must be met before the VMS system can write a complete crash dump:

- 1 You must not halt the system until the console dump messages have been printed in their entirety and the memory contents have been written to the crash dump file. Be sure to allow sufficient time for these events to take place or make sure that all disk activity has stopped before using the console to halt the system.
- 2 There must be a crash dump file in SYS\$SYSTEM: named either SYSDUMP.DMP or PAGEFILE.SYS.

This dump file must be either large enough to hold the entire contents of memory (as discussed in Section 1.1) or, if the DUMPSTYLE system parameter is set, large enough to accommodate a subset dump (see Section 1.1.1).

If SYSDUMP.DMP is not present, VMS attempts to write crash dumps to PAGEFILE.SYS. In this case, the SAVEDUMP system parameter must be 1 (the default is 0).

- 3 The system DUMPBUG parameter must be 1 (the default is 1).

8.2 Examples of How to Cause System Failures

The following examples show the sequence of console commands needed to cause a system failure on each type of VAX processor. In each instance, after halting the processor and examining its registers, you place the equivalent of -1 (for example FFFFFFFF₁₆) into the PC. The value placed in the PSL sets the processor access mode to kernel and the IPL to 31. After these commands are executed, an INVEXCEPTN bugcheck is reported on the console terminal, followed by a listing of the contents of the processor registers.

The console volume of most processors contains a command file named either CRASH.COM or CRASH.CMD that you can execute to perform these commands. Note that the console sessions recorded in this section omit much of the information the console displays in response to the listed commands.

VAX 8530/8550/8700/8800/8830/8850

The following series of console commands causes a system failure on the VAX 8530/8550/8700/8800/8830/8850 systems. (Note that the console prompt for the VAX 8830 and 8850 systems is PS-CIO-0> and not > > > .)

```
$  CTRL/P
>>> SET CPU CURRENT_PRIMARY
>>> HALT
?00      Left CPU -- CPU halted
         PC = 8001911C
>>> @CRASH
!
! Command procedure to force VMS bugcheck via access violation
!
SET VERIFY
SET CPU CURRENT_PRIMARY      !Select primary
EXAMINE PSL                  !Display PSL
                             M 00000000 00420008
EXAMINE/I/NEXT 4 0
.
.
.
DEPOSIT PC FFFFFFFF          !Set PC=-1 to force ACCVIO
DEPOSIT PSL 41F0000          !Set IPL=31, interrupt stack
CONTINUE                     !Execute from PC=-1
```

VAX 8200/8250/8300/8350 and VAX 6200 Series

The following console commands cause a system failure on a VAX 8200/8250/8300/8350 system or a VAX 6200 series system. In these systems, the HALT command is a NOP; a CTRL/P automatically halts the processor.

```
$  CTRL/P
         PC = 80008B1F
>>> E P
>>> E/I 0
>>> E/I +
>>> D/G F FFFFFFFF
>>> D P 41F0000
>>> C
```

SDA Description

VAX 8600/8650/8670

The following console commands cause a system failure on the VAX 8600 /8650/8670 systems.

```
$  CTRL/P
>>> @CRASH

      SET QUIET OFF           !Make clearer
      SET ABORT OFF          !Don't abort on E/VIR command
      HALT
      CPU stopped, INVOKED BY CONSOLE (CSM code 11)
      PC 80008B1F
      UNJAM                   !Clear the way
      E PSL                   !Display PSL
      U PSL 00000000
      E/I/N:4 0              !Display stack pointers
      .
      .
      E SP                    !Get current stack pointers
      G OE 80000C40
      E/vir/next:40 @        !Dump top of stack
      .
      .
      D PC FFFFFFFF          !Invalidate the PC
      D PSL 1F0000           !Kernel mode, IPL 31
      SET ABORT ON           !Restore abort flag
      SET QUIET ON           !Shut output off
      CONTINUE               !Force a machine check
```

VAX-11/780 and VAX-11/785

The following console commands cause a system failure on the VAX-11/780 and VAX-11/785 processors.

```
$  CTRL/P
>>> @CRASH
      HALT                   !Halt system, examine PC,
      HALTED AT 80008A89
      EXAMINE PSL            !PSL,
      00000000
      EXAMINE/INTERN/NEXT:4 0 !and all stack pointers
      DEPOSIT PC = -1        !Invalidate PC
      DEPOSIT PSL = 1F0000  !Kernel mode, IPL 31
      CONTINUE
```

VAX-11/750

The following code causes a system failure on a VAX-11/750. On a VAX-11/750 processor, the HALT command is a NOP; a CTRL/P automatically halts the processor.

```
$ CTRL/P
>>> H
>>> E P
>>> E/I 0
>>> E/I +
>>> E/I +
>>> E/I +
>>> E/I +
>>> D/G F FFFFFFFF
>>> D P 1F0000
>>> C
```

MicroVAX 3600 Series, MicroVAX II, and MicroVAX I

To force a crash of a MicroVAX, you must first halt the processor. (Once the processor is halted, press the HALT button again so that it is popped out and is not illuminated.) Then, issue the following console commands:

```
>>> E PSL
>>> E/I/N:4 0
>>> D PC FFFFFFFF
>>> D PSL 1F0000
>>> C
```

VAX-11/725 and VAX-11/730

The following console commands cause a system failure on a VAX-11/725 or a VAX-11/730 (as well as on VAXstation II systems and MicroVAX 2000 systems). CTRL/P automatically halts the processor.

```
$ CTRL/P
>>> H
>>> E PSL
>>> E/I/N:4 0
>>> D PC FFFFFFFF
>>> D PSL 1F0000
>>> C
```

SDA Usage Summary

The System Dump Analyzer is a utility that you can use to help determine the causes of system failures. This utility is also useful for examining the running system.

FORMAT

ANALYZE [*/CRASH_DUMP* [*/RELEASE*] *filespec*]
/SYSTEM
/SYMBOL=system-symbol-table

COMMAND PARAMETER

filespec

Name of the file that contains the dump you want to analyze. At least one field of the **filespec** is required, and it can be any field. The default **filespec** is the highest version of SYSDUMP.DMP in your default directory.

usage summary

You invoke SDA to analyze a system dump by issuing the command

```
$ ANALYZE/CRASH_DUMP filespec
```

If you do not specify **filespec**, SDA prompts you for it.

To analyze a crash dump, your process must have the privileges necessary for reading the dump file. This usually requires system privilege (SYSPRV), but your system manager can, if necessary, allow less privileged processes to read the dump files. Your process needs change-mode-to-kernel (CMKRNL) privilege to release page file dump blocks, whether you use the /RELEASE qualifier or the SDA COPY command.

You invoke SDA to analyze a running system by issuing the command

```
$ ANALYZE/SYSTEM
```

To examine a running system, your process must have change-mode-to-kernel (CMKRNL) privilege. You cannot specify filespec when using the /SYSTEM qualifier.

To send all output from SDA to a file, use the SDA command SET OUTPUT, specifying the name of the output file. The file produced is 132 columns wide and is formatted for output to a printer. To later redirect the output to your terminal, use the command

```
$ SET OUTPUT SYS$OUTPUT
```

To send a copy of all the commands you type and all the output those commands produce to a file use the SDA command SET LOG, specifying the name of the log file. The file produced is 132 columns wide and is formatted for output to a printer.

SDA Usage Summary

To exit from SDA, use the EXIT command. Note that the EXIT command also causes SDA to exit from display mode. Thus, if SDA is in display mode, you must use the EXIT command twice: once to exit from display mode, and a second time to exit from SDA.

SYSTEM DUMP ANALYZER

SDA Qualifiers

SDA QUALIFIERS

The qualifiers described in the following section determine whether the object of an SDA session is a crash dump or a running system and help create the environment of an SDA session.

- /CRASH_DUMP
- /RELEASE
- /SYMBOL
- /SYSTEM

/CRASH_DUMP

Invokes SDA to analyze the specified dump file.

FORMAT /CRASH_DUMP *filespec*

PARAMETER *filespec*

Name of the crash dump file to be analyzed. The default file specification is
SYS\$DISK:[default-dir]SYSDUMP.DMP

SYS\$DISK and *[default-dir]* represent the disk and directory specified in your last SET DEFAULT command. If you do not specify **filespec**, SDA prompts you for it.

DESCRIPTION

See Section 2 for additional information on crash dump analysis.

EXAMPLES

1 \$ ANALYZE/CRASH_DUMP SYS\$SYSTEM:SYSDUMP.DMP
 \$ ANAL/CRASH SYS\$SYSTEM

These commands invoke SDA to analyze the crash dump stored in
SYS\$SYSTEM:SYSDUMP.DMP.

2 \$ ANAL/CRASH SYS\$SYSTEM:PAGEFILE.SYS

This command invokes SDA to analyze a crash dump stored in the system
paging file.

SYSTEM DUMP ANALYZER

/RELEASE

/RELEASE

Invokes SDA to release those blocks in the specified system paging file occupied by a crash dump.

FORMAT **/RELEASE** *filespec*

PARAMETER ***filespec***
Name of the system page file (SYS\$SYSTEM:PAGEFILE.SYS). The default file specification is

 SYS\$DISK:[*default-dir*]SYSDUMP.DMP

SYS\$DISK and [*default-dir*] represent the disk and directory specified in your last SET DEFAULT command. If you do not specify **filespec**, SDA prompts you for it.

DESCRIPTION You use the /RELEASE qualifier to release from the system paging file those blocks occupied by a crash dump. When invoked with the /RELEASE qualifier, SDA immediately deletes the dump from the paging file and allows no opportunity to analyze its contents.

When you specify the /RELEASE qualifier in the ANALYZE command, you must also

- 1 Use the /CRASH_DUMP qualifier.
- 2 Include the name of the system paging file (SYS\$SYSTEM:PAGEFILE.SYS) as the **filespec**.

If you do not specify the system paging file or the specified paging file does not contain a dump, SDA generates the following messages:

```
%SDA-E-BLKSRLSD, no dump blocks in page file to release, or not page file
%SDA-E-NOTPAGFIL, specified file is not the page file
```

EXAMPLE

```
$ ANALYZE/CRASH_DUMP/RELEASE SYS$SYSTEM:PAGEFILE.SYS
$ ANALYZE/CRASH/RELEASE PAGEFILE.SYS
```

These commands invoke SDA to release to the paging file those blocks in SYS\$SYSTEM:PAGEFILE.SYS occupied by a crash dump.

/SYMBOL

Specifies a system symbol table for SDA to use in place of the system symbol table it uses by default (SYS\$SYSTEM:SYS.STB).

FORMAT **/SYMBOL** =*system-symbol-table*

PARAMETER ***system-symbol table***

File specification of the VMS SDA system symbol table needed to define symbols required by SDA to analyze a dump from a particular VMS system. The specified **system-symbol-table** must contain those symbols required by SDA to find certain locations in the executive image.

If you do *not* specify the /SYMBOL qualifier, SDA uses SYS\$SYSTEM:SYS.STB by default. When you *do* specify the /SYMBOL qualifier, SDA assumes the default disk and directory to be SYS\$DISK: that is, the disk and directory specified in your last SET DEFAULT command. If SDA is given a file that is not a system symbol table in the /SYMBOL qualifier, it halts with a fatal error.

DESCRIPTION

The /SYMBOL qualifier allows you to specify a system symbol table, other than SYS\$SYSTEM:SYS.STB, to load into the SDA symbol table. This may be necessary, for instance, in order to analyze a crash dump taken on a processor running a different version of VMS.

You can use the /SYMBOL qualifier whether you are analyzing a system dump or a running system.

EXAMPLE

```
$ ANALYZE/CRASH_DUMP/SYMBOL=SYS$CRASH:SYS.STB SYS$SYSTEM
```

This command invokes SDA to analyze the crash dump stored in SYS\$SYSTEM:SYSDUMP.DMP, using the system symbol table at SYS\$CRASH:SYS.STB.

SYSTEM DUMP ANALYZER

/SYSTEM

/SYSTEM

Invokes SDA to analyze a running system.

FORMAT **/SYSTEM**

PARAMETERS *None.*

DESCRIPTION See Section 3 for a full discussion of using SDA to analyze a running system.
You cannot specify the **/CRASH_DUMP** or **/RELEASE** qualifiers when you include the **/SYSTEM** qualifier in the **ANALYZE** command.

EXAMPLE

\$ ANALYZE/SYSTEM

This command invokes SDA to analyze the running system.

SYSTEM DUMP ANALYZER

SDA Commands

SDA COMMANDS

The commands described in the following section can be used in analyzing a system dump or the running system.

- @ (Execute Procedure)
- ATTACH
- COPY
- DEFINE
- EVALUATE
- EXAMINE
- EXIT
- FORMAT
- HELP
- READ
- REPEAT
- SEARCH
- SET CPU
- SET LOG
- SET OUTPUT
- SET PROCESS
- SET RMS
- SHOW CALL_FRAME
- SHOW CLUSTER
- SHOW CONNECTIONS
- SHOW CPU
- SHOW CRASH
- SHOW DEVICE
- SHOW EXECUTIVE
- SHOW HEADER
- SHOW LOCK
- SHOW PAGE_TABLE
- SHOW PFN_DATA
- SHOW POOL
- SHOW PORTS
- SHOW PROCESS
- SHOW RESOURCE
- SHOW RMS
- SHOW RSPID
- SHOW SPINLOCKS
- SHOW STACK
- SHOW SUMMARY
- SHOW SYMBOL
- SPAWN
- VALIDATE QUEUE

SYSTEM DUMP ANALYZER

@ (Execute Procedure)

@ (Execute Procedure)

Causes SDA to execute SDA commands contained in a file. Use this command to execute a set of frequently used SDA commands.

FORMAT @filespec

PARAMETER *filespec*
Name of a file that contains the SDA commands to be executed. The default file type is COM.

EXAMPLE

SDA> @USUAL

The Execute Procedure command executes the following commands, as contained in a file named USUAL.COM:

```
SET OUTPUT LASTCRASH.LIS
SHOW CRASH
SHOW PROCESS
SHOW STACK
SHOW SUMMARY
EXIT
```

This command procedure first makes the file LASTCRASH.LIS the destination for output generated by subsequent SDA commands. Next, the command procedure sends to the file information about the crash and its context, a description of the process executing at the time of the process, the contents of the stack on which the crash occurred, and a list of the processes active on the CPU that crashed. Finally, it exits from SDA.

The procedure need not exit from the utility at the end of its execution. To continue using SDA interactively after the execution of a command procedure, omit the EXIT command from the file.

ATTACH

Switches control of your terminal from your current process to another process in your job.

FORMAT **ATTACH** *process-name*

PARAMETER *process-name*
Name of the process to which you want to transfer control.

QUALIFIER ***/PARENT***
Transfers control of the terminal to the current process's parent process. When you specify this qualifier, you cannot specify the **process-name** parameter.

EXAMPLES

1 SDA> ATTACH/PARENT

This ATTACH command attaches the terminal to the parent process of the current process.

2 SDA> ATTACH DUMPER

This ATTACH command attaches the terminal to a process named DUMPER in the same job as the current process.

SYSTEM DUMP ANALYZER

COPY

COPY

Copies the contents of the dump file to another file.

FORMAT **COPY** *output-filespec*

PARAMETER *output-filespec*

Name of the device, directory, and file to which SDA copies the dump file. The default file specification is

SYS\$DISK:[default-dir]filename.DMP

SYS\$DISK and *[default-dir]* represent the disk and directory specified in your last SET DEFAULT command. You must at least supply the file name.

DESCRIPTION

Each time the system fails, the system copies the contents of physical memory and the hardware context of the current process (as directed by the DUMPSTYLE parameter) into the file SYS\$SYSTEM:SYSDUMP.DMP (or the paging file), overwriting its contents. If you do not save this crash dump elsewhere, it will be overwritten the next time that the system fails.

The COPY command allows you to preserve a crash dump by copying its contents to another file. It is generally useful to invoke SDA during system initialization (from within SYS\$MANAGER:SYSTARTUP.COM) to execute the COPY command. This ensures that a copy of the dump file is made each time the system fails.

The COPY command does not affect the contents of SYS\$SYSTEM:SYSDUMP.DMP.

If you are using the paging file (SYS\$SYSTEM:PAGEFILE.SYS) as the dump file instead of SYSDUMP.DMP, you can use the COPY command to explicitly release the blocks of the paging file that contain the dump, thus making them available for paging. Although the copy operation succeeds nonetheless, the release operation requires that your process have change-mode-to-kernel (CMKRNL) privilege. Once the dump pages have been released from the paging file, the dump information in these pages may be lost. You should perform subsequent analysis upon the copy of the dump created by the COPY command.

EXAMPLE

SDA> COPY SYS\$CRASH:SAVEDUMP

The COPY command copies the dump file into the file SYS\$CRASH:SAVEDUMP.DMP.

SYSTEM DUMP ANALYZER

DEFINE

Key Name	Key Designation
HELP	LK201 Help
DO	LK201 Do
F7 . . . F20	LK201 Function keys

command

SDA command the key is to be defined as. The command must be enclosed in quotation marks ("").

QUALIFIERS

/ECHO

/NOECHO

Determines whether the equivalence string is displayed on the terminal screen after the defined key has been pressed. The */NOECHO* qualifier functions only with the */TERMINATE* qualifier. The default is */ECHO*.

/IF_STATE=(state-name, . . .)

/NOIF_STATE

Specifies a list of one or more states, one of which must be in effect for the key definition to be in effect. States are placed in effect by the */SET_STATE* qualifier, a description of which appears below.

The **state-name** is an alphanumeric string, enclosed in quotation marks (""). By including several state names, you can define a key to have the same function in all the specified states. If you specify only one state name, you can omit the parentheses.

If you omit the */IF_STATE* qualifier—or use */NOIF_STATE*—the current state is used.

/KEY

Defines a key as an SDA command. Subsequently, you need only press the defined key and the RETURN key to issue the command. If you use the */TERMINATE* qualifier as well, you need not press the RETURN key.

When you define some keys as SDA commands, you must press CTRL/V before those keys to execute the commands. This is because of the escape sequences these keys generate, and the way the terminal driver handles those escape sequences. The following keys, when defined as SDA commands, must be preceded by a CTRL/V.

Key Name	Key Designation
LEFT	Left arrow
RIGHT	Right arrow
F7 . . . F14	LK201 function keys

/SET_STATE=state-name

Causes the key being defined to cause a key state change rather than issue an SDA command. When you use the */SET_STATE* qualifier, you supply the name of a key state in place of the **key-name** parameter. In addition, you must define the **command** parameter as a pair of quotation marks ("").

SYSTEM DUMP ANALYZER

DEFINE

The key state can be any name you think appropriate. For example, you can define the PF1 key to set the state to *gold* and use the `/IF_STATE=GOLD` qualifier to allow two definitions for the other keys, one in the *gold* state and one in the *nongold* state.

`/TERMINATE` **`/NOTERMINATE`**

Causes the key definition to include termination of the command, which causes SDA to execute the command when the defined key is pressed. Therefore, you do not have to press the RETURN key after you press the defined key if the `/TERMINATE` qualifier is specified.

DESCRIPTION

The DEFINE command causes SDA to evaluate an expression and then assign its value to a symbol. Both the DEFINE and EVALUATE commands perform computations in order to evaluate expressions. DEFINE adds symbols to the SDA symbol table but does not display the results of the computation. EVALUATE displays the result of the computation but does not add symbols to the SDA symbol table.

The DEFINE/KEY command causes an SDA command to be associated with the specified key, in accordance with any of the specified qualifiers described previously.

If the symbol or key is already defined, SDA replaces the old definition with the new one. Symbols and keys remain defined until you exit from SDA.

EXAMPLES

1 SDA> DEFINE BEGIN = 80058E00
SDA> DEFINE END = 80058E60
SDA> EXAMINE BEGIN:END

In the preceding example, DEFINE defines two addresses, called BEGIN and END. These symbols serve as reference points in memory, defining a range of memory locations that the EXAMINE command can inspect.

2 SDA> DEFINE NEXT = @PC
SDA> EXAMINE/INSTRUCTION NEXT
NEXT: MOVL @00(R6),R0

Symbol NEXT defines the address contained in the program counter, so that the symbol can be used in an EXAMINE/INSTRUCTION command.

3 SDA> DEFINE VEC SCH\$GL_PCBVEC
SDA> EXAMINE VEC
SCH\$GL_PCBVEC: 80B7D31C ".0.."

After the value of global symbol SCH\$GL_PCBVEC has been assigned to the symbol VEC, the symbol VEC is used to examine the memory location or value represented by the global symbol.

SYSTEM DUMP ANALYZER

DEFINE

```
4 SDA> DEFINE COUNT = 7
  SDA> DEFINE RESULT = COUNT * COUNT
  SDA> EVALUATE RESULT
Hex = 00000031   Decimal = 49           PR$_SBIS
                                     RESULT
```

The first DEFINE command assigns symbol COUNT the value 7. The second DEFINE command then defines RESULT to be the result of the evaluation of an arithmetic expression using the symbol COUNT. Evaluation of RESULT shows that system symbol PR\$_SBIS has an equivalent value.

```
5 SDA> DEFINE/KEY PF1 "SHOW STACK"
  SDA> [PF1] SHOW STACK [RETURN]
Process stacks (on CPU 00)
-----
Current operating stack (KERNEL):
      7FFE8DD4 00001703   SGN$C_MAXPGFL+703
      7FFE8DD8 80127920
      7FFE8DDC 00000000
      7FFE8DE0 00000000
      7FFE8DE4 00000000
      7FFE8DE8 00000000
      7FFE8DEC 7FF743E4
      7FFE8DF0 7FF743CC
SP => 7FFE8DF4 8000E646   EXE$CMODEXEC+1EE
      7FFE8DF8 7FFEDE96   SYS$CMKRNL+006
      7FFE8DFC 03C00000
```

The DEFINE/KEY command defines PF1 as the SHOW STACK command. When the PF1 key is pressed, SDA displays the command and waits for a carriage return to be entered.

```
6 SDA> DEFINE/KEY/TERMINATE PF1 "SHOW STACK"
  SDA> [PF1] SHOW STACK
Process stacks (on CPU 00)
-----
Current operating stack (KERNEL):
.
```

The DEFINE/KEY command defines PF1 as the SDA SHOW STACK command. The use of the /TERMINATE qualifier causes SDA to execute the SHOW STACK command without waiting for a carriage return to be entered.

```
7 SDA> DEFINE/KEY/SET_STATE="GREEN" PF1 ""
  SDA> DEFINE/KEY/TERMINATE/IF_STATE=GREEN PF3 "SHOW STACK"
  SDA> [PF1] [PF3] SHOW STACK
Process stacks (on CPU 00)
-----
Current operating stack (KERNEL):
.
```

The first DEFINE command defines PF1 as a key that sets a command state GREEN. The trailing pair of quotation marks is required syntax, indicating that no command is to be executed when this key is pressed.

SYSTEM DUMP ANALYZER

DEFINE

The second DEFINE command defines PF3 as the SHOW STACK command, but, using the /IF_STATE qualifier, makes the definition valid only when the command state is GREEN. Thus, the user must press PF1 before pressing PF3 to issue the SHOW STACK command. The /TERMINATE qualifier causes the command to execute as soon as the PF3 key is pressed.

SYSTEM DUMP ANALYZER

EVALUATE

EVALUATE

Computes and displays the value of the specified expression in both hexadecimal and decimal. Alternative evaluations of the expression are available with the use of the qualifiers defined for this command.

FORMAT

EVALUATE { */CONDITION_VALUE*
/PSL
/PTE
/SYMBOLS } *expression*

PARAMETER

expression

SDA expression to be evaluated. Section 5.2 describes the components of SDA expressions.

QUALIFIERS

/CONDITION_VALUE

Displays the message that the \$GETMSG system service obtains for the value of the expression.

/PSL

Evaluates the specified expression in the format of a processor status longword.

/PTE

Interprets and displays the expression as a page table entry (PTE). The individual fields of the PTE are separated and an overall description of the PTE's type is provided.

/SYMBOLS

Specifies that *all* symbols that are known to be equal to the evaluated expression are to be listed in alphabetical order. The default behavior of the EVALUATE command displays only the first several such symbols.

DESCRIPTION

If the expression is equal to the value of a symbol in the SDA symbol table, that symbol is displayed. If no symbol with this value is known, the next lower valued symbol is displayed with an appropriate offset if the offset is small enough for the selected symbol to be considered useful.

EXAMPLES

```
1 SDA> EVALUATE -1
Hex = FFFFFFFF Decimal = -1 PR$_XSID_N8NNN
```

The EVALUATE command evaluates a numeric expression, displays the value of that expression in hexadecimal and decimal notation, and displays a symbol that has been defined to have an equivalent value.

SYSTEM DUMP ANALYZER

EVALUATE

```
2 SDA> EVALUATE 1
Hex = 00000001 Decimal = 1      ACP$V_SWAPGRP
                                ACP$V_WRITECHK
                                EVT$_EVENT
                                .
                                .
```

The EVALUATE command evaluates a numeric expression and displays the value of that expression in hexadecimal and decimal notation. The preceding example also shows the symbols that have the displayed value. A finite number of symbols are displayed by default.

```
3 SDA> DEFINE TEN = A
SDA> EVALUATE TEN
Hex = 0000000A Decimal = 10     EXE$V_FATAL_BUG
                                SGN$C_MINWSCNT
                                TEN
```

The preceding example shows the definition of a symbol named TEN. The EVALUATE command then shows the value of the symbol.

Note that A, the value assigned to the symbol by the DEFINE command, could be a symbol. When SDA evaluates a string that can be either a symbol or a hexadecimal numeral, it first searches its symbol table for a definition of the symbol. If SDA finds no definition for the string, it evaluates the string as a hexadecimal number.

```
4 SDA> EVALUATE (((TEN * 6) + (-1/4)) + 6)
Hex = 00000042 Decimal = 66
```

The preceding example shows how SDA evaluates an expression of several terms, including symbols and rational fractions. SDA evaluates the symbol, substitutes its value in the expression, and then evaluates the expression. Note that the fraction $-1/4$ is truncated to 0.

```
5 SDA> EVALUATE/CONDITION 80000018
%SYSTEM-W-EXQUOTA, exceeded quota
```

The preceding example shows the output of an EVALUATE/CONDITION command.

```
6 SDA> EVALUATE/PSL 04080009
    CMP TP FPD IS CURMOD PRVMOD IPL DV FU IV T N Z V C
    0 0 0 1 KERN KERN 08 0 0 0 0 1 0 0 1
```

SDA interprets the entered value 04080009 as though it were a processor status longword (PSL) and displays the resulting field values of that longword.

SYSTEM DUMP ANALYZER

EVALUATE

7 SDA> EVALUATE/PTE ABCDFEED

```

|31      28|27      24|23      20|19      16|15      12|11      8|7
|-----|-----|-----|-----|-----|-----|-----|
|1 | 0 1 0 1 | 0 |--| 1 1 |--| 0|                                ODFEED
|-----|-----|-----|-----|-----|-----|-----|
Vld Prot= EW  M      Own=U  W                                Page Frame Number

```

Page is Active and Valid

The EVALUATE/PTE command displays the expression ABCDFEED as a page table entry (PTE) and labels the fields. It also describes the status of the page.

EXAMINE

Displays the contents of a location or range of locations in physical memory or the contents of a register. You can use location parameters to display specific locations or use qualifiers to display entire process and system regions of memory.

FORMAT **EXAMINE** *[/qualifier[,...]]* *[location]*

PARAMETER ***location***

Location in memory to be examined. A location can be represented by any valid SDA expression (see Section 5.2). To examine a range of locations, the following syntax is used:

m:n Range of locations to be examined, from *m* to *n*

m;n Range of locations to be examined, starting at *m* and continuing for *n* bytes

The default location that SDA uses is initially 0 in the program region (P0) of the process that was executing at the time the system failed (if you are examining a crash dump) or your process (if you are examining the running system). Subsequent uses of the EXAMINE command with no parameter specified increase the last address examined by 4. Use of the /INSTRUCTION qualifier increases the default address as appropriate to the translation of the instruction. To examine memory locations of other processes, you must use the SET PROCESS command.

QUALIFIERS ***/ALL***

Examines all the locations in the program and control regions and parts of the writable system region, displaying the contents of memory in hexadecimal longwords. Do not specify parameters when you use this qualifier.

/CONDITION_VALUE

Examines the specified longword, displaying the message the \$GETMSG system service obtains for the value in the longword.

/INSTRUCTION

Translates the contents of the specified range of memory locations into VAX MACRO instruction format. If more than 16 bytes are specified in the range, /INSTRUCTION processing may skip some bytes at the beginning of the range to ensure that SDA is properly synchronized with the start of each instruction. This synchronization may be overridden by specifying the /NOSKIP qualifier.

The length of the instruction displayed varies according to the opcode and addressing mode. If SDA cannot decode a memory location, it issues the following message:

%SDA-E-NOINSTRAN, cannot translate instruction

SYSTEM DUMP ANALYZER

EXAMINE

When you use this qualifier with the EXAMINE command, SDA calculates subsequent default addresses by adding the length of the last instruction, including all operands, to the last address examined.

/NOSKIP

Causes the EXAMINE command not to skip any bytes in the range when translating the contents of memory into VAX MACRO instructions. The /NOSKIP qualifier causes the execution of the /INSTRUCTION qualifier by default.

/NOSUPPRESS

Inhibits the suppression of zeros when displaying memory with one of the following qualifiers: /ALL, /P0, /P1, /SYSTEM.

/P0

Displays the entire program region for the default process. Do not specify parameters when you use this qualifier.

/P1

Displays the entire control region for the default process. Do not specify parameters when you use this qualifier.

/PSL

Examines the specified longword, displaying its contents in the format of a processor status longword. This qualifier must precede any parameters used in the command line.

/PTE

Interprets and displays the specified longword as a page table entry (PTE). The display separates individual fields of the PTE and provides an overall description of the PTE's type.

/SYSTEM

Displays portions of the writable system region. Do not specify parameters when you use this qualifier.

/TIME

Examines the specified quadword, displaying its contents in the format of a system-date-and-time quadword.

DESCRIPTION

The following sections describe how to use the EXAMINE command.

Examining Locations

When you use the EXAMINE command to look at a location, SDA displays the location in symbolic notation (symbolic name plus offset), if possible, and its contents in hexadecimal and ASCII formats:

```
SDA> EXAMINE G6605C0
806605C0: 80002119 ".!.."
```

If the ASCII character that corresponds to the value contained in a byte is not printable, SDA displays a period character (.). If the specified location does not exist in memory, SDA displays the message

```
%SDA-E-NOTINPHYS, address : not in physical memory
```

SYSTEM DUMP ANALYZER

EXAMINE

To examine a range of locations, you can designate starting and ending locations separated by a colon. For example:

```
SDA> EXAMINE G40:G200
```

Alternatively, you can specify a location and a length, in bytes, separated by a semicolon. For example:

```
SDA> EXAMINE G400;16
```

When used to display the contents of a range of locations, the EXAMINE command displays six columns of information:

- Each of the first four columns represents a longword of memory, the contents of which are displayed in hexadecimal format.
- The fifth column lists the ASCII value of each byte in each longword displayed in the previous four columns.
- The sixth column contains the address of the first, or rightmost, longword in each line. This address is also the address of the first, or leftmost, character in the ASCII representation of the longwords. Thus, you read the hexadecimal dump display from right to left, and the ASCII display from left to right.

If a series of virtual addresses does not exist in physical memory, SDA displays a message specifying the range of addresses that were not translated.

For example:

```
SDA> EXAMINE 100:200
```

```
Virtual locations 00000100 through 000001FF are not in physical memory
```

```
0130011A 0120011B 0130011E 0110011F .....0... ..0.      00000200
01200107 02300510 04310216 04210218 ..!...1...0....      00000210
01100103 01100104 01200105 01200106 .. ... ..           00000220
```

Addresses 100_{16} through $1FF_{16}$ do not exist in memory, as the message indicates. SDA displays the contents of those addresses that do exist (200_{16} through 220_{16}).

