

DSM-11 User's Guide

Order Number AA-H799B-TC

March, 1984

This document describes the use of the DSM-11 system.

This is a revised manual.

Operating System:

DSM-11 Version 3

Software:

DSM-11 Version 3

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center. Outside the United States, orders should be directed to the nearest DIGITAL Field Sales Office or representative.

Northeast/Mid-Atlantic Region

Digital Equipment Corporation
PO Box CS2008
Nashua, New Hampshire 03061
Telephone:(603)884-6660

Central Region

Digital Equipment Corporation
Accessories and Supplies Center
1050 East Remington Road
Schaumburg, Illinois 60195
Telephone:(312)640-5612

Western Region

Digital Equipment Corporation
Accessories and Supplies Center
632 Caribbean Drive
Sunnyvale, California 94086
Telephone:(408)734-4915

First Printing, October, 1980
Revised, March, 1984


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 only 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 or its affiliated companies.

Copyright © 1980, 1983 Digital Equipment Corporation.
All Rights Reserved.

The following are trademarks of Digital Equipment Corporation

DEC	DIBOL	RSTS
DECmate	DSM-11	RSX
	MASSBUS	UNIBUS
DECsystem-10	PDP	VAX
DECSYSTEM-20	P/OS	VMS
DECUS	Professional	VT
DECwriter	Rainbow	Work Processor

CONTENTS

Part 1: Installing DSM-11

CHAPTER 1 INSTALLING DSM-11

1.1	THE DSM-11 DISTRIBUTION KIT.....	1-2
1.2	THE DSM-11 BASELINE SYSTEM.....	1-2
1.3	SYSTEM OPERATION INFORMATION SOURCES.....	1-3
1.4	INSTALLATION PROCEDURE OVERVIEW.....	1-4
1.5	PREINSTALLATION CHECKLIST.....	1-5
1.6	POWER ON FOR THE CONSOLE TERMINAL.....	1-9
1.7	POWER ON FOR THE PDP-11 PROCESSOR.....	1-10
	1.7.1 Power On For The PDP-11/23-PLUS.....	1-11
	1.7.2 Power On For The PDP-11/24.....	1-12
	1.7.3 Power On For The PDP-11/44.....	1-13
1.8	MOUNTING THE DISTRIBUTION MEDIUM.....	1-14
1.9	MOUNTING THE SYSTEM OR BACKUP DISTRIBUTION DISK...	1-14
1.10	START THE SYSTEM.....	1-14
	1.10.1 Starting The PDP-11/23-PLUS.....	1-16
	1.10.2 Starting The PDP-11/24.....	1-17
	1.10.3 Starting The PDP-11/44.....	1-18
1.11	MICRO/PDP-11 INSTALLATION PROCEDURE.....	1-19
1.12	INSTALLATION DIALOGUE.....	1-21

Part 2: Introduction to DSM-11

CHAPTER 2 OVERVIEW OF DSM-11

2.1	SYSTEM OVERVIEW	2-1
2.1.1	The Language	2-1
2.1.2	System Capacity	2-2
2.1.3	Data Access	2-2
2.1.4	System Access	2-3
2.2	SYSTEM SOFTWARE	2-3
2.2.1	Operating System	2-4
2.2.2	System Utility Routines	2-5
2.2.3	Library Utility Routines	2-5
2.2.4	Library Utility Globals	2-6
2.2.5	System-Configuration Global	2-6
2.3	SYSTEM HARDWARE	2-6

CHAPTER 3 ACCESSING DSM-11

3.1	USING THE TERMINAL	3-1
3.1.1	Function Keys	3-3
3.1.2	Control Characters	3-5
3.2	LOGGING IN DSM-11	3-7
3.2.1	The User Class Identifier Code	3-8
3.2.2	The Programmer Access Code	3-8
3.2.3	Partition Size In Bytes	3-8
3.2.4	DSM-11 Modes	3-8
3.2.5	Logging In DSM-11 Application Mode	3-9
3.2.6	Logging In DSM-11 In Programmer Mode	3-9
3.2.7	Logging In At A Tied Terminal	3-10
3.2.8	Logging In Through A Modem With Autobauding	3-10
3.3	LOGGING OUT OF DSM-11	3-10

Part 3: Programming and Using the DSM-11 System

CHAPTER 4 USING DSM-11

4.1	ENTERING DSM-11 COMMANDS AND ROUTINES	4-1
4.1.1	Abbreviating DSM-11 Commands	4-2
4.1.2	Using Uppercase And Lowercase Characters	4-2
4.1.3	DSM-11 Line Structure	4-3

4.2	CREATING ROUTINES.....	4-4
4.2.1	Direct Mode	4-5
4.2.2	Entering Lines In The Routine Buffer	4-5
4.2.3	Using The ZPRINT And ZREMOVE Commands	4-6
4.3	SAVING AND LOADING ROUTINES	4-6
4.3.1	Saving A Routine In A Routine Directory	4-7
4.3.2	Loading A Routine From The Routine Directory	4-8
4.4	DELETING AND RENAMING ROUTINES	4-8
4.4.1	Deleting A Stored Routine	4-8
4.4.2	Renaming Routines.....	4-9
4.5	USING SEQUENTIAL FILES TO STORE ROUTINES.....	4-9
4.5.1	Writing A Routine Onto A Sequential File	4-10
4.5.2	Loading A Routine From A Sequential File.....	4-10
4.6	USING EDITORS.....	4-11
4.6.1	Programmer Mode Editing	4-11
4.6.2	DSM-11 Editor	4-12
4.6.3	EDI Editor (%EDI).....	4-12
4.6.4	Global Editor (%GEDIT)	4-13
4.7	STARTING AND STOPPING ROUTINES	4-13
4.7.1	Executing The Routine In Your Routine Buffer.....	4-14
4.7.2	Executing A Routine From The Routine Directory	4-14
4.7.3	Conditions For Execution To Stop.....	4-14
4.7.4	Programmer Mode Interrupt By <code>CTRL/C</code> And <code>CTRL/Z</code>	4-15
4.7.5	Recognition Of The Application Interrupt Key (<code>CTRL/C</code>)	4-16
4.7.6	Changing The Application Interrupt Key.....	4-17
4.8	USING THE MUMPS DEBUGGER.....	4-17
4.8.1	Enabling And Disabling Debugging.....	4-18
4.8.2	Breakpoints.....	4-19
4.8.2.1	Setting Breakpoints With \$ZBREAK	4-19
4.8.2.2	Setting Breakpoints With The BREAK Command.....	4-20
4.8.2.3	Breakpoint Actions.....	4-20
4.8.2.4	Examining Breakpoints.....	4-21
4.8.2.5	Clearing Breakpoints.....	4-21
4.8.3	Continuing Execution After A Breakpoint	4-21
4.9	USING DSM-11 DIRECTORIES	4-22
4.9.1	DSM-11 Routine Directories	4-23
4.9.2	DSM-11 Global Directories.....	4-23
4.9.3	Global File Protection	4-24
4.10	ERROR PROCESSING	4-25
4.10.1	Default DSM-11 Error Processing	4-25
4.10.2	Writing Error-Processing Routines	4-25
4.10.3	DSM-11 Error-Trapping Routines	4-27
4.10.4	Error Processing Within Nested Contexts	4-27

4.10.5	Exiting From An Error Handler	4-28
4.10.6	BREAK 2 Control Of Error Processing	4-29

CHAPTER 5 USING SYSTEM DEVICES

5.1	INTRODUCTION.....	5-1
5.2	I/O DEVICE SPECIFIERS	5-2
5.3	ASSIGNING I/O DEVICES	5-3
5.4	I/O COMMANDS	5-4
5.5	OUTPUT FORMATTING.....	5-5
5.5.1	Formatting Characters.....	5-6
5.5.2	The \$X And \$Y Special Variables.....	5-7
5.6	I/O ERROR PROCESSING	5-8

CHAPTER 6 I/O DEVICE CHARACTERISTICS

6.1	TERMINALS.....	6-1
6.1.1	Terminal Device Numbers	6-2
6.1.2	Terminal Commands.....	6-2
6.1.3	Terminal Error Conditions	6-16
6.2	LINE PRINTER.....	6-16
6.2.1	Line Printer Commands.....	6-16
6.2.2	Line Printer Functions And Characters.....	6-17
6.2.3	Line Printer Error Conditions.....	6-17
6.3	MAGNETIC TAPE.....	6-17
6.3.1	Magnetic Tape Device Numbers	6-18
6.3.2	Magnetic Tape Commands	6-18
6.3.3	Magnetic Tape Operations.....	6-24
6.3.4	Tape Labels And Multiple File Structures.....	6-27
6.3.5	Data Formats	6-28
6.3.6	Continuous Mode.....	6-30
6.3.7	Magnetic Tape Error Conditions	6-31
6.4	SEQUENTIAL DISK PROCESSOR	6-33
6.4.1	SDP Device Numbers	6-34
6.4.2	SDP Commands.....	6-34
6.4.3	SDP Special Characteristics And Functions	6-36
6.4.4	SDP Error Conditions	6-36
6.4.5	Examples Of SDP OPEN And USE	6-37
6.4.6	Using SDP To Read Disk Journaling Space.....	6-37
6.5	IN-MEMORY JOB COMMUNICATION	6-38

6.5.1	JOB COM Device Numbers	6-38
6.5.2	JOB COM Commands	6-38
6.6	DMC11	6-40
6.6.1	DMC11 Device Numbers	6-40
6.6.2	DMC11 Commands	6-40
6.6.3	DMC11 Message Mode	6-41
6.6.4	DMC11 Block Mode	6-41
6.6.5	DMC11 Buffer Mode	6-43
6.6.6	Using DMC On A Switched Telephone Network	6-44
6.6.7	DMC11 Error Conditions	6-45
6.7	SPOOLING DEVICE	6-46
6.7.1	Spooling Device Numbers	6-47
6.7.2	Spooling Device Commands	6-48
6.7.3	Transparent Spooling	6-50
6.7.4	Explicit Spooling	6-51
6.7.5	Spooling Device Error Conditions	6-52
6.7.6	Spool File Structure	6-52
6.8	ROUTINE INTERLOCKS	6-57
6.8.1	Routine Interlock Device Numbers	6-57
6.8.2	Routine Interlock Commands	6-57
6.9	VIEW DEVICE	6-57
6.10	RX02 DISKETTE DRIVE	6-61
6.10.1	RX02 Commands	6-61
6.10.2	Mixed Mode	6-63
6.10.3	Read Mode	6-63
6.10.4	Write Mode	6-63
6.10.5	Control Mode	6-64
6.10.6	View Buffer Mode	6-64
6.10.7	Examples Of The OPEN And USE Commands	6-64
6.10.8	Error Codes	6-65
6.11	TU58 MAGNETIC TAPE	6-66
6.11.1	TU58 Commands	6-66
6.11.2	Mixed Mode	6-68
6.11.3	Read Mode	6-68
6.11.4	Write Mode	6-69
6.11.5	Control Mode	6-69
6.11.6	View Buffer Mode	6-69
6.11.7	Examples Of The OPEN And USE Commands	6-69
6.11.8	TU58 Error Codes	6-70

CHAPTER 7 USING THE DSM-11 LIBRARY UTILITIES

7.1	INTRODUCTION TO THE LIBRARY UTILITIES	7-1
7.1.1	Running The Library Utilities	7-2
7.1.2	Library Utility Conventions	7-3

7.2	GLOBAL UTILITIES.....	7-3
7.2.1	Global Management (^%GLOMAN).....	7-4
7.2.2	Global Copy (^%GC)	7-4
7.2.3	Global Directory (^%GD).....	7-4
7.2.4	Global Efficiency (^%GE)	7-4
7.2.5	Global List (^%G)	7-5
7.2.6	Global Restore (^%GTI)	7-5
7.2.7	Global Save (^%GTO)	7-5
7.2.8	Global Selector (^%GSEL)	7-5
7.2.9	Global Subscript Filter (^%FIND).....	7-5
7.2.10	Extended Global Directory (^%EGD).....	7-6
7.3	ROUTINE UTILITIES	7-6
7.3.1	Routine Compare (^%RCMP)	7-6
7.3.2	Routine Copy (^%RCOPY)	7-7
7.3.3	Routine Directory (^%RD)	7-7
7.3.4	First Line List (^%FL).....	7-7
7.3.5	Routine Restore (^%RR)	7-7
7.3.6	Routine Save (^%RS)	7-7
7.3.7	Routine Search (^%RSE).....	7-8
7.3.8	Routine Summaries (^%SUM)	7-8
7.3.9	Routine Change Every (^%RCE).....	7-8
7.3.10	Routine Selector (^%RSEL).....	7-8
7.3.11	Routine Cross Reference (^%CRF)	7-8
7.3.12	Extended Routine Directory (^%ERD).....	7-8
7.4	MISCELLANEOUS UTILITIES.....	7-9
7.4.1	Decimal/Octal Conversion (^%DOC).....	7-9
7.4.2	List ^MENU Global (^%MENLIS)	7-10
7.4.3	Load DSM-11 Editor (LOAD^%ED)	7-10
7.4.4	Date (^%D)	7-10
7.4.5	Time (^%T)	7-10
7.4.6	Version 1 Global Restore (^%GR).....	7-11
7.4.7	Version 1 Global Save (^%GS)	7-11
7.4.8	I/O Device Selector (^%IOS)	7-11
7.4.9	Header Formatter (^%HDR).....	7-11
7.4.10	Menu Manager (^%MENU).....	7-12
7.4.11	Modem Autodialer (^%DIAL)	7-12
7.5	OTHER MISCELLANEOUS UTILITIES.....	7-12
7.5.1	Date And Time (^%H).....	7-12
7.5.2	In-line Decimal To Octal Conversion (^%DO)	7-13
7.5.3	In-Line Octal To Decimal Conversion (^%OD).....	7-13
7.5.4	Cursor Control (^%CURSOR).....	7-14
7.5.5	Get UCI (^%GUCI)	7-14
7.5.6	Magnetic Tape Status Check (^%MTCHK)	7-14

CHAPTER 8 GLOBALS

8.1	GLOBAL CONCEPTS.....	8-1
8.1.1	Global Variables	8-2
8.1.2	Sparse Arrays	8-3
8.1.3	Global Data Structures	8-4
8.1.4	Programming Strategy.....	8-5
8.2	ACCESSING GLOBALS OUTSIDE YOUR UCI WITH EXTENDED GLOBAL REFERENCES	8-6
8.2.1	UCI-to-UCI Global Access Within The Same Volume Set	8-7
8.2.2	UCI-to-UCI Global Access Across Volume Sets.....	8-8
8.2.3	Distributed Data Processing.....	8-8
8.2.4	Accessing A Global In Another UCI Through The UCI Translation Table	8-9

CHAPTER 9 OPTIMIZING DSM-11 APPLICATIONS

9.1	OPTIMIZING ROUTINE INTERPRETATION	9-1
9.1.1	Using A Direct, Top-Down Routine Execution Path.....	9-2
9.1.2	Optimizing Routines That Call Subroutines With Line Labels ...	9-5
9.1.3	Using Short Line Labels And Variable Names.....	9-6
9.1.4	Abbreviating DSM-11 Commands.....	9-6
9.1.5	Optimizing Pattern Matching	9-7
9.1.6	Optimizing Xecute And Indirection Usage	9-8
9.2	STANDARDIZING APPLICATION STRUCTURE	9-9
9.2.1	Using Programming Conventions.....	9-9
9.2.2	Programming Conventions	9-10
9.2.3	Variable Naming Conventions.....	9-11
9.2.4	Terminal Conventions	9-11

Part 4: Managing the DSM-11 System

CHAPTER 10 GENERATING DSM-11

10.1	THE SYSGEN CONCEPT	10-1
10.2	SYSTEM DEFINITION UTILITIES	10-2
10.3	THE SYSGEN PROCEDURE	10-3
10.4	THE SYSGEN SESSION	10-3
10.4.1	SYSGEN Configuration	10-4
10.4.2	Disk Information	10-4

10.4.3	System Devices	10-5
10.4.4	DMC11s Configuration	10-5
10.4.5	Software Configuration	10-6
10.4.6	Assign Device Numbers	10-7
10.4.7	Software Options	10-7
10.4.8	Memory Buffer Allocation	10-10
10.4.9	System Data Structures	10-11
10.4.10	Job Partition Definition	10-12
10.4.11	Data Base Parameters (^MBP)	10-12
10.4.12	Basic System Parameters (^MBP)	10-13
10.5	EXAMPLE SYSGEN	10-16
10.6	EXAMPLE AUTOCONFIGURATION	10-27
10.7	MODIFY SYSTEM PARAMETERS	10-29
10.7.1	Device Characteristics (^MUX)	10-29
10.7.2	Add UCIs (^UCIADD)	10-33
10.7.3	UCI Edit (^UCIEDIT)	10-34
10.7.4	UCI List (^UCILI)	10-34
10.7.5	Magnetic Tape Default Mode (^MMD)	10-34
10.7.6	Routine Mapping (^RMAP)	10-35
10.7.6.1	Disable/Enable Routine Mapping Utility (^RMDIS)	10-37
10.7.6.2	Display Loaded Routine Sets Utility (^RMSHO)	10-37
10.7.6.3	Routine Set Loading Utility (^RMLOAD)	10-38
10.7.6.4	Routine Set Management Utility (^RMBLD)	10-38
10.7.7	Tied Terminal Table Generation Questions (^TTTG)	10-39
10.7.8	UCI Translation Table (^TRANTAB)	10-41

CHAPTER 11 OPERATING AND MAINTAINING DSM-11

11.1	SYSTEM START-UP (^STU AND ^STUBLD)	11-1
11.2	PRESERVING SYSTEM INTEGRITY	11-3
11.2.1	Concept Of Physical Backup (^BACKUP And ^REST)	11-4
11.2.2	Concept Of Logical Backup	11-4
11.2.3	Concept Of Journaling	11-4
11.3	APPLYING PATCHES TO DSM-11	11-4
11.3.1	Utility Patching	11-5
11.3.2	System Patching (^AUPAT)	11-5
11.4	SYSTEM SHUTDOWN (^SSD)	11-7

CHAPTER 12 USING THE DSM-11 SYSTEM UTILITIES

12.1	INTRODUCTION TO THE SYSTEM UTILITIES	12-1
12.1.1	Running The System Utilities	12-2
12.1.2	Stopping The System Utilities	12-3
12.2	CARETAKER UTILITIES (^CARE)	12-3
12.2.1	Start System Caretaker.....	12-4
12.2.2	Stop System Caretaker	12-4
12.2.3	Print Hardware Error Log (^KTR).....	12-4
12.2.4	Erase Hardware Error Log	12-4
12.2.5	Change Error Printer.....	12-5
12.2.6	Print Disk Error Summary.....	12-5
12.2.7	Software Error Log (%ER).....	12-5
12.3	DISK MAINTENANCE UTILITIES	12-5
12.3.1	Disk Backup (^BACKUP)	12-6
12.3.2	Disk Bad Blocks (^BBTAB)	12-6
12.3.3	Disk Dismount (^DISMOUNT)	12-6
12.3.4	Disk Format/Initialize (^DISKPREP)	12-7
12.3.5	Disk Label (^LABEL)	12-7
12.3.6	Disk Mount (^MOUNT)	12-7
12.3.7	Disk Restore (^REST)	12-7
12.4	SEQUENTIAL DISK PROCESSOR UTILITY (^SDP).....	12-7
12.5	SPOOL UTILITIES	12-8
12.5.1	Allocate Spool Space (^SPLALL)	12-8
12.5.2	Deallocate Spool Space (^SPLREM).....	12-8
12.5.3	Display Spool File Structure (^SPLSTR)	12-9
12.5.4	Initialize Spool Space (^SPLINI)	12-9
12.5.5	Start Despooler Lister (DSON^SPL).....	12-9
12.5.6	Start Spooler (SPON^SPL).....	12-9
12.5.7	Stop Despooler (DSOF^SPL)	12-9
12.5.8	Stop Spooler (SPOF^SPL).....	12-9
12.6	SYSTEM REPORTS	12-10
12.6.1	Disk Reports	12-10
12.6.1.1	Block Dump (^BLDMP)	12-10
12.6.1.2	Disk Block Tally (^DBT)	12-10
12.6.1.3	Fast Disk Block Tally (^FASTDBT).....	12-10
12.6.1.4	Integrity Check (^IC)	12-10
12.6.2	Performance Statistics (^RTHIST)	12-11
12.6.3	System Status	12-11
12.6.3.1	Job Monitor (^%JOB).....	12-11
12.6.3.2	Switch Register (^SWREG).....	12-11
12.6.3.3	System Status (^STA).....	12-11
12.6.3.4	Terminal DDB (^DDR).....	12-11
12.6.3.5	Line Map Report (^LMAP).....	12-11

12.6.3.6	Active Job Report (^ACTJOB)	12-11
12.6.4	Table Utilities	12-12
12.6.4.1	Device Table (^DEVTAB)	12-12
12.6.4.2	Job Table (^JOBTAB).....	12-12
12.6.4.3	Locked Variables (^LOCKTAB)	12-12
12.6.4.4	Partition Table (^PARTAB)	12-12
12.6.4.5	System Table (^SYSTAB).....	12-12
12.6.4.6	UCI Table (^UCITAB)	12-13
12.6.4.7	Partition Vector (^PARVEC).....	12-13
12.6.4.8	Volume Set Table (^%STRTAB)	12-13
12.7	MISCELLANEOUS ROUTINES.....	12-13
12.7.1	Autopatch (^AUPAT)	12-14
12.7.2	Broadcast (^BCS)	12-14
12.7.3	Data Base Repair (^FIX).....	12-14
12.7.4	Peek (^PEEK).....	12-14
12.7.5	Restore Jobs/Devices (^RJD).....	12-14
12.7.6	Set Date (^DAT)	12-14
12.7.7	Set Time (^TIM)	12-15
12.7.8	System Shutdown (^SSD)	12-15
12.7.9	Distributed Data Base Processor (^DDP).....	12-15
12.7.10	Background Job Attacher (^ATTACH)	12-15
12.7.11	Background Job Detacher (^DETACH).....	12-15
12.7.12	Help Text Driver (^HELP).....	12-15
12.7.13	Magnetic Tape Copy (^TAPECOPY)	12-16
12.8	OTHER MISCELLANEOUS SYSTEM ROUTINES	12-16
12.8.1	Load A Driver (^LOAD).....	12-16
12.8.2	Unload A Driver (^UNLOAD)	12-16