If a range of virtual locations contains only zeros, SDA displays the message

```
Zeros suppressed from 'loc1' to 'loc2'
```

Decoding Locations

You can translate the contents of memory locations into VAX MACRO instruction format by using the /INSTRUCTION qualifier. This qualifier causes SDA to display the location in symbolic notation (if possible) and its contents in instruction format. The operands of decoded instructions are also displayed in symbolic notation.

If the specified range of locations does not begin on an instruction boundary, SDA skips bytes until it locates the next valid instruction, and issues the message

```
%SDA-W-INSKIPPED, unreasonable instruction stream - n bytes skipped
```

In this message, *n* represents the number of bytes that SDA could not translate.

SYSTEM DUMP ANALYZER

EXAMINE

Examining Memory Regions

You can display an entire region of virtual memory by using one or more of the qualifiers /ALL, /SYSTEM, /P0, and P1 with the EXAMINE command.

Other Uses

Other uses of the EXAMINE command appear in the following examples.

EXAMPLES

1 SDA> EXAMINE/SYSTEM
System Region Memory

00040039 8FBC0010 00040038 8FBC0010 ...8.....9... 80000000
.
.

The preceding example shows only the first two lines of the display generated by the EXAMINE/SYSTEM command. Note that in the dump the fifth byte from the right contains the value 38_{16} . The ASCII value of 38_{16} , the character 8, is represented in the fifth character from the left in column 5.

Likewise, the 13th byte from the right in the dump columns contains the value 39_{16} . The ASCII value of 39_{16} is 9, and 9 is represented in the ASCII column as the 13th character from the left.

2 SDA> EXAMINE/PSL G1268
CMP TP FPD IS CURMOD PRVMOD IPL DV FU IV T N Z V C
1 0 0 0 KERN KERN 00 0 1 0 1 1 1 0 0

The preceding example shows the display produced by the EXAMINE/PSL command. The address of the longword examined is 80001268_{16} .

3 SDA> EXAMINE/PTE G775F480

```
|31      28|27      24|23      20|19      16|15      12|11      8|7  
|-----|-----|-----|-----|-----|-----|-----|-----|  
|1 | 1 1 1 0 |1 |--| 0 0 |--| 0|                               OOF0F4  
+-----+-----+-----+-----+-----+-----+-----+-----+  
Vld Prot= URKW M      Own=K      W                               Page Frame Number  
Page is Active and Valid
```

The EXAMINE/PTE command displays and formats the system page table entry at $8775F480_{16}$.

4 SDA> EXAMINE/TIME EXE\$GQ_SYSTIME
18-FEB-1989 02:07:25.88

The EXAMINE/TIME command displays the formatted value of the system time quadword (EXE\$GQ_SYSTIME).

EXIT

Exits from an SDA display or exits from the SDA utility.

FORMAT	EXIT
---------------	-------------

PARAMETERS	<i>None.</i>
-------------------	--------------

QUALIFIERS	<i>None.</i>
-------------------	--------------

DESCRIPTION	<p>If SDA is displaying information on a video display terminal—and if that information extends beyond one screen—SDA displays a <i>screen overflow prompt</i> at the bottom of the screen:⁵</p>
--------------------	---

Press RETURN for more.
SDA>

If you want to discontinue the current display at this point, type EXIT. If you want SDA to execute another command, type that command. SDA discontinues the display as if you typed EXIT, and then executes the command you typed.

When the SDA> prompt is not immediately preceded by the screen overflow prompt, typing EXIT causes your process to cease executing the SDA utility.

⁵ On hardcopy terminals, SDA does not display such a prompt.

SYSTEM DUMP ANALYZER

FORMAT

FORMAT

Displays a formatted list of the contents of a block of memory.

FORMAT	FORMAT <i>[/qualifier] location</i>
---------------	--

PARAMETER	location Location of the beginning of the data block. The location can be given as any valid SDA expression.
------------------	--

QUALIFIER	/TYPE=block-type Forces SDA to characterize and format a data block at location as the specified type of data structure. The /TYPE qualifier thus overrides the default behavior of the FORMAT command in determining the type of a data block, as described below. The block-type can be the symbolic prefix of any data structure defined by VMS.
------------------	---

DESCRIPTION	<p>The FORMAT command performs the following actions:</p> <ul style="list-style-type: none">• Characterizes a range of locations as a system data block• Assigns, if possible, a symbol to each item of data within the block• Displays all the data within the block <p>Normally, you use the FORMAT command without the /TYPE qualifier. Used in this manner, it examines the byte in the structure that contains the type of the structure. In most VMS data structures, this byte occurs at an offset of 0A₁₆ into the structure. If this byte does not contain a valid block type, the FORMAT command halts with the message</p> <pre>%SDA-E-INVBLKTYP, invalid block type in specified block</pre> <p>However, if this byte does contain a valid block type, SDA checks the next byte (offset 0B₁₆) for a secondary block type. When SDA has determined the type of block, it searches for the symbols that correspond to that type of block.</p> <p>If SDA cannot find the symbols associated with the block type it has found (or you specified in the /TYPE qualifier), it issues the message</p> <pre>No "block-type" symbols found to format this block</pre> <p>If you receive this message, you may want to read additional symbols into the SDA symbol table and retry the FORMAT command. Most symbols that define VMS data structures are contained within SYS\$SYSTEM:SYSDEF.STB. Thus, you would issue the command</p> <pre>SDA> READ SYS\$SYSTEM:SYSDEF.STB</pre>
--------------------	---

SYSTEM DUMP ANALYZER

FORMAT

Certain VMS data structures do *not* contain a block type at offset $0A_{16}$. If this byte contains information other than a block type—or the byte does not contain a valid block type—SDA produces the message

Invalid block type in specified block

To format such a block, you must reissue the FORMAT command, using the /TYPE qualifier to designate a **block-type**.

The FORMAT command produces a three-column display:

- The first column shows the virtual address of each item within the block.
- The second lists each symbolic name associated with a location within the block.
- The third shows the contents of each item in hexadecimal format.

EXAMPLE

```
SDA> READ SYS$SYSTEM:SYSDEF.STB
SDA> FORMAT 800B81F0
800B81F0  UCB$L_FQFL          80000F10
          UCB$L_RQFL
          UCB$W_MB_SEED
          UCB$W_UNIT_SEED
800B81F4  UCB$L_FQBL          800026A8
          UCB$L_RQBL
800B81F8  UCB$W_SIZE          00E0
800B81FA  UCB$B_TYPE          10
800B81FB  UCB$B_FLCK          07
800B81FC  UCB$L_ASTQFL       800F80E0
          UCB$L_FPC
          UCB$T_PARTNER
800B8200  UCB$L_ASTQBL       8002CF80
          UCB$L_FR3
800B8204  UCB$L_FIRST        8002CA00
          UCB$L_FR4
          UCB$W_MSGMAX
          UCB$W_MSGCNT
```

The READ command loads into SDA's symbol table the symbols from SYS\$SYSTEM:SYSDEF.STB needed for formatting system data structures. The FORMAT command displays the data structure that begins at $800B81F0_{16}$, a unit control block (UCB). If a field has more than one symbolic name, all such names are displayed. Thus, the field that starts at $800B8204_{16}$ has three designations: UCB\$L_FIRST and UCB\$L_FR4, alternative names for the longword; and the two subfields, UCB\$W_MSGMAX and UCB\$W_MSGCNT.

The contents of each field appear to the right of the symbolic name of the field. Thus, the contents of UCB\$L_FIRST are $8002CA00_{16}$.

SYSTEM DUMP ANALYZER

HELP

HELP

Displays information about the SDA utility, its operation, and the format of its commands.

FORMAT **HELP** [*command-name*]

PARAMETER ***command-name***
Command for which you need information.

You can also specify the following keywords in place of **command-name**:

Keyword	Function
CPU_CONTEXT	Describes the concept of CPU context as it governs the behavior of SDA in uniprocessor and multiprocessor environments
EXPRESSIONS	Prints a description of SDA expressions
INITIALIZATION	Describes the circumstances under which SDA executes an initialization file when first invoked
OPERATION	Describes how to operate SDA at your terminal and by means of the site-specific startup procedure
PROCESS_CONTEXT	Describes the concept of process context as it governs the behavior of SDA in uniprocessor and multiprocessor environments

QUALIFIERS *None.*

DESCRIPTION The HELP command displays brief descriptions of SDA commands and concepts on the terminal screen (or sends these descriptions to the file designated in a SET OUTPUT command). You can request additional information by specifying the name of a topic in response to the *Topic?* prompt.

If you do not specify a parameter in the HELP command, it lists those commands and topics for which you can request help, as follows:

Information available:

ATTACH	COPY	CPU_Context	DEFINE	EVALUATE	EXAMINE
Execute_Command		EXIT	Expressions	FORMAT	HELP
Initialization		Operation	Process_Context	READ	REPEAT
SEARCH	SET	SHOW	SPAWN	Symbols	VALIDATE

Topic?

READ

Loads the global symbols contained in the specified object module into the SDA symbol table.

FORMAT

READ { */EXECUTIVE directory-spec* }
{ *[/RELOCATE=expression] filespec* }

PARAMETER

filespec

Name of the device, directory, and file that contains the object module from which you want to copy global symbols. The ***filespec*** defaults to *SYS\$DISK:[default-dir]filename.STB*, where *SYS\$DISK* and *[default-dir]* represent the disk and directory specified in your last SET DEFAULT command. You must specify a file name.

QUALIFIERS

/EXECUTIVE directory-spec

Reads into the SDA symbol table all global symbols and global entry points defined within all loadable images that make up the executive. (See Table SDA-5 for a list of these images.)

The ***directory-spec*** is the name of the directory containing the loadable images of the executive; this parameter defaults to *SYS\$LOADABLE_IMAGES*.

/RELOCATE=expression

Adds the value of ***expression*** to the value of each symbol in the symbol-table file to be read. You can use the */RELOCATE* qualifier only if you also specify a ***filespec***. The */RELOCATE* qualifier is useful for examining images that are position independent and are loaded at a base of zero.

DESCRIPTION

The READ command symbolically identifies locations in memory for which the default symbol table (*SYS\$SYSTEM:SYS.STB*) provides no definition. In other words, the required global symbols are located in modules that have been compiled and linked separately from the VMS executive.⁶

The object module file specified in the READ command can be one of the following:

- Output of a compiler or assembler (for example, an .OBJ file)
- Output generated by the linker qualifier */SYMBOL_TABLE* (for example, an .STB file)

⁶ SDA extracts no local symbols from the object module.

SYSTEM DUMP ANALYZER

READ

Most often the object module file is a file provided by VMS in directory SYS\$SYSTEM or SYS\$LOADABLE_IMAGES. Many SDA applications, for instance, need to load the definitions of system data structures by issuing a READ command specifying SYS\$SYSTEM:SYSDEF.STB. Others require the definitions of specific global entry points within the executive image that are contained within those object modules included in the executive.

Table SDA-4 lists those object module files VMS provides in SYS\$SYSTEM. Table SDA-5 lists those loadable images in SYS\$LOADABLE_IMAGES that define locations within the VMS executive image.

Table SDA-4 Modules Containing Global Symbols Used by SDA

File	Contents
CLUSTRLOA.STB	Symbols for loadable VAXcluster management code
DCLDEF.STB	Symbols for the DCL interpreter
IMGDEF.STB	Symbols for the image activator
NETDEF.STB	Symbols for DECnet data structures
RMSDEF.STB	Symbols that define RMS internal and user data structures and RMS\$_xxx completion codes
SCSDEF.STB	Symbols that define data structures for system communications services
SYSDEF.STB	Symbols that define system data structures, including the I/O database

Table SDA-5 Modules Defining Global Locations Within the Executive Image

File	Contents
CPULOA.EXE	Processor-specific data and initialization routines
ERRORLOG.EXE	Error logging routines and system services
EVENT_FLAGS_AND_ASTS.EXE	Event flag and AST delivery routines and system services
EXCEPTION.EXE	Bugcheck and exception handling routines and those system services that declare condition and exit handlers
IMAGE_MANAGEMENT.EXE	Image activator and the related system services
IO_ROUTINES.EXE	\$QIO system service, related system services (SYS\$CANCEL, SYS\$ASSIGN, etc.), and supporting routines
LMF\$GROUP_TABLE.EXE	Data for valid, licensed product groups
LOCKING.EXE	Lock management routines and system services
LOGICAL_NAMES.EXE	Logical name routines and system services

SYSTEM DUMP ANALYZER

READ

Table SDA-5 (Cont.) Modules Defining Global Locations Within the Executive Image

File	Contents
MESSAGE_ROUTINES.EXE	System message routines and system services (including SYS\$SNDJBC and SYS\$GETTIM)
PAGE_MANAGEMENT.EXE	System pager, its supporting routines, and page management system services (including SYS\$CRMPSC, SYS\$CREDEL, and SYS\$ADJSTK)
PRIMITIVE_IO.EXE	Console I/O routines
PROCESS_MANAGEMENT.EXE	Scheduler, report system event, and supporting routines and system services
RECOVERY_UNIT_SERVICES.EXE	Recovery unit system services
RMS.EXE	Global symbols and entry points for RMS
SECURITY.EXE	Security management routines and system services
SYSDEVICE.EXE	Mailbox driver and null driver
SYSGETSYI.EXE	Get System Information system service (SYS\$GETSYI)
SYSLICENSE.EXE	Licensing system service (SYS\$LICENSE)
SYSMSG.EXE	VMS system messages
SYSTEM_PRIMITIVES.EXE	Miscellaneous basic system routines, including those that allocate system memory, maintain system time, create fork processes, and control mutex acquisition
SYSTEM_SYNCHRONIZATION.EXE	Routines that enforce synchronization in a VMS multiprocessing system
WORKING_SET_MANAGEMENT.EXE	Swapper, its supporting routines, and working set management system services

EXAMPLES

1 SDA> READ SYS\$SYSTEM:SYSDEF.STB
%SDA-I-READSYM, reading symbol table SYS\$COMMON:[SYSEXE]SYSDEF.STB;1

The READ command causes SDA to add all the global symbols in SYS\$SYSTEM:SYSDEF.STB to the SDA symbol table. Such symbols are useful when you are formatting an I/O data structure, such as a unit control block or an I/O request packet.

SYSTEM DUMP ANALYZER

READ

```
2 SDA> EXAM/INST EXE$QIO+2;4
EXE$QIO+00002: CHMK #001F
EXE$QIO+00006: RET
SDA> EXAM/INST V_EXE$QIO
%SDA-E-BADSYM, unknown symbol "V_EXE$QIO"
SDA> READ/RELOCATE=IO_ROUTINES SYS$LOADABLE_IMAGES:IO_ROUTINES.EXE
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR] IO_ROUTINES.EXE;1
SDA> EXAM/INST EXE$QIO+2;4
EXE$QIO+00002: MOVZBL 04(AP),R3
EXE$QIO+00006: CMPB R3,#3F
SDA> EXAM/INST V_EXE$QIO+2;4
V_EXE$QIO+00002: CHMK #001F
V_EXE$QIO+00006: RET
```

This SDA session shows that the initial examination of the instructions at EXE\$QIO+2 and EXE\$QIO+6 produces the vector for the system service, not the system service code itself. The subsequent READ instruction brings into the SDA symbol table the global symbols defined for the system's I/O routines, including one that redefines the entry point of the system service to be the start of the routine EXE\$QIO. Thus, the second examination of the same memory locations produces the first two instructions in the routine. The READ command creates a special symbol, V_EXE\$QIO, that points to the system service vector.

```
3 SDA> SHOW STACK
Process stacks (on CPU 01)
-----
Current operating stack (KERNEL):
          7FF8F2B0  806BA870
          7FF8F2B4  7FF8F4C0
          7FF8F2B8  8016F33E      PAGE_MANAGEMENT+0053E
          .
          .
          .
SDA> READ/RELOCATE=PAGE_MANAGEMENT SYS$LOADABLE_IMAGES:PAGE_MANAGEMENT.EXE
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR] PAGE_MANAGEMENT.EXE;1
SDA> SHOW STACK
Process stacks (on CPU 01)
-----
Current operating stack (KERNEL):
          7FF8F2B0  806BA870
          7FF8F2B4  7FF8F4C0
          7FF8F2B8  8016F33E      MMG$LOCK_SYSTEM_PAGES+00188
          .
          .
          .
```

The initial SHOW STACK command contains an address that SDA resolves into an offset from the PAGE_MANAGEMENT module of the executive. The READ command loads the corresponding symbols into the SDA symbol table such that the reissue of the SHOW STACK command subsequently identifies the same location as an offset within a specific page management routine.

SYSTEM DUMP ANALYZER

READ

```
4 SDA> READ/EXEC
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]RECOVERY_UNIT_SERVICES.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]RMS.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]CPULOA.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]LMF$GROUP_TABLE.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]SYSLICENSE.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]SYSGETSYI.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]SYSDEVICE.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]MESSAGE_ROUTINES.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]EXCEPTION.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]LOGICAL_NAMES.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]SECURITY.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]LOCKING.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]PAGE_MANAGEMENT.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]WORKING_SET_MANAGEMENT.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]IMAGE_MANAGEMENT.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]EVENT_FLAGS_AND_ASTS.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]IO_ROUTINES.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]PROCESS_MANAGEMENT.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]ERRORLOG.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]PRIMITIVE_IO.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]SYSTEM_SYNCHRONIZATION.EXE;1
%SDA-I-READSYM, reading symbol table SYS$COMMON:[SYS$LDR]SYSTEM_PRIMITIVES.EXE;1
```

This READ command brings all global symbols defined in the modules of SYS\$SYSTEM:SYS.EXE (as listed in Table SDA-5) into the SDA symbol table. Included in its results is the work performed by the READ commands illustrated in the two previous examples. The READ/EXECUTIVE command, however, does not load those symbols contained in the tables described in Table SDA-4.

SYSTEM DUMP ANALYZER

REPEAT

REPEAT

Repeats execution of the last command issued. On terminal devices, the KPO key performs the same function as the REPEAT command.

FORMAT REPEAT

PARAMETERS *None.*

QUALIFIERS *None.*

DESCRIPTION The REPEAT command is useful for stepping through a linked list of data structures, or for examining a sequence of memory locations.

EXAMPLES

```
1 SDA> FORMAT @IOC$GL_DEVLIST
8000B540 DDB$L_LINK      8000B898
8000B544 DDB$L_UCB      8000B5E0
8000B548 DDB$W_SIZE     0044
.
.
8000B554 DDB$B_NAME_LEN  03
      DDB$T_NAME        "OPA"
.
.
SDA> FORMAT @.
8000B898 DDB$L_LINK      8000BBE0
8000B89C DDB$L_UCB      8000B9E0
8000B8A0 DDB$W_SIZE     0044
.
.
8000B8AC DDB$B_NAME_LEN  03
      DDB$T_NAME        "MBA"
SDA> KPO
8000BBE0 DDB$L_LINK      807F85C0
8000BBE4 DDB$L_UCB      8000BC80
8000BBE8 DDB$W_SIZE     0044
.
.
8000BBF4 DDB$B_NAME_LEN  03
      DDB$T_NAME        "NLA"
```

This series of FORMAT commands pursues the chain of device data blocks (DDBs) from the system global symbol IOC\$GL_DEVLIST. The second FORMAT command is constructed such that it refers to the contents of the

SYSTEM DUMP ANALYZER

REPEAT

address at the current location (see Section 5.2.4 for a discussion of SDA symbols). Subsequently, pressing the KP0 key—or issuing the REPEAT command—is sufficient to display each DDB in the device list.

2

```
SDA> SHOW CALL_FRAME
Call Frame Information
```

```
-----
Call Frame Generated by CALLG Instruction
```

```
Condition Handler      7FFE7D78  00000000
SP Align Bits = 00    7FFE7D7C  00000000
  Saved AP             7FFE7D80  7FFE7DC0      CTL$GL_KSTKBAS+005C0
  Saved FP             7FFE7D84  7FFE7D94      CTL$GL_KSTKBAS+00594
```

```
SDA> SHOW CALL_FRAME/NEXT_FP
Call Frame Information
```

```
-----
Call Frame Generated by CALLS Instruction
```

```
Condition Handler      7FFE7D94  00000000
SP Align Bits = 00    7FFE7D98  20FC0000
  Saved AP             7FFE7D9C  7FFED024      CTL$GL_KSTKBAS+005E4
  Saved FP             7FFE7DA0  7FFE7DE4      SYSTEM_PRIMITIVES+020AA
```

```
SDA> REPEAT
Call Frame Information
```

```
-----
Call Frame Generated by CALLG Instruction
```

```
Condition Handler      7FFE7DE4  00000000
```

The first SHOW CALL_FRAME displays the call frame indicated by the current FP value. Because the /NEXT_FP qualifier to the instruction displays the call frame indicated by the saved FP in the current call frame, you can use the REPEAT command to repeat the SHOW CALL_FRAME/NEXT_FP command and follow a chain of call frames.

SYSTEM DUMP ANALYZER

SEARCH

SEARCH

Scans a range of memory locations for all occurrences of a specified value.

FORMAT **SEARCH** *[/qualifier] range[=]expression*

PARAMETERS **range**
Location in memory to be searched. A location can be represented by any valid SDA expression (see Section 5.2). To search a range of locations, use the following syntax:

m:n Range of locations to be searched, from *m* to *n*
m;n Range of locations to be searched, starting at *m* and continuing for *n* bytes

expression
Indication of the value for which SDA is to search. SDA evaluates the **expression** and searches the specified **range** of memory for the resulting value. For a description of SDA expressions, see Section 5.2.

QUALIFIERS

/LENGTH= { **LONGWORD**
 WORD
 BYTE }

Specifies the size of the **expression** value that the SEARCH command uses for matching. If you do not specify the /LENGTH qualifier, the SEARCH command uses a longword length by default.

/STEPS= { **QUADWORD**
 LONGWORD
 WORD
 BYTE }

Specifies the granularity of the search through the specified memory **range**. After the SEARCH command has performed the comparison between the value of **expression** and memory location, it adds the specified step factor to the address of the memory location to determine the next location to undergo the comparison. If you do not specify the /STEPS qualifier, the SEARCH command uses a step factor of a longword.

DESCRIPTION SEARCH displays each location as each value is found.

SYSTEM DUMP ANALYZER

SEARCH

EXAMPLES

1 SDA> SEARCH GB81F0;500 60068
Searching from 800B81F0 to 800B86F0 in LONGWORD steps for 00060068...
Match at 800B8210
SDA>

The SEARCH command finds the value 0060068 in the longword at 800B8210.

2 SDA> SEARCH/STEPS=BYTE 80000000;1000 6
Searching from 80000000 to 80001000 in BYTE steps for 00000006...
Match at 80000A99
SDA>

The SEARCH command finds the value 00000006 in the longword at 80000A99.

3 SDA> SEARCH/LENGTH=WORD 80000000;2000 6
Searching from 80000000 to 80002000 in LONGWORD steps for 0006...
Match at 80000054
Match at 800001EC
Match at 800012AC
Match at 800012B8
SDA>

The SEARCH command finds the value 0006 in the longword locations 80000054, 800001EC, 800012AC, and 800012B8.

SYSTEM DUMP ANALYZER

SET CPU

SET CPU

Selects a processor to become the SDA current CPU.

FORMAT **SET CPU** *cpu-id*

PARAMETER *cpu-id*
Numeric value from 00₁₆ to 1F₁₆ indicating the identity of the processor to be made the current CPU. If you specify a value outside this range or a **cpu-id** of a processor that was not active at the time of the system failure, SDA displays the following message:

%SDA-E-CPUNOTVLD, CPU not booted or CPU number out of range

QUALIFIERS *None.*

DESCRIPTION When you invoke SDA to examine a system dump, the SDA current CPU context defaults to that of the processor that caused the system to fail. When analyzing a crash from a multiprocessing system, you may sometimes find it useful to examine the context of another processor in the configuration.

The SET CPU command changes the current SDA CPU context to that of the processor indicated by **cpu-id**. The CPU specified by this command becomes the current CPU for SDA until you exit from SDA or change SDA CPU context by issuing one of the following commands:

```
SET CPU cpu-id
SHOW CPU cpu-id
SHOW CRASH
```

The following commands also change SDA CPU context if the **name** or index number (**nn**) refers to a current process.

```
SET PROCESS name
SET PROCESS/INDEX=nn
SHOW PROCESS name
SHOW PROCESS/INDEX=nn
```

Changing CPU context can cause an implicit change in process context under the following circumstances:

- If there is a current process on the CPU made current, SDA changes its process context to that of that CPU's current process.
- If there is no current process on the CPU made current, SDA process context is undefined and no process-specific information is available until you set SDA process context to that of a specific process.

See Section 4 for further discussion on the way in which SDA maintains its context information.

SYSTEM DUMP ANALYZER

SET CPU

You cannot use the SET CPU command when examining the running system with SDA.

EXAMPLE

```
$ ANAL/CRASH SYS$SYSTEM:SYSDUMP.DMP
VAX/VMS System dump analyzer
```

```
Dump taken on 22-FEB-1989 14:22:17.66
NOBUFPCKT, Required buffer packet not present
```

```
SDA> SHOW CPU
CPU 01 Processor crash information
-----
```

```
CPU 01 reason for Bugcheck: NOBUFPCKT, Required buffer packet not present
```

```
SDA> SHOW STACK
CPU 01 Processor stack
-----
```

```
Current operating stack (INTERRUPT):
```

```
      80DAFB4C      8018BC20
      80DAFB50      7FFC653E
```

```
SDA> SET CPU 00
SDA> SHOW CPU
CPU 00 Processor crash information
-----
```

```
CPU 00 reason for Bugcheck: CPUEXIT, Shutdown requested by another CPU
```

```
SDA> SHOW STACK
CPU 00 Processor stack
-----
```

```
Current operating stack (INTERRUPT):
```

```
      8016ABD8      00011F4C
      8016ABDC      00010F56
```

```
SDA> SHOW CRASH
System crash information
-----
```

```
Time of system crash: 22-FEB-1989 14:22:17.66
```

```
SDA> SHOW STACK
CPU 01 Processor stack
-----
```

```
Current operating stack (INTERRUPT):
```

SYSTEM DUMP ANALYZER

SET CPU

```
80DAFB4C 8018BC20
80DAFB50 7FFC653E
```

The series of SHOW CPU and SHOW STACK commands that occurs in this example illustrates the switching of CPU context within an SDA session.

When SDA is first invoked, it is, by default, within the CPU context of the processor that caused the crash (CPU 01). This is illustrated by the first set of SHOW CPU and SHOW STACK commands.

The SET CPU 00 command explicitly changes SDA CPU context to that of CPU 00, as illustrated by the second sequence of SHOW CPU and SHOW STACK commands. Note that a SHOW CPU 00 command would have the same effect as the two commands SET CPU 00 and SHOW CPU, changing the SDA CPU context in addition to displaying the processor-specific information. Unlike the SHOW CPU **cpu-id** command, there is no display associated with the SET CPU **cpu-id** command.

Lastly, the SHOW CRASH command resets the SDA CPU context to that of the processor that caused the crash (CPU 01).

SYSTEM DUMP ANALYZER

SET LOG

SET LOG

Initiates or discontinues the recording of an SDA session in a text file.

FORMAT **SET [NO]LOG** *filespec*

PARAMETER ***filespec***

Name of the file in which you want SDA to log your commands and their output. The default **filespec** is *SYS\$DISK:[default_dir]filename.LOG*, where *SYS\$DISK* and *[default_dir]* represent the disk and directory specified in your last SET DEFAULT command. You must specify a file name.

QUALIFIERS *None.*

DESCRIPTION

The SET LOG command echoes the commands and output of an SDA session to a log file. The SET NOLOG command terminates this behavior.

There are the following differences between the SET LOG command and the SET OUTPUT command:

- When logging is in effect, your commands and their results are still displayed on your terminal. The SET OUTPUT command causes the displays to be redirected to the output file such that they no longer appear on the screen.
- If an SDA command requires that you press RETURN to produce successive screens of display, the log file produced by SET LOG will record only those screens that are actually displayed. SET OUTPUT, however, sends the entire output of all SDA commands to its listing file.
- The SET LOG command produces a log file with a default file type of LOG; the SET OUTPUT command produces a listing file whose default file type is LIS.
- The SET LOG command does not record output from the HELP command in its log file. The SET OUTPUT command can record HELP output in its listing file.
- The SET LOG command does not record SDA error messages in its log file. The SET OUTPUT command can record SDA error messages in its listing file.
- The SET OUTPUT command generates a table of contents, each item of which refers to a display written to its listing file. SET OUTPUT also produces running heads for each page of output. The SET LOG command does not produce these items in its log file.

Note that, if you have used the SET OUTPUT command to redirect output to a listing file, you cannot use a SET LOG command to direct the same output to a log file.

SYSTEM DUMP ANALYZER

SET OUTPUT

SET OUTPUT

Redirects output from SDA to the specified file or device.

FORMAT **SET OUTPUT** *filespec*

PARAMETER *filespec*
Name of the file to which SDA is to send the output generated by its commands. The default **filespec** is *SYSDISK:[default_dir]filename.LIS*, where *SYSDISK* and *[default-dir]* represent the disk and directory specified in your last SET DEFAULT command. You must specify a file name.

DESCRIPTION When you use the SET OUTPUT command to send the SDA output to a file or device, SDA continues displaying the SDA commands that you enter but sends the output generated by those commands to the file or device you specify. (See the description of the SET LOG command for a list of differences between it and the SET OUTPUT command.)

If you finish directing SDA commands to an output file and wish to return to interactive display, issue the following command:

```
SDA> SET OUTPUT TT
```

If you use the SET OUTPUT command to send the SDA output to a listing file, SDA builds a table of contents that identifies the displays you selected and places the table of contents at the beginning of the output file. The SET OUTPUT command formats the output into pages and produces a running head at the top of each page.

SET PROCESS

Selects a process to become the SDA current process.

FORMAT	SET PROCESS	$\left\{ \begin{array}{l} \textit{process-name} \\ /INDEX=nn \\ /SYSTEM \end{array} \right\}$
---------------	--------------------	---

PARAMETER	<i>process-name</i>
------------------	----------------------------

Name of the process to become the SDA current process. The **process-name** is a string containing up to 15 uppercase or lowercase characters; numerals, the dollar sign (\$) character, and the underscore (_) character can also be included in the string. If you include characters other than these, you must enclose the entire string in quotation marks (" ").

QUALIFIERS	<i>/INDEX=nn</i>
-------------------	-------------------------

Specifies the process to be made current by its index into the system's list of software process control blocks (PCBs). You can supply either of the following values for **nn**:

- The process index itself
- The process identification (PID) or extended PID longword, from which SDA extracts the correct index

To obtain these values for any given process, issue the SDA command SHOW SUMMARY.

/SYSTEM

Specifies that the system process be made the SDA current process. Each VMS system (uniprocessor or multiprocessor) uses a single system process control block (PCB) and process header (PHD) as dummy structures, located in system space, that record the system working set, global section table, global page table, and other systemwide data.

DESCRIPTION	When you issue an SDA command such as an EXAMINE command, SDA displays the contents of memory locations in its current process. To display any information about another process, you must change the current process with the SET PROCESS command.
--------------------	---

When you invoke SDA to analyze a crash dump, its process context defaults to that of the process that was current at the time of the crash. If the crash occurred on a VMS multiprocessing system, SDA sets the CPU context to that of the processor that crashed the system and the process context to that of the process that was current on that processor.

When you invoke SDA to analyze a running system, its process context defaults to that of the current process: that is, the one executing SDA.

SYSTEM DUMP ANALYZER

SET PROCESS

The SET PROCESS command changes the current SDA process context to that of the process indicated by **name** or /INDEX=**nn**. The process specified by this command becomes the current process for SDA until you exit from SDA or change SDA process context by issuing one of the following commands:

```
SET PROCESS/INDEX=nn
SET PROCESS process-name
SHOW PROCESS/INDEX=nn
```

In the analysis of a crash dump from a multiprocessing system, changing process context can involve a switch of CPU context as well. For instance, if you issue a SET PROCESS command for a process that is current on another CPU, SDA will automatically change its CPU context to that of the CPU on which that process is current. The following commands can have this effect if **process-name** or index number (**nn**) refers to a current process.

```
SET PROCESS process-name
SET PROCESS/INDEX=nn
SHOW PROCESS process-name
SHOW PROCESS/INDEX=nn
```

See Section 4 for further discussion on the way in which SDA maintains its context information.

EXAMPLES

```
1 SDA> SHOW PROCESS
Process index: 0012 Name: NETACP Extended PID: 28C00092
-----
Process status: 00149001 RES,WAKEPEN,NOACNT,PHDRES,LOGIN
PCB address      800F1140 JIB address      801FDA00
PHD address      80477200 Swapfile disk address 01000F01
.
.
```

```
2 SDA> SHOW SUMMARY
Current process summary
-----
Extended Indx Process name Username State Pri PCB PHD Wkset
-- PID --
28C00080 0000 NULL COM 0 80002100 80001F88 0
28C00081 0001 SWAPPER HIB 16 800023C8 80002250 0
28C00483 0003 KLINGON KLINGON MWAIT 6 8010FEA0 803F8600 323
28C00085 0005 ERRFMT SYSTEM COM 10 800B5A10 8061DA00 69
28C00087 0007 OPCOM SYSTEM LEF 7 800C7000 80227A00 71
.
.
```

SYSTEM DUMP ANALYZER

SET PROCESS

```
3 SDA> SET PROCESS ERRFMT
SDA> SHOW PROCESS
Process index: 0005   Name: ERRFMT   Extended PID: 28C00085
-----
Process status: 00040001   RES,PHDRES
PCB address           800B5A10   JIB address           801E5C00
.
.
.
```

The first SHOW PROCESS command shows the current process to be NETACP. The SHOW SUMMARY command shows the names of the other processes that exist. The SET PROCESS command sets the current process to ERRFMT, as shown by the second SHOW PROCESS command. Note that the SET PROCESS command could also have been issued as one of the following:

```
SDA> SET PROCESS/INDEX=5
SDA> SET PROCESS/INDEX=801E5C00
```

SYSTEM DUMP ANALYZER

SET RMS

SET RMS

Changes the options shown by the SHOW PROCESS/RMS command.

FORMAT **SET RMS** =(option[,...])

PARAMETER

option

Data structure or other information to be displayed by the SHOW PROCESS/RMS command. Table SDA-6 lists those keywords that may be used as options. The default **option** is **option=ALL:ALL,NOPIO**, designating for display by the SHOW PROCESS/RMS command all structures for all files related to the process's image I/O.

To list more than one option, enclose the list in parentheses and separate options by commas. You can add a given data structure to those displayed by ensuring that the list of keywords begins with the * (asterisk) symbol. You can delete a given data structure from the current display by preceding its keyword with "NO".

QUALIFIERS

None.

DESCRIPTION

The SET RMS command determines the data structures to be displayed by the SHOW PROCESS/RMS command. (See the examples included in the discussion of the SHOW PROCESS command for an indication of the information provided by various displays.) You can examine the options that are currently selected by issuing a SHOW RMS command.

Table SDA-6 SET RMS Command Keywords for Displaying Process RMS Information

Keyword	Meaning
[NO]ALL[:ifi] ¹	All control blocks (default)
[NO]ASB	Asynchronous context block
[NO]BDB	Buffer descriptor block
[NO]BDBSUM	BDB summary page
[NO]BLB	Buffer lock block
[NO]BLBSUM	Buffer lock summary page
[NO]CCB	Channel control block
[NO]DRC	Directory cache
[NO]FAB	File attributes block
[NO]FCB	File control block

¹The optional parameter **ifi** is an internal file identification. The default **ifi (ALL)** is all the files the current process has opened.

SYSTEM DUMP ANALYZER

SET RMS

Table SDA-6 (Cont.) SET RMS Command Keywords for Displaying Process RMS Information

Keyword	Meaning
[NO]FWA	File work area
[NO]GBDSUM	GBD summary page
[NO]GBSB	Global buffer synchronization block
[NO]GBD	Global buffer descriptor
[NO]GBH	Global buffer header
[NO]IDX	Index descriptor
[NO]IFAB[:ifi] ¹	Internal FAB
[NO]IFB[:ifi] ¹	Internal FAB
[NO]IRAB	Internal RAB
[NO]IRB	Internal RAB
[NO]JFB	Journaling file block
[NO]NAM	Name block
[NO]NWA	Network work area
[NO]PIO	Image I/O (NOPIO), the default, or process I/O (PIO) ²
[NO]RAB	Record attributes block
[NO]RLB	Record lock block
[NO]RU	Recovery unit structures, including the recovery unit block (RUB), recovery unit stream block (RUSB), and recovery unit file block (RUFB)
[NO]SFSB	Shared file synchronization block
[NO]WCB	Window control block
[NO]XAB	Extended attribute block
[NO]*	Current list of options displayed by the SHOW RMS command

¹The optional parameter **ifi** is an internal file identification. The default **ifi (ALL)** is all the files the current process has opened.

²Specifying the PIO option causes the SHOW PROCESS/RMS command to display the indicated structures for process-permanent file I/O.

EXAMPLES

```

1 SDA> SHOW RMS
RMS Display Options:  IFB,IRB,IDX,BDB,BBDSUM,ASB,CCB,WCB,FCB,FAB,RAB,NAM,XAB,RLB,
BLB,BLBSUM,GBD,GBH,FWA,GBDSUM,JFB,NWA,RU,DRC,SFSB,GBSB

Display RMS structures for all IFI values.

SDA> SET RMS=IFB
SDA> SHOW RMS

RMS Display Options:  IFB

Display RMS structures for all IFI values.

```

SYSTEM DUMP ANALYZER

SET RMS

The first SHOW RMS command shows the default selection of data structures that are displayed in response to a SHOW PROCESS/RMS command. The SET RMS command selects only the IFB to be displayed by subsequent SET/PROCESS commands.

```
2 SDA> SET RMS=(*,BLB,BLBSUM,RLB)
  SDA> SHOW RMS

RMS Display Options:  IFB,RLB,BLB,BLBSUM
Display RMS structures for all IFI values.
```

The SET RMS command adds the BLB, BLBSUM, and RLB to the list of data structures currently displayed by the SHOW PROCESS/RMS command.

```
3 SDA> SET RMS=(*,NORLB,IFB:05)
  SDA> SHOW RMS

RMS Display Options:  IFB,BLB,BLBSUM
Display RMS structures only for IFI=5.
```

The SET RMS command removes the RLB from those data structures displayed by the SHOW PROCESS/RMS command and causes only information about the file with the **ifi** of 5 to be displayed.

```
4 SDA> SET RMS=(*,PIO)
```

The SET RMS command indicates that the data structures designated for display by SHOW PROCESS/RMS be associated with process-permanent I/O instead of image I/O.

SYSTEM DUMP ANALYZER

SHOW CALL_FRAME

EXAMPLE

SDA> SHOW CALL_FRAME
Call Frame Information

Call Frame Generated by CALLG Instruction

Condition Handler	7FFE7D78	00000000	
SP Align Bits = 00	7FFE7D7C	00000000	
Saved AP	7FFE7D80	7FFE7DC0	CTL\$GL_KSTKBAS+005C0
Saved FP	7FFE7D84	7FFE7D94	CTL\$GL_KSTKBAS+00594
Return PC	7FFE7D88	8015303F	EXCEPTION+0043F

Align Stack by 0 Bytes =>
SDA> SHOW CALL_FRAME/NEXT_FP
Call Frame Information

Call Frame Generated by CALLS Instruction

Condition Handler	7FFE7D94	00000000	
SP Align Bits = 00	7FFE7D98	20FC0000	
Saved AP	7FFE7D9C	7FFED024	
Saved FP	7FFE7DA0	7FFE7DE4	CTL\$GL_KSTKBAS+005E4
Return PC	7FFE7DA4	801D58AA	MMG\$IMGRESET+00066
R2	7FFE7DA8	7FFEDD00	CTL\$GL_KSTKBAS+005D0
R3	7FFE7DAC	7FFDB9F8	
R4	7FFE7DB0	8026C720	
R5	7FFE7DB4	7FFDBA00	
R6	7FFE7DB8	7FFE6300	CTL\$A_DISPVEC+00500
R7	7FFE7DBC	00000003	

Align Stack by 0 Bytes =>
Argument List

7FFE7DC0	00000003	
7FFE7DC4	7FFE7DD0	CTL\$GL_KSTKBAS+005D0
7FFE7DC8	00000000	
7FFE7DCC	00000000	

SDA> SHOW CALL_FRAME/NEXT_FP
Call Frame Information

Call Frame Generated by CALLG Instruction

Condition Handler	7FFE7DE4	00000000	
SP Align Bits = 00	7FFE7DE8	00000000	
Saved AP	7FFE7DEC	7FFED024	
Saved FP	7FFE7DF0	7FFECFF8	
Return PC	7FFE7DF4	8015303F	EXCEPTION+0043F

Align Stack by 0 Bytes =>

The SHOW CALL_FRAME commands in this SDA session follow a chain of call frames from that specified in the FP of the SDA current process.

SYSTEM DUMP ANALYZER

SHOW CLUSTER

- Number of votes (if any) provided by the node
- Its state⁹
- Its status⁹

The **cluster block** display includes information recorded in the cluster block (CLUB), including a list of activated flags, a summary of quorum and vote information, and other data that applies to the VAXcluster from the perspective of the node for which the SDA is being run.

The **cluster failover control block** display provides detailed information concerning the cluster failover control block (CLUFCB), and the **cluster quorum disk control block** display provides detailed information from the cluster quorum disk control block (CLUDCB).

Subsequent displays provide information for each CSB listed previously in the **CSB list** display. Each display shows the state and flags of a CSB, as well as other specific node information. (See the *VMS Show Cluster Utility Manual* for information about the flags for VAXcluster nodes.)

VAXcluster as Seen by the Port Driver

The SHOW CLUSTER/SCS command provides a series of displays.

The **SCS listening process directory** lists those processes that are listening for incoming SCS connect requests. For each of these processes, this display records the following information:

- Address of its directory entry
- Connection ID
- Name
- Explanatory information, if available

The **SCS systems summary** display provides the system block (SB) address, node name, system type, system ID, and the number of connection paths for each SCS system. An *SCS system* can be a VAXcluster member, HSC, UDA, or other such device.

Subsequent displays provide detailed information for each of the system blocks and the associated path blocks. The system block displays include the maximum message and datagram sizes, local hardware and software data, and SCS poller information. Path block displays include information that describes the connection, including remote functions and other path-related data.

⁹ For information about the state and status of nodes, see the description of the ADD command in the *VMS Show Cluster Utility Manual*.

SYSTEM DUMP ANALYZER

SHOW CLUSTER

EXAMPLES

1 SDA> SHOW CLUSTER

VAXcluster data structures

```

-----
                --- VAXcluster Summary ---
    Quorum  Votes  Quorum Disk Votes  Status Summary
    -----  -----  -----  -----
             2      3             1      quorum

                --- CSB list ---
Address  Node  CSID      Votes  State  Status
-----  ----  ----      -----  ----  -----
803686F0  SOLLY  000100C8   1   open  member,qf_active
80368550  GUS    000100C9   1   open  member,qf_active
80367B90  DORIS  000100C5   1   open  member,qf_active

                --- Cluster Block (CLUB) 801C3F70 ---
Flags: 10080001 cluster,init,quorum

Quorum/Votes          2/3  Last transaction code          02
Quorum Disk Votes    1    Last trans. number            1126
Nodes                 3    Last coordinator CSID         00000000
Quorum Disk          $255$DUA2  Last time stamp                26-MAR-1986
Found Node SYSID     0000000008A0          18:52:32
Founding Time        3-DEC-1988  Largest trans. id              00000466
                                Resource Alloc. retry          0
Index of next CSID   00D2    Figure of Merit                 00000000
Quorum Disk Cntrl Block 80334E00  Member State Seq. Num          0190
Timer Entry Address  00000000  Foreign Cluster                 00000000
CSP Queue            empty

                --- Cluster Failover Control Block (CLUFCB) 801C407C ---
Flags: 00000000

Failover Step Index   00000028  CSB of Synchr. System          803686F0
Failover Instance ID  00000466

                --- Cluster Quorum Disk Control Block (CLUDCB) 80334E00 ---
State: 0001 qs_not_ready
Flags: 0000

Iteration Counter     0          UCB address                    00000000
Activity Counter      0          TQE address                     80419F40
Quorum file LBN       00000000  IRP address                     803665A0

                --- SOLLY Cluster System Block (CSB) 803686F0 ---
State: 01 open
Flags: 02020302 member,cluster,qf_active,selected,status_rcvd

```

SYSTEM DUMP ANALYZER

SHOW CLUSTER

```

Quorum/Votes      2/1   Next seq. number  0247   Send queue      00000000
Quor. Disk Vote   1     Last seq num rcvd 0314   Resend queue    00000000
CSID              000100C8  Last ack. seq num 0247   Block xfer Q.   empty
Eco/Version       0/12   Unacked messages  1       CDT address     801C28F0
Reconn. time     00000059  Ack limit        4       PDT address     801CEA20
Ref. count       2     Incarnation      18-DEC-1988  TQE address     00000000
Ref. time        18-DEC-1988 08:52:20  SB address      8041B6E0
                  08:53:58  Lock mgr dir wgt 1       Current CDRP    00000000
  
```

The preceding example shows the screen displays for the SHOW CLUSTER command. (Displays for nodes GUS and DORIS, similar to that for node SOLLY, are also included in the SHOW CLUSTER output but have been omitted from the preceding example.)

2 SDA> SHOW CLUSTER /CSID=000100C8

VAXcluster data structures

```

-----
      --- SOLLY Cluster System Block (CSB) 803686F0 ---

State: 01 open
Flags: 02020302 member,cluster,qf_active,selected,status_rcvd

Quorum/Votes      2/1   Next seq. number  0247   Send queue      00000000
Quor. Disk Vote   1     Last seq num rcvd 0314   Resend queue    00000000
CSID              000100C8  Last ack. seq num 0247   Block xfer Q.   empty
Eco/Version       0/12   Unacked messages  1       CDT address     801C28F0
Reconn. time     00000059  Ack limit        4       PDT address     801CEA20
Ref. count       2     Incarnation      18-DEC-1988  TQE address     00000000
Ref. time        18-DEC-1988 08:52:20  SB address      8041B6E0
                  08:53:58  Lock mgr dir wgt 1       Current CDRP    00000000
  
```

The preceding example shows the use of the /CSID qualifier to obtain information about a specific node (in this instance, node SOLLY). The information displayed is identical to that shown for the specified node in the SHOW CLUSTER command.

3 SDA> SHOW CLUSTER /SCS

VAXcluster data structures

```

-----
      --- SCS Listening Process Directory ---

Entry Address      Connection ID      Process Name      Information
-----
      80419D60      08EE0000          SCS$DIRECTORY
      80419E20      08EE0001          VMS$VAXcluster

      --- SCS Systems Summary ---

SB Address      Node      Type      System ID      Paths
-----
      8041A120      PINTO     HSC       00000000F10E   1
      8041AA20      DORIS     VMS       0000000008A9   1
      8041AB40      GUS       VMS       0000000008A1   1
      8041B6E0      SOLLY     VMS       0000000008A0   1
      8041D420      DODGER    HSC       00000000F00F   1
  
```

SYSTEM DUMP ANALYZER

SHOW CLUSTER

```

--- PINTO System Block (SB) 8041A120 ---
System ID          00000000F10E   Local software type      HSC
Max message size   66             Local software vers.     X301
Max datagram size  62             Local software incarn.    8355FE00
Local hardware type HS50             Local hardware type      008DA59A
Local hardware vers. 022702220222   SCS poller timeout       000F
                   022202220222   SCS poller enable mask   01

```

```

--- Path Block (PB) 8041C400 ---

```

```

Status: 0000

```

```

Remote sta. addr.  00000000000E   Remote port type         HSC
Remote state       00000000000E   Number of data paths     2
Remote hardware rev. 00000225     Cables state             A-OK B-OK
Remote func. mask  4F710200       Local state               OPEN
Resetting port     0E             Port dev. name           PABO
Handshake retry cnt. 1             SCS MSGBUF address       80390270
Msg. buf. wait queue empty       PDT address               801CEA20

```

```

--- DORIS System Block (SB) 8041AA20 ---

```

```

System ID          0000000008A9   Local software type      VMS
Max message size   112            Local software vers.     V5.0
Max datagram size  576            Local software incarn.    A9D31760
Local hardware type V780             Local hardware type      008DA59B
Local hardware vers. 010E0138207A   SCS poller timeout       000C
                   000030030E10   SCS poller enable mask   00

```

```

--- Path Block (PB) 80437E80 ---

```

```

Status: 0000

```

```

Remote sta. addr.  000000000002   Remote port type         CI780
Remote state       ENAB            Number of data paths     2
Remote hardware rev. 00040003     Cables state             A-OK B-OK
Remote func. mask  FFFFFFF0       Local state               OPEN
Resetting port     02             Port dev. name           PABO
Handshake retry cnt. 1             SCS MSGBUF address       8036FOB0
Msg. buf. wait queue empty       PDT address               801CEA20

```

The preceding example shows a subset of a typical output for the SHOW CLUSTER/SCS command. In this system, there are three VMS nodes (DORIS, GUS, and SOLLY), and there are two HSCs (PINTO and DODGER). After the summary information in the first two screen displays, specific information for each system block and its associated path block is shown.

SYSTEM DUMP ANALYZER

SHOW CONNECTIONS

SHOW CONNECTIONS

Displays information about all active connections between systems communications services (SCS) processes or a single connection.

FORMAT **SHOW CONNECTIONS** *[/ADDRESS=cdt-address]*

PARAMETERS *None.*

QUALIFIER ***/ADDRESS=cdt-address***
Displays information contained in the connection descriptor table (CDT) for a specific connection.¹⁰

DESCRIPTION The SHOW CONNECTIONS command provides a series of displays. The **CDT summary page** lists information regarding each connection on the local system, including the following:

- CDT address
- Name of the local process with which the CDT is associated
- Connection ID
- Current state
- Name of the remote node (if any) to which it is currently connected

The **CDT summary page** concludes with a count of CDTs that are free and available to the system.

SHOW CONNECTIONS next displays a page of detailed information for each active CDT listed previously.

¹⁰ You can find the *cdt-address* for any active connection on the system in the **CDT summary page** display of the SHOW CONNECTIONS command. In addition, CDT addresses are also stored in many individual data structures related to SCS connections. These data structures include class driver request packets (CDRPs) and unit control blocks (UCBs) for class drivers that use SCS and cluster system blocks (CSBs) for the connection manager.

SYSTEM DUMP ANALYZER

SHOW CONNECTIONS

EXAMPLES

1 SDA> SHOW CONNECTIONS
VAXcluster data structures

--- CDT Summary Page ---

CDT Address	Local Process	Connection ID	State	Remote Node
801C2670	SCS\$DIRECTORY	08EE0000	listen	
801C2710	VMS\$VAXcluster	08EE0001	listen	
801C27B0	VMS\$VAXcluster	08FF0002	open	DORIS
801C2850	VMS\$DISK_CL_DVR	08FD0003	open	PINTO
801C28F0	VMS\$VAXcluster	08EF0004	open	SOLLY
801C2990	VMS\$VAXcluster	08F00005	open	GUS

Number of free CDTs: 32

--- Connection Descriptor Table (CDT) 801C2670 ---

State: 0001 listen Local Process: SCS\$DIRECTORY
Blocked State: 0000

Local Con. ID	08EE0000	Datagrams sent	0	Message queue	empty
Remote Con. ID	78A30017	Datagrams rcvd	0	Send Credit Q.	empty
Receive Credit	0	Datagram discard	0	PB address	80438300
Send Credit	1	Messages Sent	0	PDT address	801CEA20
Min. Rec. Credit	0	Messages Rcvd.	0	Error Notify	8022B816
Pend Rec. Credit	0	Send Data Init.	0	Receive Buffer	00000000
Initial Rec. Credit	0	Req Data Init.	0	Connect Data	00000000
Rem. Sta.	000000000000C	Bytes Sent	0	Aux. Structure	00000000
Rej/Disconn Reason	0	Bytes rcvd	0		
Queued for BDT	0	Total bytes map	0		
Queued Send Credit	0				

The preceding example shows the *CDT summary page*, and the first page of the detailed displays for each CDT.

2 SDA> SHOW CONNECTIONS /ADDRESS=801C27B0
VAXcluster data structures

--- Connection Descriptor Table (CDT) 801C27B0 ---

State: 0002 open Local Process: VMS\$VAXcluster
Blocked State: 0000 Remote Node::Process: DORIS::VMS\$VAXcluster

Local Con. ID	08FF0002	Datagrams sent	0	Message queue	empty
Remote Con. ID	33440003	Datagrams rcvd	0	Send Credit Q.	empty
Receive Credit	4	Datagram discard	0	PB address	80437E80
Send Credit	5	Messages Sent	267	PDT address	801CEA20
Min. Rec. Credit	0	Messages Rcvd.	289	Error Notify	80227950
Pend Rec. Credit	1	Send Data Init.	0	Receive Buffer	8039AF80
Initial Rec. Credit	5	Req Data Init.	0	Connect Data	80367C0C
Rem. Sta.	0000000000002	Bytes Sent	0	Aux. Structure	80367B90
Rej/Disconn Reason	0	Bytes rcvd	0		
Queued for BDT	0	Total bytes map	0		
Queued Send Credit	0				

The preceding example shows the use of the /ADDRESS qualifier to obtain information about a specific connection.

SHOW CPU

Displays information about the state of a processor at the time of the system failure.

FORMAT **SHOW CPU** *[cpu-id]*

PARAMETER *cpu-id*

Numeric value from 00 to 1F₁₆ indicating the identity of the processor for which context information is to be displayed. If you specify a value outside this range or the **cpu-id** of a processor that was not active at the time of the system failure, SDA displays the following message:

`%SDA-E-CPUNOTVLD, CPU not booted or CPU number out of range`

If you use the **cpu-id** parameter, the SHOW CPU command performs an implicit SET CPU command, making the processor indicated by **cpu-id** the current CPU for subsequent SDA commands. (See the description of the SET CPU command and Section 4 for information on how this can affect the CPU context—and process context—in which SDA commands execute.)

QUALIFIERS *None.*

DESCRIPTION

The SHOW CPU command displays crash information about the processor specified by **cpu-id** or, by default, the SDA current CPU, as defined in Section 4. You cannot use the SHOW CPU command when examining the running system with SDA.

The SHOW CPU command produces several displays. First, there is a brief description of the crash and its environment that includes the following:

- Reason for the bugcheck
- Name of the currently executing process. If no process has been scheduled on this processor, SDA displays the following message:

`Process currently executing: no processes currently scheduled on the processor`

- File specification of the image executing within the current process (if there is a current process)
- Interrupt priority level (IPL) of the processor at the time of the system failure

Next, the **general registers** display shows the contents of the processor's general purpose registers (R0 through R11), the AP, FP, SP, PC, and PSL at the time of the crash.

SYSTEM DUMP ANALYZER

SHOW CPU

The **processor registers** display consists of the following three parts:

- Common processor registers
- Processor-specific registers
- Stack pointers and memory interconnect silos

The first section includes registers, common to all VAX processors, that are used by VMS to maintain the current process's virtual address space, system space, or other system functions. The following registers are among those displayed:

- Program region (P0 space) base register (P0BR)
- Program region length register (POLR)
- Control region (P1 space) base register (P1BR)
- Control region length register (P1LR)
- System region (S0 space) base register (SBR)
- System region length register (SLR)
- Process control block base register (PCBB)
- System control block base register (SCBB)
- Asynchronous system trap level (ASTLVL)
- Software interrupt summary register (SISR)
- Internal clock control/status register (ICCS)
- System identification register (SID)

The second section of the **processor registers** display shows those registers that are specific to the type of VAX processor being examined. (The SHOW CRASH command displays the processor type.) The contents of the register display vary according to the type of processor involved in the crash and are used primarily in hardware diagnostics.

The final section of the display includes the five stack pointers: the interrupt stack pointer (ISP) and the four pointers of the kernel, executive, supervisor, and user stacks (KSP, ESP, SSP, and USP, respectively). Certain processors, such as the VAX 8800 and VAX 8600, also display the contents of the silos of their memory interconnects in this section.

The SHOW CPU command concludes with a listing of the spin locks, if any, owned by the processor at the time of the crash, reproducing some of the information given by the SHOW SPINLOCKS command. The spin lock display includes the following information:

- Name of the spin lock
- Address of the spin lock data structure (SPL)
- IPL and rank of the spin lock
- Number of processors waiting for this processor to release the spin lock

SYSTEM DUMP ANALYZER

SHOW CPU

- Indication of the depth of this processor's ownership of the spin lock. A number greater than 1 indicates that this processor has nested acquisitions of the spin lock.

EXAMPLE

```
SDA> SHOW CPU
CPU 00 Processor crash information
-----
```

CPU 00 reason for Bugcheck: INVEXCEPTN, Exception while above ASTDEL or on interrupt stack

Process currently executing: NETACP

Current image file: \$254\$DUA200:[SYS6.SYSCOMMON.]<SYSEXE>NETACP.EXE;3

Current IPL: 8 (decimal)

General registers:

R0 = 00000008	R1 = 00080000	R2 = 8047FC40	R3 = 000003AC
R4 = 00000002	R5 = 8047FC40	R6 = 00000036	R7 = 00000000
R8 = 00000000	R9 = 00000062	R10 = 7FFE7D70	R11 = 0000747C
AP = 0000BE34	FP = 7FFE7DD0	SP = 7FFE7D30	PC = 80146682
PSL = 00080009			

Processor registers:

POBR = 816EB600	SBR = 01A6A800	ASTLVL = 00000004
POLR = 00000C0C	SLR = 00065600	SISR = 00000000
P1BR = 80FFCE00	PCBB = 008AF2A0	ICCS = 00000041
P1LR = 001FFC5F	SCBB = 01A62600	SID = 067F014F
ICR = FFFFEDEA	REVR1 = 11121111	NMIFSR = 000C0000
TODR = 2B914C0F	REVR2 = FF00FF12	NMIEAR = 2243F830
COR = 00000001	CPUINFO= 000009F7	
ISP = 8016AC00		
KSP = 7FFE7D30		
ESP = 7FFE9E00		
SSP = 7FFEDE00		
USP = 7FF8E590		

NMI bus silo:

```
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
```

SYSTEM DUMP ANALYZER

SHOW CPU

Spinlocks currently owned by CPU 00

IOLOCK8		Address	: 80185E50
Owner CPU ID	: 00	IPL	: 08
Ownership Depth	: 0001	Rank	: 14
CPUs Waiting	: 0000	Index	: 34

SDA> EXAMINE R5

R5: 8047FC40 "@G."

SDA> SHOW PROCESS

Process index: 000D Name: NETACP Extended PID: 33C0010D

Process status: 00148001 RES,NOACNT,PHDRES,LOGIN

SDA> SHOW CPU 01

CPU 01 Processor crash information

CPU 01 reason for Bugcheck: CPUEXIT, Shutdown requested by another CPU

Process currently executing: no processes currently scheduled on this CPU

Current IPL: 31 (decimal)

No spinlocks currently owned by CPU 01

SDA> EXAMINE R5

R5: 83ED5E00 ".^."

SDA> SHOW PROCESS

%SDA-E-BADPROC, no such process

This SDA session illustrates the output of the SHOW CPU command in the analysis of a crash dump from a VAX 8800 multiprocessing system with two active processors. The first SHOW CPU command displays the crash information particular to CPU 00, which initially posted an INVEXCEPTN bugcheck from within process NETACP and then requested CPU 01 to take a bugcheck (CPUEXIT) as well. That the crash occurred at IPL 8 signifies, perhaps, that a driver fork process is involved.

The second instance of the SHOW CPU command (SHOW CPU 01) corroborates that CPU 01 was requested to crash by CPU 00.

Significantly, the second SHOW CPU command changes both the SDA current CPU context and current process context. The two EXAMINE R5 commands are executed under different CPU contexts as the values they produce differ. In the CPU context of CPU 00, the current process context is that of process NETACP. There is no current process on CPU 01; thus, SDA process context is initially undefined when its CPU context is changed to that of CPU 01.

SHOW CRASH

In the analysis of a system failure, displays information about the state of the system at the time of the failure. In the analysis of a running system, provides information identifying the system.

FORMAT **SHOW CRASH**

PARAMETERS *None.*

QUALIFIERS *None.*

DESCRIPTION The SHOW CRASH command has two different manifestations, depending upon whether it is issued within the analysis of a running system or within the analysis of a system failure.

In either case, if the SDA current CPU context is not that of the processor that signaled the bugcheck, the SHOW CRASH command performs an implicit SET CPU command to make that processor the SDA current CPU. (See the description of the SET CPU command and Section 4 for a discussion of how this can affect the CPU context—and process context—in which SDA commands execute.)

When used during the analysis of a *running system*, the SHOW CRASH command produces a display that describes the system and the version of VMS that it is running. The **system crash information** display contains the following information:

- Date and time that the ANALYZE/SYSTEM command was issued (entitled "Time of system crash" in the display)
- Name and version number of the operating system
- Major and minor IDs of the operating system
- Identity of the VAX system, including an indication of its VAXcluster membership
- CPU ID of the primary CPU
- Two bit masks indicating which processors in the system are active and which are available for booting, respectively

When used during the analysis of a *system failure*, the SHOW CRASH command produces several displays that identify the system and describe its state at the time of the failure.

The **system crash information** display in this context provides the following information:

- Date and time of the system crash.
- Name and version number of the operating system.

SYSTEM DUMP ANALYZER

SHOW CRASH

- Major and minor IDs of the operating system.
- Identity of the VAX system, including an indication of its VAXcluster membership and the location of the primary CPU in a multiprocessing configuration.
- CPU IDs of both the primary CPU and the CPU that initiated the bugcheck. In a VAX uniprocessor system, these IDs are identical.
- Two bit masks indicating which processors in the system are active and which are available for booting, respectively.
- For each active processor in the system, the name of the bugcheck that caused the failure. Generally, there will be only one significant bugcheck in the system. All other processors typically display the following as their reason for taking a bugcheck:

CPUEXIT, Shutdown requested by another CPU

Subsequent screens of the SHOW CRASH command display information about the state of each active processor on the system at the time of the system failure. The information in these screens is identical to that produced by the SHOW CPU command, including the general purpose registers, processor-specific registers, stack pointers, and record of spin lock ownership. The first such screen presents information about the processor that caused the crash; others follow according to the numerical order of their CPU IDs.

EXAMPLES

```
1 $ ANALYZE/SYSTEM
VAX/VMS System analyzer

SDA> SHOW CRASH
System crash information
-----
Time of system crash: 25-FEB-1988 11:18:06.84

Version of system: VAX/VMS VERSION 5.0
System Version Major ID/Minor ID: 10/11

VAXcluster node: BIGTOP, a VAX 8800 - primary CPU (left) was booted
Primary CPU ID: 01

Bitmask of CPUs active/available: 00000003/00000003
SDA> SHOW PROCESS
%SDA-E-BADPROC, no such process
```

When issued from within the analysis of a running system, the SHOW CRASH command displays the time the ANALYZE/SYSTEM command was issued as the "Time of system crash." The display indicates that the VAX system in use is a VAX 8800 multiprocessing system, the left CPU of which is the primary CPU. The bit mask indicates that there are two processors available and both are running.

Note that there is no defined SDA current process at this time.