CHAPTER 13 GLOBAL STRUCTURE AND OPTIMIZATION

13.1	OVERVIEW OF GLOBAL MEMORY-RESIDENT TABLES AND DISK BLOCKS	13-1
13.2	DISK VOLUMES AND VOLUME SETS.....	13-2
13.3	IN-MEMORY SUPPORT FOR DISK VOLUME SETS	13-3
13.4	GENERAL DISK LAYOUT AND TERMINOLOGY	13-4
13.4.1	DSM-11 Block Layout	13-5
13.4.2	Map Blocks	13-8
13.4.3	Global Directory Block	13-10
13.4.4	Global Pointer Block	13-13
13.4.5	Global Data Block	13-15
13.5	GLOBAL GROWTH AND DEVELOPMENT	13-17
13.5.1	Global Initialization And Structure.....	13-17
13.5.2	Global Levels And Disk Access	13-19
13.5.3	Reserved Disk Space.....	13-19

13.5.4	Overflow From One Disk To Another Disk.....	13-20
13.6	GLOBAL OPTIMIZATION TECHNIQUES	13-20
13.6.1	Optimizing Disk-Cache Usage.....	13-21
13.6.2	Placing UCIs On The Data Base.....	13-22
13.6.3	Global Disk Block Allocation (^DGAM And ^%GLOMAN).....	13-26
13.6.3.1	Global Placement.....	13-27
13.6.3.2	Controlling Global Growth.....	13-27

CHAPTER 14 DSM-11 TABLES AND MEMORY-RESIDENT DATA STRUCTURES

14.1	INTRODUCTION	14-1
14.2	SYSTEM TABLE (SYSTAB)	14-4
14.3	JOB TABLE.....	14-5
14.4	LOCK TABLE.....	14-6
14.5	DEVICE TABLE.....	14-7
14.6	PARTITION TABLE.....	14-8
14.7	VOLUME SET TABLE.....	14-9
14.8	THE UCI TABLE	14-10
14.9	STORAGE ALLOCATION TABLE (SAT)	14-12
14.10	DISK TABLE.....	14-13
14.11	BAD BLOCK TABLE (BBTAB).....	14-13
14.12	DEVICE DESCRIPTOR BLOCK (DDB).....	14-14
14.13	PARTITIONS	14-17
14.13.1	Partition Sections	14-17
14.13.2	Partition Vector	14-18

CHAPTER 15 OPTIMIZING DSM-11

15.1	MAPPED ROUTINES	15-1
15.2	VARYING BUFFER SIZES	15-2
15.3	LIMITING CRITICAL DATA AREAS TO MINIMIZE SEARCHING	15-3
15.3.1	Limiting The Size Of The Global Directory.....	15-3
15.3.2	Limiting The Size Of The UCI Translation Table.....	15-4
15.4	PERFORMANCE STATISTICS.....	15-4
15.4.1	Using The Performance Statistics Utility (^RTHIST)	15-4
15.4.2	Performance Data.....	15-4
15.4.2.1	Queue Data	15-5

15.4.2.2	Data Base Event Counts.....	15-6
15.4.2.3	Derivative Ratios	15-7
15.4.2.4	Disk Usage Histogram	15-7
15.4.2.5	Routine Name Histogram	15-8
15.4.2.6	Global Name Histogram	15-8
15.4.3	Optimization Based On Performance Statistics.....	15-8

CHAPTER 16 THE DSM-11 JOURNAL PROCEDURE

16.1	INTRODUCTION	16-1
16.2	SELECTION OF GLOBALS	16-2
16.3	THE JOURNAL UTILITIES.....	16-2
16.3.1	Start Journal (^JRNSTART).....	16-3
16.3.2	Stop Journal (^JRNSTOP)	16-3
16.3.3	Show Journal Spaces (^JRNLSHOW).....	16-3
16.3.4	Allocate New Journal Space (^JRNALL).....	16-4
16.3.5	Initialize Journal Space (^JRNINIT).....	16-4
16.3.6	Deallocate Journal Space(^JRNDEALL)	16-4
16.3.7	Journal Recover (^JRNRECOV).....	16-4
16.3.8	Dejournal (^DEJRNL).....	16-5
16.4	JOURNALING GLOBALS	16-5
16.4.1	Journaling To Disk	16-5
16.4.2	Journaling To Magnetic Tape.....	16-6
16.5	THE DEJOURNAL PROCEDURE	16-6

CHAPTER 17 VOLUME SETS AND CLUSTERED SYSTEMS

17.1	VOLUME SETS.....	17-2
17.1.1	Analogy To A Set Of Books	17-3
17.1.2	DSM-11 Conventions For Volume Sets.....	17-3
17.1.3	Practical Uses Of Volume Sets	17-4
17.1.4	Logging In A Volume Set	17-5
17.2	CLUSTERED SYSTEMS	17-5
17.2.1	System Generation For Clustered Systems (^SYSGEN).....	17-5
17.2.2	The DDP Utility (^DDPUTL)	17-9
17.2.3	Explicit Syntax	17-11
17.2.4	Using The UCI Translation Table (^UCITRAN).....	17-11
17.2.5	Journaling And Backup	17-14
17.2.6	Setting Access Codes And Categories For Globals (^%GCH)	17-14
17.2.7	Dual Porting.....	17-15

17.2.8	Library UCI	17-15
17.2.9	Using \$ZR To Verify DDP	17-16

APPENDIX A DETECTING AND RECOVERING FROM ERRORS

A.1	TYPES OF ERRORS	A-1
A.2	MUMPS PROGRAMMING ERRORS	A-1
A.3	MASS STORAGE HARDWARE ERRORS.....	A-2
A.4	ERRORS THAT CAUSE SYSTEM FAILURE	A-2
A.4.1	DSM-11 System Soft Crash	A-3
A.4.2	Soft Crash Recovery	A-5
A.4.3	DSM-11 Hard Crash	A-5
A.5	DSM-11 ERROR MESSAGES	A-6

APPENDIX B ROUTINE FILE STRUCTURE

APPENDIX C DSM-11 EDITORS

C.1	THE DSM-11 EDITOR	C-1
C.2	THE %EDI EDITOR	C-4
C.2.1	Using The %EDI Editor	C-5
C.2.2	EDI Commands	C-5

APPENDIX D DSM-11 DISK CAPACITIES AND CALCULATIONS

D.1	DSM-11 DISK CAPACITIES	D-1
D.2	COMPUTING BLOCK NUMBERS.....	D-2

APPENDIX E USING THE \$ZCALL FUNCTION

E.1	CREATING THE \$ZCALL NAME TABLE.....	E-2
E.2	USING DSM-11 GENERAL PURPOSE REGISTERS	E-2
E.3	ERROR CONDITIONS	E-4

APPENDIX F USING THE OPERATIONS CONTROL REGISTER

F.1	OPERATIONS CONTROL REGISTER USE.....	F-1
-----	--------------------------------------	-----

APPENDIX G USING BIT-MASKING TECHNIQUES

APPENDIX H INTERNAL SUBSCRIPT FORMAT FOR A DSM-11 GLOBAL

GLOSSARY

INDEX

FIGURES

1-1	Power On for the VT100 Terminal	1-9
1-2	Power On for the LA120 Terminal	1-9
1-3	Power On for the PDP-11/23-PLUS	1-11
1-4	Power On for the PDP-11/24	1-12
1-5	Power On for the PDP-11/44	1-13
1-6	Starting the PDP-11/23-PLUS	1-16
1-7	Starting the PDP-11/24	1-17
1-8	Starting the PDP-11/44	1-18
3-1	Common DSM-11 Terminals	3-2
3-2	The VT100 and VT52 Keyboards.....	3-3
6-1	IBM Standard EBCDIC Label	6-28
6-2	Variable-Length Record Format	6-29
6-3	Spool File Structure.....	6-53
6-4	Structure of Spool Directory Block	6-54
6-5	Header of Spool Data Block	6-55
6-6	Open File Table Entry Format	6-56
8-1	Typical Balanced Tree	8-4
9-1	Routine Execution Path Using Nested DO Commands.....	9-3
9-2	Routine Execution Path Using Top-Down Structure.....	9-4
9-3	Optimum Code Structure of a Routine and Its Components	9-5
13-1	General Disk Layout	13-5
13-2	DSM-11 Block	13-6
13-3	Physical Structure of a Global.....	13-8
13-4	Map Block	13-9
13-5	Global Directory Block	13-11
13-6	Protection Byte Bit Assignments	13-12
13-7	A Global Key	13-13
13-8	Global Pointer Block.....	13-14
13-9	Global Data Block	13-16
13-10	Global Initialization.....	13-18
13-11	Typical Global Layout	13-24
13-12	Disk Allocation for Two UCIs.....	13-25

13-13	Global Data Growth Area Allocation for Three Globals	13-26
14-1	System Linkages	14-3
14-2	Job Table.....	14-6
14-3	Device Table.....	14-8
14-4	Partition Table.....	14-9
14-5	Volume Set Table	14-10
14-6	UCI Table.....	14-11
14-7	Device Descriptor Block for devices 1-19 and 63-191.....	14-16
14-8	Partition Layout	14-17
15-1	RTHIST: Average Counts of Jobs	15-9
15-2	RTHIST: Averages of Data Base Events Per Second	15-10
15-3	RTHIST: Derivative Ratios	15-11
15-4	RTHIST: Disk Access as Percentage of Total Time	15-12
15-5	RTHIST: Routine Execution as Percentage of Total Time	15-13
15-6	RTHIST: Global References Per Second	15-14
B-1	Routine File Structure	B-2
B-2	Routine Data Block	B-3
E-1	Memory Area for User-Written Subfunctions	E-2

TABLES

3-1	Function Keys	3-4
3-2	Control Characters	3-6
5-1	DSM-11 I/O Device Table	5-2
6-1	Device Status Word Bit Assignments.....	6-4
6-2	DH11 Line Parameter Register Bit Assignments	6-8
6-3	DZ11 Line Parameter Register Bit Assignments.....	6-9
6-4	Magnetic Tape OPEN Switches.....	6-19
6-5	Legal OPEN Switch Combinations	6-21
6-6	Magnetic Tape Control Codes Used with WRITE.....	6-23
6-7	Magnetic Tape Device \$ZA Status Bit Assignments.....	6-25
6-8	DMC11 Control Codes Used with WRITE * for Block Mode	6-42
6-9	DMC11 Control Codes Used with WRITE for Buffer Mode.....	6-44
6-10	DMC11 Device \$ZA Status Bit Assignments	6-45
6-11	VIEW Switches	6-59
6-12	RX02 OPEN Block Format Switches	6-62
6-13	RX02 OPEN I/O Mode Switches.....	6-62
6-14	RX02 Error Codes	6-65
6-15	TU58 OPEN Block Format Switches	6-67
6-16	TU58 OPEN I/O Mode Switches	6-67
6-17	TU58 Error Codes	6-70
13-1	Disk Types and Disk Codes	13-5
14-1	DSM-11 Tables and Memory-Resident Data Structures.....	14-2
14-2	Job Status Word Bit Assignment	14-19
D-1	DSM-11 Disk Capacities	D-1
F-1	Values for Operations Control Register Switches.....	F-3

Acknowledgement

Digital Standard MUMPS (DSM) is an extension of the ANSI Standard Specification (X11.1-proposed for final approval in 1984) for the Massachusetts General Hospital Utility Multi-Programming System (MUMPS). MUMPS was originally developed at the Laboratory of Computer Science at Massachusetts General Hospital, and was supported by the grant HS00240 from the National Center for Health Services Research and Development.

Preface

DSM-11 stands for Digital Standard MUMPS for the PDP-11.

MANUAL OBJECTIVES

This manual provides information for application programmers on how to use DSM-11 to develop application routines and data base application systems. In addition, it provides information for System Managers and operations personnel on how to perform system-generation and maintenance operations.

INTENDED AUDIENCE

This manual is intended as a reference for novice computer users, DSM-11 application programmers, System Managers, and all others who operate a DSM-11 system.

MANUAL STRUCTURE

This manual is structured as follows:

Part 1: Installing DSM-11

Chapter 1 describes the installation of the DSM-11 system.

Part 2: Introduction to DSM-11

Chapter 2 provides an overview of the DSM-11 system.

Chapter 3 describes how to use DSM-11 terminals and provides the information necessary to log in the system.

Part 3: Programming and Using the DSM-11 System

Chapter 4 discusses how to use DSM-11 to develop and maintain application routines. It also describes some useful programming techniques and DSM-11 directories.

Chapter 5 describes how to use system devices.

Chapter 6 describes I/O device characteristics.

Chapter 7 discusses the DSM-11 library utilities.

Chapter 8 describes globals.

Chapter 9 discusses how to optimize DSM-11 applications.

Part 4: Managing the DSM-11 System

Chapter 10 describes the DSM-11 system-generation procedure.

Chapter 11 discusses operating procedures for starting, loading, and backing up the DSM-11 system following system generation.

Chapter 12 describes the system utilities.

Chapter 13 describes global structure and data base optimization.

Chapter 14 describes the tables and other memory-resident data structures of the DSM-11 operating system.

Chapter 15 describes how to optimize the DSM-11 system.

Chapter 16 describes the use of the journaling facility and the journal utilities.

Chapter 17 describes volume sets and clustered systems.

The appendices include: Detecting and Recovering from DSM-11 Errors; Routine File Structure; The DSM-11 Editor; DSM-11 Disk Capacities and Calculations; Using the ZCALL Function; Using the Operations Control Register; Using Bit-Masking Techniques; Internal Subscript Format for a DSM-11 Global; and a glossary.

DSM-11 DOCUMENTATION

A complete set of DSM-11 documentation includes:

Manual	Order Number
The <i>DSM-11 BISYNC Programmer's Guide</i>	AA-V602A-TC
The <i>DSM-11 Language Reference Manual</i>	AA-H797B-TC
The <i>DSM-11 Summary</i>	AV-H798B-TC

The <i>DSM-11 User's Guide</i>	AA-H799B-TC
The <i>DSM-11 XDT Reference Manual</i>	AA-J701A-TC
The <i>Introduction to DSM</i>	AA-K676A-TK

RELATED DOCUMENTATION

The following documents provide additional information on PDP-11 processors:

Manual	Order Number
<i>PDP-11 Processor Handbook</i>	EB-19402-20
<i>Peripherals Handbook</i>	EB-20443-20
<i>Terminals and Communications Handbook</i>	EB-20752-20
<i>PDP-11/24 System Technical Manual</i>	EK-11024-TM
<i>PDP-11/44 System User's Guide</i>	EK-11044-UG
<i>PDP-11/23-PLUS System Manual</i>	EK-1T23B-OP
<i>MICRO/PDP-11 Handbook</i>	EB-24944-18
<i>MICRO/PDP-11 Unpacking and Installation Guide</i>	EK-OLCP5-IN
<i>MICRO/PDP-11 Owner's Manual</i>	EK-OLCP5-OM
<i>MICRO/PDP-11 Technical Manual</i>	EK-OLCP5-TM

DOCUMENTATION CONVENTIONS

This manual uses the following documentation conventions and symbols:

Convention	Meaning
<CTRL>x	A key (represented here by x) typed while the CTRL key is pressed.
<ERROR>	DSM-11 error message format.
ESC	The escape key.
PERIOD	The period key.
RET	The carriage return key.
SP	The space bar.
TAB	The TAB key (CTRL/I on some terminals).
{item}	An optional item.
	A convention to represent a break in a series.
>	The DSM-11 (Programmer Mode) prompt.

A symbol to represent a break between two illustrated lines in a routine example.

Contrasting Colors in Examples:

Black - indicates system-generated output

Red - indicates user input

Part 1: Installing DSM-11

Chapter 1

Installing DSM-11

This chapter describes the DSM-11 distribution kit and the DSM-11 installation procedure for both magnetic tape, disk cartridge, and diskettes. This chapter also describes the DSM-11 Baseline System.

This chapter provides information for installing a DSM-11 Version 3 software on a PDP-11 computer system that does not have an earlier version of the DSM-11 software previously installed. Installation means that you put the software into the hardware and make the software operate properly.

If you already have an earlier version of the DSM-11 software operating on your PDP-11 computer system, you are then upgrading your software (by replacing the earlier version DSM-11 software with Version 3 DSM-11 software). Refer to the *DSM-11 Release Notes* for a discussion of how to upgrade your DSM-11 software to Version 3.

This chapter does not describe how to customize your system using the system generation procedure. Customization is described in Chapter 10.

The discussion in this installation chapter is oriented around installing DSM-11 on one of the following systems:

- PDP-11/23-PLUS
- PDP-11/24
- PDP-11/44
- MICRO/PDP-11

The additional chapters in this manual provide information on using, operating, and managing DSM-11, as well as programming in MUMPS using DSM-11. A basic introduction to DSM-11 and MUMPS is provided in Chapter 2.

1.1 The DSM-11 Distribution Kit

The DSM-11 distribution kit contains both software and documentation. Digital distributes the DSM-11 software on all of the following:

- A single 9-track magnetic tape (800 or 1600 bits per inch)
- A single disk cartridge
- A set of diskettes

A distribution kit for the magnetic tape and the diskettes contains backup copies of distribution software. For a distribution on disk cartridge, you must provide your own blank backup distribution disk, and make your own backup copy of the distribution software at installation. The software consists of the following:

- The DSM-11 System Image
- The DSM-11 installation routines
- The DSM-11 system and library utility routines
- The DSM-11 system and library globals

The documentation consists of the following:

- The DSM-11 manuals (as listed in the preface of this manual)
- The *DSM-11 Release Notes*

1.2 The DSM-11 Baseline System

When you first install DSM-11, you have a DSM-11 Baseline System. You can proceed to the additional system generation questions and set up a full system.

The DSM-11 Baseline System supports:

- One terminal (the console terminal)
- One partition of 8K bytes
- One User Class Identifier (UCI) called MGR

- A Programmer Access Code (PAC of `(CTRL/Z)` `(CTRL/Z)` `(CTRL/Z)`)
- Disk buffers
- One magnetic tape drive (if present)
- One line printer (if present)

The primary purpose of the DSM-11 Baseline System is to allow operation of the DSM-11 system installation, system generation, and start-up procedures.

With the Baseline System, only the console terminal user can be logged into the system. The Baseline System can be useful for experimenting with the language or for performing system and data base maintenance functions. All of the features of the DSM-11 language and the DSM-11 utilities are available.

1.3 System Operation Information Sources

Chapter 11 contains system operation information, which includes starting, backing up, and shutting down DSM-11.

You can load DSM-11 directly from the distribution magnetic tape, disk cartridge, or diskettes by employing bootstrap procedures. Bootstrap procedures for the PDP-11/23-PLUS, the PDP-11/24, and the PDP-11/44 are described in Section 1.10. Additional information on these systems can be found in:

- *PDP-11/23-PLUS System Manual*
- *PDP-11/24 System Technical Manual*
- *PDP-11/44 System User's Guide*

Bootstrap information for the MICRO/PDP-11 can be found in Section 1.11. Additional information for the MICRO/PDP-11 is found in the following manuals:

- *MICRO/PDP-11 Unpacking and Installation Guide*
- *MICRO/PDP-11 Owner's Manual*
- *MICRO/PDP-11 Technical Manual*

Ordering numbers for these manuals and guides can be found in the preface.

1.4 Installation Procedure Overview

The installation procedures described in this chapter cover three major steps:

1. Installing a DSM-11 Baseline System, which gives you an operating DSM-11 system with limited capabilities
2. Using the system generation process (using autoconfiguration) to generate a software system with full capabilities (a configuration)
3. Using the start-up procedure to get your configuration started and operating

A more detailed list of the steps involved in the procedure is given later in this section.

A configuration is a software construct that allows you access to certain hardware devices (that must be physically connected to your system) and to certain software options that you have chosen during the system generation process. The system generation process is the means of making up a configuration. The Baseline System is an example of a simple and limited configuration.

The installation dialogue follows the autoconfiguration process after the Baseline System is installed. The autoconfiguration process is a system generation process that makes up a configuration that includes all of your hardware devices and most software options. The autoconfiguration process minimizes the number of questions you must answer and quickly provides you with a configuration with wide capabilities.

Before beginning the actual procedure, you should complete the installation checklist given in Section 1.5. You should also review the sample dialogue which is given later in this chapter in Section 1.12. Then you should begin the installation procedure described in Sections 1.6 through 1.10 for the PDP-11/23-PLUS, PDP-11/24, and PDP-11/44 processors. For the MICRO/PDP-11, follow the procedure described in Section 1.11.

The installation procedure for the PDP-11/23-PLUS, PDP-11/24, and PDP-11/44 contains the following steps:

1. Complete the preinstallation checklist. See Section 1.5.
2. Switch on the power for your console terminal. See Section 1.6.
3. Switch on the power for your PDP-11 processor. See Section 1.7.
4. Mount the distribution medium in one of the appropriate tape, disk, or diskette drives on your system. See Section 1.8.

5. Mount a disk, which is to be your backup distribution disk or your system disk, into a disk drive. See Section 1.9.
6. Start the system from the drive containing the distribution medium. This is sometimes referred to as "bootstrapping" the system. See Section 1.10 for more information.
7. Answer the questions contained in the installation dialogue which appears on your console terminal. An example dialogue appears in Section 1.12. These questions cover several steps:
 - a. Making a backup copy of the distribution disk, unloading the backup disk, and loading a disk which is to be your system disk (if your distribution is on disk cartridge).
 - b. Installation of a DSM-11 Baseline System.
 - c. System generation of a system configuration (by the autoconfiguration process) that allows you access to all the hardware devices and software options.
 - d. System start-up of the initial configuration. The configuration that you construct during system generation does not actually come into effect on your system until you start it up. You must complete the start-up procedure, which starts the configuration, and then log in to the configuration.

1.5 Preinstallation Checklist

Before you run the DSM-11 installation procedure, read the following checklist. Completing this checklist insures that you are ready to begin installation and helps prepare you to answer questions asked during the procedure. You should not actually mount the distribution medium, start the system, or begin the installation until after you have completed this checklist.

1. You must have your hardware installed and ready to go. You must identify which of your terminals is the console terminal. The installation must be done from this terminal.
2. DSM-11 is distributed on three types of media:
 - 9-track magnetic tape, at a density of either 800 or 1600 bits per inch
 - Disk cartridge - RL01, RL02, RK06, or RK07
 - RX50 diskettes

For magnetic tape installation, you must have a tape drive connected to your system. For disk cartridge installation, you must have an RL01,

RL02, RK06, or RK07 disk drive with your system, and the disk cartridge must match the type of disk drive that you have. RX50 diskette installation requires that your machine have the RX50 diskette drive.

3. You must know how to mount your distribution tape, disk cartridge, or diskette, and how to boot the system from the tape or disk. Refer to Sections 1.8 and 1.10 for information on how to mount your magnetic tape or disk cartridge, and how to boot from that tape or cartridge. For the RX50 installation, refer to Section 1.11 for mounting and booting information. For the RX50, you receive several diskettes. You must mount the first of the diskettes.
4. You must know the device bootstrap identifier for the disk or magnetic tape drive that is to contain your distribution medium if your system is a PDP-11/23-PLUS or PDP-11/44. This identifier is a code that informs the system where your distribution medium is, and you should start (bootstrap) the system from this medium.

Device Identifier Codes

Code	Device
DL	RL01 or RL02 Disk Drive
DM	RK06 or RK07 Disk Drive
DU	RD51, RX50, RA80, RA81, or RA60 Disk Drive
MM	TU16, TE16, TU77, or TU45 Tape Drive
MS	TSV05, TU80, or TS11 Tape Drive
MT	TU10, TE10, or TS03 Tape Drive

If there is more than one drive of the same type, then each separate drive is referred to as a unit. You can mount your distribution medium in any unit of the appropriate drive. You then refer to the drive with a 3-character code, including the unit number; for example, RL01 disk drive, unit 0 is DL0. If you do not load the distribution medium in unit 0, you must specify the unit number when you boot the system (except for the MICRO/PDP-11 which determines automatically which drive has the distribution diskette).

Device bootstrap identifier for the distribution medium _____

5. You must choose a disk drive to use to back up your distribution disk. You do not need to do this if your distribution medium is magnetic tape or diskettes; the distribution kit contains an extra copy of the distribution medium. If the distribution medium is disk cartridge, then you must provide your own disk for a backup distribution disk. You should use a new disk or a used disk in good condition. If the disk is previously used, any information currently stored on it will be lost.

The backup distribution disk is a copy of the distribution disk and should be saved in case the original distribution disk is lost or damaged. The

system disk is not the same as a distribution disk; you cannot install a new system from the system disk.

The disk drive for the backup distribution disk can be the same as the one you use for your system disk. If you do this, you must unload the distribution backup disk after it is created, and load another disk that is to become the system disk into that drive.

Distribution backup disk type _____

Unit number of disk drive _____

6. You must choose a disk drive for your system drive (except for the MICRO/PDP-11 which uses the RD51 Winchester fixed disk drive for the system disk). This disk drive must have a disk mounted in it before you start your installation. This disk becomes your system disk. You should use a new disk or a used disk in good condition. If the disk is previously used, any information currently stored on it will be lost. You should also know the unit number of the drive that this disk is installed in.

System disk type _____

Unit number of system disk drive _____

7. You should let the system check for bad disk blocks. The installation procedure also asks you a series of questions about "bad blocks." A block is a space on the disk that contains a fixed amount of information (1024 bytes). A bad block is one that has a physical defect and should not be used. The system can check for bad blocks and avoid using them.

If you have previously used DSM-11 and are using a disk that has bad blocks for the system, then you should write down the numbers of these bad blocks (if you know them). You will be given an opportunity to give the system the numbers of these bad blocks. If this does not apply to you, then answer NO to the question about additional bad blocks.

Additional bad blocks _____

8. You should decide what the system disk label should be. The label you choose is written to the disk during installation. It should be a brief phrase that makes some kind of sense, such as "TEST SYSTEM" or "SYSTEM DISK 1". It can be any string of up to 22 characters. This label is used for identification purposes only.

System Disk Label _____

9. You must choose a volume set name. This name is a 3-character name for the system disk volume. A volume set is a data base contained on a disk or set of disks.

Volume Set Label _____

10. You must choose a name for your initial configuration. At this point, you are setting up a configuration. A configuration is a set of hardware devices (from those connected to your system) and a set of software options. You can set up different choices of devices and options in each of several configurations. When you start up your system, you can choose among the different configurations you have set up.

You do not need to concern yourself at this time with these choices. You can take the autoconfiguration option as indicated in the installation procedure to set up an initial configuration. See Part 4 of this manual for more information on the operating system, and Chapter 10 specifically for more information on system generation and setting up additional configurations.

The configuration you select should be a short string of characters such as "TEST", "FIRST", your initials, or a number.

Configuration name _____

11. You must indicate what your power line frequency is. In the United States, it is 60 hertz. In many other countries, it is 50 hertz. You must specify the frequency correctly, or your time-of-day clock will give incorrect values.

Power line frequency _____

12. You must determine what you want the 3-character programmer access code (PAC) to be. This code is a password which is requested when you log in to DSM-11. Do not choose a code that is obvious or easy to guess, such as your own initials.

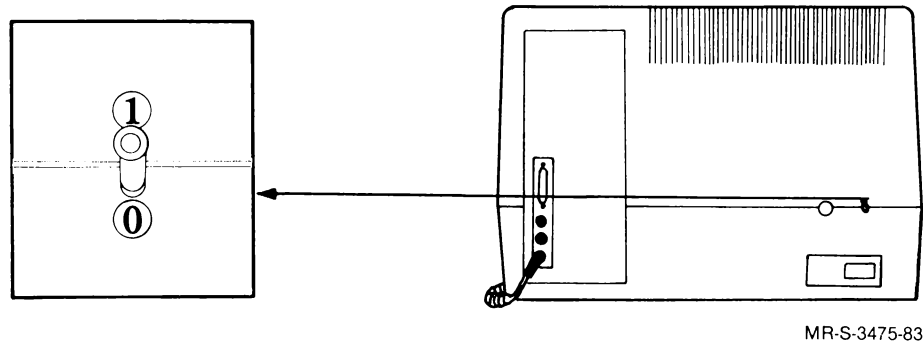
Programmer Access Code _____

1.6 Power On For The Console Terminal

Your console terminal can be either a VT100 video terminal or an LA120 hard-copy terminal. The console terminal is the main or primary terminal for the system.

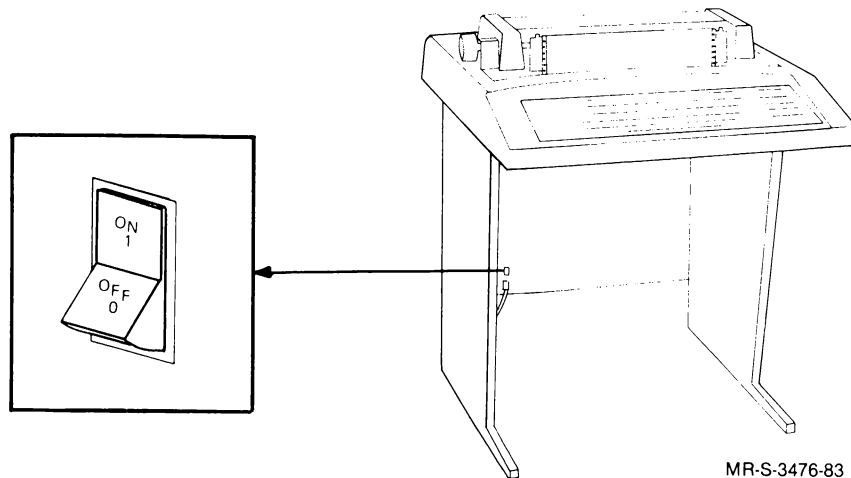
If you have a VT100 video terminal, refer to Figure 1-1. Lift the power switch to the (1) position.

Figure 1-1: Power On for the VT100 Terminal



If you have an LA120 hard-copy terminal, refer to Figure 1-2. Press the power switch to the ON position.

Figure 1-2: Power On for the LA120 Terminal



Proceed to the next section about switching on power for your PDP-11 processor.

1.7 Power On For The PDP-11 Processor

This section describes how to switch on the power for three processors:

- PDP-11/23-PLUS
- PDP-11/24
- PDP-11/44

If you are installing DSM-11 on a different PDP-11 processor, refer to your PDP-11 user's, system installation, or system operating guide for information on how to switch on power for the processor. If you are installing DSM-11 on a MICRO/PDP-11, refer to the *MICRO/PDP-11 Owner's Manual* for information on how to switch on power for the processor.

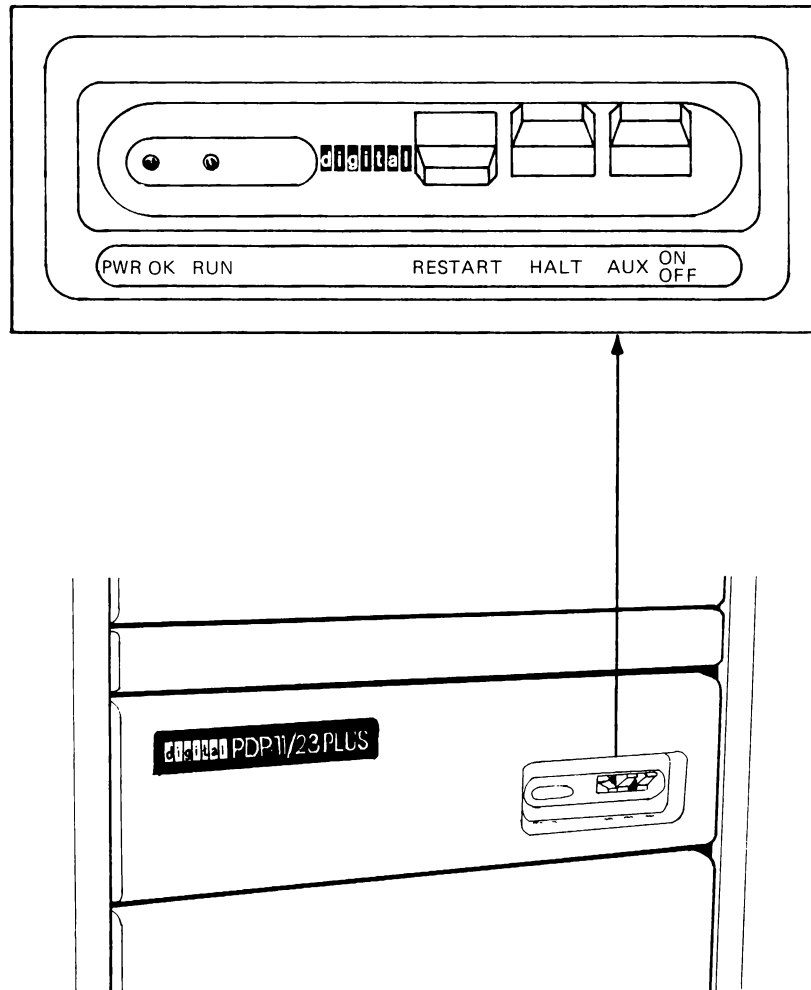
Once you have switched on the processor, see Section 1.8 for information on mounting the distribution medium.

1.7.1 Power On For The PDP-11/23-PLUS

Refer to Figure 1-3.

1. Lift the AUX toggle switch to the ON (up) position. The red PWR OK indicator light on the front panel lights up.
2. Lift the HALT toggle switch to the up position.

Figure 1-3: Power on for the PDP-11/23-PLUS



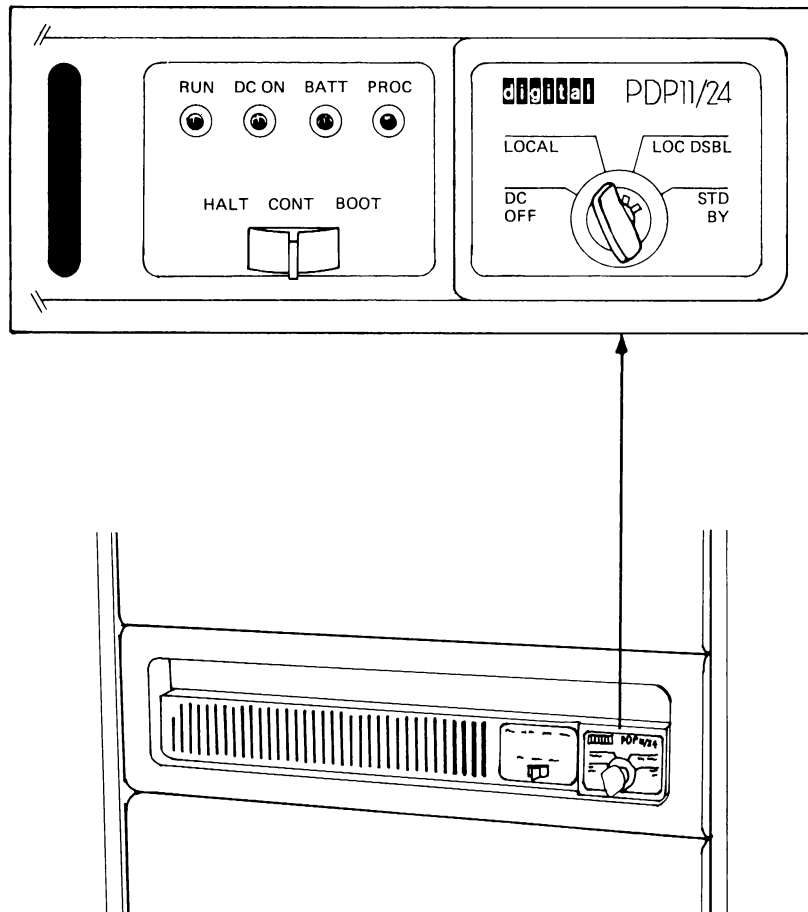
MR-S-3478-83

1.7.2 Power On For The PDP-11/24

Refer to Figure 1-4.

Turn the status selector key to the LOCAL position. The red DC ON indicator on the front panel lights up.

Figure 1-4: Power On for the PDP-11/24



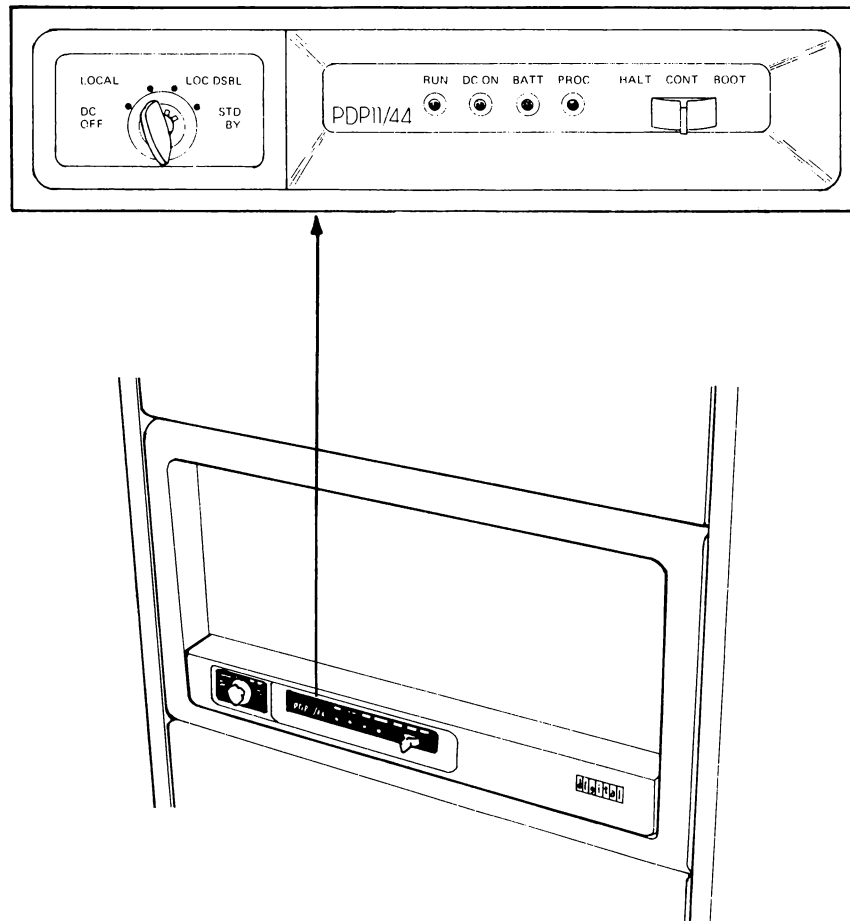
MR-S-3477-83

1.7.3 Power On For The PDP-11/44

Refer to Figure 1-5.

Turn the status selector key to the LOCAL position. The red DC ON indicator on the front panel lights up.

Figure 1-5: Power On for the PDP-11/44



MR-S-3479-83

1.8 Mounting The Distribution Medium

When there is more than one drive of a given type, each drive is referred to as a unit. For instance, if there are two disk drives, they can be referred to as unit 0 and unit 1 of that type of drive. In general, the distribution medium should be mounted in unit 0. If the distribution medium is tape, it must be mounted in unit 0. If you mounted a disk in a unit other than 0, you must inform the system which unit the disk is in when you boot the system.

- Magnetic Tape - Different models of tape drives have different mounting procedures. Instructions for mounting the tape are contained on the inside of the door to the tape-reading mechanism. Open the door and read the instructions. Mount the tape following the instructions.
- Disk Cartridge - Refer to your disk drive user's guide for information on how to mount the disk cartridge for your particular drive.
- Diskettes - Refer to Section 1.11.

1.9 Mounting The System Or Backup Distribution Disk

Refer to the disk drive user's guide for information on how to mount the disk into the disk drive. You must provide your own blank disk to be used as the system disk. It is not provided with the distribution kit.

For distribution on diskettes or magnetic tape, backup copies of the distribution medium come with the distribution kit. If your distribution medium is on disk cartridge, then you should provide a disk of your own to back up the distribution disk. If this is the case, you should mount the backup distribution disk in the disk drive. At the beginning of the installation dialogue, you can copy the distribution disk onto a backup disk. You must then unload the backup disk and load a another blank disk, which becomes your system disk, into the drive.

1.10 Start The System

You must now start (or boot) the system. You boot the system from the disk or tape drive that contains the distribution medium. If your system is a PDP-11/23-PLUS or PDP-11/44, you need to know how to identify the drive that contains the distribution medium by a code. This code is described in item number 4 of the preinstallation checklist in Section 1.5.

The following sections describe the bootstrap procedure for these systems:

- PDP-11/23-PLUS
- PDP-11/24
- PDP-11/44

If you are installing DSM-11 on a different PDP-11 processor, refer to your PDP-11 user's, system installation, or system operating guide for information on how to bootstrap the processor. If you are installing DSM-11 on the MICRO/PDP-11, refer to Section 1.11.

Do not actually bootstrap your system until you are ready to go through the DSM-11 installation dialogue, for this dialogue begins automatically once the system has been booted.

1.10.1 Starting The PDP-11/23-PLUS

Refer to Figure 1-6.

1. Lift the RESTART toggle switch. The switch springs back to the center position, and the read RUN indicator lights up. The following messages appear on your terminal:

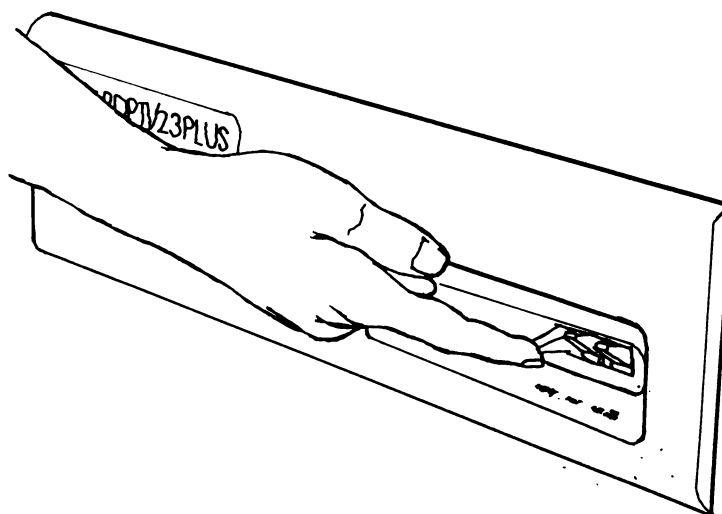
```
TESTING MEMORY  
xxxx.kw  
START?
```

2. Type on your terminal the device bootstrap identifier for the drive that contains your distribution medium, for example:

```
DL1 RET
```

3. Press the carriage return key on your terminal. Wait for instructions to be typed out on the terminal.
4. The DSM-11 installation dialogue now begins. See Section 1.12.

Figure 1-6: Starting the PDP-11/23-PLUS



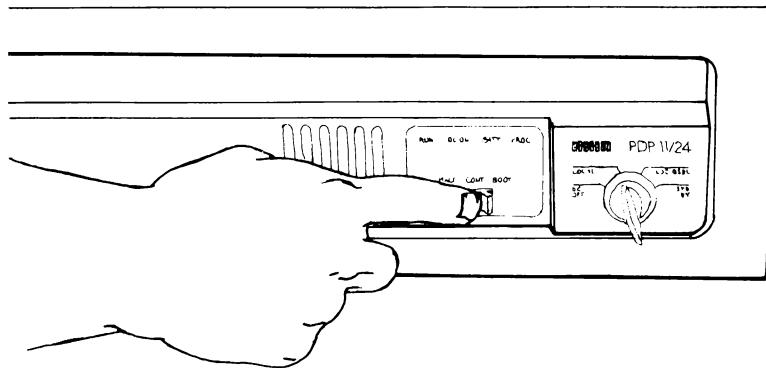
MR-S-3480-83

1.10.2 Starting The PDP-11/24

Refer to Figure 1-7.