SYSTEM DUMP ANALYZER

SHOW CRASH

```
2 $ ANALYZE/CRASH SYS$SYSTEM
VAX/VMS System dump analyzer

Dump taken on 23-FEB-1988 12:44:30.23
INVEXCEPTN, Exception while above ASTDEL or on interrupt stack

SDA> SHOW CRASH
System crash information①
-----
Time of system crash: 23-FEB-1988 12:44:30.23

Version of system: VAX/VMS VERSION 5.0
System Version Major ID/Minor ID: 10/11

VAXcluster node: MOOSE, a VAX 8800 - primary CPU (left) was booted
Crash CPU ID/Primary CPU ID: 00/01
Bitmask of CPUs active/available: 00000003/00000003

CPU bugcheck codes:②
CPU 00 -- INVEXCEPTN, Exception while above ASTDEL or on interrupt stack
1 other -- CPUEXIT, Shutdown requested by another CPU

CPU 00 Processor crash information
-----

CPU 00 reason for Bugcheck: INVEXCEPTN, Exception while above ASTDEL or on interrupt stack③

Process currently executing on this CPU: NETACP③

Current image file: $254$DUA200:[SYS6.SYSCOMMON.][SYSEXE]NETACP.EXE;3

Current IPL: 8 (decimal)④

General registers:
RO = 00000008 R1 = 00080000 R2 = 8047FC40 R3 = 000003AC
R4 = 00000002 R5 = 8047FC40 R6 = 00000036 R7 = 00000000
R8 = 00000000 R9 = 00000062 R10 = 7FFE7D70 R11 = 0000747C
AP = 0000BE34 FP = 7FFE7DD0 SP = 7FFE7D30 PC = 80146682
PSL = 00080009

Processor registers:
POBR = 816EB600 SBR = 01A6A800 ASTLVL = 00000004
POLR = 00000C0C SLR = 00065600 SISR = 00000000
P1BR = 80FFCE00 PCBB = 008AF2A0 ICCS = 00000041
P1LR = 001FFC5F SCBB = 01A62600 SID = 067F014F

ICR = FFFEFEDEA REVR1 = 11121111 NMIFSR = 000C0000
TODR = 2B914COF REVR2 = FFO0FF12 NMIEAR = 2243F830
COR = 00000001 CPUINFO= 000009F7 MEMCSRO= 000700F0
NBIA0 CSRO = 00203810 NBIA1 CSRO = 00000000

ISP = 8016AC00
KSP = 7FFE7D30
ESP = 7FFE9E00
SSP = 7FFEDE00
USP = 7FF8E590
```

SYSTEM DUMP ANALYZER

SHOW CRASH

NMI bus silo:

00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000

Spinlocks currently owned by CPU 00

IOLCK8		Address	: 80185E50
Owner CPU ID	: 00	IPL	: 08
Ownership Depth	: 0001	Rank	: 14
CPUs Waiting	: 0000	Index	: 34

CPU 01 Processor crash information

CPU 01 reason for Bugcheck: CPUEXIT, Shutdown requested by another CPU

Process currently executing on this CPU: None

Current IPL: 31 (decimal)

General registers:

R0	= 00000020	R1	= 00000000	R2	= 8000CA78	R3	= 80DAF000
R4	= 80487000	R5	= 83ED5E00	R6	= 7FFA4188	R7	= 7FF28EB8
R8	= 7FF28E68	R9	= 7FFA2808	R10	= 7FFA4000	R11	= 7FFE0070
AP	= 7FF28D90	FP	= 7FF28D98	SP	= 80DAFBF8	PC	= 80765465
PSL	= 041F0000						

Processor registers:

POBR	= 83EE8E00	SBR	= 01A6A800	ASTLVL	= 00000004
POLR	= 000001C1	SLR	= 00065600	SISR	= 00000000
P1BR	= 837FA600	PCBB	= 00BB62A0	ICCS	= 00000041
P1LR	= 001FF935	SCBB	= 01A62600	SID	= 06FF014F
ICR	= FFFFE7C1	REVR1	= 11121111	NMIFSR	= 000C0000
TODR	= 2B914C0F	REVR2	= FF00FF12	NMIEAR	= 24080000
COR	= 00000001	CPUINFO	= 000009F7	MEMCSRO	= 000700F0
NBIA0 CSRO	= 00203810	NBIA1 CSRO	= 00000000		
ISP	= 80DAFBF8				
KSP	= 7FFE7E00				
ESP	= 7FFE9E00				
SSP	= 7FFED04E				
USP	= 7FF28D90				

SYSTEM DUMP ANALYZER

SHOW CRASH

NMI bus silo:

```
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
```

No spinlocks currently owned by CPU 01

This long display reflects the output of the SHOW CRASH command within the analysis of a system failure that occurred on a VAX 8800 multiprocessing system.

The first part of the display includes the following information:

- ① Identification of the system and the version of VMS it was running at the time of the crash
- ② Indication that the failed processor (CPU 00) was not the primary (CPU 01), but requested CPU 01 to take a CPUEXIT bugcheck. (CPU 01 was, in fact, idle at the time of the crash.)

The next part of the display shows information particular to CPU 00:

- ③ CPU 00 encountered an INVEXCEPTN bugcheck while executing the NETACP process.
- ④ Although the next step in the analysis may be to examine the interrupt stack of CPU 00, the fact that the failure occurred at IPL 8 may be grounds to suspect that an I/O driver is involved.

At the end of the example, SDA CPU context remains that of CPU 00; its current process context is that of the NETACP process.

SYSTEM DUMP ANALYZER

SHOW DEVICE

SHOW DEVICE

Displays a list of all devices in the system and their associated data structures, or displays the data structures associated with a given device or devices.

FORMAT

SHOW DEVICE { *device-name*
/ADDRESS=*ucb-address* }

PARAMETER

device-name

Device or devices for which data structures are to be displayed. There are several uses of the **device-name** parameter:

To display the structures for . . .	Action
All devices in the system	Do <i>not</i> specify a device-name (for example, SHOW DEVICE).
A single device	Specify an entire device-name (for example, SHOW DEVICE VTA20).
All devices of a certain type on a single controller	Specify only the device type and controller designation (for example, SHOW DEVICE RTA or SHOW DEVICE RTB).
All devices of a certain type on any controller	Specify only the device type (for example, SHOW DEVICE RT).
All devices whose names begin with a certain character or character string	Specify the character or character string (for example, SHOW DEVICE D).
All devices on a single node or HSC	Specify only the node name or HSC name (for example, SHOW DEVICE GREEN\$).

In a VAXcluster environment, device information is displayed for each device in the VAXcluster with the specified **device-name**. You can limit the display to those devices that are on a particular node or HSC by specifying the node name or HSC name as part of the **device-name** (for example, GREEN\$D or GREEN\$DB).

QUALIFIER

/ADDRESS=*ucb-address*

Indicates the device for which data structure information is to be displayed by the address of its unit control block (UCB). The /ADDRESS qualifier is thus an alternate method of supplying a device name to the SHOW DEVICE command. If both the **device-name** parameter and the /ADDRESS qualifier appear in a single SHOW DEVICE command, SDA responds only to the parameter or qualifier that appears first.

SYSTEM DUMP ANALYZER

SHOW DEVICE

DESCRIPTION

The SHOW DEVICE command produces several displays taken from system data structures that describe the devices in the system configuration.

If you use the SHOW DEVICE command to display information for more than one device or one or more controllers, it initially produces the **DDB list** display to provide a brief summary of the devices for which it renders information in subsequent screens.

Information in the **DDB list** appears in six columns, the contents of which are as follows:

- 1 Address of the device data block (DDB)
- 2 Controller name
- 3 Name of the ancillary control process (ACP) or extended QIO processor (XQP) associated with the device
- 4 Name of the device driver
- 5 Address of the driver prologue table (DPT)
- 6 Size of the DPT

The SHOW DEVICE command then produces a display of information pertinent to the device controller. This display includes information gathered from the following structures:

- Device data block (DDB)
- Primary channel request block (CRB)
- Interrupt dispatch block (IDB)
- Driver dispatch table (DDT)

If the controller is an HSC controller, SHOW DEVICE also displays information from its system block (SB) and each path block (PB).

Many of these structures contain pointers to other structures and driver routines. Most notably, the DDT display points to various routines located within driver code, such as the start I/O routine, unit initialization routine, and cancel I/O routine.

For each device unit subject to the SHOW DEVICE command, SDA displays information taken from its unit control block, including a list of all I/O request packets (IRPs) in its I/O request queue. For certain mass-storage devices, SHOW DEVICE also displays information from the primary class driver data block (CDDB), the volume control block (VCB), and the ACP queue block (AQB). For units that are part of a shadow set, SDA displays a summary of shadow set membership.

For a detailed explanation of I/O data structures displayed by SDA, consult the *VMS Device Support Manual*.

SYSTEM DUMP ANALYZER

SHOW DEVICE

EXAMPLES

```

1 SDA> SHOW DEVICE VTA20
VTA20 ==> LTA20                                VT200_Series      UCB address: 8042E4C0

Device status: 00010110 online,bsy,deleteucb
Characteristics: 0C040007 rec,ccl,trm,avl,idv,odv
                  00000200 nnm

Owner UIC [000001,000004]  Operation count      5793  ORB address  8042E590
      PID      00010064  Error count          0  DDB address  80CEF2E0
Class/Type      42/6E  Reference count      2  DDT address  807696FB
Def. buf. size   80  BOFF                0155  CRB address  80BC8B00
DEVDEPEND      180093A0  Byte count          0100  IRP address  80BE2B00
DEVDEPN2       7962100C  SVAPTE             804801C0  I/O wait queue  empty
FLCK/DLCK      00000012  DEVSTS             0000

                                I/O request queue
                                -----
STATE  IRP      PID  MODE CHAN  FUNC   WCB   EFN   AST   IOSB  STATUS
C      80BE2B00 00010064 E   FFC0  C000  00000000  29  80127458  7FFA800C  0003
      nop bufio,func
  
```

This example reproduces the SHOW DEVICE display for a single device unit, VTA20. Whereas this display lists information from the UCB for VTA20, including some addresses of key data structures and a list of pending I/O requests for the unit, it does not display information about the controller or its device driver. To display the latter sort of information, specify the **device-name** as VTA (for example, SHOW DEVICE VTA).

```

2 SDA> SHOW DEVICE DU
I/O data structures
-----

                                DDB list
                                -----

      Address      Controller      ACP      Driver      DPT      DPT size
      -----      -
      80D0B3C0     BLUES$DUA     F11XQP    DSDRIVER    807735B0  679D
      8000B2B8     RED$DUA       F11XQP    DSDRIVER    807735B0  679D
      80D0B9C0     RED$DUS       F11XQP    DSDRIVER    807735B0  679D
      80D08BA0     BIGTOP$DUA    F11XQP    DSDRIVER    807735B0  679D
      80D08AEO     TIMEIN$DUA    F11XQP    DSDRIVER    807735B0  679D
  
```

Press RETURN for more.

This excerpt from the output of the SHOW DEVICE DU command illustrates the format of the **DDB list** display. In this case, the **DDB list** concerns itself with those devices whose device type begins with DU (that is, DUA and DUS). It displays devices of these types attached to various HSCs (RED\$ and BLUES\$) and systems in a VAXcluster (BIGTOP\$ and TIMEIN\$).

Following the **DDB list**, SHOW DEVICE DU produces displays for each controller and each unit on each controller, as illustrated in the next example.

SYSTEM DUMP ANALYZER

SHOW DEVICE

```

Owner UIC [100001,000063] Operation count      55595 ORB address  803B9D90
                          PID          00000000 Error count      0 DDB address  80D0B9C0
Alloc. lock ID 00010161 Reference count    3 DDT address  80773640
Alloc. class   254 Online count      2 VCB address  8044D940
Class/Type     01/15 BOFF          0000 CRB address  80BF7000
Def. buf. size 512 Byte count      0A00 PDT address  803B38D0
DEVDEPEND     04E00E33 SVAPTE          835C7738 CDDB address 803B4150
DEVDEPEND2    00000000 DEVSTS          0004 I/O wait queue empty
FLCK/DLCK     00000012 RWAITCNT        0000
  
```

--- Primary Class Driver Data Block (CDDB) 803B4150 ---

```

Status:          1040 alcls_set,bshadow
Controller Flags: 80D6 cf_shadw,cf_mlths,cf_this,cf_misc,cf_attn,cf_replc

Allocation class 254 CDRP Queue      80BD1170 DDB address  8000B2B8
System ID        0000FFF2 Restart Queue empty CRB address  80BF7000
                  0000 DAP Count      1 CDDB link   803C01C0
Contrl. ID       0000FFF2 Contr. timeout 75 PDT address  803B38D0
                  01010000 Reinit Count  0 Original UCB 00000000
Response ID      00000000 Wait UCB Count 0 UCB chain   803B89A0
MSCP Cmd status FFFFFFFF
  
```

*** I/O request queue is empty ***

--- Volume Control Block (VCB) 8044D940 ---

```

Volume: VMSCMSMASTER Lock name: VMSCMSMASTER
Status: AO extfid,system
Status2: 15 writethru,mountver,nohighwater
Shadow status: 21 shadmast,mvbegin
  
```

```

Mount count      1 Rel. volume      0 AQB address  80D0BAE0
Transactions     3 Max. files      111384 RVT address  803B9C60
Free blocks      205989 Rsvd. files    9 FCB queue   80BD87B0
Window size      7 Cluster size    3 Cache blk.  8044DA30
Vol. lock ID     00010167 Def. extend sz. 5 Shadow mem. FL 80CF5C40
Block. lock ID   01A50139 Record size    0 Shadow mem. BL 80CF5BE0
Shadow lock ID   00010168
  
```

--- Shadow set \$254\$DUS3 member summary ---

Volume: JAZZLORE

Physical unit	Primary path	Secondary path	Member status
\$254\$DUA129	RED	-- none --	Shadow set member
\$254\$DUA139	RED	-- none --	Shadow set member

--- ACP Queue Block (aqb) 80D0BAE0 ---

ACP requests are serviced by the eXtended Qio Processor (XQP)

Status: 14 defsys,xqioproc

```

Mount count      56 ACP type      f11v2 Request queue 00000000
                  ACP class      0
  
```

*** ACP request queue is empty ***

```

RED$DUS5          RA80          UCB address: 803B9DF0
  
```

SYSTEM DUMP ANALYZER

SHOW DEVICE

```
Device status: 00021810 online,valid,unload,lcl_valid  
Characteristics: 1C4D4008 dir,fod,shr,avl,mnt,elg,idv,odv,rnd  
000002A1 clu,mscp,svr,nm
```

The previous example illustrates the output of the command SHOW DEVICE DUS command, where two shadow sets (RED\$DUS3 and RED\$DUS5) are associated with the HSC RED\$. There is a controller display for RED\$DUS, and a unit display for each of the two shadow sets.

SYSTEM DUMP ANALYZER

SHOW EXECUTIVE

SHOW EXECUTIVE

Displays the location and size of each loadable image that makes up the VMS executive.

FORMAT	SHOW EXECUTIVE
---------------	-----------------------

PARAMETERS	<i>None.</i>
-------------------	--------------

QUALIFIERS	<i>None.</i>
-------------------	--------------

DESCRIPTION	<p>The VMS executive consists of a fixed portion and a loadable portion. The fixed portion is known as SYS\$SYSTEM:SYS.EXE and consists of three parts:</p>
--------------------	---

- System service dispatch vectors
- Universal executive routine vectors
- Globally referenced data cells

The loadable portion consists of a number of independent images that perform the work of the VMS operating system.

The SHOW EXECUTIVE command lists the location and size of each image within the loadable portion of the executive image. It can thus enable you to determine whether a given memory address falls within the range occupied by a particular loadable image. (Table SDA-5 describes the contents of each loadable image.)

By default, SDA displays each location within the loadable portion of the executive as an offset from the beginning of one of the loadable images: for instance, EXCEPTION+00282. Similarly, those symbols that represent system services point to the vector region and not the system service's loadable code. When tracing the course of a system failure through the listings of modules contained within a given loadable executive image, you may find it useful to load into the SDA symbol table all global symbols and global entry points defined within one or all modules that make up the loadable portion of the executive image. See the description of the READ command for additional information.

The SHOW EXECUTIVE command usually shows all components of the executive image, as illustrated in the following example. In rare circumstances, you may obtain a partial listing. For instance, once it has loaded the EXCEPTION module (in the INIT phase of system initialization), the system can successfully post a bugcheck exception and save a crash dump. Later, if the system should fail sometime during initialization, it may not have been able to load some of the modules that appear above EXCEPTION in the SHOW EXECUTIVE display (see the following example).

SYSTEM DUMP ANALYZER

SHOW EXECUTIVE

EXAMPLE

```
SDA> SHOW EXECUTIVE
VMS Executive Layout
```

```
-----
Image                Base                End                Length
SYSMSG               8015AA00           80183600           00028C00
RECOVERY_UNIT_SERVICES 80211400           80212000           00000C00
RMS                  80183600           801A7E00           00024800
CPULOA              801B2800           801B3200           00000A00
LMF$GROUP_TABLE     801B3800           801B3C00           00000400
SYSLICENSE          801B4000           801B5400           00001400
SYSGETSYI           801B5A00           801B7000           00001600
SYSDEVICE           801B7400           801B8A00           00001600
MESSAGE_ROUTINES    801B9000           801BB600           00002600
EXCEPTION           801CBA00           801D3E00           00008400
LOGICAL_NAMES       801D4600           801D6000           00001A00
SECURITY            801D6600           801D7C00           00001600
LOCKING             801D8200           801DA800           00002600
PAGE_MANAGEMENT     801DAE00           801E2600           00007800
WORKING_SET_MANAGEMENT 801E2E00           801E7200           00004400
IMAGE_MANAGEMENT    801E7C00           801EA400           00002800
EVENT_FLAGS_AND_ASTS 801EAA00           801EBE00           00001400
IO_ROUTINES         801EC400           801F2C00           00006800
PROCESS_MANAGEMENT  801F3200           801F9400           00006200
ERRORLOG            80204C00           80205600           00000A00
PRIMITIVE_IO        80205C00           80206C00           00001000
SYSTEM_SYNCHRONIZATION 80207000           80208C00           00001C00
SYSTEM_PRIMITIVES   80209200           8020C400           00003200
```

The SHOW EXECUTIVE command displays the location and length of the loadable images included in the VMS executive.

SYSTEM DUMP ANALYZER

SHOW HEADER

SHOW HEADER

Displays the header of the dump file.

FORMAT SHOW HEADER

PARAMETERS *None.*

QUALIFIERS *None.*

DESCRIPTION The SHOW HEADER command produces a 10-column display, each line of which displays both the hexadecimal and ASCII representation of the contents of the dump file header in 32-byte intervals. Thus, the first eight columns, when read right to left, represent the hexadecimal contents of 32 bytes of the header; similarly, the ninth column, when read left to right, records the ASCII equivalent of the contents. (Note that the period character (.) in this column indicates an ASCII character that cannot be displayed.)

After it displays the contents of the first header block, the SHOW HEADER command displays the hexadecimal contents of the saved error log buffers.

See the *VAX/VMS Internals and Data Structures* manual for a discussion of the information contained in the dump file header.

EXAMPLE

SDA> SHOW HEADER

Dump file header

```
-----  
7FF03944 7FFED04E . . . 000000C1 00000000 . . . . . N . . D9 . . 00000000  
00000000 00000000 . . . 00040000 80185200 . R . . . . . 00000020  
00000000 00000000 . . . 00000000 00000000 . . . . . 00000040  
00020000 00000000 . . . 15000011 00000000 . . . . . 00000060  
414E454C 45480800 . . . 0000012C 00000000 . . . . . GARNER 00000080  
FE9E007F F74D7C0A . . . 00000000 00002020 . . . . . % . @ . o41 . . . . M . . . . 000000A0  
.  
.  
.
```

Saved error log messages

```
-----  
00000000 00000009 . . . 801D8739 00000300 . . . 9 . . . . . 5 . . . . . 801D8600  
7B0090AC 2FCBCEC2 . . . 414E454C 45480800 . . GARNER . . . . . & . zxcv . 0 . . . 801D8620  
00202041 4E454C45 . . . 01080100 0000C30A . A . . . . . d . . . . . GARNER . . 801D8640  
.  
.  
.
```

SYSTEM DUMP ANALYZER

SHOW HEADER

The SHOW HEADER command displays the contents of the dump file's header from address $6B0_{16}$ to address $C90_{16}$. Ellipses indicate hexadecimal information omitted from the display.

SYSTEM DUMP ANALYZER

SHOW LOCK

Table SDA-7 (Cont.) Contents of the SHOW LOCK and SHOW PROCESS/LOCKS Displays

Display Element	Description
LKB	Address of the lock block (LKB). If a blocking AST has been enabled for this lock, the notation "BLKAST" appears next to the LKB address.
Resource	Dump of the resource name. The two leftmost columns of the dump show its contents as hexadecimal values, the least significant byte being represented by the rightmost two digits. The rightmost column represents its contents as ASCII text, the least significant byte being represented by the leftmost character.
Status	Status of the lock, information used internally by the VMS lock manager.
Length	Length of the resource name.
---	Processor access mode of the name space in which the resource block (RSB) associated with the lock resides.
---	Owner of the resource. Certain resources owned by the VMS operating system list "System" as the owner. Resources owned by a group have the number (in octal) of the owning group in this field.
---	Indication of whether the lock is mastered on the local system or is a process copy.

EXAMPLE

```
SDA> SHOW LOCK
Lock database
-----
```

```
Lock id: 00010001  PID: 00000000  Flags: NOQUEUE SYNCSTS SYSTEM
Par. id: 00000000  Granted at  EX          CVTSYS
Sublocks: 1
LKB: 80D0B8A0
Resource: 5F535953 24535953  SYS$SYS_  Status: NOQUOTA
Length 16 00000000 4C774449  IDwL....
Exec. mode 00000000 00000000  ....
System 00000000 00000000  ....
Local copy
```

```
Lock id: 00010004  PID: 00000000  Flags: CONVERT SYNCSTS CVTSYS
Par. id: 00000000  Granted at  CR
Sublocks: 16
LKB: 80D091A0  BLKAST
Resource: 4D567624 42313146  F11B$VM  Status: NOQUOTA
Length 18 20204E41 4A353153  S15JAN
Kernel mode 00000000 00002020  ....
System 00000000 00000000  ....
Local copy
```

SYSTEM DUMP ANALYZER

SHOW LOCK

Lock id: 00280009 PID: 00000000 Flags: VALBLK CONVERT SYNCSTS
Par. id: 00000000 Granted at CR NOQUOTA CVTSYS
Sublocks: 0
LKB: 80CDA880
Resource: 52414B5F 24535953 SYS\$_KAR Status: MSTCPY
Length 17 30415544 24455441 ATE\$DUAO
Kernel mode 00000000 0000003A :.....
System 00000000 00000000 :.....
Master copy of lock 001C00F5 on system 000100A1

SDA> SHOW RESOURCE/LOCK=280009

Resource database

Address of RSB: 80BD2150 Group grant mode: CR
Parent RSB: 00000000 Conversion grant mode: CR
Sub-RSB count: 0 BLKAST count: 0
Value block: 00000000 00000000 00000000 00000019 Seq. #: 0000002D
Resource: 52414B5F 24535953 SYS\$_KAR
Length 17 30415544 24455441 ATE\$DUAO CSID: 00000000
Kernel mode 00000000 0000003A :.....
System 00000000 00000000 :.....

Granted queue (Lock ID / Gr mode):

00DA1269	CR	00280009	CR	0094054D	CR
00270B9F	CR	00D70BFE	CR	00D0F4F	CR
000D1017	CR	00601418	CR	01131450	CR
000F1964	CR	000200DF	CR		

Conversion queue (Lock ID / Gr/Rq mode):

*** EMPTY QUEUE ***

Waiting queue (Lock ID / Rq mode):

*** EMPTY QUEUE ***

This SDA session shows the output of the SHOW LOCK command for several locks. The SHOW RESOURCE command, executed for the last displayed lock, verifies that the lock is in the resource's granted queue, among many other locks given concurrent read (CR) access to the resource. (See Table SDA-13 for a full explanation of the contents of the display of the SHOW RESOURCE command.)

SHOW PAGE_TABLE

Displays a range of system page table entries, the entire system page table, or the entire global page table.

FORMAT **SHOW PAGE_TABLE** *[/qualifier[...]] [range]*

PARAMETER *range*
Range of virtual addresses for which SDA is to display page table entries. You can express a range using the following syntax:

m:n Range of virtual addresses from *m* to *n*

m;n Range of virtual addresses starting at *m* and continuing for *n* bytes

QUALIFIERS **/GLOBAL**
Lists the global page table.

/SYSTEM
Lists the system page table.

/ALL
Lists both the global and system page tables. This is default behavior of SHOW PAGE_TABLE.

DESCRIPTION For each virtual address displayed by the SHOW PAGE_TABLE command, the first six columns of the listing provide the associated page table entry and describe its location, characteristics, and contents (see Table SDA-8). SDA obtains this information from the system page table.

If the virtual page has been mapped to a physical page, the last nine columns of the listing include information from the page frame number (PFN) database (see Table SDA-9). Otherwise, the section is left blank.

SDA indicates pages are inaccessible by displaying the following message:

----- n NULL PAGES

Here, *n* indicates the number of inaccessible pages.

SYSTEM DUMP ANALYZER

SHOW PAGE_TABLE

Table SDA-8 Virtual Page Information in the SHOW PAGE_TABLE Display

Value	Meaning																		
ADDRESS	System virtual address that marks the base of the virtual page																		
SVAPTE	System virtual address of the page table entry that maps the virtual page																		
PTE	Contents of the page table entry, a longword that describes a system virtual page																		
Type	Type of virtual page. There are the following eight types:																		
	<table border="1"> <thead> <tr> <th>Type</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>VALID</td> <td>Valid page (in main memory)</td> </tr> <tr> <td>TRANS</td> <td>Transitional page (between main memory and page lists)</td> </tr> <tr> <td>DZERO</td> <td>Demand-allocated, zero-filled page</td> </tr> <tr> <td>PGFIL</td> <td>Page within a paging file</td> </tr> <tr> <td>STX</td> <td>Section table's index page</td> </tr> <tr> <td>GPTX</td> <td>Index page for a global page table</td> </tr> <tr> <td>IOPAG</td> <td>Page in I/O address space</td> </tr> <tr> <td>NXMEM</td> <td>Page not represented in physical memory. The page frame number (PFN) of this page is not mapped by any of the system's memory controllers. This indicates an error condition.</td> </tr> </tbody> </table>	Type	Meaning	VALID	Valid page (in main memory)	TRANS	Transitional page (between main memory and page lists)	DZERO	Demand-allocated, zero-filled page	PGFIL	Page within a paging file	STX	Section table's index page	GPTX	Index page for a global page table	IOPAG	Page in I/O address space	NXMEM	Page not represented in physical memory. The page frame number (PFN) of this page is not mapped by any of the system's memory controllers. This indicates an error condition.
Type	Meaning																		
VALID	Valid page (in main memory)																		
TRANS	Transitional page (between main memory and page lists)																		
DZERO	Demand-allocated, zero-filled page																		
PGFIL	Page within a paging file																		
STX	Section table's index page																		
GPTX	Index page for a global page table																		
IOPAG	Page in I/O address space																		
NXMEM	Page not represented in physical memory. The page frame number (PFN) of this page is not mapped by any of the system's memory controllers. This indicates an error condition.																		
PROT	Protection: a code, derived from bits in the PTE, that designates the type of access (read and/or write) granted to processor access modes (kernel, executive, supervisor, or user)																		
Bits	Letters that represent the setting of a bit or a combination of bits in the PTE. These bits indicate attributes of a page. The following codes are listed:																		
	<table border="1"> <thead> <tr> <th>Code</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>M</td> <td>Page has been modified.</td> </tr> <tr> <td>L</td> <td>Page is locked into a working set.</td> </tr> <tr> <td>K</td> <td>Owner can access the page in kernel mode.</td> </tr> <tr> <td>E</td> <td>Owner can access the page in executive mode.</td> </tr> <tr> <td>S</td> <td>Owner can access the page in supervisor mode.</td> </tr> <tr> <td>U</td> <td>Owner can access the page in user mode.</td> </tr> </tbody> </table>	Code	Meaning	M	Page has been modified.	L	Page is locked into a working set.	K	Owner can access the page in kernel mode.	E	Owner can access the page in executive mode.	S	Owner can access the page in supervisor mode.	U	Owner can access the page in user mode.				
Code	Meaning																		
M	Page has been modified.																		
L	Page is locked into a working set.																		
K	Owner can access the page in kernel mode.																		
E	Owner can access the page in executive mode.																		
S	Owner can access the page in supervisor mode.																		
U	Owner can access the page in user mode.																		

SYSTEM DUMP ANALYZER

SHOW PAGE_TABLE

Table SDA-9 Physical Page Information in the SHOW PAGE_TABLE Display

Category	Meaning																		
PAGTYP	<p>Type of physical page, one of the following six types:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Page Type</th> <th style="text-align: left;">Meaning</th> </tr> </thead> <tbody> <tr> <td>PROCESS</td> <td>Page is part of process space.</td> </tr> <tr> <td>SYSTEM</td> <td>Page is part of system space.</td> </tr> <tr> <td>GLOBAL</td> <td>Page is part of a global section.</td> </tr> <tr> <td>PPGTBL</td> <td>Page is part of a process's page table.</td> </tr> <tr> <td>GPGTBL</td> <td>Page is part of a global page table.</td> </tr> <tr> <td>GBLWRT</td> <td>Page is part of a global, writable section.</td> </tr> </tbody> </table>	Page Type	Meaning	PROCESS	Page is part of process space.	SYSTEM	Page is part of system space.	GLOBAL	Page is part of a global section.	PPGTBL	Page is part of a process's page table.	GPGTBL	Page is part of a global page table.	GBLWRT	Page is part of a global, writable section.				
Page Type	Meaning																		
PROCESS	Page is part of process space.																		
SYSTEM	Page is part of system space.																		
GLOBAL	Page is part of a global section.																		
PPGTBL	Page is part of a process's page table.																		
GPGTBL	Page is part of a global page table.																		
GBLWRT	Page is part of a global, writable section.																		
LOC	<p>Location of the page within the system, one of the following eight types:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Location</th> <th style="text-align: left;">Meaning</th> </tr> </thead> <tbody> <tr> <td>ACTIVE</td> <td>Page is in a working set.</td> </tr> <tr> <td>MDFYLIST</td> <td>Page is in the modified page list.</td> </tr> <tr> <td>FREELIST</td> <td>Page is in the free page list.</td> </tr> <tr> <td>BADLIST</td> <td>Page is in the bad page list.</td> </tr> <tr> <td>RELPEND</td> <td>Release of the page is pending.</td> </tr> <tr> <td>RDERROR</td> <td>Page has had an error during an attempted read operation.</td> </tr> <tr> <td>PAGEOUT</td> <td>Page is being written into a paging file.</td> </tr> <tr> <td>PAGEIN</td> <td>Page is being brought into memory from a paging file.</td> </tr> </tbody> </table>	Location	Meaning	ACTIVE	Page is in a working set.	MDFYLIST	Page is in the modified page list.	FREELIST	Page is in the free page list.	BADLIST	Page is in the bad page list.	RELPEND	Release of the page is pending.	RDERROR	Page has had an error during an attempted read operation.	PAGEOUT	Page is being written into a paging file.	PAGEIN	Page is being brought into memory from a paging file.
Location	Meaning																		
ACTIVE	Page is in a working set.																		
MDFYLIST	Page is in the modified page list.																		
FREELIST	Page is in the free page list.																		
BADLIST	Page is in the bad page list.																		
RELPEND	Release of the page is pending.																		
RDERROR	Page has had an error during an attempted read operation.																		
PAGEOUT	Page is being written into a paging file.																		
PAGEIN	Page is being brought into memory from a paging file.																		
STATE	Byte that describes the state of the physical page.																		
TYPE	Byte that describes the type of virtual page. The types in this column are the hexadecimal codes that stand for the page types that appear in column PAGTYP of this display, described previously.																		
REFCOUNT	Count of the processes that are referencing this PFN. If the value of REFCOUNT is nonzero, the page is used in at least one working set. If the value is zero, the page is not used in any working set.																		
BAK	Address of the backing store; location on a disk device to which pages can be written.																		
SVAPTE	Virtual address associated with this page frame. The two SVAPTEs indicate a valid link between physical and virtual address space.																		
FLINK	Forward link within PFN database that points to the next virtual page; this longword also acts as the count of the number of processes that are sharing this global section.																		
BLINK	Backward link within PFN database; also acts as an index into the working set list.																		