1. Slide the HALT/CONT/BOOT toggle switch to the BOOT position. The switch springs back to the center position, and the red RUN indicator lights up. Numbers indicating the amount of memory appear on your terminal. DSM-11 is automatically bootstrapped.
2. The DSM-11 installation dialogue now begins. See Section 1.12.

Figure 1-7: Starting the PDP-11/24



MR-S-3481-83

1.10.3 Starting The PDP-11/44

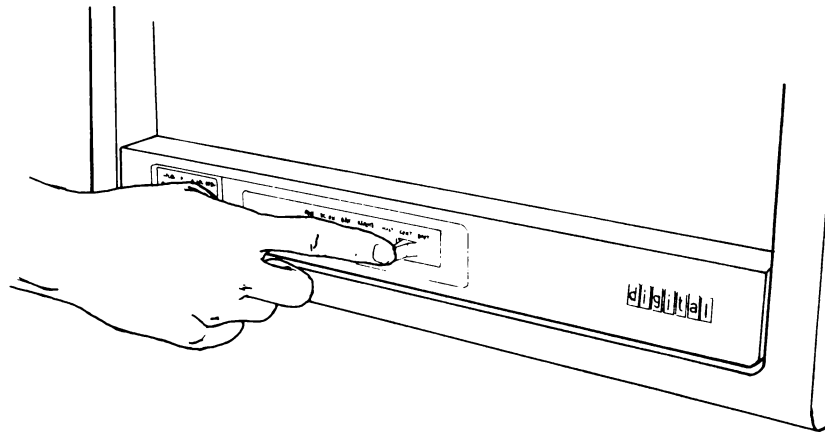
Refer to Figure 1-8.

1. Slide the HALT/CONT/BOOT toggle switch to the BOOT position. The switch springs back to the center position, and the red RUN indicator lights up. The prompt >>> appears on the terminal.
2. Type on your terminal the bootstrap command and the device bootstrap identifier, for example:

```
B DL1 RET
```

3. Press the carriage return key on your terminal. Wait for instructions to appear on your terminal.
4. The DSM-11 installation dialogue now begins. See Section 1.12.

Figure 1-8: Starting the PDP-11/44



MR-S-3482-83

1.11 MICRO/PDP-11 Installation Procedure

The procedure for installing DSM-11 on the MICRO/PDP-11 is as follows:

1. Unpack and install the MICRO/PDP-11 hardware. Follow the instructions indicated in your *MICRO/PDP-11 Unpacking and Installation Guide*. Your MICRO/PDP-11 model must include the RD51 Winchester fixed disk drive.

Do not switch on power for your MICRO/PDP-11 system unless you want to check out the system by running the MICRO/PDP-11 user test diskettes. See the *MICRO/PDP-11 Unpacking and Installation Guide* for information on the test diskettes.

CAUTION

Make sure that any cardboard shipping inserts have removed from the disk drives before switching on your MICRO/PDP-11. NEVER switch on the system while these cardboard inserts are in the disk drives.

2. Complete the preinstallation checklist given 1.5.
3. Switch on the power for your console terminal, if you have not already done so during the unpacking and installation procedure described in your *MICRO/PDP-11 Unpacking and Installation Guide*. Section 1.6 also gives information about switching on your console terminal.
4. Set the power switch on your MICRO/PDP-11 to on (1). See your *MICRO/PDP-11 Unpacking and Installation Guide* or *MICRO/PDP-11 Owner's Manual* for more information. The following message is displayed on the terminal:

```
KDF11B-BE ROM V1.0  
  
128K MEMORY  
9 STEP MEMORY TEST  
STEP 1 2 3 4 5 6 7 8 9  
TOTAL MEMORY ERRORS = 0  
CLOCK ENABLED
```

The first line of this text may be different for different models of the MICRO/PDP-11. If the terminal is not an ANSI terminal, it displays "6n" twice. If it is an ANSI terminal, there are two carriage returns.

The system automatically tries to find bootable media on the system. The following messages are printed (since the system cannot find any bootable media):

```
ERROR UNIT DU2  
ERR 16 NOT BOOTABLE
```

```
ERROR UNIT DU1  
ERR 16 NOT BOOTABLE
```

```
ERROR UNIT DU0  
ERR 16 NOT BOOTABLE  
WISH TO REBOOT [[Y,(N)]?
```

5. Insert the first DSM-11 RX50 diskette into one of the RX50 diskette drives. If you are unfamiliar with how to insert an RX50 diskette into a drive, refer to the *MICRO/PDP-11 Owner's Guide*.

Type:

```
Y(RET)
```

The following message appears:

```
BOOTING FROM DU1  
Booting DSM-11...
```

The message can refer to DU1 or DU2, depending on which drive you put the diskette in. It does not matter which drive you use.

6. The DSM-11 installation dialogue now begins. See Section 1.12.

1.12 Installation Dialogue

Before beginning the dialogue, you should have completed the preinstallation checklist, and loaded a disk that is to be your system disk (except for the MICRO/PDP-11 which has a fixed Winchester disk, which becomes the system disk).

You now begin the installation dialogue. This is a series of questions that you must answer. After each question there is a short comment in this manual. You can also obtain helpful information by typing a question mark (?) as your answer to the question. The computer types out a short message about the question on the console or terminal, and then restates the question.

Each question ends with the prompt (>). When the prompt appears, type your answer to the question, and then press the carriage return. If you type only a carriage return, you get the default answer, which follows the question and is contained within angle brackets, < >.

The computer may type out messages for your information which you do not need to answer. These messages do not end in a prompt.

If a question demands a YES or NO, answer with a lowercase or uppercase Y or N, followed by a carriage return. If you type an inappropriate answer or a ?, then you get a help message.

SAMPLE INSTALLATION:

```
28
Start?
DL0 RET
Booting DSM-11...
```

It takes approximately 15 seconds to boot the system. If you loaded the distribution medium into a unit number different from unit 0, be sure to indicate this when booting, for example, DL1 for unit 1 of an RL02 drive.

```
DSM-11 Version 3
```

```
Now running the Baseline System.
```

```
This is a DSM-11 Version 3 Distribution Disk.
```

```
You can use this specialized DSM-11 system either to create your own DSM-11 system, or to copy this disk onto another disk (as a backup).
```

You should make a copy of the distribution disk to serve as a backup. Later in this dialogue you can install the system on your system disk. You need

separate disks for the backup distribution disk and the system disk. If your distribution medium is diskettes or tape, you do not need to back them up because you receive a backup copy of the diskettes or tape with the distribution kit.

Do you wish to make a copy [Y or N] <N> YES

Answer YES if you want to make a copy of the distribution disk.

Make copy on which disk unit ? >DL1

Indicate the disk unit number that contains the disk that can be overwritten with a copy of the distribution disk. Overwriting means that any data on the disk is destroyed and replaced with data provided from the distribution disk. If you answer ? to this question, you get a list of the disk drives that are available. After you indicate which disk unit to use, it takes about five minutes to make the copy.

Begin copy...

End copy...

Now you must remove the backup copy of the distribution disk and load a disk that you wish to become your system disk.

Begin DSM-11 Version 3 System Installation

Answer with a question mark (?) any time you wish more information.

Please enter today's date [dd-mmm-yy] ? >3-JUL-83
and time [hh:mm:ss] ? >11:45

Possible disk units are: DU0 (RD51 unit 0)
DL1 (RL02 unit 1)

Install DSM-11 on which disk ? >DL1

This question is asked only if there is more than one disk drive of more than 2 megabytes capacity connected to the system. Each separate disk drive is referred to as a unit. For the MICRO/PDP-11 system, DSM-11 is installed on DU0, the RD51 Winchester fixed disk drive.

Do you wish to format this disk? Y

This question only appears for RK05, RP04, RP05, or RP06 disks. In general, you should answer YES to this question (if it appears), unless you know for certain that this disk was already formatted.

Do you wish to run a comprehensive test for bad blocks on this disk ? [Y/N] >Y

You should answer yes to this question to uncover any bad blocks. Bad blocks have some physical defect that prevents them from being used. A block can be thought of as a unit of data. In DSM-11, a block is equal to 1024 bytes. Each byte is equal to 8 bits. A bit is a binary piece of data represented by either a "0" or a "1".

The DSM-11 system lists these blocks in an internal Bad Block Table, and avoids using them. Finding bad blocks (unless there are a very large number of them) does not prevent the system from continuing with the installation. If the system finds a large number of bad blocks and fills the Bad Block Table, the system halts the installation. You then have a problem with either the disk or the disk drive.

The testing for bad blocks does not actually occur until the next four questions are answered.

```
Test pattern 177777 octal ? [Y/N] > N
Test pattern 125125 octal ? [Y/N] > N
Test pattern 525525 octal ? [Y/N] > N
Test pattern 000000 octal ? [Y/N] > Y
```

These questions on test patterns are asked only if you answered YES to the previous question about having a bad block check done. The numbers represent patterns of data that will be written to every block on the disk. The "000000" pattern is one commonly used to check DSM-11 disks. The system requires some time to complete this test (for example, about two minutes for an RL02 disk). This time varies depending on the size of the disk involved.

(You can hit the "ESC" key at any time to determine the number of blocks processed so far.)

```
1:20:45      Begin test pattern 000000
```

```
1:23:07      Testing complete
```

```
RL02 Unit 1 Bad Block Table
```

```
The Bad Block Table is empty
```

```
Do you know of any other bad blocks on this disk ? > Y
```

If you know of any bad blocks on this disk (from using it before on a DSM-11 system), then answer Y to this question. If you do not know of any bad blocks, answer N for NO. For a new installation with new disks, you answer N.

```
What is the block number of the bad block? 5432
```

```
Now adding block 5432 to DL1 Bad Block Table.
```

What do you want the label of this disk to be ?
(up to 22 characters; enclosed in quotes) ?) "TEST SYSTEM"

Each disk has a label or name for the disk. This name is written to a block on the disk.

What 3-character uppercase name do you wish to give this volume set ? SYS

Now initializing DL1 for use as a DSM-11 volume...

The system has now collected the appropriate information from you about the disk, and must initialize the disk as a DSM-11 disk. This must be done because DSM-11 has its own format for a disk. When the system begins copying information onto the disk, it expects the disk to be in DSM-11 format.

Loading the DSM-11 Version 3 system utilities onto the system disk:

ZBN ZCRF ZCRF1 ...

The system then begins to copy information from the distribution media to the disk. The distribution media is the disk, magnetic tape, or diskette that contains the DSM-11 software. The disk that is receiving information is referred to as the system disk. It takes about three minutes to copy all of the system utilities onto the disk.

The next question is asked only if your distribution media is RX50 diskettes. If this is the case, you have additional diskettes that must be mounted in diskette drive 2 in the correct sequence. If you booted from drive 1, DSM-11 asks you to mount additional diskettes in drive 2 (DU2). Note that the point at which this question is asked (just before the routine MBPH) can vary from what is shown here.

Please mount the second installation diskette in diskette drive DU2.

MBPH MDAT MMD ...

... UNLOAD UTL V3TAPEGL V3UTILS VALID

The codes such as "MBPH" or "MMD" are the names of utilities that are being transferred to the disk. You do not need to respond to them in any way. Note that the actual list of routines that you see is longer than the list given here.

Please mount the third installation diskette in diskette drive DU2.

Transferring the system globals:

% %EDI %EDIHELP %MENU %Q ...

A global in DSM-11 is equivalent to a data file. The system is now copying basic files that the DSM-11 system uses when it is running. It takes about one minute to transfer these globals.

Now copying the system image onto your new disk, making it a bootable DSM-11 Version 3 system disk...

Now the operating system is copied onto the disk. When this is completed, the system disk is complete; and the following message is given:

DL1 is now a bootable DSM-11 Version 3 system.
You can dismount the distribution disk or diskettes now.

You should remove your distribution tape, diskette, or disk and store it in a safe place. In case of some accident or (unusual) disk problem, you may need to install the system again at some point in the future.

You have no defined configuration.
Do you wish to proceed directly to SYSGEN ? <Y>

You can answer N to this question to have the Baseline System, which is a limited running system - but you should proceed and answer the SYSGEN questions.

System generation for DIGITAL Standard MUMPS

Type ? for HELP at any time.

SYSGEN means "system generation," the next step after system installation, which you have just completed. You can establish different configurations of SYSGEN that have different characteristics (such as different software options), but use the same system disk. You can produce a well optimized system that includes all your hardware devices by simply taking the default values for all of the SYSGEN questions. This approach results in an autoconfiguration that minimizes the number of questions that you are asked.

PART 1: SYSGEN

1.1 Would you like extended help [Y or N] ? <N>

If you answer YES to this question you will get a paragraph (or more) of help text with each question.

1.2 Enter the configuration identifier <1> TEST

You must give this configuration a unique name. At a later time, you may want to set up a different configuration.

1.3 Do you wish to autoconfigure the current system [Y or N] ? <Y> Y

By answering Y to this question, you call the autoconfiguration procedure. On some of the older PDP-11s, the autoconfiguration procedure can fail. The procedure can hang indefinitely. If your procedure does not complete the next step in two minutes, it is probably hanging. If this happens, you must reboot, and then do a system generation without autoconfiguration. In this case, you need to know the CSR and Vector addresses for the hardware devices connected to your system.

Configuring Host System...

This process takes about 30 to 60 seconds.

Processor Type: PDP-11/23

Memory Size: 128 KB

Processor/Memory Options:

Extended Instruction Set

Name	Vector	CSR	Unit	Type	Description
DLA	160	174400		RL11	Disk Controller
			0		
			1		
MSA	224	172520		TSV05	Tape Controller
YLA	300	176500		DL11	Asynch Single Line Controller
YLB	310	176510		DL11	Asynch Single Line Controller
YLC	320	176520		DL11	Asynch Single Line Controller

1.4 Do you wish to modify this configuration information [Y or N]? <N> N

You should answer N to this question to include all of these devices in your configuration. If you answer Y to this question, then you will be asked individual questions on all hardware devices.

PART 2: DISK INFORMATION

Disk information supplied by AUTOCONFIGURE

PART 3: SYSTEM DEVICES

System Device information supplied by AUTOCONFIGURE

PART 4: CONFIGURE DMC-11s

PART 5: SOFTWARE CONFIGURATION

5.1 Do you wish to use the STANDARD SOFTWARE OPTIONS [Y or N] ? <Y>

Answer YES to this question to make the software options available. The standard options that you receive are given in Part 7. If you answer YES to this question, SYSGEN proceeds automatically through Part 12.

PART 6: ASSIGN DEVICE NUMBERS

The following single-line device assignments have been made:

Device Number	Controller-Number
3	DL11-1
4	DL11-2
5	DL11-3

PART 7: SOFTWARE OPTIONS

SEQUENTIAL DISK PROCESSOR support:	Included
JOURNAL support: 2 buffers	Included
INTERJOB COMMUNICATIONS support: With 16 communication channels and a 64-byte default ring buffer size	Included
EBCDIC-ASCII TRANSLATION TABLES support:	Not Included
LOADABLE or USER DRIVER SPACE support:	Not Included
EXECUTIVE DEBUGGING TOOL support:	Not Included
SPOOLING SUPPORT:	Not Included
MAPPED ROUTINES support:	Not Included
UCI TRANSLATION TABLES support:	Included
MOUNTABLE DATA BASE VOLUME SETS support:	Included
Total System Exec size:	59.40 K Bytes

PART 8: MEMORY BUFFER ALLOCATION

Default terminal RING BUFFER size:	64 Bytes Total
Total space allocated to RING BUFFERS:	1536 Bytes
Total number of 1K byte DISK-TAPE cache buffers:	33

PART 9: SYSTEM DATA STRUCTURES

Space allocated for DISK-MAP and BAD BLOCK TABLE:	448 Bytes
Space allocated to UCI TRANSLATION TABLE:	1024 Bytes
Space allocated to LOCK TABLE:	512 Bytes
Number of mountable DATA BASE VOLUME SETS	3

PART 10: JOB PARTITION DEFINITION

PARTITIONS are allocated in 1024-byte increments.

The following PARTITIONs have been defined:

JOURNAL system job	1K bytes
GARBAGE COLLECTOR JOB system job	1K bytes
Job 1 (to guarantee one 8K byte PARTITION)	8K bytes

Default partition size:	8K bytes
-------------------------	----------

Space remaining for PARTITION allocation:	18.10K bytes
---	--------------

The remainder of memory is assigned to the DYNAMIC PARTITION POOL

PART 11: DATA BASE PARAMETERS

WRITE CHECK after WRITE on disks:	Not Included
-----------------------------------	--------------

System default global characteristics are:

8-Bit Subscripts:	Yes
Journaling:	Yes
Collating sequence:	Numeric

PART 12: BASIC SYSTEM PARAMETERS

View buffer device protection:	Included
ZUSE command protection:	Included
LOG-IN SEQUENCE CHARACTERS:	Echoed
Default APPLICATION INTERRUPT key:	3 ((CTRL/C))
Default PROGRAMMER ABORT key:	25 ((CTRL/Y))
Time delay for POWER FAIL RESTART:	40 seconds
Time delay for TELEPHONE DISCONNECT:	15 seconds
Number of significant DIGITS for DIVISION:	20

12.9 Is the LINE FREQUENCY 60 HZ [Y or N] ? <Y>

By pressing the carriage return key, the default value of 60 hertz is selected. This is the correct value for the United States and Canada. You should enter N if you are connected to an electrical system in a country with a 50-hertz frequency.

12.10 Enter the 3-character Programmer Access Code (PAC) > ZZZ

This code is a password. You must give this code each time that you log in to DSM-11. Choose a code that is not obvious.

Please enter your initials> JSS

The system keeps a record of who generated this SYSGEN by storing your initials.

Enter comment (max. 200 chars.) > Test Autoconfiguration

Here you can enter a brief comment about this particular SYSGEN session.

The system global ^SYS has been built by SYSGEN.
^SYS is a reserved global and should not be altered.

If you wish to customize your new configuration by modifying:

- Terminal speed settings or other parameters
- Magnetic tape default format
- UCIs or data base VOLUME SETS
- TIED TERMINAL table
- UCI Translation Table
- Default GLOBAL CHARACTERISTICS/PLACEMENT
- Routine maps

then log in to the manager's UCI and type "D^SYSDEF".

Through ^SYSDEF you can have access to utilities that can do the modifications just described. You cannot log in at this time. You can log in once you build a start-up command file and start the configuration running. You are currently still running the Baseline System. Chapter 10 contains detailed information on the ^SYSDEF utilities.

```
You do not have a start-up command file.  
Do you wish to remain in the Baseline System: <N>
```

DSM-11 uses the start-up command file, in addition to the system configuration, when starting up the system. You can change (at a later date) this command file by using the utilities Define System Parameters, ^STUBLD, or when using the System Start-up utility, ^STU. Both of these utilities are described in Chapter 11. To establish a start-up command file for this configuration, answer NO to this question.

You can remain in the Baseline System by answering Y to this question. If you do so, remember that the UCI is MGR, and the PAC is ~~CTRL/X~~ ~~CTRL/X~~ ~~CTRL/X~~. The new PAC that you entered in question 12.10 applies only to the new configuration that you just created.

Begin defining a new start-up command file.

The following questions are the same as those asked by the system utility ^STUBLD.

```
Configuration ? <TEST>  
Apply patches to memory [Y/N] ? <N>  
Start up the Journal [Y/N] ? <N>  
Enable the Spool device (device 2) [Y/N] ? <N>  
Start the caretaker background job [Y/N] ? <Y>  
Enter printer number for system error messages <1>  
Automatic logging of DSM errors [Y/N] ? <N>
```

Mount additional disk volumes [Y or N] ? <N>
Make this the new start-up command file for configuration TEST [Y/N] ? <Y>

By pressing the carriage return, you take the default values for each of these questions. In this way, you quickly build a start-up command file.

Reconfiguring memory...
Memory reconfigured

The system takes about 20 seconds to reconfigure memory.

Now mounting volume set SYS in table slot S0

Volume 1 on DL1 has	10000 blocks	8951 available
Total in volume set	10000 blocks	8951 available

Building terminal control blocks...

Caretaker is now running as job number 2.

The caretaker job is a "background" job that is transparent to the user and does basic housekeeping functions for the system.

DSM-11 Version 3 TEST is now up and running!
Exit

The autoconfiguration is now complete, and you have a running system with the default values that have been indicated. By pressing **RET**, you get the DSM-11 log-in prompt. You can now log in and use the system. You can log in to the System Manager's UCI by typing MGR: and then ZZZ for the PAC (which is the password as indicated in the answer to the earlier question about the PAC). Now is the time to run ^SYSDEF to further define and modify your configuration.

```
RET
DSM-11 Version 3 Device 1 UCI: MGR:ZZZ RET
>
```


Part 2: Introduction to DSM-11

Chapter 2

Overview of DSM-11

This chapter introduces the basic building blocks of DSM-11: the language, the data base, the operating system modules, the system and library utilities, and the hardware.

2.1 System Overview

DSM-11 is a multiuser, time-sharing system that runs on the PDP-11 computer. Basic system features include:

- Use of Digital's version of the ANSI standard MUMPS language
- Variable-length data elements and variable-length logical records
- Random access of data through variables
- Utilities for system management and application program development
- Support for a variety of terminal and peripheral devices

2.1.1 The Language

The DSM-11 language is a high-level, interpretive language.

The language has many capabilities. Its basic orientation is procedural. Its capabilities are primarily directed to the processing of variable-length string data. It also provides standard algebraic and Boolean operations.

DSM-11 allows you to write a routine as well as to debug, edit, run, and modify it in one interactive session at the terminal. This single-session capability reduces problem-solving time, the computer time to check the routine, and, most importantly, the time required to obtain a final running application.

The language interpreter has two main user modes: Programmer Mode and Application Mode. In Programmer Mode, you can execute commands and create, modify, debug, and store routines. In Application Mode, you can execute routines (which are applications).

2.1.2 System Capacity

The DSM-11 system software requires between 48K and 72K bytes of memory. The exact requirement depends on the hardware and software configuration at a particular site. During system generation, DSM-11 subdivides the remaining memory into user partitions and input/output buffers.

A partition holds users' routines, as well as local data and partition overhead information. DSM-11 systems can have as many as 63 partitions depending on partition size and available memory. The System Manager can determine partition sizes either at system-generation time or at log-in time. The recommended (and default) size for a partition is 8K bytes (to run system utilities), but it is not necessary for all partitions to be the same size. When you access the system and do not specify the partition size, DSM-11 assigns the default size.

2.1.3 Data Access

The data access capabilities of DSM-11 allow you to reference data symbolically through *variables*. A variable represents a variable-length character string that is either a numeric value or an alphanumeric string.

DSM-11 allows two types of data: local data and disk data; and thus two types of variables: local variables and global variables.

Local data is defined solely for the routine or routines residing in the partition. This type of data is not intended for permanent storage, but only for temporary use. DSM-11 allocates storage space for this data only as needed.

All data stored on disk are referenced symbolically; their names are similar to those of local variable names in a routine. Data stored on disk comprise an external structure of arrays. This structure provides a common data base available to all routines within a user class or group of user classes. The arrays that make up this external structure are called global arrays. Each global array is identified by a unique name. Elements of an array are comprised of records stored on disk. All elements are referenced by subscripts.

2.1.4 System Access

The DSM-11 protection scheme controls the availability of routines and global data to users. This protection scheme divides users into user classes. There can be as many as thirty classes of users for each volume set defined within the system. Each user class has access only to those routines residing within that class, and to the library utility routines. All user classes can have access to global arrays.

To access a user class, you must have a password. The password is called a User Class Identifier (UCI). If you know a UCI, you can enter DSM-11 in Application Mode. Application Mode allows you to run routines that read or write global data but cannot write new routines or modify existing ones. If you have an additional password called the Programmer Access Code (PAC), you can enter DSM-11 in Programmer Mode. Programmer Mode allows you to run and modify existing routines or write new routines. The System Manager assigns the PAC and UCI codes at system generation time or with the Modify Basic Parameters (^MBP) Utility.

Some applications do not require users to have programming access to the system. In such cases, DSM-11 allows you to "tie" a terminal to a particular routine. DSM-11 invokes this routine whenever you access that terminal. In this "tied" mode, you bypass the DSM-11 login procedure and immediately begin executing an application routine.

2.2 System Software

The DSM-11 system software consists of:

- The DSM-11 operating system
- The system utility routines and global arrays
- The library routines and global arrays

The following sections describe each element of the DSM-11 system software.

2.2.1 Operating System

The operating system contains all the software necessary to operate DSM-11 on the PDP-11. The software is completely memory-resident, and consists of four operating system functions. All four functions constantly interact with each other. For purposes of a general overview, however, each one is described as a separate function.

Executive

The executive is a system supervisor that controls the time-sharing operations of the DSM-11 system by assigning each new user the next available partition. The executive passes control from one user to another optimizing the use of the central processor.

I/O Monitor

The I/O monitor supervises peripheral devices such as magnetic tape, line printer(s), and terminals. It initiates and processes input/output activity through its interrupt handlers. The I/O monitor also handles terminal I/O by communicating with the interpreter through buffers. By supervising the filling and emptying of these buffers, the monitor can overlap output with the processing of the routine that requested I/O.

Language Interpreter

The language interpreter implements and controls routines written in the DSM-11 language. Specifically, it examines and analyzes each language statement and executes the specified operation. The interpreter also files and loads routines to and from the disk. The interpreter is particularly useful in three respects:

- You do not need to compile or assemble routines.
- The interpreter performs error checking during program execution, and reports all language errors at the terminal.
- You can perform debugging at the terminal.

Data Base Supervisor

The data base supervisor controls the logical and physical allocation of disk structures that form the DSM-11 data base. The disk structures that store the data base are called *blocks*. Blocks store data records that you reference symbolically through global variables. Subscripted global variables form arrays whose data records are stored in blocks in a tree-like configuration.

The data base supervisor allots only the space necessary to store the elements of the global array that are defined. When a global array no longer requires the space (or part of the space) allocated to it, the data base supervisor recovers the space and returns it to the system's pool of available disk space. (Refer to Chapter 13 for detailed information about the data base supervisor.)

2.2.2 System Utility Routines

The DSM-11 system utility routines are a part of the DSM-11 software package. The utility routines are written in the DSM-11 language. They are provided to help the application programmer and the System Manager develop and maintain the software and data for their application.

You use the DSM-11 utility routines to maintain and service the system. These routines are stored on the disk under control of the manager's UCI (UCI 1), and are accessible only to individuals who know the manager's UCI code. Their functions fall into the following categories:

- System backup and global journaling
- System parameter modification
- Miscellaneous utility operations

System backup utilities allow you to create physical backup of globals from disk on any mass storage media supported by DSM-11. These utilities also format and initialize disk packs as DSM-11 volumes.

The journaling utilities provide additional backup by allowing you to save and restore all SET and KILL operations you perform on specified globals. System parameter change utilities allow you to modify the system configuration. The miscellaneous utilities provide services for system management functions that are frequently performed.

2.2.3 Library Utility Routines

The library utility routines are part of the DSM-11 software package, and provide general services for all users regardless of UCI. These services include utilities that allow you to save and restore individual routines and globals. These routines also reside under the manager's UCI, but use a naming convention that distinguishes them from system utilities. Names of library utility routines begin with the percent (%) character. Their functions fall into the following categories:

- Routine related
- Global related
- Miscellaneous

2.2.4 Library Utility Globals

Some of these routines hold data, and others hold MUMPS command lines that perform routine editing functions.

2.2.5 System-Configuration Global

The System-Configuration global (^SYS) contains the necessary information to describe the hardware and software environment for the DSM-11 system. Using the system generation utility package, you can construct the configuration to suit your application in terms of software options, hardware components, program partitions, and user classes. You can also generate alternate configurations that enable your system to run with different hardware and software features. Each time you start your system, you can choose the configuration under which you want to run.

2.3 System Hardware

The hardware for DSM-11 is based on the PDP-11 central processing unit and its associated peripheral devices. The total hardware configuration for a DSM-11 system can vary. Important factors to consider when configuring the hardware for a system are:

- Memory size requirements
- Types and numbers of terminals
- Capacity and number of disk and tape drives

The system is distributed on 9-track magnetic tape, 5 1/4-inch floppy diskettes, and disk cartridges. One of the supported disk types must be available for use as a system disk. The system disk is the disk where the DSM-11 operating system is stored.

**Part 3: Programming and Using the DSM-11
System**

Chapter 3

Accessing DSM-11

This chapter provides information about operating terminals and accessing the DSM-11 system.

3.1 Using the Terminal

You use a *terminal* to communicate with the PDP-11 computer and the DSM-11 operating system. A terminal is a device that looks, and sometimes behaves, like a typewriter.

You type commands on the terminal keyboard to tell the computer what you want it to do. The computer either responds as instructed or displays a message telling you that it does not understand your request.

DSM-11 supports several different terminals. These fall into two general categories:

- Hard-copy Terminals
- Video Display Terminals

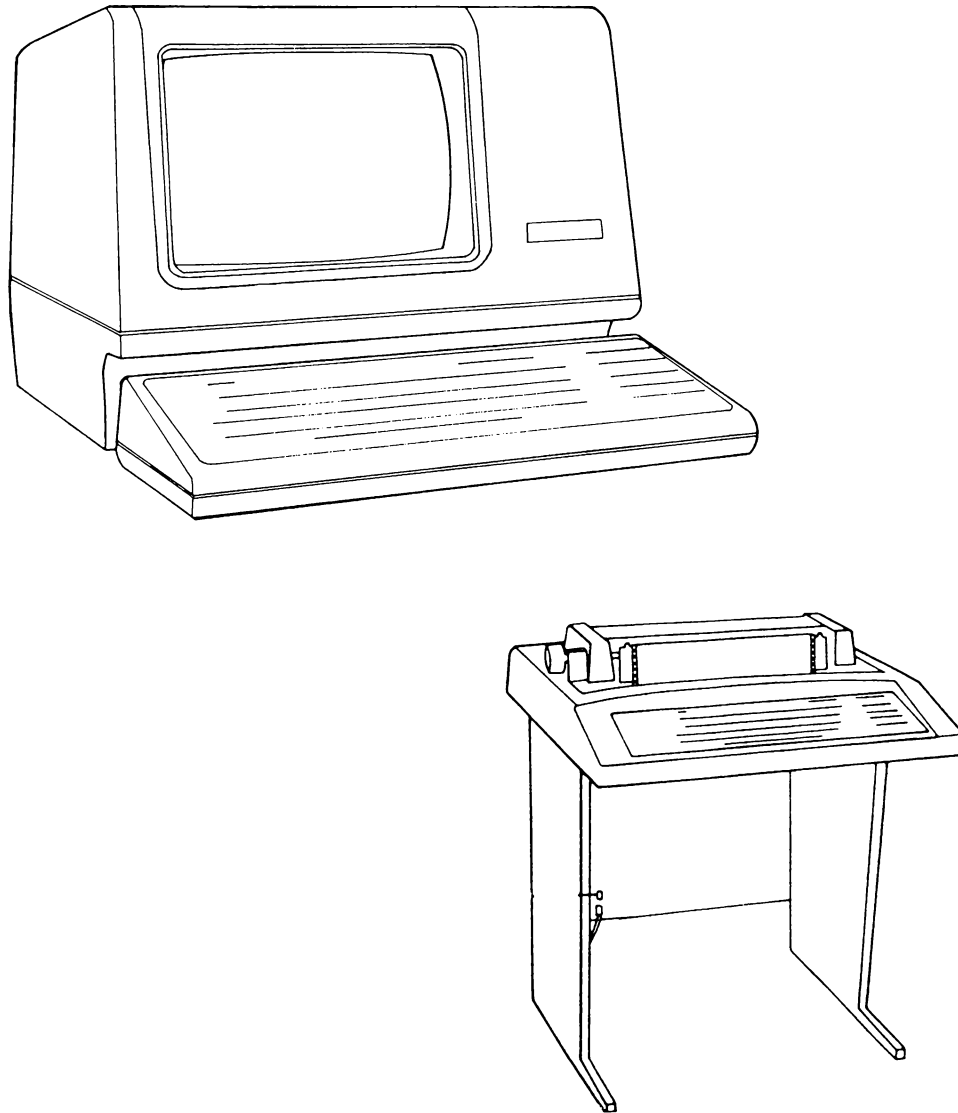
Hard-copy terminals print on continuous forms of paper. This type of terminal is particularly useful when you want a permanent record of your terminal session. The LA120 is an example of a hard-copy terminal.

Video display terminals print on a television-like screen called a Cathode Ray Tube (CRT). Generally, you cannot get a permanent record of your terminal

session from a video display terminal. However, in many cases you will find that a video display terminal offers a faster and more effective work environment than a hard-copy device. The VT100 and the VT52 are common video display terminals used with DSM-11.

Figure 3-1 illustrates both hard-copy and video display terminals.

Figure 3-1: Common DSM-11 Terminals



MR-S-3483-83

A terminal has a keyboard that resembles a conventional typewriter. The keyboard contains alphabetic and numeric characters. Information that you type on the terminal keyboard is sent to the computer for processing.

3.1.1 Function Keys

Most terminals have keys that allow you to send special signals to the computer. These keys are called function keys.

Figure 3-2 shows the keyboards of two different terminals: the VT100 and the VT52. Arrows point to the most commonly used function keys. The symbols used in these diagrams are used throughout this text as a shorthand notation to indicate that you should press these keys.

Note that function keys are not always in the same position on different types of terminals, so check the keyboard layout each time you use a terminal for the first time.

Figure 3-2: The VT100 and VT52 keyboards

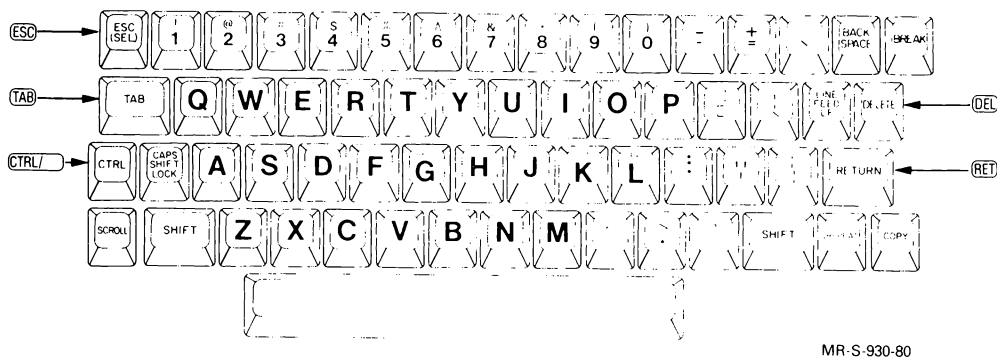
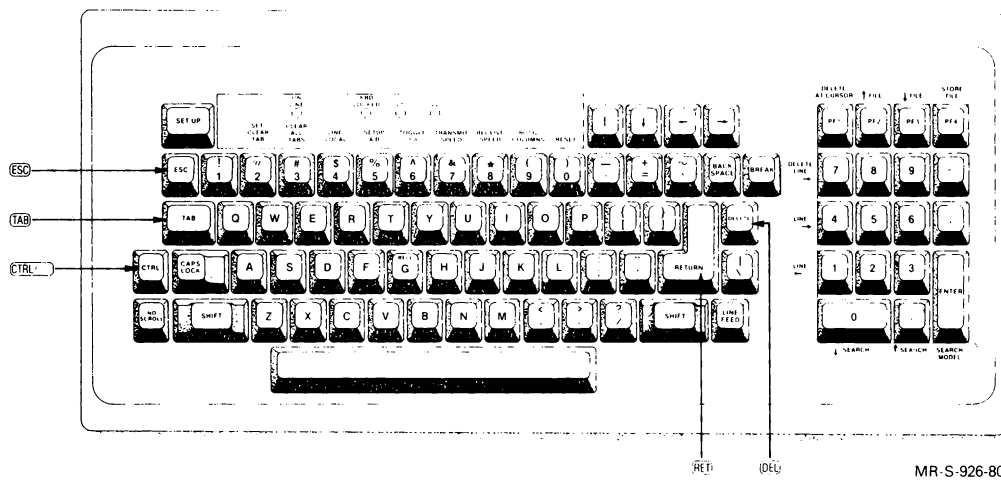


Table 3-1 describes the important function keys that appear on your terminal.

Table 3-1: Function Keys

Function Key	Description
␣ (CTRL)	<p>Is part of many 2-key combinations that perform a variety of defined functions.</p> <p>The control characters used most often are described in Table 3-2.</p>
␣ (DEL)	<p>Deletes the last character entered on a video terminal and backspace over it. On a hard-copy terminal, DELETE instructs the system to ignore the last character typed, and prints a backslash (\).</p>
␣ (ESC)	<p>Terminates input lines from the terminal if escape sequence processing is not enabled. For more information about escape sequence processing, refer to Table 6-1.</p>
␣ (RET) (carriage return)	<p>Transmits the current line to the system for processing and advances the cursor to the leftmost position of the next line.</p>
␣ (TAB)	<p>Moves the printing position on the terminal to the next tab stop. Each tab position represents eight spaces.</p>

Terminal keyboards do not always represent function characters in the same way. The following are some common variations:

- The ESCAPE key can be labeled ALT MODE or ESC (SEL)
- The circumflex (^) can be represented as an uparrow
- The underscore (__) can be represented as a backarrow
- The DELETE key can be labeled RUBOUT, or DEL
- The RETURN key can be labeled CR for carriage return

If you do not have a Digital terminal, consult your terminal user's guide for information about the representation of function characters.

3.1.2 Control Characters

A *control character* is a character sequence entered by pressing a letter key and the <CTRL> key, simultaneously. Control characters are sometimes displayed on the terminal by a circumflex (^) followed by the selected letter. Other control characters such, as CTRL/L, are not displayed. When you enter a control character, the system performs one of several tasks. Table 3-2 summarizes the functions of the most commonly used control characters.

Table 3-2: Control Characters

Control Character	Function
␣	When in Programmer Mode, turns on the MUMPS debugger. If a routine is running at the time ␣ is struck, the routine will be interrupted by a break. Equivalent to ZBREAK ON followed by BREAK. See Section 4.8 for more information on the debugger.
␣	When enabled, causes DSM-11 to generate an <INRPT> error; depending on how error handling is set up, execution of a routine may continue. See Section 4.10 for a discussion of error handling. See the discussion of ␣ that follows for unconditional command execution interruption. Refer to Section 4.7 for a complete description of DSM-11 responses to ␣.
	Initiates the log-in sequence, if a terminal is not set up for baud rate detection (autobaud).
␣	When you type ␣ at the console terminal, the entire system's operation is interrupted, and the system debugger, DSM-11 XDT, is started (assuming XDT is included in the system's configuration.) Type G to exit from the system debugger. Refer to the <i>DSM-11 XDT Reference Manual</i> for additional information.
␣	Duplicates the function of the TAB key.
␣	Causes a form feed.
␣	Duplicates the function of the RETURN key.
␣	Suppresses display of terminal output the first time you press it, and resumes output to the terminal (at a further point in the display) the second time you press it.
␣	When you type it at the console terminal of a processor equipped with a remote diagnostic console, enters console Programmer Mode. To exit from console Programmer Mode, type "Z".
␣	Restarts terminal output that was stopped by ␣.
␣	Retypes the current line and leaves the cursor positioned at the end of the line. This is particularly useful on a hard-copy terminal.
␣	Stops terminal output until you press ␣. No data is lost.
␣	Discards the current input line.
␣	In Programmer Mode causes an <ABORT> error, and interrupts command execution. Ignored in Run Mode. See Section 4.7 for more discussion of ␣ and ␣ interruption.

3.2 Logging in DSM-11

The procedure you use to access DSM-11 from the terminal is called *login*. The log-in procedure identifies you to the DSM-11 operating system. You let the system know you want to log in by pressing a carriage return or `CTRL/C`, if the terminal is not autobauded. For an autobauded terminal, see Section 3.2.8. The system responds by asking you to identify yourself:

```
DSM-11 Version n.n Device nnn UCI:
```

If you do not respond to the log-in prompt within 30 seconds, DSM-11 displays:

```
TIME-OUT
```

and prints the log-out message. You must then reinitiate the log-in sequence.

In general, login involves entering two or three access parameters:

- The User Class Identifier Code (UCI)
- The Programmer Access Code (PAC) or routine name
- The partition size (in bytes)

3.2.1 The User Class Identifier Code

DSM-11 allows 30 classes of system users in each volume set to be defined. The password necessary to access one of these user classes is called the *User Class Identifier* (UCI). The UCI can be any combination of three alphabetic characters.

NOTE

You must have a UCI before you can log into the system. UCIs are set up by the System Manager or whoever authorizes the use of your system.

3.2.2 The Programmer Access Code

A privileged DSM-11 user is given the system *Programmer Access Code* in addition to a UCI. There is only one PAC for a DSM-11 system. The PAC can be a combination of any three characters (including control characters).

3.2.3 Partition Size In Bytes

Partition size in bytes is an optional parameter. If you do not specify partition size, DSM-11 assigns a partition of the default size specified during SYSGEN. (See Chapter 10 for more information on SYSGEN.) The SYSGEN default partition size is 8K bytes.

3.2.4 DSM-11 Modes

DSM-11 has several modes:

- Application Mode - gives you access to a specific routine (an application).
- Programmer Mode - allows you to write routines; includes two other modes described in Section 4.2:
 - Direct Mode.
 - Indirect Mode.
- Break Mode - is used for debugging purposes. See Section 4.8.
- I/O Modes - accessed by the OPEN or USE command, refers to input and output modes through a specific device. See Chapter 6.

The two user modes that you can access when logging in are:

- Application Mode
- Programmer Mode

By itself, the UCI lets you operate in Application Mode.

In Programmer Mode, you can execute commands, and create, modify, debug, and store routines. See Sections 4.2 for more discussion of Direct, Indirect, and Programmer Modes. In combination with a UCI, the PAC allows you to operate DSM-11 in Programmer Mode.

Sections 3.2.5 through 3.2.8 describe the log-in procedure for each of these user modes and for tied terminals.

3.2.5 Logging In DSM-11 Application Mode

After DSM-11 displays the log-in prompt, select DSM-11 Application Mode by typing your UCI, followed by a colon (:), a routine name, and an optional colon followed by a partition size in bytes. This type of login permits access to the routines and globals listed in the routine and global directories assigned to your UCI. If you are limited to running selected routines, you will only be permitted to know a specific UCI and the names of the routines that you can run.

To let the system know you want to log in, press the carriage return. (If your line is not an autobaud line, then press a carriage return or `CTRL/C`.)

Then respond to UCI: by typing your UCI followed by a colon (:), and a routine name. Terminate the line with a carriage return. For example:

```
DSM-11 Version n.n Device #64 UCI:MGR:DATA:8192 RET
```

This login sequence gives you access to DSM-11, and executes the routine DATA. Note that there are no spaces between the UCI and the DSM-11 routine name.

3.2.6 Logging In DSM-11 In Programmer Mode

After DSM-11 displays the log-in prompt, select DSM-11 Programmer Mode by typing your UCI followed by a colon (:), and the Programmer Access Code (PAC). For example, the following log-in sequence grants you access to the UCI "DEM" in Programmer Mode:

```
RET
```

```
DSM-11 Version n.n Device #64 UCI:DEM:XXX RET
```

```
>
```

(If your line is not an autobaud line, then press a carriage return or `CTRL/C`.) The system responds by displaying the DSM-11 Programmer Mode prompting symbol, the right angle bracket (>). This prompt indicates that the system is ready to accept commands to create, edit, delete, or run DSM-11 routines.