SYSTEM DUMP ANALYZER

SHOW PAGE_TABLE

EXAMPLE

SDA> SHOW PAGE_TABLE

System page table

```
-----
ADDRESS  SVAPTE  PTE  TYPE  PROT  BITS  PAGTYP  LOC STATE TYPE  REFCNT  BAK          SVAPTE  FLINK      BLINK
.
8AD22E00 F8020725  VALID UR      K
8AD22E04 F8020726  VALID UR      K
8AD22E08 F8020727  VALID UR      K
8AD22E0C F8020728  VALID UR      K
8AD22E10 F8020729  VALID UR      K
8AD22E14 EC02072A  VALID UREW  M  K
8AD22E18 F402072B  VALID URKW  M  K
.
8AD22FEC F801F10E  VALID UR      K SYSTEM ACTIVE 07 01 1 0040FFF8 8AD22FEC 00000000 00000258
8AD22FF0 F801F10F  VALID UR      K SYSTEM ACTIVE 07 01 1 0040FFF8 8AD22FF0 00000000 00000257
8AD22FF4 F801F173  VALID UR      K SYSTEM ACTIVE 07 01 1 0040FFF8 8AD22FF4 00000000 000004B1
8AD22FF8 F801F172  VALID UR      K SYSTEM ACTIVE 07 01 1 0040FFF8 8AD22FF8 00000000 00000301
8AD22FFC F801F17F  VALID UR      K SYSTEM ACTIVE 07 01 1 0040FFF8 8AD22FFC 00000000 000000F5
8AD23000 F801F17E  VALID UR      K SYSTEM ACTIVE 07 01 1 0040FFF8 8AD23000 00000000 00000174
8AD23004 7801EBC6  TRANS UR      K SYSTEM FREELST 00 01 0 0040FFF8 8AD23004 0000D38B 0001EBC7
.
.
```

SHOW PFN_DATA

Displays information that is contained in the page lists and PFN database.

FORMAT **SHOW PFN_DATA** *[pfn] [/qualifier]*

PARAMETER *pfn*
Page frame number (PFN) of the physical page for which information is to be displayed.

QUALIFIERS */ALL*
Displays the free page list, modified page list, and bad page list. This is the default behavior of the SHOW PFN_DATA command. SDA precedes each list with a count of the pages it contains and its low and high limits.

/BAD
Displays the bad page list. SDA precedes the list with a count of the pages it contains, its low limit, and high limit.

/FREE
Displays the free page list. SDA precedes the list with a count of the pages it contains, its low limit, and high limit.

/MODIFIED
Displays the modified page list. SDA precedes the list with a count of the pages it contains, its low limit, and high limit.

/SYSTEM
Displays the entire PFN database in order by page frame number, starting at PFN 0000.

DESCRIPTION For each page frame number it displays, the SHOW PFN_DATA command lists information used in translating physical page addresses to virtual page addresses. Table SDA-10 lists the contents of the display.

SYSTEM DUMP ANALYZER

SHOW PFN_DATA

Table SDA-10 Page Frame Number Information in the SHOW PFN_DATA Display

Item	Contents																		
PFN	Page frame number																		
PTE ADDRESS	System virtual address of the page table entry that describes the virtual page mapped into this physical page																		
BAK	Place to find information on this page when all links to this PTE are broken: either an index into a process section table or the number of a virtual block in the paging file																		
REFCNT	Number of references being made to this page																		
FLINK	Address of the next page in the list in which this virtual page currently resides																		
BLINK	Address of the previous page in the list in which this virtual page currently resides																		
TYPE	Type of virtual page, one of the following: <table border="1"><thead><tr><th>Code</th><th>Meaning</th></tr></thead><tbody><tr><td>00</td><td>Process page</td></tr><tr><td>01</td><td>System page</td></tr><tr><td>02</td><td>Global, read-only page</td></tr><tr><td>03</td><td>Global, read/write page</td></tr><tr><td>04</td><td>Process page-table page</td></tr><tr><td>05</td><td>Global page-table page</td></tr></tbody></table>	Code	Meaning	00	Process page	01	System page	02	Global, read-only page	03	Global, read/write page	04	Process page-table page	05	Global page-table page				
Code	Meaning																		
00	Process page																		
01	System page																		
02	Global, read-only page																		
03	Global, read/write page																		
04	Process page-table page																		
05	Global page-table page																		
STATE	State of the virtual page, the low nibble of which can be one of the following: <table border="1"><thead><tr><th>Code</th><th>Meaning</th></tr></thead><tbody><tr><td>0</td><td>Page is on the free page list.</td></tr><tr><td>1</td><td>Page is on the modified page list.</td></tr><tr><td>2</td><td>Page is on the bad page list.</td></tr><tr><td>3</td><td>Release of the page to the free or modified page list is pending.</td></tr><tr><td>4</td><td>Error occurred as the page was being read from the disk.</td></tr><tr><td>5</td><td>Modified page writer is currently writing the page to the disk.</td></tr><tr><td>6</td><td>Page fault handler is currently reading the page from the disk.</td></tr><tr><td>7</td><td>Page is active and valid.</td></tr></tbody></table>	Code	Meaning	0	Page is on the free page list.	1	Page is on the modified page list.	2	Page is on the bad page list.	3	Release of the page to the free or modified page list is pending.	4	Error occurred as the page was being read from the disk.	5	Modified page writer is currently writing the page to the disk.	6	Page fault handler is currently reading the page from the disk.	7	Page is active and valid.
Code	Meaning																		
0	Page is on the free page list.																		
1	Page is on the modified page list.																		
2	Page is on the bad page list.																		
3	Release of the page to the free or modified page list is pending.																		
4	Error occurred as the page was being read from the disk.																		
5	Modified page writer is currently writing the page to the disk.																		
6	Page fault handler is currently reading the page from the disk.																		
7	Page is active and valid.																		

SYSTEM DUMP ANALYZER

SHOW PFN_DATA

EXAMPLE

SDA> SHOW PFN_DATA

Free page list

Count: 225
Low limit: 57
High limit: 1073741824

PFN	PTE ADDRESS	BAK	REFCNT	FLINK	BLINK	TYPE	STATE
----	-----	-----	-----	-----	-----	-----	-----
1329	8047AF3C	03002A83	0	1963	0000	00 PROCESS	00 FREELST
1963	8047AB10	03002A43	0	017C	1329	00 PROCESS	00 FREELST
017C	8047B3F8	03002A84	0	14B4	1963	00 PROCESS	00 FREELST
14B4	8047B464	03002A85	0	1529	017C	00 PROCESS	00 FREELST
1529	8047AA34	03002A87	0	1485	14B4	00 PROCESS	00 FREELST
1485	8047AC80	030010B3	0	1707	1529	00 PROCESS	00 FREELST

The SHOW PFN_DATA command displays the information shown previously for the free page list, the modified page list, and the bad page list, and then all of the PFN database, including the first three lists.

SYSTEM DUMP ANALYZER

SHOW POOL

SHOW POOL

Displays information about the disposition of memory, paged and nonpaged, including the lookaside lists (SRP, IRP, and LRP), nonpaged dynamic storage pool, and paged dynamic storage pool.

FORMAT

SHOW POOL [*/FREE*
/HEADER
/SUMMARY
/TYPE=block-type] [*range*
/ALL
/IRP
/LRP
/NONPAGED
/PAGED
/SRP]

PARAMETERS

range

Range of virtual addresses in pool that SDA is to examine. You can express a range using the following syntax:

m:n Range of virtual addresses in pool from *m* to *n*

m;n Range of virtual addresses in pool starting at *m* and continuing for *n* bytes

QUALIFIERS

/ALL

Displays the entire contents of allocated pool, including the lookaside lists, nonpaged dynamic storage pool, and paged dynamic storage pool. This is the default behavior of the SHOW POOL command.

/FREE

Displays the entire contents, both allocated and free, of the specified region or regions of pool. You cannot use the */FREE* qualifier when you use a **range** to indicate a region of pool to be displayed.

/HEADER

Displays only the first 16 longwords of each data block found within the specified region or regions of pool.

/IRP

Displays the contents of the I/O request packets (IRPs) currently in use.

/LRP

Displays the contents of the large request packets (LRPs) currently in use.

/NONPAGED

Displays the contents of the nonpaged dynamic storage pool currently in use.

/PAGED

Displays the contents of the paged dynamic storage pool currently in use.

SYSTEM DUMP ANALYZER

SHOW POOL

/SRP

Displays the contents of the small request packets (SRPs) currently in use.

/SUMMARY

Displays *only* an allocation summary for each specified region of pool.

/TYPE=block-type

Displays the blocks within the specified region or regions of pool that are of the indicated **block-type**. If SDA finds no blocks of that type in the pool region, it displays a blank screen, followed by an allocation summary of the region.

DESCRIPTION

The SHOW POOL command displays information about the contents of any specified region of pool in an 8-column format. The contents of the full display, from left to right, are listed as follows:

Column 1 contains the type of control block that starts at the virtual address in pool indicated in column 2. If SDA cannot interpret the block type, it displays a block type of "UNKNOWN." Column 3 lists the number of bytes (in decimal) of memory allocated to the block. The block size is fixed for SRPs, IRPs, and LRPs, and is variable in the paged and nonpaged pools.

The remaining columns contain a dump of the contents of the block, in 4-longword intervals, until the block is complete. Columns 4 through 7 display, from right to left, the contents in hexadecimal; column 8 displays, from left to right, the contents in ASCII. If the ASCII value of a byte is not a printing character, SDA displays a period character (.) instead.

For each region of pool it examines, the SHOW POOL command displays an allocation summary. This 4-column table lists, in column 2, the types of control block identified in the region and records the number of each in column 1. The last two columns represent the amount of the pool region occupied by each type of control block: column 3 records the total number of bytes and column 4 the percentage. The summary concludes with an indication of the number of bytes used within the particular pool region, as well as the number of bytes remaining. It provides an estimate of the percentage of the region that has been allocated.

SYSTEM DUMP ANALYZER

SHOW POOL

EXAMPLES

1 SDA> SHOW POOL
IRP lookaside list

```
-----
                          Dump of blocks allocated from IRP lookaside list

CIMSG  80BADE00  208
        001000DA 003C0090 0000A900 00036FF0 .o.....<....
        D9B3001C 00000000 A0B5001D 35E60017 ...5.....
        41414141 00000600 65EA0004 00000600 .....e....AAAA
        41414141 41414141 41414141 41414141 AAAAAAAAAAAAAAAAAA
        .
        .
FCB     80BAE070  208
        00000000 000700D0 80BAECA0 80BD9B30 0.....
        00000000 00000000 80BAE080 80BAE080 .....
        00000001 00000002 000C017B 00010000 ....{.....
        00000001 00000003 0000023E 0000606C 1'...>.....
        .
        .
IRP     80BAE140  208
        000100D1 410A00C4 0015D9A0 0015D9A0 .....A....
        80431820 00000000 7FFA2808 80127458 Xt.....C.
        80422450 1019FF80 7FFA2814 1C1D0032 2.....P$B.
        61616161 80431820 00000000 000601B9 .....C.aaaa
        .
        .
```

The SHOW POOL command lists the used contents of the IRP lookaside list, among which are the CI message block (CIMSG), file control block (FCB), and I/O request packet (IRP). When it completes the listing for the IRP lookaside list, it displays a summary of IRP allocation information.

The SHOW POOL command continues, displaying similar information for the other regions of pool—the LRP and SRP lookaside lists, nonpaged dynamic storage pool, and paged dynamic storage pool.

2 SDA> SHOW POOL GOBADE00;260
Non-paged dynamic storage pool

```
-----
                          Dump of blocks allocated from non-paged pool

CIMSG  80BADE00  144
        001000DA 003C0090 0000A900 00036FF0 .o.....<....
        D9B3001C 00000000 A0B5001D 35E60017 ...5.....
        41414141 00000600 65EA0004 00000600 .....e....AAAA
        41414141 41414141 41414141 41414141 AAAAAAAAAAAAAAAAAA
        41414141 41414141 41414141 41414141 AAAAAAAAAAAAAAAAAA
        .
        .
```

SYSTEM DUMP ANALYZER

SHOW POOL

```

UNKNOWN 80BADE90 112
41414141 41414141 41414141 41414141 AAAAAAAAAAAAAAAAAA

```

```

CIDG 80BADEDO 144
807708BB 003B0090 0004D7E0 000008F0 .....;...w.
61616161 61616161 61616161 016CE87C ..l.aaaaaaaaaaaa
61616161 61616161 61616161 61616161 aaaaaaaaaaaaaaaaaa
61616161 61616161 61616161 61616161 aaaaaaaaaaaaaaaaaa

```

```

UNKNOWN 80BADF60 64
61616161 61616161 61616161 61616161 aaaaaaaaaaaaaaaaaa

```

```

CIDG 80BADFA0 144
807708BB 003B0090 0003FFC0 0004B1B0 .....;...w.
61616161 61616161 61616161 016CE94C L.l.aaaaaaaaaaaa
61616161 61616161 61616161 61616161 aaaaaaaaaaaaaaaaaa
61616161 61616161 61616161 61616161 aaaaaaaaaaaaaaaaaa

```

```

UNKNOWN 80BAE030 48
61616161 61616161 61616161 61616161 aaaaaaaaaaaaaaaaaa
61616161 61616161 61616161 61616161 aaaaaaaaaaaaaaaaaa
61616161 61616161 61616161 61616161 aaaaaaaaaaaaaaaaaa

```

Summary of non-paged pool contents

```

3 UNKNOWN = 176 (29%)
2 CIDG = 288 (48%)
1 CIMSG = 144 (24%)

```

Total space used = 608 out of 608 total bytes, 0 bytes left

Total space utilization = 100%

The preceding example examines 608 (260₁₆) bytes of nonpaged pool, starting at address 80BADE00₁₆, which happens to be the starting address of the CIMSG block listed in the previous example's output. SDA attempts to identify allocated blocks as it proceeds through the specified region of pool, and displays an allocation summary when it completes the listing.

SYSTEM DUMP ANALYZER

SHOW POOL

```

3 SDA> SHOW POOL/FREE
  IRP lookaside list
  -----
                                Dump of blocks allocated from IRP lookaside list

CIMSG   80BADE00   208
          001000DA 003C0090 0000A900 00036FF0 .o.....
          D9B3001C 00000000 A0B5001D 35E60017 ...5.....
          41414141 00000600 65EA0004 00000600 .....e....AAAA
          41414141 41414141 41414141 41414141 AAAAAAAAAAAAAAAAAA
          .
          .
[Free]   80BADE00   208
          807708BB 003B0090 0004D7E0 000008F0 .....;...w.
          61616161 61616161 61616161 016CE87C ..l.aaaaaaaaaaaa
          61616161 61616161 61616161 61616161 aaaaaaaaaaaaaaaaaa
          61616161 61616161 61616161 61616161 aaaaaaaaaaaaaaaaaa
          .
          .
[Free]   80BADFA0   208
          807708BB 003B0090 0003FFC0 0004B1B0 .....;...w.
          61616161 61616161 61616161 016CE94C Ll.aaaaaaaaaaaa
          61616161 61616161 61616161 61616161 aaaaaaaaaaaaaaaaaa
          61616161 61616161 61616161 61616161 aaaaaaaaaaaaaaaaaa
          .
          .

```

The SHOW POOL/FREE command produces a display similar in format and extent to that presented in Example 1. However, it displays the unallocated portions of pool in addition to those that are used.

```

4 SDA> SHOW POOL/PAGED/HEADER
  Paged dynamic storage pool
  -----
                                Dump of blocks allocated from paged pool

RSHT    8024FE00   528
          802DC710 00380210 00000000 FFFFFFF80 .....8...-
LNM     80250010   96
          8015B847 00400060 802D75A0 00000000 .....u-'.@.G...
LNM     80250070   48
          8015B847 01400030 802500A0 802D7400 .t-...%.O.@.G...
LNM     802500A0   96
          8015B847 02400060 802DC170 80250070 p.%.p-'.@.G...
LNM     80250100   48
          8015B847 00400030 802DC510 802E1B60 '.....-O.@.G...
          .
          .

```

The SHOW POOL/PAGED/HEADER command displays only the name of each block allocated from paged pool, its starting address, its size, and the first four longwords of its contents.

SHOW PORTS

Displays those portions of the port descriptor table (PDT) that are port independent.

FORMAT **SHOW PORTS** *[/ADDRESS=*pdt-address*]*

PARAMETERS *None.*

QUALIFIER ***/ADDRESS=pdt-address***
Displays the specified port descriptor table (PDT).¹¹

DESCRIPTION The SHOW PORTS command provides port-independent information from the port descriptor table (PDT) for those CI ports with full SCS connections. This information is used by all system communications services (SCS) port drivers.

Note that the SHOW PORTS command does *not* display similar information about UDA ports, BDA ports, and similar controllers.

The SHOW PORTS command produces several displays. The initial display, the **PDT summary page**, lists the PDT address, port type, device name, and driver name for each PDT. Subsequent displays provide information taken from each PDT listed on the summary page.

You can use the */ADDRESS* qualifier to the SHOW PORTS command to produce more detailed information about a specific port. The first display of the SHOW PORTS/*ADDRESS* command duplicates the last display of the SHOW PORTS command, listing information stored in the port's PDT. Subsequent displays list information about the port blocks and virtual circuits associated with the port.

¹¹ You can find the **pdt-address** for any active connection on the system in the **PDT summary page** display of the SHOW PORTS command. In addition, CDT addresses are also stored in many individual data structures related to SCS connections: for instance in the path block displays of the SHOW CLUSTER/SCS command.

SYSTEM DUMP ANALYZER

SHOW PORTS

EXAMPLES

1 SDA> SHOW PORTS
VAXcluster data structures

```
-----  
          --- PDT Summary Page ---  
  
PDT Address      Type      Device      Driver Name  
-----  
803B38D0         pa        PAAO        PADRIVER  
803CD6D0         pe        PEA0        PEDRIVER  
  
          --- Port Descriptor Table (PDT) 803B38D0 ---  
  
Type: 01 pa  
Characteristics: 0000  
  
Msg Header Size      32 Connect      807725A0 Recyclh_Msg_Buf 80772FEE  
Max Xfer Bcnt        FFFFFFFF Dealloc_Dg_Buf 807731E6 Request_Data    8077338D  
DG Header Size       136 Disconnect  80772661 Send_Data       807733D6  
Poller Sweep         21 Unmap        8077344A Send_Dg_Buf     8077324A  
Fork Block W.Q.      empty Map         807732E5 Send_Msg_Buf    80773108  
UCB Address          803B34F0 Map_Bypass   807732CC Send_Cnt_Msg_Buf 8077310F  
ADP Address          80BF7F70 Map_Irp      807732D5 Read_Count      8076FE90  
Accept               807725ED Map_Irp_Bypass 807732C4 Rls_Read_Count  8076FEFE  
Alloc_Dg_Buf         807731D2 Queue_Dg_Buf 807731EC Mreset          807722B7  
Alloc_Msg_Buf        80772F69 Queue_Mult_Dgs 807731F4 Mstart          807722C1  
Dealloc_Msg_Buf      80773047 Recycl_Msg_Buf 80772FF8 Stop_Vcs        80772304  
Dealloc_Msg_Buf_Reg 8077305A Reject      8077262C Send_Dg_Reg     8077323D  
  
          --- Port Descriptor Table (PDT) 803CD6D0 ---  
  
Type: 03 pe  
Characteristics: 0000  
  
Msg Header Size      32 Connect      803C9DFC Recyclh_Msg_Buf 803CA84A  
.  
.  
.
```

The SHOW PORTS command first lists the two SCS ports that exist within the VAXcluster and then displays information obtained from each port descriptor table (PDT).

SYSTEM DUMP ANALYZER

SHOW PORTS

```

2 SDA> SHOW PORTS/ADDRESS=G3CD6D0
VAXcluster data structures
-----

      --- Port Descriptor Table (PDT) 803CD6D0 ---

Type: 03 pe
Characteristics: 0000

Msg Header Size          32 Connect          803C9DFC Recyclh_Msg_Buf 803CA84A
.
.
.

```

```

      --- Port Block 80B7CCA0 ---

Status: 0001 authorize
VC Count: 12
Secs Since Last Zeroed: 77188

SCXB Size                268                LCXB Size                1780
SCXB Count               26                LCXB Count               30
SCXB Max                 26                LCXB Max                 30
SCXB Queue Empty        4553                LCXB Queue Empty        0
No SCXB for ACK         47

```

Bus	Error Count	Last Error	Time of Last Error
0	0		
0	2	00390334	23-FEB-1988 12:43:14.99

```

      --- Virtual Circuit (VC) 803C0030 ---

Remote System Name: BLUES                Remote System ID: AA-00-04-00-77-4C
Local System ID: 223 (DF)                Status: 0005 open,path

Msg Xmt          6  Msg Rcv          25736  Auth fail          0
Unsequence       3  Unsequence         3  No RcvChan         0
Sequence         0  Sequence           0  No XmtChan         0
ReXmt            0  ReRcv              0  Ill ACK            0
ACK              0  ACK                0  SeqMsg TMO         0
Chan Cntl        3  Cache              0  Listen TMO         0
Bytes Xmt        165  Ill Seq            0  CC HS TMO          0
                  Chan Cntl          25733  TR DFQ Empty       0
                  Bad Chksum          0  TR MFQ Empty       0
                  TR Short            0  CC DFQ Empty       0
                  CC Short            0  CC MFQ Empty       0
                  CC Bad ECO          0
                  Bytes Rcv          1182733

```

```

      --- Virtual Circuit (VC) 803DE9A0 ---

Remote System Name: GREENS                Remote System ID: AA-00-04-00-79-4C
Local System ID: 222 (DE)                Status: 0005 open,path
.
.
.

```

The SHOW PORTS/ADDRESS command first provides a summary of port information obtained from the PDT located at 803CD6D0₁₆. It then displays information from the associated port block and a screen describing each of its 12 virtual circuits.

SYSTEM DUMP ANALYZER

SHOW PROCESS

SHOW PROCESS

Displays the software and hardware context of any process in the balance set.

FORMAT

SHOW PROCESS *[/qualifier[,...]]* [*ALL*
process-name
/INDEX=nn
/SYSTEM]

PARAMETERS

ALL

Shows information about all processes that exist in the system.

process-name

Name of the process for which information is to be displayed.¹²

You can determine the names of the processes in the system by issuing a SHOW SUMMARY command.

The **process-name** can contain up to 15 letters and numerals, including the underscore (_) and dollar sign (\$) characters. If it contains any other characters, you must enclose the **process-name** in quotation marks (" ").

QUALIFIERS

/ALL

Displays all information shown by the following qualifiers: /PCB, /PHD, /REGISTERS, /WORKING_SET, /PROCESS_SECTION_TABLE, /PAGE_TABLES, and /CHANNEL.

/CHANNEL

Displays information about the I/O channels assigned to the process.

/INDEX=nn

Specifies the process for which information is to be displayed by its index into the system's list of software process control blocks (PCBs). You can supply either of the following values for **nn**:

- The process index itself
- The process identification (PID) or extended PID longword, from which SDA extracts the correct index

To obtain these values for any given process, issue the SDA command SHOW SUMMARY.

¹² Use of the **process-name** parameter, the /INDEX qualifier, or the /SYSTEM qualifier causes the SHOW PROCESS command to perform an implicit SET PROCESS command, making the indicated process the current process for subsequent SDA commands. (See the description of the SET PROCESS command and Section 4 for information on how this can affect the process context—and CPU context—in which SDA commands execute.)

SYSTEM DUMP ANALYZER

SHOW PROCESS

/LOCKS

Displays the lock management locks owned by the current process.

The */LOCKS* qualifier produces a display similar in format to that produced by the *SHOW LOCKS* command. See Table SDA-7 for additional information.

/P0

Displays the page tables for P0 space. See the description of the */PAGE_TABLES* qualifier.

/P1

Displays the page tables for P1 space. See the description of the */PAGE_TABLES* qualifier.

/PAGE_TABLES $\left[\begin{array}{l} \textit{range} \\ \textit{/P0} \\ \textit{/P1} \end{array} \right]$

Displays the page tables of the process's program (P0) and control (P1) regions, or, optionally, either page table or the page table entries for a **range** of addresses.

You can express a **range** using the following syntax:

m:n Displays the page table entries that correspond to the range of virtual addresses from *m* to *n*

m;n Displays the page table entries that correspond to a range of *n* pages, starting with page *m*

/PCB

Displays the information contained in the software process control block (PCB). This is the default behavior of the *SHOW PROCESS* command.

/PHD

Lists information included in the process header (PHD).

/PROCESS_SECTION_TABLE

Lists the information contained in the process section table (PST).

/REGISTERS

Lists the hardware context of the process, as reflected in the process's registers stored in the hardware PCB and—if the process is current on a processor in the VAX system—the processor's registers.

/RMS[=option[...]]

Displays certain specified RMS data structures for each image I/O or process permanent I/O file the process has open. To display RMS data structures for process-permanent files, specify the *PIO* option to this qualifier.

SYSTEM DUMP ANALYZER

SHOW PROCESS

SDA determines the structures to be displayed according to either of the following methods:

- If you provide the name of a structure or structures in the **option** parameter, SHOW PROCESS/RMS displays information from only the specified structures. (See Table SDA-6 for a list of keywords that may be supplied as options.)
- If you do not specify an **option**, SHOW PROCESS/RMS displays the current list of options as shown by the SHOW RMS command and set by the SET RMS command.

/SYSTEM

Displays the system process control block.¹³ The system PCB and process header (PHD) are dummy structures that are located in system space. These structures contain the system working set, global section table, global page table, and other systemwide data.

/WORKING_SET

Displays the process's working set list.

DESCRIPTION

The SHOW PROCESS command displays information about the process specified by **process-name**, the process specified in the /INDEX qualifier, the system process, or all processes. The SHOW PROCESS command performs an implicit SET PROCESS command under certain uses of its qualifiers and parameters, as noted above. By default, the SHOW PROCESS command produces information about the SDA current process, as defined in Section 4.

The default of the SHOW PROCESS command provides information taken from the software process control block (PCB).¹⁴ This information describes the following characteristics of the process:

- Software context
- Condition-handling information
- Information on interprocess communication
- Information on counts, quotas, and resource usage

Among the displayed information are the process's PID, EPID, priority, job information block (JIB) address, and process header (PHD) address. SHOW PROCESS also describes the resources owned by the process, such as event flags and mutexes. The "State" field records the process's current scheduling state; in a VMS multiprocessing system, the display indicates the CPU ID of any process whose state is CUR.

The SHOW PROCESS/ALL command displays additional process-specific information, also provided by several of the individual qualifiers to the command.

¹³ Use of the **process-name** parameter, the /INDEX qualifier, or the /SYSTEM qualifier causes the SHOW PROCESS command to perform an implicit SET PROCESS command, making the indicated process the current process for subsequent SDA commands. (See the description of the SET PROCESS command and Section 4 for information on how this can affect the process context—and CPU context—in which SDA commands execute.)

¹⁴ This is the first display provided by the /ALL qualifier and the only display provided by the /PCB qualifier.

SYSTEM DUMP ANALYZER

SHOW PROCESS

The **process header** display, also produced by the /PHD qualifier, provides information taken from the process header (PHD), which is swapped into memory when the process becomes part of the balance set. Each item listed in the display reflects a quantity, count, or limit for the process's use of the following resources:

- Process memory
- The pager
- The scheduler
- Asynchronous system traps
- I/O activity
- CPU activity

The **process registers** display, also produced by the /REGISTERS qualifier, describes the process's hardware context, as reflected in its registers.

There are two places where a process's hardware context is stored, as described below:

- If the process is currently executing on a processor in the VAX system (that is, in the CUR scheduling state), its hardware context is contained in that processor's registers. (That is, the process's registers and the processor's registers contain identical values, as illustrated by a SHOW CPU command for that processor or a SHOW CRASH command if the process was current at the time of the system failure.)
- If the process is not executing, its hardware context is stored in the part of the PHD known as the hardware PCB.

The **process registers** display first lists those registers stored in the hardware PCB ("Saved process registers"). If the process to be displayed is currently executing on a processor in the VAX system, the display then lists the processor's registers ("Active registers for the current process"). In each section, the display lists the registers in the following groups:

- General purpose registers (R0 through R11, the AP, FP, and PC)
- Stack pointers (KSP, ESP, SSP, and USP)
- Special-purpose registers (PC and PSL)
- Base and length registers (P0BR, P1BR, P0LR, and P1LR)

The **working set information** and **working set list** displays, also produced by the /WORKING_SET qualifier, describe those virtual pages that the process can access without a page fault. After a brief description of the size,

SYSTEM DUMP ANALYZER

SHOW PROCESS

scope, and characteristics of the working set list itself, SDA displays the following information for each entry in the working set list:

Column	Contents
INDEX	Index into the working set list at which information for this entry can be found
ADDRESS	Virtual address of the page in the process address space that this entry describes
STATUS	Three columns that list the following status information: <ul style="list-style-type: none"> • Page type • Location of the page in physical memory • Indication of whether the page is locked into the working set

When SDA locates one or more unused working set entries, it issues the following message:

```
--- n empty entries
```

In this message, *n* is the number (in decimal) of contiguous, unused entries.

The **process section table information** and **process section table** displays, also produced by the /PROCESS_SECTION_TABLE qualifier, list each entry in the process section table (PST) and display the offsets to the first free entry and last used entry.

SDA displays the information listed in Table SDA-11 for each PST entry.

Table SDA-11 Process Section Table Entry Information in the SHOW PROCESS Display

Part	Definition
INDEX	Offset into the PST at which the entry is found. Because entries in the process section table begin at the highest location in the table, and the table expands toward lower addresses, the following expression determines the address of an entry in the table: $PHD + PSTBASOFF - INDEX$.
ADDRESS	Virtual address that marks the beginning of the first page of the section described by this entry.
PAGES	Length, in pages, of the process section.
VBN	Virtual block number, the number of the file's virtual block that is mapped into the section's first page.
CLUSTER	Cluster size used when faulting pages into this process section.
REFCNT	Number of pages of this section that are currently mapped.
FLINK	Forward link, the pointer to the next entry in the PST list.
BLINK	Backward link, the pointer to the previous entry in the PST list.
FLAGS	Flags that describe the access that processes have to the process section.

SYSTEM DUMP ANALYZER

SHOW PROCESS

The **P0 page table** and **P1 page table** displays, also produced by the /PAGE_TABLES qualifier, display listings of the process's page table entries in the same format as that produced by the SHOW PAGE_TABLE command (see Tables SDA-8 and SDA-9).

The **process active channels** display, the last produced by SHOW PROCESS /ALL and the only one produced by the /CHANNEL qualifier, displays the following information for each I/O channel assigned to the process:

Column	Contents
Channel	Number of the channel
Window	Address of the window control block (WCB) for the file if the device is a file-oriented device; zero otherwise
Status	Status of the device: "Busy" if the device has an I/O operation outstanding; blank otherwise
Device/file accessed	Name of the device and, if applicable, name of the file being accessed on that device

The information listed under the heading "Device/file accessed" varies from channel to channel and from process to process. SDA displays certain information according to the conditions listed in Table SDA-12.

Table SDA-12 Process I/O Channel Information in the SHOW PROCESS Display

Information Displayed ¹	Type of Process
<i>dcuu:</i>	SDA displays this information for devices that are not file structured, such as terminals, and for processes that do not open files in the normal way.
<i>dcuu:filespec</i>	SDA displays this information only if you are examining a running system, and only if your process has enough privilege to translate the <i>file-id</i> into the <i>filespec</i> .
<i>dcuu:(file-id)filespec</i>	SDA displays this information only when you are examining a dump. The <i>filespec</i> corresponds to the <i>file-id</i> on the device listed. If you are examining a dump from your own system, the <i>filespec</i> is probably valid. If you are examining a dump from another system, the <i>filespec</i> is probably meaningless in the context of your system.

¹This table uses the following formulas to identify the information displayed:

dcuu:(file-id)filespec

where:

dcuu: is the name of the device.

file-id is the RMS file identification.

filespec is the full file specification, including directory name.

SYSTEM DUMP ANALYZER

SHOW PROCESS

Table SDA-12 (Cont.) Process I/O Channel Information in the SHOW PROCESS Display

Information Displayed ¹	Type of Process
<i>dcuu:(file-id)</i>	The <i>file-id</i> no longer points to a valid <i>filespec</i> , as when you look at a dump from another system; or the process in which you are running SDA does not have enough privilege to translate the <i>file-id</i> into the corresponding <i>filespec</i>

¹This table uses the following formulas to identify the information displayed:

dcuu:(file-id)filespec

where:

dcuu: is the name of the device.

file-id is the RMS file identification.

filespec is the full file specification, including directory name.

EXAMPLES

1 SDA> SHOW PROCESS

```

Process index: 001B   Name: PUTP1   Extended PID: 27E0011B
-----
Process status: 00044001  RES,BATCH,PHDRES

PCB address      803C7710   JIB address      806B9100
PHD address      81F5C400   Swapfile disk address 02002FA1
Master internal PID 0001001B   Subprocess count      0
Internal PID     0001001B   Creator internal PID  00000000
Extended PID     27E0011B   Creator extended PID  00000000
State            CUR 00   Termination mailbox  0000
Current priority 3       AST's enabled        KES
Base priority    3       AST's active         E
UIC              [00011,000176]  AST's remaining      39
Mutex count      0       Buffered I/O count/limit 12/12
Waiting EF cluster 0     Direct I/O count/limit 18/18
Starting wait time 1B001C1C  BUFIO byte count/limit 31968/31968
Event flag wait mask BFFFFFFF   # open files allowed left 90
Local EF cluster 0 20000001   Timer entries allowed left 9
Local EF cluster 1 C0000000   Active page table count 0
Global cluster 2 pointer 00000000  Process WS page count 1020
Global cluster 3 pointer 00000000  Global WS page count 233

```

The SHOW PROCESS command displays information taken from the software PCB of PUTP1, the SDA current process. According to the "State" field in the display, process PUTP1 is current on CPU 00 in the VMS multiprocessing system.

SYSTEM DUMP ANALYZER

SHOW PROCESS

2 SDA> SHOW PROCESS/ALL

Process index: 00AD Name: GLOBE Extended PID: 462002AD

Process status: 02040001 RES,PHDRES

PCB address 8044E650 JIB address 806E0010

Process header

First free P0 address	0007D600	Accumulated CPU time	00000559
Free PTEs between P0/P1	276902	CPU since last quantum	FFEE
First free P1 address	7FEF2200	Subprocess quota	8
Free page file pages	24234	AST limit	50
Page fault cluster size	16	Process header index	0020
Page table cluster size	2	Backup address vector	00003E12
Flags	0002	WSL index save area	00003980
Direct I/O count	509	PTs having locked WSLs	5
Buffered I/O count	827	PTs having valid WSLs	20
Limit on CPU time	00000000	Active page tables	21
Maximum page file count	25600	Maximum active PTs	26
Total page faults	7589	Guaranteed fluid WS pages	20
File limit	50	Extra dynamic WS entries	698
Timer queue limit	10	Locked WSLE counts array	1CD8
Paging file index	06000000	Valid WSLE counts array	2564

Saved process registers

R0 = 00000001	R1 = 00000000	R2 = 8000CA78	R3 = 8044E6A0
R4 = 8044E650	R5 = 00000000	R6 = 00000000	R7 = 00000003
R8 = 00001F60	R9 = 7FF9FB38	R10 = 7FF9FA08	R11 = 7FFE0070
AP = 7FEF4AE4	FP = 7FEF4AEC	PC = 801622B4	PSL = 03C00000
KSP = 7FFE7E00	ESP = 7FFE9E00	SSP = 7FFED04E	USP = 7FEF4AE4

POBR = 82D43600 POLR = 000003EB P1BR = 82654E00 P1LR = 001FF792

Active registers for current process

R0 = 00000001	R1 = 80002398	R2 = 00000000	R3 = 00000000
R4 = 7FFA05A0	R5 = 00000000	R6 = 0007D400	R7 = 00000010
R8 = 00001F60	R9 = 7FF9FB38	R10 = 7FF9FA08	R11 = 7FFE0070
AP = 7FFE9D70	FP = 7FFE9D58	PC = 801620A5	PSL = 01400000
KSP = 7FFE7E00	ESP = 7FFE9D58	SSP = 7FFED04E	USP = 7FEF4AE4

Working set information

First WSL entry	0074	Current authorized working set size	2048
First locked entry	00A6	Default (initial) working set size	512
First dynamic entry	00B9	Maximum working set allowed (quota)	2048
Last entry replaced	018C		
Last entry in list	0561		

Working set list

INDEX	ADDRESS	STATUS
0074	7FFE7C00	VALID PROCESS WSLOCK
0075	7FFE7A00	VALID PROCESS WSLOCK
0076	7FFE7800	VALID PROCESS WSLOCK

SYSTEM DUMP ANALYZER

SHOW PROCESS

Process section table information

```

Last entry allocated   FFA0
First free PST entry  0000
  
```

Process section table

INDEX	ADDRESS	PAGES	WINDOW	VBN	CLUSTER	CHANNEL	REFCNT	FLINK	BLINK	FLAGS
FFF8	00000200	0000000A	8082C400	00000002	0	7FFCCFDO	10	FFE8	FFFO	
FFFO	00001600	00000007	8082C400	0000000C	0	7FFCCFDO	0	FFF8	FFE8	WRT CRF
FFE8	00002400	00000012	8082C400	00000013	0	7FFCCFDO	18	FFFO	FFF8	

P0 page table

ADDRESS	SVAPTE	PTE	TYPE	PROT	BITS	PAGTYP	LOC	STATE	TYPE	REFCNT	BAK	SVAPTE	FLINK	BLINK
----- 1 NULL PAGE														
00000200	82D43604	F9804F73	VALID	UR		U PROCESS	ACTIVE	07	00	1	0040FFF8	82D43604	0000	0153
00000400	82D43608	F9806905	VALID	UR		U PROCESS	ACTIVE	07	00	1	0040FFF8	82D43608	0000	0154
00000600	82D4360C	F9807569	VALID	UR		U PROCESS	ACTIVE	07	00	1	0040FFF8	82D4360C	0000	0155

P1 page table

ADDRESS	SVAPTE	PTE	TYPE	PROT	BITS	PAGTYP	LOC	STATE	TYPE	REFCNT	BAK	SVAPTE	FLINK	BLINK
7FEF2400	82E52C48	21800000	DZERO	UW		U								
7FEF2600	82E52C4C	21800000	DZERO	UW		U								
7FEF2800	82E52C50	21800000	DZERO	UW		U								

Process active channels

Channel	Window	Status	Device/file accessed
0010	00000000		ROCK\$DJA233:
0020	8082C400		ROCK\$DJA233:(1008,48490,0)
0030	807F2260		LOVE\$DUA200:(209,1,0)[V5COMMON.SYSLIB]SMGSHR.EXE;1 (section file)
0040	00000000		VTA71:
0050	00000000		VTA71:
0060	807EFFE0		LOVE\$DUA200:(195,1,0)[V5COMMON.SYSLIB]LIBRTL.EXE;1 (section file)
0070	807EECC0		LOVE\$DUA200:(199,1,0)[V5COMMON.SYSLIB]MTHRTL.EXE;1 (section file)
0080	80838E80		LOVE\$DUA200:(196,1,0)[V5COMMON.SYSLIB]LIBRTL2.EXE;1
0090	807E4880		LOVE\$DUA200:(210,1,0)[V5COMMON.SYSLIB]SORTSHR.EXE;1
00A0	80818720		LOVE\$DUA200:(191,1,0)[V5COMMON.SYSLIB]FDLSHR.EXE;1
00B0	8083CFC0		LOVE\$DUA200:(169,1,0)[V5COMMON.SYSLIB]CONVSHR.EXE;1
00C0	8083DECO		ROCK\$DJA233:(1026,16,0)

The SHOW PROCESS/ALL command displays information taken from the software PCB of process GLOBE, and then proceeds to display the process header, the process's registers, the process section table, the P0 page table, the P1 page table, and information about the I/O channels owned by the process. These displays may also be obtained by the /PCB, /PHD, /REGISTERS, /PROCESS_SECTION_TABLE, /P0, /P1, and /CHANNEL qualifiers, respectively.

SYSTEM DUMP ANALYZER

SHOW PROCESS

3 SDA> SHOW PROCESS/LOCKS/INDEX=0A

Process index: 000A Name: JOB_CONTROL Extended PID: 4620010A

Lock data:

Lock id: 09960A0F PID: 0001000A Flags: VALBLK CONVERT SYNCSTS
Par. id: 00000000 Granted at PW SYSTEM
Sublocks: 100
LKB: 8082B0E0 BLKAST
Resource: 003C0248 24534D52 RMS\$H.< Status: ASYNC
Length 26 444B4C4F 46020000 ...FOLKD
Exec. mode 00202020 20202024 \$.
System 00000000 00000000

Local copy

Lock id: 043C0491 PID: 0001000A Flags: NOQUEUE
Par. id: 00000000 Granted at EX
Sublocks: 0
LKB: 8083FAE0 BLKAST
Resource: 4C41434F 2443424A JBC\$ROCK Status:
Length 10 00000000 00002041 A
User mode 00000000 00000000
System 00000000 00000000

Local copy

The SHOW PROCESS/LOCKS/INDEX=0A command displays information about the locks held by process JOB_CONTROL, whose PCB is at index 0A into the system's PCB list. This command implicitly makes JOB_CONTROL the SDA current process for subsequent commands that display process context information. It has no effect on SDA CPU context as JOB_CONTROL is not current on any processor in the VMS multiprocessing system.

SYSTEM DUMP ANALYZER

SHOW PROCESS

4 SDA> SHOW RMS

RMS Display Options: IFB, IRB, IDX, BDB, BDBSUM, ASB, CCB, WCB, FCB, FAB, RAB, NAM, XAB, RLB, BLB, BLBSUM, GBD, GBH, FWA, GBDSUM, JFB, NWA, RU, DRC, SFSB, GBSB
 Display RMS structures for all IFI values.

SDA> SHOW PROCESS/RMS

Process index: 0032 Name: BEASSEM_MTHRTL_ Extended PID: 27200132

 IFAB Address: 7FF9C808 IFI: 0002 Organization: Sequential

PRIM_DEV:	1C4D4108	DIR, FOD, SHR, AVL, ELG, IDV, ODV, RND		
BKPBITS:	00080020	ACCESSED, NORECLK		
BLN:	3A 58.	BID:	0B	11.
EFN:	00	MODE:	03	
IOS:	00000001	ASEADDR:	00000000	
IOS2:	0000	WAIT_Q_FLINK:	00000000	
IOS4:	00000000	ARGLST:	7FF21418	
ATJNLBUF:	00000000	WAIT_Q_BLINK:	00000000	
FSPTR:	00000000	AGENT_MODE:	03	
SHR:	02 SHRGET			
IRAB_LNK:	7FF9C958	CHNL:	00C0	
FAC:	02 GET			
ORGCASE:	00	Sequential		
LAST_FAB:	00081FDO	NWA_PTR:	00000000	
IFI:	0002	ECHO_ISI:	0000	
FWA_PTR:	7FF9CC00			
BDB_FLNK:	7FF9CB80	DEVBUFSIZ:	00000200	512.
BDB_BLNK:	7FF9CB60	RTDEQ:	0000	0.
RFMORG:	02	VAR		
RAT:	02	CR		
LRL:	004C 76.	HBK_DISK:	000C0000	
FFB:	0084 132.	EBK_DISK:	000C0000	
FSZ:	00 0.	BKS:	00	0.
DEQ:	0000 0.	MRS:	0000	0.
HBK:	0000000C 12.	GBC:	0000	0.
EBK:	0000000C			
LAST_GOOD_EBK:	00000000 0.	LAST_GOOD_FFB:	0000	0.
RNS_LEN:	00000000	LOCK_BDB:	00000000	
SFSB_PTR:	00000000	AVLCL:	0003	3.
GBSB_PTR:	00000000	AVGPPB:	0000	0.
PAR_LOCK_ID:	00000000			
AS_DEV:	1C4D4108	BLBFLNK:	00000000	
ASDEVBSIZ:	00000200	BLBBLNK:	00000000	
GBH_PTR:	00000000	AI_JFB:	00000000	
BI_JFB:	00000000	AT_JFB:	00000000	
JOURNAL:	00 0.	BUFFER_OFFSET:	0000	0.
RUSB_LNK:	00000000	RU_ACTIVE:	00	0.
RUFBLNK:	00000000	RU_HANDLE:	00000000	
ATJNLBUF:	00000000	JNLBDB:	00000000	
JNLFLG:	00			
RECVRFLGS:	00			
JNLFLG2:	00			
EXTJNLBUF:	00000000	RLB_FL_FLINK:	7FF9C858	

FAB Address: 00081FDO

BID:	03	3.		
BLN:	50	80.		
IFI:	0002			
FOP:	00000000			
STS:	00010001	ALQ:	0000000C	
STV:	000000C0	DEQ:	0000	
FAC:	02 GET			
SHR:	02 SHRGET			
CTX:	00000000	RTV:	00	
ORG:	00	Sequential		
RAT:	02	CR		

SYSTEM DUMP ANALYZER

SHOW PROCESS

```
RFM:          02      VAR
JOURNAL:      00
RU_FACILITY:  00
XAB:          0008235C      NAM:          000822D0
FNA:          00062894      DNA:          00000000
FNS:          2B           DNS:          00
File name:    VMSTESTRES$:[MTHRTL.SRC]MTH$ATAN2TST.B32;1
MRS:          0000          MRN:          00000000
BLS:          0200          512.         BKS:          00           0.
FSZ:          00           0.           DEV:          4108
SDC:          1C4D4108      GBC:          0000          0.
ACMODES:      00
```

XABFHC Address: 0008235C

```
-----
COD:          1D           29.         BLN:          2C           44.
NXT:          00082330      RVN:          0202
RFO:          02
ATR:          02      CR
LRL:          004C          76.
HBK:          0000000C      FFB:          0084
EBK:          0000000C      HSZ:          00
MRZ:          0000          0.           DXQ:          0000          0.
GBC:          0000          0.
SBN:          00000000
```

XABDAT Address: 00082330

```
-----
COD:          12           18.         BLN:          2C           44.
NXT:          00000000      RVN:          0002
RDT:          CDT:
  RDT0:        2AB640E0      CDT0:        69508580
  RDT4:        008F8F1B      CDT4:        008F8EEA
EDT:          BDT:
  EDT0:        00000000      BDT0:        00000000
  EDT4:        00000000      BDT4:        00000000
```

NAM Address: 000822D0

```
-----
BID: 02      2.      RFS:          00      NO_RFS
BLN: 60      96.
NOP: 00
WCC 00000000
FNB: 002030C7 PPF,NODE,QUOTED,GRP_MBR,WILD_DIR,
      DIR_LVL5,WILD_UFD,WILD_SFD1,WILD_SFD3,WILD_SFD5
Expanded String:  ESL: 2B / ESS: FF @ ESA: 00091A98
                  VMSTESTRES$:[MTHRTL.SRC]MTH$ATAN2TST.B32;1
Resultant String: RSL: 2B / RSS: FF @ RSA: 00091A98
                  VMSTESTRES$:[MTHRTL.SRC]MTH$ATAN2TST.B32;1
FID: [1CA4,0003,0001]
NODE: 00/00091A98
DEV: 0D/00091A98      VMSTESTRES$:
DIR: 0C/00091AA5      [MTHRTL.SRC]
NAME: 0C/00091AB1      MTH$ATAN2TST
TYPE: 04/00091ABD      .B32
VER: 02/00091AC1      ;1
```

CCB Address: 7FFCAD30

```
-----
UCB:          80487E70      WIND:          80B7A1C0
STS:          00
AMOD:         02           IOC:          0001
DIRP:         00000000
```

SYSTEM DUMP ANALYZER

SHOW PROCESS

WCB Address: 80B7A1C0

```

-----
WFL:      804A2AA0      SIZE:      0060
WLBL:     804A2AA0      TYPE:      12
ACCESS:   01          READ
PID:      00010032      ORGUCB:    80487E70
ACON:     0001          NOWRITE
NMAP:     0005          FCB:       804A2A90
RVT:      808F9140      LINK:      00000000
READS:    00000001      WRITES:    00000000
STVBN:    00000001      P1_COUNT:  0003
P1_LBN:   010C568E      P2_COUNT:  0001
P2_LBN:   010C5695
  
```

FCB Address: 804A2A90

```

-----
FCBFL:    804BFBF0      SIZE:      00D0
FCBBL:    804788A0      TYPE:      07
EXFCB:    00000000      WFL:      80B7A1C0
REFCNT:   0001          1.          ACNT:      0001          1.
WCNT:     0000          0.          LCNT:      0001          1.
TCNT:     0000          0.          ACCLKMODE: 03
STATUS:   0000
SEGN:     0000          0.          STVBN:    00000001
HDLBN:    0000CE22      STLBN:    00000000
FILESIZE: 0000000C      EFBLK:    0000000C
VERSIONS: 0000          0.          DIRINDX:  00000000
DIRSEQ:   0000          ACCLKID:  00283A1F
LOCKBASIS: 01001CA4      TRUNCVBN: 00000000
CACHELKID: 00000000      HIGHWATER: 0000000D
HWM_UPDATE: 0000          0.          HWM_PARTIAL: 0000          0.
HWM_ERASE: 0000          0.          HWM_WAITFL: 804A2AF8
FID:      [1CA4,0003,0001]
HWM_WAITBL: 804A2AF8

ORB
FILEOWNER: [007F,0009]      ACMODE:    FFFFFFFF/FFFFFFF
SYS_PROT:  0000EE0C
OWN_PROT:  00000000
GRP_PROT:  00000000
WOR_PROT:  00000000
ACLFL:     00000000
ACLBL:     00000000
  
```

FWA Address: 7FF9CC00

```

-----
FLAGS:    20C7F810
          06000003

PASSFLGS: 10          FNA_PASS
FLDFLGS:  F8          VERSION, TYPE, NAME, DIR, DEVICE
FLDFLGS:  C7          EXP_VER, EXP_TYPE, EXP_NAME, EXP_DIR, EXP_DEV
PARSEFLGS: 20
DIRFLGS:  03          DIR1, DIR2
DIRWFLGS: 00
LNFLGS:   00
SLFLGS:   06          CONCEAL_DEV, ROOT_DIR

DIRTERM:   5D          "]"
ESCSTRING: 00000000      ESCFLG:    00
ESCTYP:    00          ESCIFI:    0000
BUFFLG:    00          0.          DIRBDB:    00000000
XLTMODE:   03          XLTSIZE:   000C          12.
XLTBUFF1:  7FF9D600      XLTBUFF2:  7FF9D700
DEVBUFSIZ: 00000000      DEV_CLASS: 00000001
  
```

SYSTEM DUMP ANALYZER

SHOW PROCESS

FIB: 00000040 / 7FF9CDF4

 FIB fields within the FWA:

```

WSIZE:          00          0.      WCC:          00000000
ACCTL:          00000001      NOWRITE
FID:            (1CA4,0003,0001)    DID:      (148B,0002,0001)
NUMCTL:         0000
EXCTL:          0000
CNTRLFUNC:     0000      REWINDFIL
EXSZ:           00000000          CNTRLVAL:     00000000
EXVBN:         00000000          ALALIGN:      00
ALOPTS:         00
VERLIMIT:      7FFF          ACLCTX:          00000000
STATUS:         00000000
ALT_ACCESS:    00000000
  
```

```

UIC:            [0,0]          LOOKUP:        00000000
DEVNODADR:     7FFE03F0          PRO:          0000
LEVEL:         03          3.      UCHAR:         0000
DIRLEN:        0B          11.     SUBNODCNT:     00          0.
SWB_PTR:       00000000
SLBH_PTR:      7FF9C8F8          SLB_PTR:       7FF9C8F8
SLBH_FLINK:    7FF9C8F8          SLBH_BLINK:    7FF9C938
ITM_INDEX:     00010004 / 7FF9CCC4 / 00000000
ITM_ATTR:      00030004 / 7FF9CCC4 / 00000000
ITM_STRING:    000200FF / 7FF9D600 / 7FF9CC9E
ITM_MAX_INDEX: 00070004 / 7FF9CCC8 / 00000000
ITM_END:       00000000
LOGNAM:        0000000B / 7FF9D600 .....
DIR:           00000030 / 7FF9D468 SRCRTLDDUA101.....
NODE:          00000000 / 7FF9D3E9
DEVICE:        0000000B / 7FF9D1E9 $254$DUA101
CONCEAL_DEV:  0000000C / 7FF9D2E8 VMSTESTRES$
CDIR1:         00000007 / 7FF9CF7E VMSTEST
CDIR2:         00000000 / 7FF9CFA5
CDIR3:         00000000 / 7FF9CFCC
CDIR4:         00000000 / 7FF9CFF3
CDIR5:         00000000 / 7FF9D01A
CDIR6:         00000000 / 7FF9D041
CDIR7:         00000000 / 7FF9D068
CDIR8:         00000000 / 7FF9D08F
DIR1:          00000006 / 7FF9CE46 MTHRTL
DIR2:          00000003 / 7FF9CE6D SRC
DIR3:          00000000 / 7FF9CE94
DIR4:          00000000 / 7FF9CEBB
DIR5:          00000000 / 7FF9CEE2
DIR6:          00000000 / 7FF9CF09
DIR7:          00000000 / 7FF9CF30
DIR8:          00000000 / 7FF9CF57
NAME:          00000012 / 7FF9D0B6 MTH$ATAN2TST.B32;1
TYPE:          00000003 / 7FF9D0C3 B32
RNS:           0000012E / 7FF9D0B6 MTH$ATAN2TST.B32;1
VERSION:       00000001 / 7FF9D0C7 1
SHRFIL:        0000000C / 7FF9D498 _$254$DUA101
AS_SHRFIL:     0000000C / 7FF9D4B8 _$254$DUA101
SHRFIL_LCK:    00000010 / 7FF9D4A8 .RES28JUN ....
NODE1:         00000000 / 00000000
NODE2:         00000000 / 00000000
NODE3:         00000000 / 00000000
NODE4:         00000000 / 00000000
NODE5:         00000000 / 00000000
NODE6:         00000000 / 00000000
NODE7:         00000000 / 00000000
NODE8:         00000000 / 00000000
  
```

BI Journaling ACE:
 (UNKNOWN=%X00,SIZE=%DO,FLAGS=%X0000,ACCESS=%X00000000,DATA)

AI Journaling ACE:
 (UNKNOWN=%X00,SIZE=%DO,FLAGS=%X0000,ACCESS=%X00000000,DATA)

AT Journaling ACE:
 (UNKNOWN=%X00,SIZE=%DO,FLAGS=%X0000,ACCESS=%X00000000,DATA)

RU Journaling ACE:
 (UNKNOWN=%X00,SIZE=%DO,FLAGS=%X0000,ACCESS=%X00000000,DATA)

SYSTEM DUMP ANALYZER

SHOW PROCESS

IRAB Address: 7FF9C958 ISI: 0001

```

-----
IFAB_LNK:      7FF9C808
BKPBITS:      00000400      RAHWBH
BLN:          27          39.  BID:          0A          10.
EFN:          00          MODE:          03
IOS:          00000000      ASBADDR:      7FF9CA00
IOS4:         00000000      ARGST:        7FF21418
IRAB_LNK:     00000000      CURBDB:       00000000
LAST_RAB:    00081DA4      NXTBDB:       7FF9CB00
JNLBDB:       00000000
RLB_FLINK:    7FF9C99C      WAIT_Q_FLINK: 00000000
RLB_BLINK:    7FF9C99C      WAIT_Q_BLINK: 00000000
ISI:          0001          IDENT:         00000230
JNLFLG:       00
ATJNL_PTR:    00000000
RP_VBN:       00000000      0.  NRP_VBN:      00000001      1.
RP_OFF:       00000000      0.  NRP_OFF:       00000000      0.
CURVBN:       00000000      0.
CACHEFLGS:    01          LOCK
SRCHFLGS:     0000
SPL_BITS:     00
STOPLEVEL:    00          0.
POS_INS:      0000          0.  SPLIT:         0000          0.
LST_REC:      00000000      PRT_VBN:      00000000      0.
SPLIT_1:      0000          0.  SPLIT_2:       0000          0.
OWNER_ID:     00010000      BCNT:         03          3.
OWN_ISI:      0001          1.  MBC:          0B          11.
RU_HANDLE:    00000000      MBF:          03          3.
PPF_ISI:      01          1.  OWN_ID:        0000
MBC_P1:       0C          12.
TEMPO:        00000000      CSIZ:         0000          0.
ROVHDSZ:     0000          0.  RTOTLSZ:       0000          0.
PRE_CCTL:     00          0.  POST_CCTL:     00          0.
CURBLKADR:    00000000      PPF_STRLEN:   0000
ENDBLKADR:    00000000
PPF_STR:      00000000
    
```

RAB Address: 00081DA4

```

-----
BID:          01          1.  ISI:          0001
BLN:          44          68.
ROP:          00010600      RAH, WBH, LOC
CTX:          00000000      RAC:          00          SEQ
STS:          00000000      RFA:          00000000, 0000
STV:          00010001
TMO:          00          0.  RHB:          00000000
USZ:          0084          132.  UBF:          00091F38
RSZ:          0000          0.  RBF:          00000000
KBF:          00000000      KSZ:          00          0.
PBF:          00000000      PSZ:          00          0.
KRF:          00          0.  MBC:          00          0.
MBF:          00          0.  BKT:          00000000
FAB:          00081FD0      DCT:          00000000
XAB:          00000000
    
```

ASB Address: 7FF9CA00

```

-----
ARGCNT: 00          0.
ARGLST: 00000000      BID:          0D          13.
FABRAB: 00000000      BLN:          44          68.
ERR:     00000000      STKLEN:       00E0          224.
SUC:     00000000      STKSIZ:       0000          0.

R6:     00000000
R7:     00000000
R8:     00000000
R10:    00000000
R11:    00000000
    
```

SYSTEM DUMP ANALYZER

SHOW PROCESS

Saved Stack:

 SP => (STACK IS EMPTY)

BDB/GBPB Summary

Address	USERS	SIZE	NUMB	VBN	BLB_PTR	ADDR	VAL	ID	FLGS
7FF9CB80	0	6144	6144	00000001	00000000	7FFA0C00	0	BDB	IOP
7FF9CB10	0	6144	0	00000000	00000000	7FF9DC00	0	BDB	
7FF9CB60	0	6144	0	00000000	00000000	7FF9F400	0	BDB	

3. BDBs were processed
 0. GBPBs were processed

BDB Address: 7FF9CB80

```

-----
FLINK:      7FF9CB10          BID:          0C          12.
BLINK:      7FF9C850          BLN:          14          20.
FLGS:       04          IOP
USERS:      0000          0.          BLB_PTR:      00000000
CACHE_VAL:  00          0.          BUFF_ID:      0000          0.
SIZE:       1800          6144.        NUMB:         1800          6144.
ADDR:       7FFA0C00          VBN:          00000001
VBNSEQNO:   00000000          WAIT:         00000000
WK1:        00000000          CURBUFADR:    00000000
REL_VBN:    00          0.          PRE_CCTL:     00
ASB:        00000000
ALLOC_ADDR: 7FFA0C00          BI_BDB:       00000000
ALLOC_SIZE: 1800          6144          AI_BDB:       00000000
VAL_VBNS:   00          0.          POST_CCTL:    00
IOSB:       00000000          WAIT_Q_FLINK: 00000000
REUSE_COUNT: 00000000          WAIT_Q_BLINK: 00000000
  
```

BDB Address: 7FF9CB10

```

-----
FLINK:      7FF9CB60          BID:          0C          12.
BLINK:      7FF9CB80          BLN:          14          20.
FLGS:       00
USERS:      0000          0.          BLB_PTR:      00000000
CACHE_VAL:  00          0.          BUFF_ID:      0000          0.
SIZE:       1800          6144.        NUMB:         0000          0.
ADDR:       7FF9DC00          VBN:          00000000
VBNSEQNO:   00000000          WAIT:         00000000
WK1:        00000000          CURBUFADR:    00000000
REL_VBN:    00          0.          PRE_CCTL:     00
ASB:        00000000
ALLOC_ADDR: 7FF9DC00          BI_BDB:       00000000
ALLOC_SIZE: 1800          6144          AI_BDB:       00000000
VAL_VBNS:   00          0.          POST_CCTL:    00
IOSB:       00000000          WAIT_Q_FLINK: 00000000
REUSE_COUNT: 00000000          WAIT_Q_BLINK: 00000000
  
```

SYSTEM DUMP ANALYZER

SHOW PROCESS

BDB Address: 7FF9CB60

FLINK:	7FF9C850		BID:	0C	12.
BLINK:	7FF9CB10		BLN:	14	20.
FLGS:	00				
USERS:	0000	0.	BLB_PTR:	00000000	
CACHE_VAL:	00	0.	BUFF_ID:	0000	0.
SIZE:	1800	6144.	NUMB:	0000	0.
ADDR:	7FF9F400		VBN:	00000000	
VBNSQNO:	00000000		WAIT:	00000000	
WK1:	00000000		CURBUFADR:	00000000	
REL_VBN:	00	0.	PRE_CCTL:	00	
ASB:	00000000				
ALLOC_ADDR:	7FF9F400		BI_BDB:	00000000	
ALLOC_SIZE:	1800	6144	AI_BDB:	00000000	
VAL_VBNS:	00	0.	POST_CCTL:	00	
IOSB:	00000000		WAIT_Q_FLINK:	00000000	
	00000000		WAIT_Q_BLINK:	00000000	
REUSE_COUNT:	00000000				

The SHOW PROCESS/RMS command displays those RMS data structures associated with the image files process GLOBE, the SDA current process, is accessing.

SYSTEM DUMP ANALYZER

SHOW RESOURCE

Table SDA-13 (Cont.) Resource Information in the SHOW RESOURCE Display

Field	Contents														
Group grant mode	<p>Indication of the most restrictive mode in which a lock on this resource has been granted. This field can contain the following values (shown in order from the least restrictive mode to the most restrictive):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>NL</td> <td>Null mode</td> </tr> <tr> <td>CR</td> <td>Concurrent-read mode</td> </tr> <tr> <td>CW</td> <td>Concurrent-write mode</td> </tr> <tr> <td>PR</td> <td>Protected-read mode</td> </tr> <tr> <td>PW</td> <td>Protected-write mode</td> </tr> <tr> <td>EX</td> <td>Exclusive mode</td> </tr> </tbody> </table>	Value	Meaning	NL	Null mode	CR	Concurrent-read mode	CW	Concurrent-write mode	PR	Protected-read mode	PW	Protected-write mode	EX	Exclusive mode
Value	Meaning														
NL	Null mode														
CR	Concurrent-read mode														
CW	Concurrent-write mode														
PR	Protected-read mode														
PW	Protected-write mode														
EX	Exclusive mode														
Conversion grant mode	<p>Indication of the most restrictive lock mode to which a lock on this resource is waiting to be converted. This does not include the mode for which the lock at the head of the conversion queue is waiting.</p>														
BLKAST count	<p>Number of locks on this resource that have requested a blocking AST.</p>														
Value block	<p>Hexadecimal dump of the 16-byte block value block associated with this resource.</p>														
Sequence #	<p>Sequence number associated with the resource's value block. If the number indicates that the value block is not valid, the words "Not valid" appear to the right of the number.</p>														
CSID	<p>Cluster system identification number (CSID) of the node that owns the resource</p>														
Resource	<p>Dump of the name of this resource, as stored at the end of the RSB. The first two columns are the hexadecimal representation of the name, with the least significant byte represented by the rightmost two digits in the rightmost column. The third column contains the ASCII representation of the name, the least significant byte being represented by the leftmost character in the column. Periods in this column represent values that correspond to nonprinting ASCII characters.</p>														
Length	<p>Length in bytes of the resource name.</p>														

SYSTEM DUMP ANALYZER

SHOW RESOURCE

Table SDA-13 (Cont.) Resource Information in the SHOW RESOURCE Display

Field	Contents
---	Processor mode of the name space in which this RSB resides.
---	Owner of the resource. Certain resources, owned by the VMS operating system, list "System" as the owner. Locks owned by a group have the number (in octal) of the owning group in this field.
Granted queue	List of locks on this resource that have been granted. For each lock in the list, SDA displays the number of the lock and the lock mode in which the lock was granted.
Conversion queue	List of locks waiting to be converted from one mode to another. For each lock in the list, SDA displays the number of the lock, the mode in which the lock was granted, and the mode to which the lock is to be converted.
Waiting queue	List of locks waiting to be granted. For each lock in the list, SDA displays the number of the lock and the mode requested for that lock.