3.2.7 Logging In At A Tied Terminal

Tied terminals force the automatic and protected start-up of a DSM-11 application. Tied terminals restrict user access to preselected routines and globals. To log in at a tied terminal, type:

```
RET
```

(If your line is not an autobaud line, then press a carriage return or CTRL/C.)

3.2.8 Logging In Through A Modem With Autobauding

DSM-11 has an option for automatically detecting the baud rate of your terminal when you log in. Baud rate refers to the speed at which information is transmitted over the line between your terminal and the computer. Different terminals can have different baud rates, or the same terminal can be set up for different baud rates.

Autobauding is especially useful for terminals connected with modems. If your system is using autobauding, you must log in with a carriage return. You may need to press the carriage return more than once. Allow a brief pause (one or two seconds) between each time you press the key, for example:

```
RET  
RET
```

The system responds with:

```
DSM-11 Version n.n Device #64 UCI:
```

You should respond appropriately depending on whether you are logging in in Programmer Mode or Application Mode.

3.3 Logging out of DSM-11

The log-out procedure ends a terminal session. DSM-11 provides three ways to log out. Each procedure is associated with one of the log-in procedures described in Sections 3.2.4 through 3.2.8.

1. If you are in Programmer Mode, that is, if you have logged in with a PAC, type HALT in response to the Programmer Mode prompt:

> HALT RET

The system displays an EXIT message, and logs out the terminal.

NOTE

If a routine executes a HALT command, DSM-11 also logs out the terminal.

2. If you are in Application Mode, that is, if you have with a routine name or through a tied terminal, logout is accomplished in either of the following ways:
 - If the application has enabled `CTRL/C` recognition with a BREAK 1 command, type `CTRL/C`, which generates an <INRPT> error. If this error is not trapped by the error-handling routines of the application, you will be logged out. Refer to Section 4.8 and the *DSM-11 Language Reference Manual* for more information about the BREAK command.
 - If the application has not enabled `CTRL/C` recognition, the routine is unbreakable; and you must wait for the program to terminate normally when it encounters a QUIT or HALT command. After normal termination, the system displays the EXIT message, and logs out the terminal.

NOTE

If you shut off your terminal while you are logged in, the system does not log you out.

Chapter 4

Using DSM-11

This chapter describes how to use DSM-11 to develop your application routines. It also provides information about DSM-11 directories, and describes useful programming techniques to process errors. It describes:

- Entering DSM-11 commands
- Creating routines
- Saving and loading routines
- Deleting and renaming routines
- Storing routines on sequential files
- Editing routines
- Starting and stopping routines
- Debugging routines
- Using directories
- Error processing

To use this chapter effectively, you should be familiar with the syntax of the DSM-11 language as described in the *DSM-11 Language Reference Manual*.

4.1 Entering DSM-11 Commands and Routines

After you have logged in to DSM-11 in Programmer Mode, the DSM-11 interpreter is ready to accept commands from the terminal keyboard. You must adhere to strict DSM-11 syntax when you enter commands in Programmer Mode.

The following sections describe the aspects of DSM-11 syntax that you need to know to enter commands and create routines. The *DSM-11 Language Reference Manual* describes all DSM-11 commands in detail.

4.1.1 Abbreviating DSM-11 Commands

When you use a DSM-11 command, you can either enter its full spelling or an abbreviation of the command word. Commands that begin with any character except "Z" can be abbreviated to their first character. Commands that begin with "Z" can be abbreviated to their first two characters. You cannot use any intermediate abbreviations. For example, the following two DSM-11 statements are valid:

```
>GOTO PARA
```

```
>G PARA
```

However, the following statement is not:

```
>GOT PARA
```

4.1.2 Using Uppercase And Lowercase Characters

You can use uppercase or lowercase characters in DSM-11 language elements, variable names, routine names, and line labels. DSM-11 considers commands, functions, and special variables written in uppercase or lowercase letters as equivalent. For example, DSM-11 considers the following three commands identical:

```
WRITE "HELLO"
```

```
write "HELLO"
```

```
WrITe "HELLO"
```

However, DSM-11 does not consider uppercase and lowercase variable names, routine names, and line labels as equivalent. For example, the following statements are not equivalent.

```
SET XX="HELLO"
```

```
SET Xx="HELLO"
```

```
SET xx="HELLO"
```

For more information on using uppercase and lowercase letters in language elements, variable names, routine names, and line labels, refer to the *DSM-11 Language Reference Manual*.

You can use mixed lowercase and uppercase characters in the following:

String Literals

String literals are strings of characters enclosed in quotation marks, embedded in DSM-11 code.

String Data

String data is a contiguous series of characters that is considered a single data entry. String data can consist of both uppercase and lowercase characters. The response to an interactive READ command is an example of string data. For instance, if an application prompts for a name, it can be entered with mixed characters:

```
LAST NAME> Webster
```

Comments

Comments are brief explanations of programming action included in routine code. A comment is preceded by a semicolon (;). Unlike string literals, comments are not enclosed in quotation marks.

4.1.3 DSM-11 Line Structure

DSM-11 code is organized into lines. Each line contains one or more statements and/or a comment, and ends with a carriage return.

The DSM-11 interpreter recognizes two types of lines:

- The command line
- The routine line

A *command line* consists of one or more DSM-11 statements. The DSM-11 interpreter translates and executes a command line as soon as you terminate the line with a carriage return.

The DSM-11 interpreter recognizes a command line by its format. The following are examples of DSM-11 command lines:

```
>DO STAT  
  
>ZINSERT "TITLE ;DEMOGRAPHIC DATA SORT ROUTINE":+0  
  
>XECUTE:$D(A(I))=1 A(I)
```

A *routine line* consists of one or more statements preceded by a tab, or a line label and a tab, and entered for later execution. After you press the carriage return at the end of the line, the DSM-11 interpreter stores the line and waits for your next command. The following are examples of routine lines:

```
OUTPUT(TAB)F I=1:1:6 W ?20,"1",?25,B(0),!!  
  
(TAB)S A(1)="W $(27,89,35,42),S,EL R B(I)"  
  
B7(TAB)S Z="Zip Code" W Z ;correct zip is vital
```

- ✎ A collection of routine lines that you enter and run as a unit is called a *routine*. When you run a routine (by typing the DO command, for example), the DSM-11 interpreter executes the routine lines in the order in which they are stored (unless you use statements such as GOTO that alter the flow of control).

DSM-11 allows more than one space to be inserted between commands in a line. For example, the following statement is an acceptable command line for DSM-11:

```
0(SP)3(SP)(SP)(SP)U(SP)3(SP)(SP)(SP)ZP(SP)(SP)(SP)C(SP)3
```

You still must have one space between the command and its argument, or two spaces following commands with no arguments. See the *DSM-11 Language Reference Guide* for a more detailed discussion.

4.2 Creating Routines

The DSM-11 language can be used in two ways:

1. In *Direct Mode*, in which all commands entered at the terminal are executed (interpreted) immediately. Programmer Mode includes the Direct Mode; references to Programmer Mode usually mean the Direct Mode of Programmer Mode.

2. Through *routines* made up of multiple lines of DSM-11 code. Routines can be stored and used in future DSM sessions. When you are typing in the routine lines, they are not executed immediately. You are in *Indirect Mode*. Programmer Mode also includes Indirect Mode.

Introduction to DSM, included in your documentation set, provides a tutorial in developing routines. The following sections summarize these subjects for the purpose of reference.

4.2.1 Direct Mode

When you enter DSM-11 in Programmer Mode, you may want to use the language in Direct Mode. Direct Mode is useful for learning how the language works, and for getting quick answers to computational problems. It is also useful for testing out command lines before putting them into a routine.

In Direct Mode, you enter commands after the Programmer Mode prompt (>). DSM-11 executes the command immediately. The following example shows your input to DSM-11 in Direct Mode and the interpreter's response:

```
> WRITE "Hello"  
Hello
```

4.2.2 Entering Lines In The Routine Buffer

Direct Mode is limited because you must wait for DSM-11's response after each line of code. For more sophisticated programming tasks, use routines instead of individual lines of code. You signal to DSM-11 that a line of code belongs to a routine by starting the line with a `(TAB)` or a label followed by a `(TAB)`. When you are entering routine lines in this manner, you are in Indirect Mode. The routine lines are *not* executed when you enter them.

The following are examples of routine lines:

```
> START(TAB)WRITE "Powers of 2",!  
> (TAB)SET X=1  
> (TAB)FOR I=1:1:10 SET X=X*2 WRITE X, " "
```

The routine lines shown above make up a routine that calculates and displays on the terminal the first 10 powers of 2. You execute these lines by entering the DO command, using the first label in the routine as the point at which execution begins:


```
>DO START
```

DSM-11 responds by displaying the output of the routine:

```
Powers of 2  
2 4 8 16 32 64 128 256 512 1024
```

Because they start with a `(TAB)` or a label and a `(TAB)`, the lines stay in your *routine buffer*. You can execute them again by repeating the DO command.

4.2.3 Using The ZPRINT And ZREMOVE Commands

You can display the lines in your routine buffer by issuing the ZPRINT command. The ZPRINT command leaves the pointer (showing your location in the buffer) at the end of the routine.

If you specify a line label with ZPRINT, DSM-11 displays the line specified and positions the pointer after that line.

The ZREMOVE command deletes the contents of the routine buffer. If you specify a line label with ZREMOVE, that line is deleted.

After you delete a line with ZREMOVE, you can replace the line by entering a new line starting with `(TAB)` or with a label and `(TAB)`. The following lines show how ZREMOVE is used in deleting and replacing a routine line:

```
>ZREMOVE START  
>START TAB WRITE "Powers of two"
```

To insert a line without removing one, you first move the pointer to the position after a line by using ZPRINT. Then, you enter the new line, as shown in the following example:

```
>ZPRINT START  
START WRITE "Powers of two"  
> TAB WRITE !,"from 1 to 10",!
```

To insert a line at the beginning of the routine buffer (the area in which your routine lines are kept), type ZPRINT +0 to position the pointer before the first line in the buffer. Then enter the new line.

4.3 Saving and Loading Routines

If you want to save a DSM-11 routine for later use, you can store it in the *routine directory*.

You must restore the routine to your routine buffer if you want to add more lines, remove lines, or edit the routine with the DSM-11 editor (described in Section 4.6.2).

To display a list of the routines in your routine directory, invoke the `^%RD` (Routine Directory) utility, as follows:

```
>DO ^ZRD

      Routine Directory  19-Jan-83
            of JSS      3:06

      ABC  DEF  XXX

      3 routines

>
```

4.3.1 Saving A Routine In A Routine Directory

The `ZSAVE` command places the routine currently in your routine buffer into your DSM-11 routine directory.

The argument of the `ZSAVE` command is the name under which you want to save the routine. For example, the following command stores the contents of the routine buffer in the routine directory under the name `ABC`:

```
>ZSAVE ABC
```

You use the same name when you load the routine back into your routine buffer, as described in the next section. You also use the routine name when you execute the routine from the routine directory. To execute a routine that is in your routine directory, you issue the `DO` command with the name of the routine, preceded by a circumflex (`^`), as shown in the following example:

```
>DO ^ABC

Powers of two
from 1 to 10
2 4 8 16 32 64 128 256 512 1024
```

If you try to `ZSAVE` an unnamed routine (that is, if you enter `ZSAVE` without an argument while code is in your routine buffer), DSM-11 gives you a `<NOPGM>` error message.

A routine in your routine buffer may already have a name because it was already stored and then loaded back into the routine buffer. You determine

the name (if any) of the current routine by issuing the following WRITE command:

```
>W $TEXT(+0)
```

The function \$TEXT(+0) always contains the string DSM-11 uses to name the current routine.

4.3.2 Loading A Routine From The Routine Directory

To restore a routine to the routine buffer, enter the ZLOAD command with the name of the routine as the argument:

```
>ZLOAD ABC
```

ZLOAD loads the routine from the routine directory which is stored on the disk. You can then execute the routine using a DO statement. The routine can be recovered from the disk and executed in one statement by typing:

```
>DO ^ABC
```

Notice that, if routine ^ABC is in the mapped routine area, ZLOAD recovers the routine from the disk; while the DO ^ABC statement recovers the routine from the mapped routine area in memory. The mapped routine area is described in more detail in Section 15.1.

4.4 Deleting and Renaming Routines

As described in Section 4.2.3, the ZREMOVE command deletes lines from your routine buffer. In conjunction with other commands, the ZREMOVE command can be used to delete routines stored in your routine directory, and to rename routines. The following sections describe these uses of ZREMOVE.

4.4.1 Deleting A Stored Routine

To delete a routine stored in your routine directory, first enter the ZREMOVE command to clear your routine buffer. Then, enter the ZSAVE command with the name of the routine that you want to delete as its argument. This sequence saves an empty routine buffer under the same name as the routine that you want to delete. After you delete a routine from the routine directory, DSM-11 also removes that routine's name from the list displayed by ^%RD.

The following command sequence deletes from disk a routine called BOOKSTAT that you once saved:

```
>ZREMOVE  
>ZSAVE BOOKSTAT
```

4.4.2 Renaming Routines

The procedure for renaming a routine stored on disk is similar to the procedure for deleting a routine stored on disk. First, use the ZLOAD command to load the routine into your routine buffer. Then, refile the routine using the ZSAVE command with the new name as its argument. This procedure produces two copies of the same routine in the routine directory.

To keep only a single copy of the routine under the new name, you must delete the routine under its original name. You do this by saving an empty routine buffer using the old routine name. The following example shows how to change the name of the routine BOOKSTAT to BKCHK, and then delete the original version of the routine:

```
>ZL BOOKSTAT  
>ZS BKCHK  
>ZR  
>ZS BOOKSTAT
```

4.5 Using Sequential Files to Store Routines

In addition to storing routines in a routine directory for execution, you can write routines to a sequential file for later printing or backup, as described in the following sections. However, the DSM-11 utilities ^%RS (Routine Save) and ^%RR (Routine Restore) move routines to and from a sequential storage medium, so you need not execute the steps in Sections 4.5.1 or 4.5.2 manually.

NOTE

Remember that you cannot call (use the DO command on) a routine from a sequential file.

Files stored on a magnetic tape, a DECTape II, a disk cart accessed by the SDP (Sequential Disk Processor), or an RX01/RX02 diskette are sequential files.

4.5.1 Writing A Routine Onto A Sequential File

To write a routine to a sequential file, follow these steps:

1. Issue the OPEN command to gain ownership of the device.
2. Issue the USE command to make the device current.
3. Issue the ZPRINT command followed by two spaces to write the routine.

The following command sequence writes the routine currently in your routine buffer to an SDP file.

```
>0 59:(0:1600:"DL1") U 59 ZP  W ! C 59
```

Note the two blanks after the ZP command in this command sequence. Two blanks are required because the ZP command has no argument.

The command sequence refers to byte 0 in block 1600 on disk DL1 as the starting point for the file. The SDP unit number is 59. Refer to Section 6.4 for more discussion about determining the correct format and block locations when using the SDP.

You can also invoke the DSM routine $\wedge\%RS$ to save a routine in a sequential file.

4.5.2 Loading A Routine From A Sequential File

To load a routine from a sequential file manually, use the following procedure:

1. Issue the OPEN command to gain ownership of the device.
2. Issue the USE command to make the device current.
3. Enter an argumentless ZLOAD command to load the first routine in the file into your routine buffer. ZLOAD reads lines into your buffer until it encounters a null line or an end-of-file.

For example:

```
>0 59:(0:1600:"DL1") U 59 ZL
```

If the routine you want to load is not the first record of the file, (that is, in the first physical position), use multiple ZLOAD commands separated by two spaces to load the desired routine. The number of ZLOAD commands you use should match the position of the routine in the file. For example, two

ZLOADs loads the second routine in the file; three ZLOADs loads the third, and so forth.

The DSM-11 utility ^%RR (RESTORE) loads a routine from a sequential file.

4.6 Using Editors

DSM-11 has several methods of editing routines and data.

1. Programmer Mode Editing
2. DSM-11 Editor
3. EDI Editor
4. Global Editor

Programmer Mode editing, the DSM-11 editor, and the EDI editor are useful for editing text and routines. The global editor is useful for editing global variables.

4.6.1 Programmer Mode Editing

The DSM-11 language has two commands that allow you to edit routines directly:

- ZINSERT
- ZREMOVE

ZINSERT and ZREMOVE allow you to manipulate DSM-11 routines in your partition by directly invoking system commands.

ZINSERT inserts new lines into a routine at the point you specify in the argument of the ZINSERT command.

ZREMOVE without an argument deletes the current routine in your partition. ZREMOVE with an argument deletes the specified line or lines from the current routine. Section 4.2.3 describes how to use ZREMOVE. The *DSM-11 Language Reference Manual* describes ZINSERT and ZREMOVE in detail, and provides examples of their editing capabilities. *Introduction to DSM* also provides step-by-step examples of editing with the ZREMOVE command.

4.6.2 DSM-11 Editor

DSM-11 has an interactive text editor that help you to edit the code in your routine buffer. The DSM-11 Editor can change text on one line or in a range of lines.

This editor has a very terse syntax, which makes it ideal for low baud rate terminals. The editor displays a sequence of question-like prompts. At each prompt, you enter the information required to do your editing task. If you enter a question mark (?) at an editor prompt, the editor displays a list of your options.

Before using the DSM-11 Editor, bring the routine that you want to edit into the routine buffer (if it is not already there). The following command invokes the routine editor:

```
>XECUTE ^Z
```

The abbreviated form is:

```
>X ^Z
```

The editor's initial response to this command is:

```
LINE>
```

This response indicates that you have entered the DSM-11 editor and that the editor is waiting for you to respond to its first query. To exit the editor, simply type a period (.).

The DSM-11 Editor displays seven prompts. Appendix C describes each prompt in detail. See *Introduction to DSM* for tutorial information on using the DSM-11 Editor.

4.6.3 EDI Editor (^%EDI)

The EDI Editor is a line editor that can be used to edit a routine that is currently in the routine buffer. To use the editor, type:

```
>X ^ZEDI
```

This editor is based on a series of commands such as ADD, BEGIN, CHANGE, FIND, and SAVE. While you are using the EDI editor, you can obtain information on a specific command, such as ADD, by typing:

```
*HELP ADD
```

Appendix C provides additional information on this editor.

4.6.4 Global Editor (^%GEDIT)

The Global Editor can be used to edit data in global variables. It allows you to perform node-by-node edits on one or more globals or parts of globals. The editor only allows you to alter the data associated with a global node, not the node reference itself.

The Global Editor can be accessed through the utilities menu or directly by typing:

```
> DO ^%GEDIT
```

You then receive a prompt for the global that you want to edit:

```
Global ^
```

Help information can be obtained by typing a question mark (?) after accessing the editor. To exit %GEDIT, type a carriage return.

4.7 Starting and stopping Routines

In Programmer Mode, you can execute the routine that is currently in your routine buffer or a routine that is stored on disk, as described in Sections 4.2.2 and 4.3.1. In Application Mode, a routine is executed immediately after DSM-11 image start-up.

Section 4.7.1 summarizes the rules for executing the routine that is currently in your routine buffer. Section 4.7.2 describes how you execute a routine stored in your routine directory.

Section 4.7.3 lists the conditions under which a routine stops executing.

4.7.1 Executing The Routine In Your Routine Buffer

To execute a routine that is currently in your routine buffer, you use the DO or GOTO command.

```
>DO PARA+3
```

```
>GOTO PARA+3
```

PARA + 3 is a routine line specified by a label plus an offset.

DSM-11 executes the routine starting with line PARA + 3.

4.7.2 Executing A Routine From The Routine Directory

To load and execute a routine stored in your routine directory, you issue the DO command with an argument. The argument is the name of the routine in your routine directory that you wish to execute. The argument is always preceded by a circumflex (^).

For example, the following command line loads the routine TRACE into your routine buffer, and executes it:

```
>DO ^TRACE
```

You can also specify a routine line in the following syntax:

```
>DO GG+1^TRACE
```

This command loads the routine TRACE into your routine buffer and executes it starting from line GG + 1.

4.7.3 Conditions For Execution To Stop

Several conditions or situations can cause execution of a routine to stop. These conditions can differ between Application Mode and Programmer Mode.

In Application Mode, the following conditions cause the results indicated:

- HALT command - causes a logoff.
- QUIT or ZQUIT command - causes a logoff when DSM-11 executes the QUIT or ZQUIT command at the lowest or first nesting level.

- **BREAK** command - has no effect.
- **CTRL/Y** - has no effect.
- **CTRL/B** - has no effect.
- **Error** - passes control to the most recently declared error handler within the application or routine; if no error handler is found, causes a logoff.
- **CTRL/C** - if **CTRL/C** recognition is enabled, generates an **<INRPT>** error, with the same effect as an error. (See Section 4.10 for a discussion of error handling.)

In Programmer Mode, the following conditions cause the results indicated:

- **HALT** command - causes a logoff.
- **QUIT** or **ZQUIT** command - when DSM-11 executes a **QUIT** or **ZQUIT** command at the lowest or first nesting level, returns control to Programmer Mode.
- **BREAK** command - if the debugger is off, has no effect. If the debugger is on, generates a break, indicated by **<BREAK>**. (See Section 4.8 in this manual or the *DSM-11 Language Reference Manual* for more discussion of the **BREAK** command).
- **CTRL/Y** - passes control to Programmer Mode with all stacked contexts discarded. (An **<ABORT>** message appears.)
- **CTRL/B** - at the initiation of the next command, generates a break.
- **Error** - passes control to the most recently declared error handler. If no error handler is found, discards current contexts and passes control into Programmer Mode with an error message displayed. (See Section 4.10 for a discussion of error handling.)
- **CTRL/C** - generates an error, **<INRPT>**, with the same effect as an error. (See Section 4.7.4 for more discussion of **CTRL/C** and **CTRL/Y**.)

4.7.4 Programmer Mode Interrupt By **CTRL/C** And **CTRL/Y**

In Programmer Mode, routine execution can be interrupted by either **CTRL/C** or **CTRL/Y**. The two keys function differently.