EXAMPLES

```

1 SDA> SHOW RESOURCE
Resource database
-----
Address of RSB: 807F6120 Group grant mode: NL
Parent RSB: 806EA180 Conversion grant mode: NL
Sub-RSB count: 0 BLKAST count: 0
Value block: 806CE510 00000000 00000002 00000002 Seq. #: 00000008
Resource: 09ED7324 42313146 F11B$s.
Length 10 00000000 00000200 ..... CSID: 00020041
Kernel mode 00000000 00000000 .....
System 00000000 00000000 .....

Granted queue (Lock ID / Gr mode):
006801AE NL

Conversion queue (Lock ID / Gr/Rq mode):
*** EMPTY QUEUE ***

Waiting queue (Lock ID / Rq mode):
*** EMPTY QUEUE ***

Address of RSB: 807EB9E0 Group grant mode: PW
Parent RSB: 00000000 Conversion grant mode: EX
Sub-RSB count: 0 BLKAST count: 1
Value block: 00000000 00000003 00000000 0000FFF2 Seq. #: 0000027F Not valid
Resource: 32245F24 44414853 SHAD$_$2
Length 16 3A31534A 44243435 54$DJS1: CSID: 0002001A
Kernel mode 00000000 00000000 .....
System 00000000 00000000 .....

Granted queue (Lock ID / Gr mode):
00020301 CR

```

SYSTEM DUMP ANALYZER

SHOW RESOURCE

Conversion queue (Lock ID / Gr/Rq mode):
095B00F2 PW/EX

Waiting queue (Lock ID / Rq mode):
054400BC EX

The SHOW RESOURCE command displays information taken from the RSBs of all resources in the system. For instance, the RSB at 807EB9E0₁₆ is a parent block with no sub_RSBs. The most restrictive lock granted on this resource is in protected-write (PW) mode. There is a lock on the conversion queue waiting to be converted from PW mode to exclusive (EX) mode.

2

SDA> SHOW PROCESS/LOCKS

Process index: 001C Name: STARTQ Extended PID: 4800011C

Lock data:

Lock id: 0117054F PID: 0001001C Flags: VALBLK SYNCSTS SYSTEM
Par. id: 00000000 Granted at PW NOQUOTA
Sublocks: 0
LKB: 808091A0
Resource: 45527624 42313146 F11B\$vRE Status: NOQUOTA
Length 18 20205241 4D323053 SO2MAR
Kernel mode 00000000 00002020
System 00000000 00000000
Process copy of lock 008209CF on system 0002001

SDA> SHOW RESOURCE/LOCKID=117054F

Resource database

Address of RSB: 806BB050 Group grant mode: NL
Parent RSB: 00000000 Conversion grant mode: NL
Sub-RSB count: 4 BLKAST count: 0
Value block: 00960102 0000330B 000735AA 5A020005 Seq. #: 00006D9F
Resource: 45527624 42313146 F11B\$vRE
Length 18 20205241 4D323053 SO2MAR CSID: 0002001A
Kernel mode 00000000 00002020
System 00000000 00000000

Granted queue (Lock ID / Gr mode):
0117054F PW 00060545 CR

Conversion queue (Lock ID / Gr/Rq mode):
*** EMPTY QUEUE ***

Waiting queue (Lock ID / Rq mode):
*** EMPTY QUEUE ***

The SHOW PROCESS/LOCKS command lists all locks associated with the SDA current process, STARTQ. Its display is identical to that of the SHOW LOCK command, illustrated in Table SDA-7. The SHOW RESOURCE/LOCKID=117054F command determines that this particular lock is on the granted queue in protected-write mode for the resource at 806BB050₁₆.

SHOW RMS

Displays the RMS data structures selected by the SET RMS command to be included in the default display of the SHOW PROCESS/RMS command.

FORMAT SHOW RMS

PARAMETERS *None.*

QUALIFIERS *None.*

DESCRIPTION The SHOW RMS command lists the names of the data structures selected for the default display of the SHOW PROCESS/RMS command.

For a description of the significance of the options listed in the SHOW RMS display, see the description of the SET RMS command and Table SDA-6.

For an illustration of the information displayed by the SHOW PROCESS/RMS command, see the examples included in the description of the SHOW PROCESS command.

EXAMPLES

1 SDA> SHOW RMS

RMS Display Options: IFB,IRB,IDX,BDB,BDBSUM,ASB,CCB,WCB,FCB,FAB,RAB,NAM,XAB,RLB, BLB,BLBSUM,GBD,GBH,FWA,GBDSUM,JFB,NWA,RU,DRC,SFSB,GBSB
Display RMS structures for all IFI values.

The SHOW RMS command displays the full set of options available for display by the SHOW PROCESS/RMS command. SDA, by default, selects the full set of RMS options at the beginning of an analysis.

2 SDA> SET RMS=(IFAB,CCB,WCB)
SDA> SHOW RMS

RMS Display Options: IFB,CCB,WCB
Display RMS structures for all IFI values.

The SET RMS command establishes the IFB, CCB, and WCB as the structures to be displayed when the SHOW PROCESS/RMS command is issued. The SHOW RMS command verifies this selection of RMS options.

SYSTEM DUMP ANALYZER

SHOW RSPID

SHOW RSPID

Displays information about response IDs (RSPIDs) of all SCS connections or, optionally, a specific SCS connection.

FORMAT **SHOW RSPID** *[/CONNECTION=cdt-address]*

PARAMETERS *None.*

QUALIFIER ***/CONNECTION=cdt-address***
Displays RSPID information for the specific SCS connection whose connection descriptor table (CDT) address is provided in **cdt-address**.¹⁵

DESCRIPTION Whenever a local system application (SYSAP) requires a response from a remote SYSAP, a unique number, called an RSPID, is assigned to the response by the local system. The RSPID is transmitted in the original request (as a means of identification), and the remote SYSAP returns the same RSPID in its response to the original request.

The SHOW RSPID command displays information taken from the response descriptor table (RDT), which lists the currently open local requests that require responses from SYSAPs at a remote node. For each RSPID, SDA displays the following information:

- RSPID value
- Address of the class driver request packet (CDRP) which generally represents the original request
- Address of the CDT that is using the RSPID
- Name of the local process using the RSPID
- Remote node from which a response is required (and has not yet been received).

¹⁵ You can find the **cdt-address** for any active connection on the system in the **CDT summary page** display of the SHOW CONNECTIONS command. In addition, CDT addresses are also stored in many individual data structures related to SCS connections. These data structures include class driver request packets (CDRPs) and unit control blocks (UCBs) for class drivers that use SCS and cluster system blocks (CSBs) for the connection manager.

SYSTEM DUMP ANALYZER

SHOW RSPID

EXAMPLES

1 SDA> SHOW RSPID
VAXcluster data structures

--- Summary of Response Descriptor Table(RDT) 8037A4A8 ---

RSPID	CDRP Address	CDT Address	Local Process Name	Remote Node
04C30000	803917B0	8037AB50	VMS\$DISK_CL_DRVR	SOWHAT
06260001	80804FA0	8037AF10	VMS\$VAXcluster	WALKIN
0C390002	807E0460	8037AD30	VMS\$VAXcluster	OLEO

The SHOW RSPID command shows the response IDs that are currently open for all local connections in the VAXcluster.

2 SDA> SHOW RSPID/CONNECTION=G37B7D0
VAXcluster data structures

--- Summary of Response Descriptor Table(RDT) 8037A4A8 ---

RSPID	CDRP Address	CDT Address	Local Process Name	Remote Node
08B8001C	807F0300	8037B7D0	VMS\$VAXcluster	METEOR
0915001D	807F08A0	8037B7D0	VMS\$VAXcluster	METEOR

The SHOW RSPID/CONNECTION=G37B7D0 command displays only those RSPIDs in use that are associated with the SCS connection whose CDT is at address 8037B7D0₁₆.

SYSTEM DUMP ANALYZER

SHOW SPINLOCKS

SHOW SPINLOCKS

Displays information taken from the data structures that provide system synchronization in a VMS multiprocessing environment.

FORMAT

SHOW SPINLOCKS *[/OWNED]* *[/BRIEF]*
[/FULL]
[/DYNAMIC]
[/STATIC]
[name]
[/ADDRESS=expression]
[/INDEX=expression]

PARAMETER

name

Name of the spin lock, fork lock, or device lock structure to be displayed. You can obtain the names of the static system spin locks and fork locks from Table SDA-14. Device lock names are of the form *[node\$]lock*, where *node* optionally indicates the VAXcluster node name (allocation class) and *lock* indicates the device and controller identification (for example, HAETAR\$DUA).

QUALIFIERS

/ADDRESS=expression

Displays the lock at the address specified in **expression**. You can use the */ADDRESS* qualifier to display a specific device lock; however the name of the device lock is listed as "Unknown" in the display.

/BRIEF

Produces a condensed display of the lock information displayed by default by the SHOW SPINLOCKS command, including the following: address, spin lock name or device name, IPL or device IPL, rank, index, ownership depth, number of waiting CPUs, CPU ID of the owner CPU, and interlock status (depth of ownership).

/DYNAMIC

Displays information for all device locks in the system.

/FULL

Displays full descriptive and diagnostic information for each displayed spin lock, fork lock, or device lock.

/INDEX=expression

Displays the system spin lock whose index is specified in **expression**. You cannot use the */INDEX* qualifier to display a device lock.

SYSTEM DUMP ANALYZER

SHOW SPINLOCKS

/OWNED

Displays information for all spin locks, fork locks, and device locks owned by the SDA current CPU. If a processor does not own any spin locks, SDA displays the following message:

No spinlocks currently owned by CPU xx

xx represents the CPU ID of the processor.

/STATIC

Displays information for all system spin locks and fork locks.

DESCRIPTION

The SHOW SPINLOCKS command displays status and diagnostic information about the VMS multiprocessing synchronization structures known as spin locks.

A *static spin lock* is a spin lock the data structure for which is permanently assembled into the system. Static spin locks are accessed as indexes into a vector of longword addresses called the *spin lock vector*, the address of which is contained in SMP\$AR_SPNLKVEC. System spin locks and fork locks are static spin locks. Table SDA-14 lists the static spin locks.

A *dynamic spin lock* is a spin lock that is created based on the configuration of a particular system. One such dynamic spin lock is the device lock SYSGEN creates when configuring a particular device. This device lock synchronizes access to the device's registers and certain UCB fields. VMS creates a dynamic spin lock by allocating space from nonpaged pool, rather than assembling the lock into the system as it does in creating a static spin lock.

See the *VMS Device Support Manual* for a full discussion of the role of spin locks in maintaining synchronization of kernel mode activities in a VMS multiprocessing environment.

Table SDA-14 Static Spin Locks

Name	Description
QUEUEAST	Fork lock for queuing ASTs at IPL 6
FILSYS	Lock on file system structures
IOLOCK8	Fork lock for executing a driver fork process at IPL 8
PR_LK8	Primary CPU's private lock for IPL 8
TIMER	Lock for adding and deleting timer queue entries and searching the timer queue
JIB	Lock for manipulating job nonpaged pool quotas as reflected by the fields JIB\$_BYTCNT and JIB\$_BYTLM in the job information block (JIB)
MMG	Lock on VMS memory management, PFN database, swapper, modified page writer, and creation of per-CPU database structures
SCHED	Lock on process control blocks (PCBs), scheduler database, and mutex acquisition and release structures
IOLOCK9	Fork lock for executing a driver fork process at IPL 9
PR_LK9	Primary CPU's private lock for IPL 9
IOLOCK10	Fork lock for executing a driver fork process at IPL 10
PR_LK10	Primary CPU's private lock for IPL 10

SYSTEM DUMP ANALYZER

SHOW SPINLOCKS

Table SDA-14 (Cont.) Static Spin Locks

Name	Description
IOLOCK11	Fork lock for executing a driver fork process at IPL 11
PR_LK11	Primary CPU's private lock for IPL 11
MAILBOX	Lock for sending messages to mailboxes
POOL	Lock on nonpaged pool database
PERFMON	Lock for I/O performance monitoring
INVALIDATE	Lock for system space translation buffer (TB) invalidation
VIRTCONS	Lock for ownership of the virtual console
HWCLK	Lock on hardware clock database, including the quadword containing the due time of the first timer queue entry (EXE\$GQ_1ST_TIME) and the quadword containing the system time (EXE\$GQ_SYSTIME)
MEGA	Lock for serializing access to fork-wait queue
MCHECK	Lock for synchronizing certain machine error handling
EMB	Lock for allocating and releasing error logging buffers

For each spin lock, fork lock, or device lock in the system, SHOW SPINLOCKS provides the following information:

- Name of the spin lock (or device name for the device lock)
- Address of the spin lock data structure (SPL)
- The owner CPU's CPU ID
- IPL at which allocation of the lock is synchronized on a local processor
- Number of nested acquisitions of the spin lock by the processor owning the spin lock ("Ownership Depth")
- Rank of the spin lock
- Number of processors waiting to obtain the spin lock
- Spin lock index
- Timeout interval for spin lock acquisition (in terms of 10 milliseconds)

SHOW SPINLOCKS/BRIEF produces a condensed display of this same information.

If the VAX system under analysis was executing with full-checking multiprocessing enabled (according to the setting of the MULTIPROCESSING system parameter), SHOW SPINLOCKS/FULL adds to the spin lock display the last eight PCs at which the lock was acquired or released. If applicable, SDA also displays the PC of the last release of multiple, nested acquisitions of the lock.

SYSTEM DUMP ANALYZER

SHOW SPINLOCKS

EXAMPLES

```
1 SDA> SHOW SPINLOCKS
System static spinlock structures
-----
EMB                               Address : 801B9EF8
Owner CPU ID      : None          IPL      : 1F
Ownership Depth   : 0000          Rank     : 00
CPUs Waiting      : 0000          Index    : 20
Timeout interval  002DC60

MCHECK                            Address : 801B9F48
Owner CPU ID      : None          IPL      : 1F
Ownership Depth   : 0000          Rank     : 01
CPUs Waiting      : 0000          Index    : 21
Timeout interval  002DC60

.
.
IOLOCK8                          Address : 801BA538
Owner CPU ID      : 02           IPL      : 08
Ownership Depth   : 0001          Rank     : 14
CPUs Waiting      : 0000          Index    : 34
Timeout interval  002DC60

.
.
System dynamic spinlock structures
-----
HAETAR$MBA                       Address : 801BA178
Owner CPU ID      : None          IPL      : 0B
Ownership Depth   : 0000          Rank     : 08
CPUs Waiting      : 0000          Index    : 37
Timeout interval  002DC60

HAETAR$NLA                       Address : 801BA178
Owner CPU ID      : None          IPL      : 08
Ownership Depth   : 0000          Rank     : 08
CPUs Waiting      : 0000          Index    : 37
Timeout interval  002DC60

HAETAR$PAA                       Address : 801BA538
Owner CPU ID      : 02           IPL      : 14
Ownership Depth   : 0001          Rank     : 14
CPUs Waiting      : 0000          Index    : 37
Timeout interval  002DC60

.
.
```

This excerpt illustrates the default output of the SHOW SPINLOCKS command. Note that the fork lock IOLOCK8 is owned by the CPU whose CPU ID is 2. CPU 2 must be executing at at least IPL 8, which is the acquisition IPL of the fork lock. CPU 2 has no nested ownership of the fork lock. The rank of IOLOCK8 is 14₁₆, indicating that CPU 2 could not own any locks with a logical rank of 15₁₆ or higher when it acquired IOLOCK8.

SYSTEM DUMP ANALYZER

SHOW SPINLOCKS

Similarly, while owning IOLOCK8, CPU 2 cannot obtain any additional spin locks with a logical rank of 14₁₆ or below.

No CPUs are waiting for the fork lock; its index is 34₁₆.

2

SDA> SHOW SPINLOCKS/BRIEF

Address	Spinlock Name	IPL	Rank	Index	Depth	#Waiting	Owner CPU	Interlock
801B9EF8	EMB	1F	00	20	00	0000	None	Free
801B9F48	MCHECK	1F	01	21	00	0000	None	Free
801B9F98	MEGA	1F	02	22	00	0000	None	Free
801B9FE8	HWCLK	16	03	23	00	0000	None	Free
801BA038	VIRTCONS	14	04	24	00	0000	None	Free
801BA088	INVALIDATE	13	05	25	00	0000	None	Free
801BA0D8	PERFMON	0F	06	26	00	0000	None	Free
801BA128	POOL	0B	07	27	00	0000	None	Free
801BA178	MAILBOX	0B	08	28	00	0000	None	Free
801BA1C8	PR_LK11	0B	09	29	00	0000	None	Free
801BA218	IOLOCK11	0B	0A	2A	00	0000	None	Free
801BA268	PR_LK10	0A	0B	2B	00	0000	None	Free
801BA2B8	IOLOCK10	0A	0C	2C	00	0000	None	Free
801BA308	PR_LK9	09	0D	2D	00	0000	None	Free
801BA358	IOLOCK9	09	0E	2E	00	0000	None	Free
801BA3A8	SCHED	08	0F	2F	00	0000	None	Free
801BA3F8	MMG	08	10	30	00	0000	None	Free
801BA448	JIB	08	11	31	00	0000	None	Free
801BA498	TIMER	08	12	32	00	0000	None	Free
801BA4E8	PR_LK8	08	13	33	00	0000	None	Free
801BA538	IOLOCK8	08	14	34	01	0000	02	00
801BA588	FILSYS	08	15	35	00	0000	None	Free
801BA5D8	QUEUEAST	06	16	36	00	0000	None	Free

Address	Device Name	DIPL	Rank	Index	Depth	#Waiting	Owner CPU	Interlock
801BA178	HAETAR\$MBA	0B	08	37	00	0000	None	Free
801BA178	HAETAR\$NLA	08	08	37	00	0000	None	Free
801BA538	HAETAR\$PAA	14	14	37	01	0000	02	00
8063C5C0	HAETAR\$XEA	15	FF	37	00	0000	None	Free
8063C4A0	HAETAR\$XGA	15	FF	37	00	0000	None	Free
8063C380	HAETAR\$PEA	14	FF	37	00	0000	None	Free
8063AC40	HAETAR\$TXA	15	FF	37	00	0000	None	Free
8063A520	HAETAR\$LCA	15	FF	37	00	0000	None	Free
801BA538	HAETAR\$CNA	08	14	37	01	0000	02	00

This excerpt illustrates the condensed form of the display produced in the previous example.

SYSTEM DUMP ANALYZER

SHOW SPINLOCKS

```
3 SDA> SHOW SPINLOCKS/OWNED
System static spinlock structures
-----
IOLOCK8                               Address : 801BA538
Owner CPU ID       : 02                 IPL      : 08
Ownership Depth   : 0001                Rank     : 14
CPUs Waiting      : 0000                Index    : 34
Timeout interval  002DC60

.

System dynamic spinlock structures
-----
HAETAR$PAA                               Address : 801BA538
Owner CPU ID       : 02                 IPL      : 14
Ownership Depth   : 0001                Rank     : 14
CPUs Waiting      : 0000                Index    : 34
Timeout interval  002DC60

HAETAR$CNA                               Address : 801BA538
Owner CPU ID       : 02                 IPL      : 08
Ownership Depth   : 0001                Rank     : 14
CPUs Waiting      : 0000                Index    : 34
Timeout interval  002DC60

HAETAR$NET                               Address : 801BA538
Owner CPU ID       : 02                 IPL      : 08
Ownership Depth   : 0001                Rank     : 14
CPUs Waiting      : 0000                Index    : 34
Timeout interval  002DC60

HAETAR$NDA                               Address : 801BA538
Owner CPU ID       : 02                 IPL      : 08
Ownership Depth   : 0001                Rank     : 14
CPUs Waiting      : 0000                Index    : 34
Timeout interval  002DC60

.
.
```

The SHOW SPINLOCKS/OWNED command shows all owned spin locks in the system.

SYSTEM DUMP ANALYZER

SHOW SPINLOCKS

```
4 SDA> SHOW SPINLOCKS/FULL
System static spinlock structures
-----
EMB                                Address : 801B9EF8
Owner CPU ID      : None           IPL       : 1F
Ownership Depth   : 0000           Rank      : 00
CPUs Waiting      : 0000           Index     : 20
Timeout interval  002DC60

Spinlock EMB was last acquired or released from:
(Most recently)      80195146 ERL$WAKE+00089
                    801950EF ERL$WAKE+00032
                    .
                    80195146 ERL$WAKE+00089
                    .
                    801950EF ERL$WAKE+00032
                    .
                    80195146 ERL$WAKE+00089
                    .
                    801950EF ERL$WAKE+00032
                    .
                    80195146 ERL$WAKE+00089
                    .
                    801950EF ERL$WAKE+00032
(Least recently)

Last release of multiple acquisitions occurred at:
                    801194F9 EXE$INSIOQ+00044

IOLOCK8                                Address : 801BA538
Owner CPU ID      : 02              IPL       : 08
Ownership Depth   : 0001           Rank      : 14
CPUs Waiting      : 0000           Index     : 34
Timeout interval  002DC60

Spinlock IOLOCK8 was last acquired or released from:
(Most recently)      801BBE08 EXE$FORKDSPH+0007E
                    80198EBF EXE$QIOACPPKT+00052
                    .
                    80198E7E EXE$QIOACPPKT+00011
                    .
                    80199BB2 IOC$CHECK_HWM+0032D
                    .
                    80182DE5 LCK$QUEUED_EXIT+0001D
                    .
                    80182884 LCK$AR_COMPAT_TBL+0007C
                    .
                    8018357E EXE$DEQ+00189
                    .
                    80183428 EXE$DEQ+00033
(Least recently)
```

The SHOW SPINLOCKS/FULL command displays a list of the last eight PCs that have accessed the spin lock. For instance, the fork dispatcher contains the code that most recently acquired the fork lock.

SYSTEM DUMP ANALYZER

SHOW STACK

SDA provides the following information in each stack display:

Section	Contents
Identity of stack	SDA indicates whether the stack is a process stack (user, supervisor, executive, or kernel) or the processor interrupt stack. If the interrupt stack is being displayed, SDA displays the CPU ID of the processor that owns it. Similarly, if the SDA current process is currently scheduled on a processor in the VAX system, SHOW STACK also specifies the CPU ID of the processor on which the process is scheduled.
Stack pointer	The stack pointer identifies the top of the stack. The display indicates the stack pointer by the symbol SP => .
Stack address	SDA lists all the virtual addresses that the operating system has allocated to the stack. The stack addresses are listed in a column that increases in increments of 4 bytes (one longword).
Stack contents	SDA lists the contents of the stack in a column to the right of the stack addresses.
Symbols	SDA attempts to display the contents of a location symbolically, using a symbol and an offset. If the address is not within FFF_{16} of the value of any existing symbol, this column is left blank.

If a stack is empty, the display shows the following:

```
SP => (STACK IS EMPTY)
```

EXAMPLE

```
SDA> SHOW STACK
Process stacks (on CPU 00)
```

```
-----
Current operating stack (USER):
```

```
       7FF73278 200C0000
       7FF7327C 00001518      SGN$C_MAXPGFL+518
       7FF73280 7FF732F0
       7FF73284 000187A7      RMS$_ECHO+72E
SP => 7FF73288 0000060A      BUG$_NOHDJMT+002
       7FF7328C 00000000
       7FF73290 00000003
       7FF73294 7FF73800
       7FF73298 7FF73800
```

The SHOW STACK command displays a user stack which was the current operating stack for a process scheduled on CPU 00. The data shown above the stack pointer may not be valid. The symbol to the right of the columns, `BUG$_NOHDJMT+002`, is the result of the SDA attempt to interpret the contents of the longword at the top of the stack as a symbol meaningful to the user. In this case the value on the stack and the value of `BUG$_NOHDJMT` are unrelated.

SYSTEM DUMP ANALYZER

SHOW SUMMARY

SHOW SUMMARY

Displays a list of all active processes and the values of the parameters used in swapping and scheduling these processes.

FORMAT **SHOW SUMMARY** *[/IMAGE]*

PARAMETERS *None.*

QUALIFIER ***/IMAGE***
Causes SDA to display, if possible, the name of the image being executed within each process.

DESCRIPTION The SHOW SUMMARY command displays the information in Table SDA-15 for each active process in the system.

Table SDA-15 Process Information in the SHOW SUMMARY Display

Column	Contents																				
Extended PID	32-bit number that uniquely identifies the process																				
Indx	Index of this process into the PCB array																				
Process name	Name assigned to the process																				
Username	Name of the user who created the process																				
State	Current state of the process, one of the following 14 states:																				
	<table border="1"><thead><tr><th>State</th><th>Meaning</th></tr></thead><tbody><tr><td>COM</td><td>Computable and resident in memory</td></tr><tr><td>COMO</td><td>Computable, but outswapped</td></tr><tr><td>CUR</td><td>Currently executing¹</td></tr><tr><td>CEF</td><td>Waiting for a common event flag</td></tr><tr><td>LEF</td><td>Waiting for a local event flag</td></tr><tr><td>LEFO</td><td>Outswapped and waiting for a local event flag</td></tr><tr><td>HIB</td><td>Hibernating</td></tr><tr><td>HIBO</td><td>Hibernating and outswapped</td></tr><tr><td>SUSP</td><td>Suspended</td></tr></tbody></table>	State	Meaning	COM	Computable and resident in memory	COMO	Computable, but outswapped	CUR	Currently executing ¹	CEF	Waiting for a common event flag	LEF	Waiting for a local event flag	LEFO	Outswapped and waiting for a local event flag	HIB	Hibernating	HIBO	Hibernating and outswapped	SUSP	Suspended
State	Meaning																				
COM	Computable and resident in memory																				
COMO	Computable, but outswapped																				
CUR	Currently executing ¹																				
CEF	Waiting for a common event flag																				
LEF	Waiting for a local event flag																				
LEFO	Outswapped and waiting for a local event flag																				
HIB	Hibernating																				
HIBO	Hibernating and outswapped																				
SUSP	Suspended																				

¹For a process in the CUR state executing in a multiprocessing environment, SDA indicates the CPU ID of the processor on which the process is current. This information, however, may not be accurate in SHOW SUMMARY displays produced in the analysis of a running system.

SYSTEM DUMP ANALYZER

SHOW SUMMARY

Table SDA-15 (Cont.) Process Information in the SHOW SUMMARY Display

Column	Contents												
	<table border="1"> <thead> <tr> <th>State</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>SUSPO</td> <td>Suspended and outswapped</td> </tr> <tr> <td>PFW</td> <td>Waiting for a page that is not in memory (page-fault wait)</td> </tr> <tr> <td>FPG</td> <td>Waiting to add a page to its working set (free-page wait)</td> </tr> <tr> <td>COLPG</td> <td>Waiting for a page collision to be resolved (collided-page wait); this usually occurs when several processes cause page faults on the same shared page</td> </tr> <tr> <td>MWAIT</td> <td>Waiting for a system resource (miscellaneous wait)</td> </tr> </tbody> </table>	State	Meaning	SUSPO	Suspended and outswapped	PFW	Waiting for a page that is not in memory (page-fault wait)	FPG	Waiting to add a page to its working set (free-page wait)	COLPG	Waiting for a page collision to be resolved (collided-page wait); this usually occurs when several processes cause page faults on the same shared page	MWAIT	Waiting for a system resource (miscellaneous wait)
State	Meaning												
SUSPO	Suspended and outswapped												
PFW	Waiting for a page that is not in memory (page-fault wait)												
FPG	Waiting to add a page to its working set (free-page wait)												
COLPG	Waiting for a page collision to be resolved (collided-page wait); this usually occurs when several processes cause page faults on the same shared page												
MWAIT	Waiting for a system resource (miscellaneous wait)												
Pri	Current scheduling priority of the process												
PCB	Address of the process control block												
PHD	Address of the process header												
Wkset	Number (in decimal) of pages currently in the process's working set												

EXAMPLE

SDA> SHOW SUMMARY/IMAGE

Current process summary

```

-----
Extended  Indx Process name      Username      State  Pri  PCB      PHD      Wkset
--  PID  --  -----
33C00101  0001 SWAPPER                SIVAD        HIB    16  8000C3C0 8000C200  0
33C00205  0005 _RTA5:                SYSTEM       LEF    4  80482FE0 82120E00 293
33C00106  0006 ERRFMT              SYSTEM       HIB    8  80432950 80DB4600 126
          $254$DUA200: [SYS6.SYSCOMMON.] [SYSEXE]ERRFMT.EXE;1
33C00107  0007 CACHE_SERVER       SYSTEM       HIB    16  80432AC0 81121E00 120
          $254$DUA200: [SYS6.SYSCOMMON.] [SYSEXE]FILESERV.EXE;400
33C00108  0008 CLUSTER_SERVER     SYSTEM       HIB    10  804331F0 81246600 313
          $254$DUA200: [SYS6.SYSCOMMON.] [SYSEXE]CSP.EXE;300
          .
          .
          .
33C0010D  000D NETACP              DECNET       CUR    00 10 8044C6D0 816D8600 1500
          $254$DUA200: [SYS6.SYSCOMMON.] <SYSEXE>NETACP.EXE;3
33C0010E  000E EVL                 DECNET       HIB    4  8044CD60 817FCE00  68
          $254$DUA200: [SYS6.SYSCOMMON.] <SYSEXE>EVL.EXE
          .
          .
          .

```

The SHOW SUMMARY/INDEX command describes all active processes in the VAX system at the time of the system failure. Note that the process NETACP is in the CUR state on CPU 00 of a VAX multiprocessor at the time of the failure.

SYSTEM DUMP ANALYZER

SHOW SYMBOL

SHOW SYMBOL

Displays the hexadecimal value of a symbol and, if the value is equal to an address location, the contents of that location.

FORMAT **SHOW SYMBOL** *[/ALL] symbol-name*

PARAMETER *symbol-name*
Name of the symbol to be displayed. You must provide a **symbol-name**.

QUALIFIER */ALL*
Displays information on all symbols whose names begin with the characters specified in **symbol-name**.

DESCRIPTION The SHOW SYMBOL/*ALL* command is useful for determining the names of symbols that belong to a symbol set, as illustrated in the following example.

EXAMPLE

```
SDA> SHOW SYMBOL G
G = 80000000 : 8FBC0FFC
```

The SHOW SYMBOL command evaluates the symbol G as 80000000₁₆ and displays the contents of address 80000000₁₆ as 8FBC0FFC₁₆.

```
SDA> SHOW SYMBOL/ALL BUG
Symbols sorted by name
-----
BUG$BUILD_HEADE 80002038 => 24A89F16      BUG$_CONSOLRX50 00000640 => 10A2020E
BUG$DUMP_REGIST 80002040 => 24A89F16      BUG$_CONTRACT   000000C0
BUG$FATAL       80002048 => 24A89F16      BUG$_CPUBUSYWAI 00000780 => 6501FB30
BUG$L_BUGCHK_FL 80004108 => 00000001      BUG$_CPUCEASED  000005E8 => 5EDD0000
BUG$L_FATAL_SPS 8000410C => 7FFE7C6C      BUG$_CPUEXIT    000006B8 => 218FD007
BUG$READ_ERR_RE 80002050 => 24A89F16      BUG$_CPUSANITY  00000778 => 8A031164
BUG$REBOOT      80002058 => 6E9E9F17      BUG$_CTERM      00000678 => 00000004
BUG$TABLE       8000D09E => 00280001      BUG$_CWSERR     00000698 => 004C414E
```

The preceding example shows the display produced by the SHOW SYMBOL/*ALL* command. SDA searches its symbol table for all symbols that begin with the string "BUG" and displays the symbols and their values. Although certain values equate to memory addresses, it is doubtful that the contents of those addresses are actually relevant to the symbol definitions in this instance.

SYSTEM DUMP ANALYZER

SPAWN

SPAWN

Creates a subprocess of the process currently running SDA, copying the context of the current process to the subprocess and, optionally, executing within the subprocess a specified command.

FORMAT **SPAWN** *[/qualifier[,...]] [command]*

PARAMETER *command*
Name of the command that you want executed by the subprocess.

QUALIFIERS ***/INPUT=filespec***
Specifies an input file containing one or more command strings to be executed by the spawned subprocess. If you specify a command string with an input file, the command string is processed before the commands in the input file. Once processing is complete, the subprocess is terminated.

/NOLOGICAL_NAMES
Specifies that the logical names of the parent process are not to be copied to the subprocess. The default behavior is that the logical names of the parent process are copied to the subprocess.

/NOSYMBOLS
Specifies that the DCL global and local symbols of the parent process are not to be passed to the subprocess. The default behavior is that these symbols are passed to the subprocess.

/NOTIFY
Specifies that a message is to be broadcast to SYS\$OUTPUT when the subprocess completes processing or aborts. The default behavior is that such a message is not sent to SYS\$OUTPUT.

/NOWAIT
Specifies that the system is not to wait until the subprocess is completed before allowing more commands to be specified. This qualifier allows you to specify new commands while the spawned subprocess is running. If you specify /NOWAIT, you should use /OUTPUT to direct the output of the subprocess to a file in order to prevent more than one process from simultaneously using your terminal.

The default behavior is that the system waits until the subprocess is completed before allowing more commands to be specified.

/OUTPUT=filespec
Specifies an output file to which the results of the SPAWN operation are written. You should specify an output other than SYS\$OUTPUT whenever you specify /NOWAIT to prevent output from the spawned subprocess from being displayed while you are specifying new commands. If you omit the /OUTPUT qualifier, output is written to the current SYS\$OUTPUT device.

SYSTEM DUMP ANALYZER

SPAWN

/PROCESS=process-name

Specifies the name of the subprocess to be created. The default name of the subprocess is *USERNAME_n*, where *USERNAME* is the user name of the parent process.

EXAMPLE

```
SDA> SPAWN
$ MAIL
.
.
.
$ DIR
.
.
.
$ LO
Process SYSTEM_1 logged out at 5-MAR-1989 15:42:23.59
SDA>
```

The previous example shows a general use of the SPAWN command to create a subprocess that issues DCL commands to invoke the Mail Utility and list the contents of a directory before logging off to return to the parent process executing SDA.

SYSTEM DUMP ANALYZER

VALIDATE QUEUE

VALIDATE QUEUE

Validates the integrity of the specified queue by checking the pointers in the queue.

FORMAT **VALIDATE QUEUE** *[address] [/SELF-RELATIVE]*

PARAMETER ***address***

Address of an element in a queue.

If you specify the period character (.) as the **address**, SDA uses the last evaluated expression as the queue element's address.

If you do not specify an **address**, the VALIDATE QUEUE command determines the address from the last issued VALIDATE QUEUE command in the current SDA session.

If you do not specify an **address**, and no queue has previously been specified, SDA displays the following error message:

%SDA-E-NOQUEUE, no queue has been specified for validation

QUALIFIER ***/SELF_RELATIVE***

Specifies that the selected queue is a self-relative queue.

DESCRIPTION

The VALIDATE QUEUE command uses the forward and backward pointers in each element of the queue to make sure that all such pointers are valid and that the integrity of the queue is intact. If the queue is intact, SDA displays the following message:

Queue is complete, total of *n* elements in the queue

In these messages, *n* represents the number of entries the VALIDATE QUEUE command has found in the queue.

If SDA discovers an error in the queue, it displays one of the following error messages:

Error in forward queue linkage at address nnnnnnnn after tracing x elements

Error comparing backward link to previous structure address (nnnnnnnn)

Error occurred in queue element at address oooooooo after tracing pppp elements

These messages can appear frequently when the VALIDATE QUEUE command is used within an SDA session that is analyzing a running system. In a running system, the composition of a queue can change while the command is tracing its links, thus producing an error message.

If there are no entries in the queue, SDA displays this message:

The queue is empty

SYSTEM DUMP ANALYZER

VALIDATE QUEUE

EXAMPLE

```
SDA> VALIDATE QUEUE/SELF_RELATIVE IOC$GL_IRPFL  
Queue is complete, total of 159 elements in the queue
```

The previous example validates the self-relative queue that is the IRP lookaside list. The validation is successful and determines that there are 159 IRPs in the list.

Index

A

Access violation • SDA-16, SDA-19
ACP (ancillary control process) • SDA-99
Addition operator (+) • SDA-12
Address
 examining • SDA-51
/ADDRESS qualifier • SDA-87, SDA-98,
 SDA-123
/ALL qualifier • SDA-51, SDA-108, SDA-111,
 SDA-115, SDA-126, SDA-143, SDA-157,
 SDA-161
ANALYZE command • SDA-32
 /CRASH_DUMP qualifier • SDA-35
 /RELEASE qualifier • SDA-36
 /SYMBOL qualifier • SDA-37
 /SYSTEM qualifier • SDA-38
ANALYZE/CRASH_DUMP command • SDA-6,
 SDA-32
ANALYZE/CRASH_DUMP/RELEASE command •
 SDA-3
ANALYZE/SYSTEM command • SDA-2, SDA-32
Analyzing a crash dump
 See also Crash dump, System failure
 privileges required • SDA-32
 requirements • SDA-6
Analyzing a running system • SDA-38
 See also System
 privileges required • SDA-8, SDA-32
AND operator (&) • SDA-12
AP symbol • SDA-13
AQB (ACP queue block) • SDA-99
Argument pointer (AP) • SDA-13
Arithmetic operator • SDA-12
Arithmetic shifting operator (@) • SDA-13
ASB (asynchronous save block) • SDA-76
ASTLVL register
 displaying • SDA-90
AST routines
 global symbols • SDA-60
ATTACH command • SDA-41

B

Backup Utility (BACKUP)
 copying system dump file • SDA-4
Bad page list
 displaying • SDA-115
/BAD qualifier • SDA-115
BDB (buffer descriptor block) • SDA-76
BDB summary page (BDBSUM) • SDA-76
Binary operator • SDA-12 to SDA-13
BLB (buffer lock block) • SDA-76
BMB summary page (BLBSUM) • SDA-76
Bugcheck
 fatal conditions • SDA-16 to SDA-20
 halt/restart • SDA-7
 identifying • SDA-21
Bugcheck code • SDA-15
Bugcheck handling routines
 global symbols • SDA-60
Bugcheck reason • SDA-94

C

Call frame
 displaying in SDA • SDA-79
 following a chain • SDA-79
Cancel I/O routine • SDA-99
CCB (channel control block)
 displaying in SDA • SDA-76
CDDB (class driver data block) • SDA-99
CDRP (class driver request packet) • SDA-87,
 SDA-148
CDT (connection descriptor table) • SDA-87,
 SDA-148
/CHANNEL qualifier • SDA-131
CLUB (cluster block) • SDA-83
CLUDCB (cluster quorum disk control block) •
 SDA-83
CLUFCB (cluster failover control block) • SDA-83
Cluster management code
 global symbols • SDA-60
CLUSTRLOA.STB • SDA-60
CLUSTRLOA symbol • SDA-13
Command • SDA-10 to SDA-14

Index

Condition handling routines
 global symbols • SDA-60

Condition value
 evaluating • SDA-48
 examining • SDA-51

/CONDITION_VALUE qualifier • SDA-48

Connection
 displaying SDA information • SDA-87,
 SDA-123, SDA-148

Connection manager
 displaying SDA information • SDA-82

/CONNECTION qualifier • SDA-148

Context
 SDA CPU • SDA-10
 SDA process • SDA-9

Control block
 formatting • SDA-56

Control region • SDA-14
 base register • SDA-14
 examining • SDA-52
 length register • SDA-14

Control region operator (H) • SDA-12

Control region page table
 displaying • SDA-127

COPY command • SDA-3, SDA-4, SDA-42

CPU context
 changing • SDA-68, SDA-74, SDA-89,
 SDA-93, SDA-126
 displaying • SDA-89

CPU ID (CPU identification number) • SDA-89

CPULOA.EXE
 global symbols • SDA-60

Crash dump
 See also System failure
 analysis • SDA-1 to SDA-165
 incomplete • SDA-7
 short • SDA-7

Crash dump file
 header • SDA-106

/CRASH_DUMP qualifier • SDA-6

CRB (channel request block) • SDA-99

CREATE command • SDA-2

CSB (cluster system block) • SDA-82, SDA-87

CSID (cluster system identification number) •
 SDA-82, SDA-144

/CSID qualifier • SDA-82

Current location symbol (.) • SDA-13

D

Data structure
 formatting • SDA-56
 stepping through a linked list • SDA-64

Data structures
 global symbols • SDA-60

DCLDEF.STB • SDA-60

DCL interpreter
 global symbols • SDA-60

DDB (device data block) • SDA-99

DDT (driver dispatch table) • SDA-99

Decimal value of an expression • SDA-48

DECnet data structures
 global symbols • SDA-60

DEFINE command • SDA-43

Device
 displaying SDA information • SDA-98

Device driver
 base address of driver prologue table (DPT) •
 SDA-13
 locating • SDA-13
 locating a failing instruction • SDA-24

Device driver routine
 address • SDA-99

Division operator (/) • SDA-13

DPT (driver prologue table) • SDA-99

DPT base address • SDA-24

DUMP
 subset • SDA-4

DUMPBUG parameter • SDA-2, SDA-28

Dump file
 analyzing • SDA-32
 copying the contents • SDA-42

DUMPSTYLE parameter • SDA-4

E

/ECHO qualifier • SDA-44

ERRORLOG.EXE • SDA-60

ERRORLOGBUFFERS parameter • SDA-3

Error logging routines
 global symbols • SDA-60

ESP symbol • SDA-13

EVALUATE command • SDA-48

EVALUATE/PSL command • SDA-22

Event flag routines
 global symbols • SDA-60

EVENT_FLAGS_AND_ASTS.EXE
 global symbols • SDA-60
 EXAMINE command • SDA-16, SDA-24, SDA-51
 EXAMINE/INSTRUCTION command • SDA-23
 Exception
 fatal • SDA-16
 identifying causes of • SDA-21
 EXCEPTION.EXE
 global symbols • SDA-60
 Exception handling routines
 global symbols • SDA-60
 Execute procedure • SDA-40
 Executive image
 contents • SDA-60, SDA-104
 /EXECUTIVE qualifier • SDA-59, SDA-157
 Executive stack pointer • SDA-13
 EXIT command • SDA-55
 Exiting from SDA • SDA-55
 Expression • SDA-11 to SDA-14
 evaluating • SDA-48

F

FAB (file attributes block) • SDA-76
 Fatal exception • SDA-16
 FATALEXCPT bugcheck • SDA-16
 FCB (file control block) • SDA-76
 Floating point emulation code
 base address • SDA-13
 FORMAT command • SDA-26, SDA-56, SDA-64
 FPEMUL symbol • SDA-13
 FP symbol • SDA-13
 Frame pointer • SDA-13
 Free page list
 displaying • SDA-115
 /FREE qualifier • SDA-115, SDA-118
 FWA (file work area) • SDA-77

G

GBD (global buffer descriptor) • SDA-77
 GBD (global buffer descriptor) summary page • SDA-77
 GBH (global buffer header) • SDA-77
 GBSB (global buffer synchronization block) • SDA-77
 Global page table
 displaying • SDA-111

/GLOBAL qualifier • SDA-111
 G operator • SDA-12
 G symbol • SDA-14

H

Header
 crash dump • SDA-106
 /HEADER qualifier • SDA-118
 HELP command • SDA-58
 recording output • SDA-71
 Hexadecimal value of an expression • SDA-48
 H operator • SDA-12
 H symbol • SDA-14

I

I/O database
 displaying SDA information • SDA-98
 global symbols • SDA-60
 ICCS register
 displaying • SDA-90
 IDB (interrupt dispatch block) • SDA-99
 IDX (index descriptor) • SDA-77
 IFAB (internal file access block) • SDA-77
 IFB (internal file access block) • SDA-77
 IFI (internal file identifier) • SDA-76
 /IF_STATE qualifier • SDA-44
 Image activator
 global symbols • SDA-60
 Image I/O structures • SDA-77
 /IMAGE qualifier • SDA-159
 IMAGE_MANAGEMENT.EXE
 global symbols • SDA-60
 IMGDEF.STB • SDA-60
 /INDEX qualifier • SDA-73, SDA-126
 /INPUT qualifier • SDA-162
 /INSTRUCTION qualifier • SDA-51
 Interlocked queue
 validating • SDA-164
 /INTERRUPT qualifier • SDA-157
 Interrupt stack
 displaying contents • SDA-157
 INVEXCEPTN bugcheck • SDA-16
 IO_ROUTINES.EXE
 global symbols • SDA-60

Index

IPL\$_ASTDEL
PGFIPLHI bugcheck • SDA-19
IRAB (internal record access block) • SDA-77
IRB (internal record access block) • SDA-77
IRP (I/O request packet) • SDA-99, SDA-118
IRP lookaside list
 displaying contents • SDA-118
/IRP qualifier • SDA-118

J

JFB (journaling file block) • SDA-77
JIB (job information block) • SDA-128

K

/KERNEL qualifier • SDA-157
Kernel stack
 displaying contents • SDA-157
Kernel stack pointer • SDA-14
Key
 defining for SDA • SDA-43
/KEY qualifier • SDA-44
KSP symbol • SDA-14

L

Linker map
 use in crash dump analysis • SDA-15
LKB (lock block) • SDA-108
LMF\$GROUP_TABLE.EXE
 global symbols • SDA-60
Location
 examining • SDA-51
 SDA default • SDA-51
 translating to VAX MACRO instruction • SDA-51
Lock
 displaying SDA information • SDA-143
/LOCKID qualifier • SDA-143
LOCKING.EXE • SDA-60
Lock management routines
 global symbols • SDA-60
Lock manager
 displaying SDA information • SDA-108
Lock mode • SDA-144

/LOCKS qualifier • SDA-127
Logical AND operator (&) • SDA-12
Logical NOT operator (#) • SDA-12
Logical operator • SDA-12
Logical OR operator (|) • SDA-12
Logical XOR operator (\) • SDA-13
LOGICAL_NAMES.EXE
 global symbols • SDA-60
Lookaside lists
 displaying contents • SDA-118
LRP (large request packet) • SDA-118
LRP lookaside list
 displaying • SDA-118
/LRP qualifier • SDA-118

M

MA780 multiport memory
 configuring a dump file for • SDA-3
Machine check code
 base address • SDA-14
MCHK symbol • SDA-14
Mechanism array • SDA-17, SDA-22
Memory
 examining • SDA-51
 formatting • SDA-56
Memory location
 decoding • SDA-53
 examining • SDA-52
Memory region
 examining • SDA-54
MESSAGE_ROUTINES.EXE
 global symbols • SDA-61
MicroVAX 2000
 inducing a crash • SDA-31
MicroVAX 3600 series
 inducing a crash • SDA-31
MicroVAX I
 inducing a crash • SDA-31
MicroVAX II
 inducing a crash • SDA-31
Modified page list
 displaying • SDA-115
/MODIFIED qualifier • SDA-115
Module
 finding a failing • SDA-24
MSCP server code
 base address • SDA-14
MSCP symbol • SDA-14

Multiplication operator (*) • SDA-12
 Multiprocessing
 global symbols • SDA-61
 Multiprocessor
 analyzing crash dumps • SDA-9
 displaying synchronization structures • SDA-150

N

NAM (name block) • SDA-77
 Negative operator (-) • SDA-12
 NETDEF.STB • SDA-60
 nnDRIVER symbol • SDA-13
 /NOLOGICAL_NAMES qualifier • SDA-162
 Nonpaged dynamic storage pool
 displaying contents • SDA-118
 /NONPAGED qualifier • SDA-118
 /NOSKIP qualifier • SDA-52
 /NOSUPPRESS qualifier • SDA-52
 /NOSYMBOLS qualifier • SDA-162
 /NOTIFY qualifier • SDA-162
 NOT operator (#) • SDA-12
 /NOWAIT qualifier • SDA-162
 NWA (network work area) • SDA-77

O

Operator • SDA-12
 precedence of • SDA-12, SDA-13
 OR operator (!) • SDA-12
 /OUTPUT qualifier • SDA-162

P

POBR register
 displaying • SDA-90
 POBR symbol • SDA-14
 POLR register
 displaying • SDA-90
 POLR symbol • SDA-14
 P0 page table
 displaying • SDA-127
 /P0 qualifier • SDA-127
 P0 region
 examining • SDA-52

P1BR register
 displaying • SDA-90
 P1BR symbol • SDA-14
 P1LR register
 displaying • SDA-90
 P1LR symbol • SDA-14
 P1 page table
 displaying • SDA-127
 /P1 qualifier • SDA-52, SDA-127
 P1 region
 examining • SDA-52
 Paged dynamic storage pool
 displaying contents • SDA-118
 /PAGED qualifier • SDA-118
 Page fault
 illegal • SDA-19
 Page table
 displaying • SDA-111, SDA-127
 Page table entry
 evaluating • SDA-48
 examining • SDA-52
 PAGE_MANAGEMENT.EXE
 global symbols • SDA-61
 /PAGE_TABLES qualifier • SDA-127
 Paging file
 See also SYS\$SYSTEM:PAGEFILE.SYS
 as system dump file • SDA-5
 Parenthesis
 as precedence operator • SDA-13
 /PARENT qualifier • SDA-41
 PB (path block) • SDA-99
 PCB (process control block) • SDA-160
 displaying • SDA-127
 hardware • SDA-129
 PCBB register
 displaying • SDA-90
 /PCB qualifier • SDA-127
 PC symbol • SDA-14
 PDT (port descriptor table) • SDA-123
 PFN (page frame number) database • SDA-111
 displaying • SDA-115
 PGFIPLHI bugcheck • SDA-19
 PHD (process header) • SDA-160
 displaying • SDA-127
 /PHD qualifier • SDA-127
 Port
 displaying SDA information • SDA-123
 Port driver
 displaying SDA information • SDA-82
 Positive operator (+) • SDA-12
 Precedence of operators • SDA-12

Index

Precedence operator • SDA-13
PRIMITIVE_IO.EXE
 global symbols • SDA-61
Process
 channel • SDA-126
 displaying SDA information • SDA-126,
 SDA-159
 examining a hung • SDA-8
 image • SDA-159
 listening • SDA-83
 lock • SDA-127
 scheduling state • SDA-129, SDA-159
 spawning a subprocess • SDA-162
Process context
 changing • SDA-68, SDA-73, SDA-93,
 SDA-126
Process control region • SDA-14
Process control region operator (H) • SDA-12
Process identification • SDA-126
Process index • SDA-126
Process name • SDA-126
Processor context
 changing • SDA-68, SDA-74, SDA-89,
 SDA-93, SDA-126
Processor-specific loadable code
 base address • SDA-14
Processor status longword
 See PSL
Processor type
 displaying • SDA-90
Process-permanent I/O structures • SDA-77
 /PROCESS qualifier • SDA-163
PROCESS_MANAGEMENT.EXE
 global symbols • SDA-61
 /PROCESS_SECTION_TABLE qualifier • SDA-127
Program counter • SDA-14
Program counter (PC)
 in a crash dump • SDA-15
Program region
 base register • SDA-14
 examining • SDA-52
 length register • SDA-14
Program region page table
 displaying • SDA-127
PSL • SDA-14
PSL (processor status longword)
 evaluating • SDA-22, SDA-48
 examining • SDA-52
 /PSL qualifier • SDA-52
PSL symbol • SDA-14

PST (process section table)
 displaying • SDA-127
 /PTE qualifier • SDA-48, SDA-52

Q

Queue
 stepping through • SDA-64
 validating • SDA-164
Quorum • SDA-82

R

RAB (record attributes block) • SDA-77
Radix
 default • SDA-12
Radix operator • SDA-12
RDT (response descriptor table) • SDA-148
READ command • SDA-59
 SYS\$DISK • SDA-60
READ/EXECUTIVE command • SDA-16
Recovery unit system services
 global symbols • SDA-61
RECOVERY_UNIT_SERVICES.EXE
 global symbols • SDA-61
Register
 displaying • SDA-89, SDA-127
 general • SDA-14
 /REGISTERS qualifier • SDA-127
 /RELEASE qualifier • SDA-3
 /RELOCATE qualifier • SDA-59
REPEAT command • SDA-64
Report system event
 global symbols • SDA-61
Resource
 displaying SDA information • SDA-143
RLB (record lock block) • SDA-77
RMS.EXE • SDA-61
RMSDEF.STB • SDA-60
RMS image
 base address • SDA-14
 /RMS qualifier • SDA-127
RMS symbol • SDA-14
RSB (resource block) • SDA-109, SDA-143
RSPID (response ID)
 displaying SDA information • SDA-148
RUB (recovery unit block) • SDA-77

RUFB (recovery unit file block) • SDA-77
 RUBS (recovery unit stream block) • SDA-77

S

S0 region
 examining • SDA-52
 SAVEDUMP parameter • SDA-3, SDA-28
 SB (system block) • SDA-83, SDA-99
 SBR register
 displaying • SDA-90
 SCBB register
 displaying • SDA-90
 Scheduler
 global symbols • SDA-61
 SCS (system communications services)
 base address • SDA-14
 displaying SDA information • SDA-82,
 SDA-83, SDA-87, SDA-123, SDA-148
 global symbols • SDA-60
 SCSDEF.STB • SDA-60
 SCSLOA symbol • SDA-14
 /SCS qualifier • SDA-82
 SDA\$INIT logical name • SDA-8
 SDA current CPU • SDA-10, SDA-68, SDA-74,
 SDA-89, SDA-93, SDA-126, SDA-157
 SDA current process • SDA-9, SDA-10, SDA-68,
 SDA-73, SDA-93, SDA-126, SDA-157
 SDA symbol table • SDA-13
 building • SDA-7
 expanding • SDA-8
 SEARCH command • SDA-66
 SECURITY.EXE
 global symbols • SDA-61
 Self relative queue
 validating • SDA-164
 /SELF_RELATIVE qualifier • SDA-164
 SET CPU command • SDA-10, SDA-68
 analyzing a running system • SDA-9
 SET LOG command • SDA-71
 compared with SET OUTPUT command •
 SDA-71
 SET NOLOG command • SDA-71
 SET OUTPUT command • SDA-72
 compared with SET LOG command • SDA-71
 SET PROCESS command • SDA-9, SDA-73
 SET RMS command • SDA-76
 /SET_STATE qualifier • SDA-45
 SFSB (shared file synchronization block) • SDA-77
 Shadow set
 displaying SDA information • SDA-99
 Shifting operator (@) • SDA-13
 SHOW CALL_FRAME command • SDA-65,
 SDA-79
 SHOW CLUSTER command • SDA-82
 SHOW CLUSTER/SCS command • SDA-123
 SHOW CONNECTIONS command • SDA-87
 SHOW CPU command • SDA-10, SDA-68,
 SDA-89
 analyzing a running system • SDA-9
 SHOW CRASH command • SDA-10, SDA-15,
 SDA-16, SDA-68, SDA-93
 analyzing a running system • SDA-9
 SHOW DEVICE command • SDA-15, SDA-24,
 SDA-98
 SHOW EXECUTIVE command • SDA-15,
 SDA-104
 SHOW HEADER command • SDA-106
 SHOW LOCK command • SDA-108
 SHOW MEMORY command • SDA-3
 SHOW PAGE_TABLE command • SDA-23,
 SDA-111
 SHOW PFN_DATA command • SDA-115
 SHOW POOL command • SDA-118
 SHOW PORTS command • SDA-123
 SHOW PROCESS/ALL command • SDA-128
 SHOW PROCESS command • SDA-74, SDA-126
 SHOW PROCESS/LOCKS command • SDA-108
 SHOW PROCESS/RMS command • SDA-147
 selecting display options • SDA-76
 SHOW RESOURCE command • SDA-108,
 SDA-143
 SHOW RMS command • SDA-147
 SHOW RSPID command • SDA-148
 SHOW SPINLOCKS command • SDA-151
 SHOW STACK command • SDA-21, SDA-157
 SHOW SUMMARY command • SDA-126,
 SDA-159
 SHOW SYMBOL command • SDA-161
 Shutdown
 operator-requested • SDA-5
 SID register
 displaying • SDA-90
 Signal array • SDA-18
 SISR register
 displaying • SDA-90
 Site-specific startup procedure
 See SYS\$MANAGER:SYSTARTUP.COM
 SLR register
 displaying • SDA-90
 SPAWN command • SDA-162

Index

- Spin lock
 - displaying SDA information • SDA-150
 - owned • SDA-90
- SPR (Software Performance Report) • SDA-2, SDA-28
- SP symbol • SDA-14
- SRP (small request packet) • SDA-119
- SRP lookaside list
 - displaying contents • SDA-119
- /SRP qualifier • SDA-119
- SSP symbol • SDA-14
- SSRVEXCEPT bugcheck • SDA-16
- Stack
 - displaying contents • SDA-157
- Stack frame
 - displaying in SDA • SDA-79
 - following a chain • SDA-79
- Stack pointer
- Start I/O routine • SDA-99
- Subprocess • SDA-162
- Subtraction operator (-) • SDA-12
- /SUMMARY qualifier • SDA-119
- /SUPERVISOR qualifier • SDA-157
- Supervisor stack
 - displaying contents • SDA-157
- Supervisor stack pointer • SDA-14
- Swapper
 - global symbols • SDA-61
- Symbol • SDA-13 to SDA-14, SDA-23
 - defining for SDA • SDA-43
 - displaying • SDA-14
 - evaluating • SDA-161
 - listing • SDA-161
 - loading into the SDA symbol table • SDA-59
 - name • SDA-13, SDA-43
 - representing executive modules • SDA-104
 - user-defined • SDA-43
- /SYMBOLS qualifier for EVALUATE • SDA-48
- Symbol table
 - See also SDA symbol table, System symbol table
 - specifying an alternate SDA • SDA-37
- Symbol table file
 - reading into SDA symbol table • SDA-59
- SYSDISK as SDA output • SDA-72
- SYSDISK global read • SDA-60
- SYSDMANAGER:SYSTARTUP.COM
 - invoking SDA • SDA-5
 - producing an SDA listing • SDA-5
 - releasing page file blocks • SDA-3
- SYSDSYSTEM:OPCCRASH.COM
 - involvement in writing crash dump • SDA-5
- SYSDSYSTEM:PAGEFILE.SYS • SDA-5, SDA-28
 - See also System dump file as dump file • SDA-3
 - releasing blocks containing a crash dump • SDA-36
- SYSDSYSTEM:REQSYSDEF.STB • SDA-6, SDA-7
- SYSDSYSTEM:SHUTDOWN.COM
 - involvement in writing crash dump • SDA-5
- SYSDSYSTEM:SYS.EXE • SDA-59
 - contents • SDA-60, SDA-104
- SYSDSYSTEM:SYS.STB • SDA-6, SDA-7, SDA-9, SDA-15
- SYSDSYSTEM:SYSDEF.STB • SDA-8
- SYSDSYSTEM:SYSDUMP.DMP • SDA-28
 - See also System dump file protection • SDA-5
 - size of • SDA-3
- SYSAP (system application) • SDA-148
- SYSDEVICE.EXE
 - global symbols • SDA-61
- SYSGETSYI.EXE
 - global symbols • SDA-61
- SYSLICENSE.EXE
 - global symbols • SDA-61
- SYSLOA symbol • SDA-14
- SYSMSG.EXE
 - global symbols • SDA-61
- System
 - analyzing a running • SDA-2, SDA-8 to SDA-9, SDA-32
 - investigating performance problems • SDA-8
- System dump file • SDA-2 to SDA-3
 - copying • SDA-4
 - header • SDA-5
 - mapping physical memory to • SDA-7
 - requirements for analysis • SDA-6
 - saving • SDA-4
 - size • SDA-3
- System failure
 - analyzing • SDA-15 to SDA-28
 - causing • SDA-28 to SDA-31
 - diagnosing from PC contents • SDA-15
 - example • SDA-21 to SDA-28
 - summary • SDA-93
- System hang • SDA-28
- System image
 - contents • SDA-60, SDA-104

System management
 creating a crash dump file • SDA-2

System map • SDA-15

System message routines
 global symbols • SDA-61

System page table (SPT)
 displaying • SDA-23, SDA-111
 in system dump file • SDA-2, SDA-7

System paging file
 as dump file • SDA-3
 releasing blocks containing a crash dump • SDA-36

System PCB (process control block)
 displaying • SDA-128

System process • SDA-73
 /SYSTEM qualifier • SDA-52, SDA-73, SDA-111, SDA-115, SDA-128

System region
 examining • SDA-52

System space
 base address • SDA-14

System space operator (G) • SDA-12

System symbol table • SDA-6, SDA-13

System time quadword
 examining • SDA-52

SYSTEM_PRIMITIVES.EXE
 global symbols • SDA-61

SYSTEM_SYNCHRONIZATION.EXE
 global symbols • SDA-61

T

Terminal key
 defining for SDA • SDA-43
 /TERMINATE qualifier • SDA-45
 /TIME qualifier • SDA-52
 /TYPE qualifier • SDA-56, SDA-119

U

UCB (unit control block) • SDA-87

Unary operator • SDA-12
 /USER qualifier • SDA-157

User stack
 displaying contents • SDA-157

User stack pointer • SDA-14

USP symbol • SDA-14

V

VALIDATE QUEUE command • SDA-164

VAX-11/725
 inducing a crash • SDA-31

VAX-11/730
 inducing a crash • SDA-31

VAX-11/750
 inducing a crash • SDA-31

VAX-11/780
 inducing a crash • SDA-30

VAX-11/785
 inducing a crash • SDA-30

VAX 6200 series
 inducing a crash • SDA-29

VAX 8200
 inducing a crash • SDA-29

VAX 8230
 inducing a crash • SDA-29

VAX 8250
 inducing a crash • SDA-29

VAX 8300
 inducing a crash • SDA-29

VAX 8350
 inducing a crash • SDA-29

VAX 8530
 inducing a crash • SDA-29

VAX 8550
 inducing a crash • SDA-29

VAX 8600
 inducing a crash • SDA-30

VAX 8650
 inducing a crash • SDA-30

VAX 8700
 inducing a crash • SDA-29

VAX 8800
 inducing a crash • SDA-29

VAX 8830
 inducing a crash • SDA-29

VAX 8850
 inducing a crash • SDA-29

VAXcluster
 base address of loadable code • SDA-13
 displaying SDA information • SDA-82

VAX MACRO instruction
 formatting memory with SDA • SDA-51

VAXstation II
 inducing a crash • SDA-31

VCB (volume control block) • SDA-99

Virtual address operator (@) • SDA-12

Index

Virtual address space
 sufficient for system dump analysis • SDA-6
VIRTUALPAGECNT parameter • SDA-6
VMS executive image
 global symbols • SDA-59
VMS Record Management Services (RMS)
 data structures shown by SDA • SDA-76
 displaying data structures • SDA-127,
 SDA-147
 global symbols • SDA-60, SDA-61
VMS system image
 global symbols • SDA-59
Vote • SDA-82

W

WCB (window control block) • SDA-77
Working set list
 displaying • SDA-128
/WORKING_SET qualifier • SDA-128
WORKING_SET_MANAGEMENT.EXE
 global symbols • SDA-61

X

XAB (extended attribute block) • SDA-77
XOR operator (\) • SDA-13
XQP (extended QIO processor) • SDA-99

Reader's Comments

VMS System Dump
Analyzer Utility Manual
AA-LA87A-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Completeness (enough information)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Clarity (easy to understand)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Organization (structure of subject matter)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Figures (useful)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Examples (useful)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Index (ability to find topic)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Page layout (easy to find information)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

I would like to see more/less _____

What I like best about this manual is _____

What I like least about this manual is _____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____
Company _____ Date _____
Mailing Address _____
_____ Phone _____

-- Do Not Tear - Fold Here and Tape --

digital™



No Postage
Necessary
if Mailed
in the
United States



BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987



-- Do Not Tear - Fold Here --