You can enable and disable **CTRL/C** recognition in Programmer Mode by using the **BREAK** command.

You can return unconditionally to the Programmer Mode prompt (>) with `CTRL/Y`. The message, <ABORT>, is given when you use `CTRL/Y`.

Using `CTRL/C` generates an error. It gives the message <INRPT>. The use of `CTRL/C` may not halt the execution of the routine; `CTRL/C` will be handled by any error-handling routines that you have established by setting \$ZTRAP within the routine. If the error generated by `CTRL/C` can be handled by your error-trap routine, execution may continue at another level.

4.7.5 Recognition Of The Application Interrupt Key (`CTRL/C`)

The BREAK command with an argument allows you to enable or disable `CTRL/C` recognition. The BREAK command with an argument has the forms:

B 1 Enable `CTRL/C` application interrupt recognition

B 0 Disable `CTRL/C` application interrupt recognition

In Application Mode, DSM-11 disables `CTRL/C` recognition by default. Therefore, the executing DSM-11 routine must contain the BREAK 1 command if you want `CTRL/C` recognition to be enabled in Application Mode for that job.

DSM-11 brings up tied terminals in Application Mode. Thus, `CTRL/C` is always disabled on tied terminals.

In general, disabling `CTRL/C` is useful when you do not wish to have a routine interrupted by the operator. For example, in some applications, an interrupt can cause the data base to become inconsistent. In these instances, inserting a BREAK 0 at the appropriate place in your routine allows the routine to execute a series of SET commands without the possibility of interruption.

A DSM-11 application can poll terminal devices to determine if a user has entered a `CTRL/C`. The application should issue a USE command for the device to be tested and examine the status of bit 15 in the \$ZA special variable.

DSM-11 does not automatically clear a `CTRL/C`. If the application detects `CTRL/C`, the application should clear `CTRL/C`, using BREAK 0, so that it can poll the device again at a later point in processing. (See Section 6.1.2 for more information.)

4.7.6 Changing The Application Interrupt Key

During the SYSGEN procedure, you can define the systemwide default interrupt key (see Section 10.4.12). You can select a different default key from `(CTRL/C)`. This key is in effect whenever you log in. A program can change the interrupt keys for a specific device (such as a terminal) by a USE parameter. When that device is closed, then the default interrupt key is reset.

The default value of the application interrupt key is set to ASCII control value 3 (`(CTRL/C)`). You can set the default application interrupt key to any ASCII control value (0 to 37 octal). For example, if you set the interrupt key to ASCII control value 32 (octal), you enable only `(CTRL/Z)` recognition.

Some terminals have a key labeled BREAK. To enable interrupt recognition of the BREAK key, set the default application interrupt key to ASCII control value zero (0).

The default interrupt key can also be used to log in from a terminal without autobauding in place of `(CTRL/C)`.

You cannot use any of the control characters reserved for the DSM-11 system for the alternate application interrupt key. You also cannot use some nonreserved combinations, such as `(CTRL/M)`, that generate the same code as the RETURN key for the alternate application interrupt key. (See Section 3.1.2 for more information on control keys.)

4.8 Using the Mumps Debugger

The MUMPS debugger is a facility you use in Programmer Mode to monitor the execution of routines. It is a separate debugger from the system debugger, XDT; for more information on XDT, see the *DSM-11 XDT Reference Manual*.

The debugger allows you to stop the execution of your routine at any point you desire and enter *Break Mode*. Break Mode is like Programmer Mode except that the state of the interrupted routine is preserved so that you can start execution of your routine again. You can interrupt execution as often as you like.

You can enter Break Mode in one of four ways:

1. Press the `(CTRL/B)` key while your routine is running. You then enter Break Mode after the execution of the next command even if the debugger is off.
2. Insert BREAK commands in your routine. When DSM-11 encounters the BREAK command while the routine is running, DSM-11 enters Break Mode. The debugger must be on for this to occur.
3. Set up breakpoints (up to 9) using the \$ZBREAK special variable. The debugger must be on for this to occur.

4. Start execution of a routine with the ZGO command and an entry reference.

Once you have entered Break Mode, you can also execute your routine in *single-step mode* in which execution is interrupted after every command in the routine.

The debugger is controlled by a set of DSM-11 commands and a special variable. The commands, described in full in the *DSM-11 Language Reference Manual*, are:

- BREAK
- ZBREAK
- ZGO

Using these commands is described later in this chapter.

The special variable \$ZBREAK, described in detail in the *DSM-11 Language Reference Manual*, is used to contain a reference to the location where you set a breakpoint. Section 4.8.2.1 below describes how to use \$ZBREAK.

4.8.1 Enabling And Disabling Debugging

The ZBREAK command sets an internal flag that turns the debugger on or off. When this flag is ON, DSM-11 recognizes breakpoints set by means of the \$ZBREAK special variable, and the BREAK command.

The flag is set to OFF until you set it to ON by entering the following command:

```
>ZBREAK ON  
D>
```

This command is abbreviated ZB ON. The prompt (D>) indicates that the debugger is on.

To disable recognition of the BREAK command and breakpoints set with \$ZBREAK, you set the flag to OFF with the following command:

```
>ZBREAK OFF  
>
```

The prompt (>) indicates that the debugger is off, and that you have returned to Programmer Mode. When the debugger is off, DSM-11 still recognizes the **CTRL/B** key.

Both ZBREAK ON and ZBREAK OFF can be specified with a postconditional argument. Using a postconditional argument allows you to turn debugging off or on depending on conditions that exist at run time.

4.8.2 Breakpoints

DSM-11 allows you to set nine breakpoints. (These nine breakpoints do not include the breakpoints set with the BREAK command, described in Section 4.8.2.2.)

The nine breakpoints are defined as the values of \$ZBREAK(1) through \$ZBREAK(9).

You set these breakpoints, examine their contents, and clear them, as described in the following sections.

When DSM-11 encounters a breakpoint in a routine, and the ZBREAK flag (described in Section 4.8.1) is ON, DSM-11 displays the following message:

```
<BREAK>entry-ref:command number routine-line  
DB>
```

The arguments entry-ref and routine-line identify the location in the routine where the breakpoint was encountered. The DB> prompt indicates that you are in Break Mode (B), and that the debugger is on (D).

The argument, command number, is the number of the command on the routine line before which the breakpoint took place. The argument, routine-line, is the remainder of the interrupted routine line.

4.8.2.1 Setting Breakpoints With \$ZBREAK — You set a breakpoint at a particular location in your routine by issuing the DSM-11 SET command with the breakpoint variable \$ZBREAK as its argument. (You can set, modify, or kill elements in \$ZBREAK regardless of the value of the ZBREAK flag.) With the \$ZBREAK variable, you specify the entry reference and command number of the desired location in the following format:

```
SET $ZBREAK = "entry reference:command number"
```

entry reference Must be a complete routine reference in the form:

```
label + offset ^routine
```

The label must be the closest preceding label to the line you are specifying. If the line you are specifying has its own label, you must specify that label, not an offset from an earlier label.

command number Must be included; there is no default command number. DSM-11 does not recognize breakpoints on lines with no commands (comment lines, for example).

DSM-11 assigns this location to the first unused breakpoint, beginning with breakpoint 1. If all nine breakpoints are in use, DSM-11 clears breakpoint 9 and assigns the value you specify to breakpoint 9.

You set a particular breakpoint by specifying a breakpoint number with the \$ZBREAK variable, as follows:

```
SET $ZBREAK(n) = "entry reference:command number"
```

You can specify for n any number from 1 to 9 for normal execution.

4.8.2.2 Setting Breakpoints With The BREAK Command — You can also set a breakpoint in a routine by including the BREAK command in the routine, as follows:

```
BREAK
```

The BREAK command also takes a postconditional expression. For example, the following command string suspends execution only if the value of M is less than 30:

```
BREAK:M<30
```

Execution of the BREAK command is dependent on the status of the ZBREAK flag, set with the ZBREAK command.

See the *DSM-11 Language Reference Manual* for more information on the BREAK command.

4.8.2.3 Breakpoint Actions — A breakpoint action string is a command or set of commands that is executed whenever a particular breakpoint is encountered. You can specify an action string as part of the \$ZBREAK value for the breakpoint you specify, in the following format:

```
SET $ZBREAK(n) = "entry reference:command number > action"
```

The action string can contain any legal DSM-11 commands. The commands are executed when the breakpoint is reached, before the BREAK message is printed.

You can use an action string to perform tasks that you would otherwise have to type every time a breakpoint is reached. For example, you could use an action string to examine the contents of a variable. Note that any DSM-11 command is legal, including DO, GOTO, ZBREAK, and ZGO.

You can set the special variable, `$ZBREAK(0)`, as an action string that executes each time you single step in Break Mode. The special variable, `$ZBREAK(0)`, is not a breakpoint but only an action string.

This causes the breakpoint action to be performed after each step in your program. Note that you do not specify an entry reference or command number in this command.

4.8.2.4 Examining Breakpoints — To examine the contents of all existing breakpoints, enter the following command:

```
ZWRITE $ZBREAK
```

To examine the contents of any particular breakpoint, enter the `WRITE` command, specifying which breakpoint you want to look at:

```
WRITE $ZBREAK(n)
```

In response, DSM-11 displays the contents of the breakpoint as in the following example:

```
$ZB(4)="START+1^TEST:3)W A"
```

4.8.2.5 Clearing Breakpoints — You can clear a particular breakpoint by setting the breakpoint to the null string, as follows:

```
SET $ZBREAK(n)=""
```

To clear all existing breakpoints, enter the following command:

```
KILL $ZBREAK
```

Note that breakpoints do take space from your partition (128 bytes each), and the space is not reclaimed until you issue a `KILL $ZBREAK`.

4.8.3 Continuing Execution After A Breakpoint

The two DSM-11 commands for continuing execution after a breakpoint are `ZGO` and `ZBREAK`.

You use the `ZGO` command (abbreviated as `ZG`) to continue execution from Break Mode. Execution is resumed and continues until another breakpoint is reached.

You use the various forms of the `ZBREAK` command to continue execution in single-step mode. These forms are:

ZBREAK OVER

This form of ZBREAK treats any DO or XECUTE command and the subroutine associated with the DO or XECUTE command as one unit. Execution is not halted at any of the lines in the subroutine. Execution is halted when another line is reached in the "top" routine.

ZBREAK IN

or:

```
(RET)
```

This form of ZBREAK steps to the next command and issues a <BREAK> message.

ZBREAK OUT

This form of ZBREAK steps to the first command following the QUIT from the current routine. You usually use ZBREAK OUT when you step into a routine with ZBREAK IN, execute a number of steps, and then decide that you do not need to monitor the remainder of the subroutine.

You can also step through a routine from the beginning without setting a breakpoint. The ZGO command allows you to do this. You call the routine using the ZGO command, which brings you to the beginning of the routine. A breakpoint is issued at the first command, for example:

```
> ZGO ^TEST  
<BREAK>TEST+1^TEST:1 SET X=1,Y=50  
DB>
```

You can now use one of the forms of the ZBREAK command to step through the routine. Note that you cannot use the ZGO command to call up another routine once you are already in the debugger. See the *DSM-11 Language Reference Manual* for more information on the ZGO command.

4.9 Using DSM-11 Directories

DSM-11 provides two directories for each user class (UCI):

- A routine directory
- A global directory

DSM-11 catalogues all application routines and globals for each UCI in these directories. Privileged applications, such as system and library utilities and globals, are catalogued in the directories of the manager's UCI.

The following sections describe the use of DSM-11 directories. For a detailed description of DSM-11 directory structure, refer to Section 13.4 and Appendix B.

4.9.1 DSM-11 Routine Directories

The system places routines in the routine directory of the current UCI whenever you ZSAVE the contents of a partition by name. For example, typing the following places the routine XTR in the routine directory of the current UCI:

```
>ZS XTR
```

In general, each user class has access only to those routines in the routine directory of its UCI. Library utilities, however, which reside in the routine directory of the manager's UCI, are accessible "run only" from all user classes; users with access to the manager's UCI can also modify the library utilities. Accessing the library utilities across UCIs is transparent to the user.

To examine the contents of a routine directory, run the Routine Directory utility, `^%RD`. This routine prints a complete list of all routines in the UCI under which you logged in.

4.9.2 DSM-11 Global Directories

DSM-11 provides a separate directory for the globals created in each UCI. Globals comprise the DSM-11 data base (refer to Chapters 8 and 13 for a detailed description of globals). They are sparse arrays, stored on disk. A sparse array contains only nodes that have been explicitly defined.

The system enters globals in the global directory of the current UCI whenever you execute a routine or command line that defines a global variable with the SET command. After the system places an entry in a global directory, you can only delete it by issuing the KILL command. For a complete description of the KILL command and the global KILL procedure, refer to the *DSM-11 Language Reference Manual*.

A user in a user class can access any global (regardless of its directory) if the user class has access privileges. Section 4.9.3 describes global access privileges in detail. Globals that the library utilities use are accessible, read only, from all user classes. You can also write to globals within the context of the routine. Access to library globals across UCIs is transparent to the user.

To examine the contents of a global directory, run the Global Directory routine, `^%GD`. This routine prints a complete list of the names of all globals in the UCI under which you logged in.

4.9.3 Global File Protection

DSM-11 provides controlled access to the globals in the global directory of each UCI. This allows you to specify privileges for the following user categories:

SYSTEM	Users logged into the manager's UCI (UCI #1).
USER	Users logged into the same UCI in which the global resides.
GROUP	Users logged into any UCI on the volume set on which the global resides.
WORLD	Users logged into any UCI on any volume set, and across DDP.

Each user category can be permitted or denied the following types of access:

READ	The right to examine a global
WRITE	The right to modify a global
DELETE	The right to KILL a global

Access default for newly created globals is READ/WRITE/DELETE for User and System categories; NO ACCESS for Group and World categories.

You can override the default protection for globals through the Global Management utility, `^%GLOMAN`. This routine uses four access codes to specify a range of access privileges for each global. The access codes are:

1. No Access
2. Read Access
3. Read/Write Access
4. Read/Write/Delete Access

If you have access to the manager's UCI, you can override any global access restrictions specified by users in nonprivileged user classes.

4.10 Error Processing

While your routines are running, conditions can arise that are either erroneous or so unusual that the DSM-11 interpreter cannot continue. These conditions cause interpretation to stop, and the \$ZERROR special variable to be set to a 5-character error message surrounded by angle brackets. The 5-character message identifies the type of the error. In addition, the exact location (routine name, line reference, and command number) where the error occurred is included with the \$ZERROR message. (Appendix A gives a listing of DSM-11 error messages.)

What happens next depends on the state of another special variable, \$ZTRAP. (See Section 4.10.2)

4.10.1 Default DSM-11 Error Processing

If \$ZTRAP is set to the null string, default DSM-11 error processing is in effect. The default procedure considers all language syntax, routine, or system operation errors fatal. DSM-11 normally reports errors by:

- Setting the \$IO special variable to the principal I/O device number. (When you log in, the principal device is your terminal.)
- Printing an error message on the principal I/O device. This error message is the current value of the \$ZERROR special variable, followed by the remainder of the routine line where the error occurred. (Appendix A lists error messages, and describes their meaning in detail.)

NOTE

When there is not enough partition space to process the error, DSM-11 deletes the routine from the routine buffer to allow room for the error message.

- Returning you to the Programmer Mode, if you in Programmer Mode. If you logged in in Application Mode, DSM-11 automatically logs you out.

4.10.2 Writing Error-Processing Routines

You can write a routine that interprets errors and continues execution, using the \$ZTRAP special variable. An error-processing routine prevents DSM-11 from aborting a job when an error occurs.

You set the \$ZTRAP special variable (using the SET command) to an error-handling routine for your DSM-11 application. The value of \$ZT, as described in the *DSM-11 Language Reference Manual*, is a reference to the line and/or routine to which control passes when an error occurs in your application.

If you set \$ZTRAP to "LABEL^ROU" and an error occurs, the system:

- Sets the \$ZERROR special variable to a string that consists of the error message plus routine name, label, and command number to indicate where the error occurred.
- Transfers control to the routine called "ROU" at the entry point "LABEL" as specified by \$ZTRAP.
- Preserves the current Call Stack of nested DO or XECUTE contexts.

Note that control is returned to the same context level as the routine where \$ZTRAP was set; control does not automatically return to the lowest level context. See Section 4.10.4 for more discussion of error processing in a situation of several nested contexts.

Your routine can examine \$ZERROR to determine which error occurred and where it occurred. The following example shows an error-processing routine in a simple situation, one that does not involve nested DO or XECUTE commands. The routines print explicit error messages for an application routine. The application routine "PRG" calls the part of routine "ERR" that corresponds to the error index in \$ZERROR.

Routine PRG:

```
A    S $ZTRAP="^ERR"  
  
    . (contains error)
```

Routine ERR:

```
ERR    ;Error Processor  
    U 0 W !, "SORRY BUT LINE ", $P($P($ZERROR, " " ), 2), "^", 1)  
    W " IN ROUTINE ", $P($P($ZERROR, "^", 2), " " ), 1)  
    W " GENERATED AN ", $E($P($ZERROR, " " ), 1), 2, 99), " ERROR"
```

4.10.3 DSM-11 Error-Trapping Routines

If you do not wish to allow your application routines to continue after an error occurs, but you would like to record the fact that an error has occurred, use the following DSM-11 routines:

1. %ET - this routine can log each error as it occurs. Set your error trap to routine %ET:

```
S $ZT="^%ET"
```

Each time an error occurs in your application, the location of the error and the local symbol table are stored in global %ER.

2. %ER - use this routine to investigate errors. The routine %ER reports which errors have occurred, and can reload the routine and its saved symbol table, so that you can debug the error. You can run this routine directly, or you can access it through the CARETAKER option of the SYSTEMS UTILITIES menu (^SYS). Section 12.2 contains more discussion of the ^%ET and ^%ER routines.

4.10.4 Error Processing Within Nested Contexts

A context is created by a DO or an XECUTE statement. Many levels of contexts can be established, each of which requires a Call Stack "frame" of approximately 50 bytes. Programmer Mode is level 1. Successive nested DO and XECUTE commands create higher numbered contexts (up to 255) while stacking a call frame. The QUIT command (explicit or implicit) removes a call frame and reduces the context level by one. The level that is currently executing is called the current context.

The \$ZTRAP special variable can be set at the lowest context level, as well as at higher levels (within successive DOs). Thus, each routine in an application can establish its own error-handling routine by setting \$ZTRAP to the name of an error handler to which DSM-11 passes control in the event of an error.

If an error occurs at a higher context level during the execution of a routine, the procedure is as follows:

- The system error processor first stores the error code, the routine reference, and the command count in \$ZERROR.
- The system error processor then examines the current \$ZT. If the current \$ZT is null, then the current context is discarded; and the next higher context becomes the current context. Any variables saved by the NEW command are restored.

- This procedure is repeated until a context containing a nonnull \$ZT is found. A GOTO the routine indicated in this \$ZT is performed, and \$ZT is set to null.
- If the system processor reaches the end of the Call Stack without finding a nonnull \$ZT, then the default DSM-11 error processing goes into effect. (See Section 4.10.1.)

The following example explains how a sequence of \$ZTRAP settings works:

- Routine D (which does not set \$ZTRAP)
- Routine C (which contains the DO command to execute routine D) sets \$ZTRAP to "^CERR"
- Routine B sets \$ZTRAP to "^BERR" (and contains the DO command to execute routine C)
- Routine A sets \$ZTRAP to "^AERR" (and contains the DO command to execute routine B)

If an error occurs in routine D (the context of which contains no handler), DSM-11 transfers control to CERR to process the error condition, and discards the DO context for routine D from the Call Stack. If an error occurs in routine C, DSM-11 transfers control to ^CERR to process the error condition. But DSM-11 does not discard any context, since the error occurred in the same context in which \$ZT was set to ^CERR.

4.10.5 Exiting From An Error Handler

If an error handler can correct an error condition, it can exit using any of the following commands:

- QUIT
- GOTO
- ZQUIT

From an error handler, QUIT transfers control to the context prior to the one in which the error handler was declared. Thus, if routine D, called from routine C, contains an error that transfers control to ^CERR, the QUIT command in ^CERR transfers control to the statement in routine B that follows the DO ^C command.

You can use the ZQUIT command in your error-processing routines to transfer control to the previously declared error handler, that is, to the error handler of the last subroutine to set \$ZTRAP.

When used together, \$ZTRAP and ZQUIT provide a mechanism for linking all the error handlers in an application into a hierarchy of handlers, systematically passing an error from one handler to another until the error condition is resolved.

Usually, you terminate an error-processing routine that can correct an error condition with GOTO. In this way, the application can proceed to the next routine (or to any appropriate destination).

If your error handler cannot correct an error condition, you can terminate that error-processing routine with ZQUIT. Control is then transferred to the error handler of the last subroutine to set \$ZTRAP. For example, if an error occurs in routine D (in the application described in Section 4.10.4) and the error handler ^CERR cannot correct the error condition, ZQUIT transfers control to ^BERR and discards routine C from the Call Stack. If ^BERR cannot correct the error condition, ZQUIT transfers control to ^AERR and discards the call to routine B from the Call Stack. If the last declared error handler issues a ZQUIT, the DSM-11 error is handled as if no handler were present.

4.10.6 BREAK 2 Control Of Error Processing

Applications written under Version 2 of DSM-11 may not handle errors properly under later versions of DSM-11. In Version 2 mode, the DSM-11 error processor discards the entire Call Stack of nested DOs and XECUTEs whenever an error occurs. As a result, nested error processing is not possible in Version 2 mode. The BREAK command can be used to make DSM-11 simulate the simpler error processing available prior to Version 3.0.

The BREAK command can be used as follows:

B -2 Disables Version 2 compatible DSM-11 error processing

B 2 Enables Version 2 compatible DSM-11 error processing

The default value is for disabling Version 2 error processing (B -2).

Version 2 compatibility has been included so that existing DSM-11 applications that have been upgraded to later versions can handle errors as they had been doing with Version 2. The error-processing parts of these applications do not need to be rewritten; however, the applications must be changed to include "B 2" as the first command.

Chapter 5

Using System Devices

This chapter describes input/output (I/O) under DSM-11 using the system devices.

5.1 Introduction

The DSM-11 system includes multiple I/O devices. Some are physical devices such as terminals, disks, and line printers. Others are logical devices, such as in-memory job communication and routine interlocks.

Because of DSM-11's timesharing environment, many routines can compete for the use of a device. Thus, before you attempt read or write operations to a device other than the terminal with which your routine is associated, you must issue the OPEN command with a specifier of that device as an argument. This establishes exclusive ownership of the device for your routine.

A routine can own more than one device at a time; however, it can access only one of the devices that it owns to perform input and output operations at any particular instant. To gain access to a device, issue the USE command with a specifier of that device as an argument.

After you establish access to a device, you can issue DSM-11 commands to write to or read from the device. Because DSM-11 data transfers consist of ASCII strings of up to 255 characters, you normally do not need to be concerned with specific device characteristics.

When you finish I/O operations with a device, you must issue the CLOSE command with a specifier of that device as an argument. CLOSE ends your ownership and use of the specified device.

5.2 I/O Device Specifiers

Each physical and logical I/O device has a unique device specifier. That specifier is an identification number. Table 5-1 lists the DSM-11 I/O device assignments.

Table 5-1: DSM-11 I/O Device Table

Number	Device
1	Console Terminal
2	Spooling Device
3	Start of Single Line Terminals
	· ·
	· ·
	· ·
19	End of Single Line Terminals
20	Routine Interlock 1
	· ·
	· ·
	· ·
45	Routine Interlock 26
46	Routine Interlock 27
47	Magnetic tape Drive 0
48	Magnetic tape Drive 1
49	Magnetic tape Drive 2
50	Magnetic tape Drive 3
51	Reserved for loadable drivers
	· ·
	· ·
	· ·
58	Reserved for loadable drivers
59	Sequential Disk Processor 0
60	Sequential Disk Processor 1
61	Sequential Disk Processor 2
62	Sequential Disk Processor 3
63	VIEW Command Device (disk and memory)
64	Start of Multiplexer Terminals
	· (all DH11 terminals followed by
	· all DZ11 terminals)
	· ·
	· ·
	· ·
191	End of Multiplexer Terminals

Table 5-1: DSM-11 I/O Device Table (Cont.)

Number	Device
192	Reserved for Digital . . .
199	Reserved for Digital
200	Routine Interlock 28 . . .
223	Routine Interlock 51
224	In-memory job communication, receiver for Unit 0,
225	In-memory job communication, transmitter for Unit 0 . . .
254	In-memory job communication, receiver for Unit 15
255	In-memory job communication, transmitter for Unit 15

5.3 Assigning I/O Devices

The \$IO special variable contains the device number of the current device. DSM-11 directs each read and write operation to this device. When you log in to the system, DSM-11 grants you ownership of and sets \$IO to the device number of your terminal. DSM-11 designates that terminal as the principal device for your job. (By convention, DSM-11 interprets device 0 to mean the principal device of the current job.) You can then reference \$IO in any expression, but you can only change its value with a USE command.

You must use the OPEN command to establish ownership of each device that a routine uses other than the principal device. This command can reserve any number of I/O devices for a routine.

The OPEN command obtains ownership of the device specified in its argument. When you issue an OPEN command, it removes the specified device from the system's pool of available devices and assigns it to your job's pool of devices.

The USE command assigns a previously opened device as your current device, called \$IO. DSM-11 directs all I/O requests to the device whose number is in \$IO. This device does not change until you issue another USE command, or until you close the \$IO device.

The following example obtains ownership of devices 3,4, and 5 and directs any subsequent I/O operations to device 5. Device 5 is the current device.

```
> O 3,4,5 U 5
```

NOTE

The OPEN and USE commands can accept arguments that allow you to set device characteristics. Any unusual use of these commands is included in the description of each device beginning in Chapter 6.

When you no longer require a device, issue the CLOSE command release it for ownership by other users. The following example opens device 3, directs I/O to it, and then closes it. The example sends the string HI to device 3, not to your terminal.

```
> O 3 U 3 W "HI" C 3
```

For output devices, DSM-11 does not execute I/O commands until the current output operation terminates. Also, when you HALT a routine, or when you end a terminal session, DSM-11 closes all currently owned devices unless it is processing output. In this case, the I/O completes before DSM-11 closes the device and halts the job.

5.4 I/O Commands

The commands for input and output operations are:

- READ
- WRITE
- ZLOAD
- ZPRINT
- ZWRITE

You can use these commands with any applicable device. You cannot, however, perform operations inappropriate to a device; for example, reading from a line printer.

The READ command accepts data from the current device and stores it in local variables. The READ command can also include a prompt in the form of

text literals and formatting characters. You can have a timed READ for most devices.

The WRITE command writes variables and string literals to the current device. You can also use the WRITE command for special features of certain I/O devices, which are specified by nonprinting ASCII codes. The WRITE command accepts numeric arguments prefixed by an asterisk (*). The low-order eight bits of these arguments are taken as the decimal representation of an ASCII code. WRITE transmits the code without conversion to the currently assigned I/O device.

Examples:

1. To ring the bell on a terminal, type:

```
> WRITE *7
```

2. To WRITE a Line Feed without a carriage return, type:

```
> WRITE *10
```

3. To rewind magnetic tape unit 0, type:

```
> 0 47 U 47 W *5
```

The ZLOAD command loads a routine into your partition. If you specify a routine name, ZLOAD transfers the routine from your routine directory on disk. If you omit a routine name, ZLOAD transfers the routine from the current device.

The ZPRINT command writes the contents of the routine buffer to the current device.

The ZWRITE command writes the contents of the local symbol table to the current device.

5.5 Output Formatting

DSM-11 has three facilities that allow you to format the data you write:

1. The formatting characters
2. The OPEN and USE commands
3. \$X and \$Y special variables

5.5.1 Formatting Characters

You can use any of the following formatting characters as arguments to the READ and WRITE commands:

#	Page Feed (Form Feed) - Recognized on hard-copy devices and Spooling devices.
!	Carriage-return/line-feed sequence.
?n	Horizontal tabulation - positions the next output character n spaces from the left margin (where n is an integer argument of the tabulation character), but cannot go to the left of the current position on the line.

When these characters immediately follow one another in an argument list, no intervening commas are necessary (that is, #!?3!!! is legal).

Examples:

The following example writes two Form Feeds followed by: 65 spaces, the string PAGE, the value of the variable PAG, and a carriage-return/line-feed sequence.

```
> WRITE ##?65, "PAGE ", PAG, !
```

When writing to a terminal, the following example writes two carriage return/line feed sequences, followed by two rings of the bell, and the letter A.

```
> WRITE !!, *7, *7, "A"
```

The horizontal tabulation character (?) is especially useful to format columns of data. Tabulation is relative to the left margin, so that each successive tabulation on a line must indicate the number of spaces from the left margin, not from the last character. For example:

```
> WRITE ?10, "A", ?10, "H"
```

results in

(10 spaces)AH

not

(10 spaces)A(10 spaces)H

However,

```
> WRITE ?10, "A", ?21, "H"
```

produces the second result.

In any line of text, if one string overlaps the starting position for a ?n formatted string, the ?n string starts at the next available character position. The following example illustrates this:

```
> SET A=17
```

```
> WRITE A, ?6, "METERS"
```

```
17    METERS
```

and:

```
> SET A="SEVENTEEN "
```

```
> WRITE A, ?6, "METERS"
```

produces:

```
SEVENTEEN METERS
```

To write a line with quotation marks in it, such as:

```
WHEN A "?" IS READ, THE ROUTINE PRINTS A "HELP" MESSAGE
```

you must use a double set of quotation marks, because the system drops one set when it writes the text. Therefore, the input line should look like this:

```
> W "WHEN A ""?"" IS READ, THE ROUTINE PRINTS A ""HELP"" MESSAGE"
```

5.5.2 The \$X And \$Y Special Variables

The \$X and \$Y special variables provide the following information to help you format output lines:

\$X	Contains the total number of characters you have written since a carriage return or form feed on the current I/O device. \$X processes only printable (graphic) characters (a W *7 is not a printable character). DSM-11 resets \$X to 0 when its value exceeds 255.
------------	--

\$Y Contains the total number of line feeds you have issued since the last page or form feed on the current output device. DSM-11 resets \$Y to 0 when its value exceeds 255.

Refer to the *DSM-11 Language Reference Manual* for a complete description of the \$X and the \$Y special variables.

5.6 I/O Error Processing

The following categories of errors can occur for I/O:

- Errors that always trap. See Section 4.10 for what happens on an error trap.
- Errors that only trap if you have set \$ZTRAP to a routine reference. If you set \$ZTRAP to a null string, you can only detect these errors by examining the \$ZA special variable.
- Errors that never trap. The routine must detect these errors by examining the \$ZA special variable.

I/O device errors can fall into any category. To determine how to handle a particular error, you must know the specific device characteristics. Chapter 6 describes the device characteristics for each device.

Chapter 6

I/O Device Characteristics

The following sections describe the programming characteristics of DSM-11 I/O devices. For specific hardware characteristics, refer to the *PDP-11 Peripherals Handbook* and the *Terminals and Communications Handbook*.

6.1 Terminals

Terminals in DSM-11 systems include, but are not limited to, VT52, VT55, VT100 video terminals, and DECwriters. The system can have up to 17 remote or local single-line terminals, and up to 128 local or remote terminals on any combination of DH11 multiplexers and DZ11 multiplexers. Any of the single-line devices 3 to 19 can be terminal devices.

6.1.1 Terminal Device Numbers

The terminal device numbers are as follows:

Number	Device
1	Console Terminal
2	Spooling Device*
3	Start of Single Line Devices**
.	.
.	.
.	.
19	End of Single Line Devices**
64	Start of Multiplexer Devices
.	(all DH-11 devices followed by
.	all DZ-11 devices)
.	.
.	.
.	.
.	.
191	End of Multiplexer Devices

* Note that the spooling device is a terminal type device but is described separately, under Section 6.7.

** DMC11 device numbers are in the range of 3 through 19 but are described separately, under Section 6.6.

6.1.2 Terminal Commands

The commands used for terminal I/O are:

Assignment	Output	Input
OPEN	WRITE	READ
CLOSE	ZPRINT	
USE	ZWRITE	
ZUSE		

OPEN Command

The format of the OPEN command for terminals is:

```
OPEN{:postcond} DEV:{(param1:param2:...param10)}{:timeout}
```

When you issue the OPEN command to establish ownership of a terminal device, you can enter any of the 10 optional parameters to override the default terminal characteristics. If you specify parameters, they remain in effect until you issue a USE, CLOSE, or another OPEN command. (In some cases, you change the parameter when you access the device.) To skip the specification of parameters, use a colon for each position, for example:

```
Ø DEV:(:::param4)
```

Output Margin - (*param1*)

The Output Margin parameter allows you to change the setting of the right margin. Whenever the value of \$X (the horizontal carriage or cursor position on the current line) for the device exceeds the value of the right margin, DSM-11 issues a carriage return/line feed. The default is either the setting you specified at SYSGEN or the value you specified with the last OPEN or USE command. A value of zero disables the automatic carriage return/line feed feature.

Output Ring Buffer - (*param2*)

The Output Ring Buffer parameter specifies the size, in bytes, of the output ring buffer. If the terminal is READ/WRITE, this parameter also specifies the size of the input ring buffer unless you also set parameter 4. The value you specify for *param2* must be in the range of 2 to 255 bytes.

Input Field Length - (*param3*)

The Input-Field-Length parameter allows you to enable automatic field termination (without using carriage return) for input strings from 1 to 255 characters. This parameter remains in effect for a single READ only. DSM-11 then resets the Input-Field-Length value to 255 characters. Notice that this parameter duplicates the field length specification now allowed on READ commands. When it is used, the READ command takes precedence over the OPEN command. This parameter (*param3*) is included here only for backward compatibility of existing routines written before Version 3 of DSM-11; the READ command is now the preferred method of specifying field length.

Input Ring Buffer - (*param4*)

The Input-Ring-Buffer parameter allows you to set the size of the Input Ring Buffer. The value you specify in the Input Ring Buffer parameter must be in the range of 2 to 255 bytes.

Table 6-1: Device Status Word Bit Assignments (Cont.)

Bit	Area of Effect	Meaning
6	Escape-enabled	<p>0= Escape processing is disabled. 1= Escape processing is enabled. When you strike an escape key during a READ, DSM-11 waits for the next character that is anything other than ESC "?", or "[". DSM-11 places that character + 16 modulo 64 into the high byte of \$ZB and terminates the READ.</p>
7	Cursor Control	<p>0= DSM-11 does not transmit cursor control sequence when <i>param7</i> is specified. 11= DSM-11 transmits cursor control sequence when <i>param7</i> is specified.</p>
8	Login-control	<p>0= Login enabled. 1= Login disabled.</p> <p>You can use bit 8 to prevent login on a terminal. If bit 8 is set, DSM-11 ignores the <code>CTRL/C</code> character from that terminal. If the terminal is modem controlled, DSM-11 ignores a ring signal unless the terminal is owned.</p>
9	Connect	<p>0= Either the terminal line is not modem-controlled, or the line is modem-controlled but there is no terminal connected. 1= There is a modem-controlled connection to a terminal.</p>
10	Line Printer	<p>0= Device is not a line printer (LP11). 1= Device is a line printer (LP11).</p>
11	Carrier	<p>0= Either the terminal line is not modem-controlled, or the line is modem-controlled, but carrier is not present. 1= Carrier is present.</p>
12	<code>TAB</code>	<p>0= The terminal does not recognize the 8-space tab character. DSM-11 prints spaces instead to bring the value of \$X to the next higher multiple of 8. 1= The terminal recognizes the tab character and DSM-11 writes it to the terminal.</p>
13	DMC11	<p>0= Device is not a DMC11. 1= Device is a DMC11.</p>

Table 6-1: Device Status Word Bit Assignments (Cont.)

Bit	Area of Effect	Meaning
14	Lowercase Characters	0= DSM-11 accepts alphabetic lowercase characters on input. 1= DSM-11 converts alphabetic lowercase characters to uppercase on input.
15	Interrupt Received	0= CTRL/C or alternate application interrupt key has been received. 1= No CTRL/C or alternate application interrupt key has been received since last clear status command cleared bit 15.
16	Terminal Type	0= VT52 cursor control sequences used if cursor control is in effect (bit 7 = 1) and <i>param7</i> is specified. 1= VT100 cursor control sequences used if cursor control is in effect (bit 7 = 1) and <i>param7</i> is specified.
17	Autobaud	0= Terminal is not autobauded. 1= Terminal is autobauded.
18	Data Length	0= 7-bit data is passed (bit 8 is stripped). 1= 8-bit data is passed.
19	Empty String Delete	0= Empty string delete is ignored. 1= When the delete key is struck, and there are no characters to be deleted, the READ command is terminated. \$ZB low-byte = 127.
20	Nonprogrammed Control Keys	0= All nonprogrammed control keys are accepted as normal characters. 1= All nonprogrammed control keys are ignored on input. They are not echoed, and they are not placed in the input buffer.
21	CTRL/O	0= CTRL/O is enabled. 1= CTRL/O is disabled.
22	Buffer Control	0= XOFF is transmitted by DSM-11 if the input buffer is full. XON is transmitted when the input buffer has been emptied. 1= XOFF and XON are not automatically transmitted by DSM-11.

Table 6-1: Device Status Word Bit Assignments (Cont.)

Bit	Area of Effect	Meaning
23	Pass All	0 = Received control characters are filtered and interpreted. 1 = All control character codes are passed to the program without filtering.
24	ZUSE	0 = A ZUSE of this terminal is allowed. 1 = A ZUSE of this terminal is ignored.
25	Type Ahead	0 = Type ahead enabled. 1 = Type ahead disabled.

If you attempt to set the following bits in the Set-Status parameter, DSM-11 reports a syntax error:

- Bit 1 (output only)
- Bit 3 (modem control)
- Bit 8 (login control)
- Bit 10 (line printer)
- Bit 11 (carrier)
- Bit 13 (DMC11)

You can set bits 1, 3, 8, 10, and 13 only by using the Device Characteristics utility, [^]MUX, see Section 10.7.1. Only the DSM-11 system can set bit 11.

You can set bit 9 (connect), on no-login terminals only, for equipment that requires a Data Terminal Ready (DTR) signal to be asserted prior to connection. You can set bit 4 (CTRL/O) to stop terminal output and purge the output buffer.

Clear Status - (*param6*)

The Clear-Status parameter, like the Set-Status parameter, allows you to change device characteristics. The constraints on clearing status bits are the same as on setting them. When you clear bit 9, DSM-11 disconnects a modem-controlled terminal.

Set \$X and \$Y - (*param7*)

The Set X-Y parameter allows you to change the \$X and \$Y settings for the terminal. The values you should enter for the Set X-Y parameters should equal:

$$(Y \text{ coordinate} * 256) + X \text{ coordinate}$$

If you have set bit 7 (cursor control) in *param 5*, you can then use the Set X-Y parameter for direct cursor control. DSM-11 then moves the cursor to the coordinates you specify in the Set X-Y parameter with the USE or ZUSE commands.

Line Parameter Register - (*param8*)

The values for the Line Parameter Register depend on whether you have a DH11 or a DZ11 controller. Tables 6-2 and 6-3 show the Line Parameter Register bit assignments for both DH11 and DZ11 multiplexers. This parameter is inapplicable for single-line terminals.

Table 6-2: DH11 Line Parameter Register Bit Assignments

Bit(s)	Function	Assignments
0,1	Character Length	Bit Bit 1 0 — — 0 0 = 5 Bits 0 1 = 6 Bits 1 0 = 7 Bits 1 1 = 8 Bits
2	Number of Stop Bits	0 = 1 Stop Bit 1 = 2 Stop Bits for 6 to 8 bit characters; 1.5 Stop Bits for 5 bit characters
3	Not Used	—
4	Parity	0 = Disable 1 = Enable
5	Type Parity (ignored if bit 4 = 0)	0 = Even 1 = Odd
6, 7, 8, 9	Receiver Speed	(same as transmitter assignments)

Table 6-2: DH11 Line Parameter Register Bit Assignments (Cont.)

Bit(s)	Function	Assignments
10,11,12,13	Transmitter Speed	Bit Bit Bit Bit 13 12 11 10 — — — — 0 0 0 0 = Zero Baud 0 0 0 1 = 50 0 0 1 0 = 75 0 0 1 1 = 110 0 1 0 0 = 134.5 0 1 0 1 = 150 0 1 1 0 = 200 0 1 1 1 = 300 1 0 0 0 = 600 1 0 0 1 = 1200 1 0 1 0 = 1800 1 0 1 1 = 2400 1 1 0 0 = 4800 1 1 0 1 = 9600 1 1 1 0 = External Input A* 1 1 1 1 = External Input B*
14	Duplex	0 = Full Duplex 1 = Half Duplex
15	Auto-Echo	0 = Disable 1 = Enable

*Special Order Hardware

Table 6-3: DZ11 Line Parameter Register Bit Assignments

Bit(s)	Function	Assignments
0,1,2	Line Number	—
3,4	Character Length	Bit Bit 4 3 — — 0 0 = 5 Bits

Table 6-3: DZ11 Line Parameter Register Bit Assignments (Cont.)

Bit(s)	Function	Assignments																																																																																															
		0 1 = 6 Bits 1 0 = 7 Bits 1 1 = 8 Bits																																																																																															
5	Number of Stop Bits	0 = 1 Stop Bit 1 = 2 Stop Bits 1.5 Stop Bits for 5 bit characters																																																																																															
6	Parity	0 = Disable 1 = Enable																																																																																															
7	Type Parity (ignored if bit 6 = 0)	0 = Even 1 = Odd																																																																																															
8,9,10,11	Speed Select (Transmitter/Receiver)	<table border="0"> <tr> <td>Bit</td> <td>Bit</td> <td>Bit</td> <td>Bit</td> <td></td> </tr> <tr> <td>11</td> <td>10</td> <td>9</td> <td>8</td> <td></td> </tr> <tr> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td></td> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>= 50 Baud</td> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>= 75</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>0</td> <td>= 110</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>1</td> <td>= 134.5</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>= 150</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>1</td> <td>= 300</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>0</td> <td>= 600</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>1</td> <td>= 1200</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>= 1800</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>1</td> <td>= 2000</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>0</td> <td>= 2400</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>1</td> <td>= 3600</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>0</td> <td>= 4800</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>1</td> <td>= 7200</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>0</td> <td>= 9600</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>= Reserved</td> </tr> </table>	Bit	Bit	Bit	Bit		11	10	9	8		—	—	—	—		0	0	0	0	= 50 Baud	0	0	0	1	= 75	0	0	1	0	= 110	0	0	1	1	= 134.5	0	1	0	0	= 150	0	1	0	1	= 300	0	1	1	0	= 600	0	1	1	1	= 1200	1	0	0	0	= 1800	1	0	0	1	= 2000	1	0	1	0	= 2400	1	0	1	1	= 3600	1	1	0	0	= 4800	1	1	0	1	= 7200	1	1	1	0	= 9600	1	1	1	1	= Reserved
Bit	Bit	Bit	Bit																																																																																														
11	10	9	8																																																																																														
—	—	—	—																																																																																														
0	0	0	0	= 50 Baud																																																																																													
0	0	0	1	= 75																																																																																													
0	0	1	0	= 110																																																																																													
0	0	1	1	= 134.5																																																																																													
0	1	0	0	= 150																																																																																													
0	1	0	1	= 300																																																																																													
0	1	1	0	= 600																																																																																													
0	1	1	1	= 1200																																																																																													
1	0	0	0	= 1800																																																																																													
1	0	0	1	= 2000																																																																																													
1	0	1	0	= 2400																																																																																													
1	0	1	1	= 3600																																																																																													
1	1	0	0	= 4800																																																																																													
1	1	0	1	= 7200																																																																																													
1	1	1	0	= 9600																																																																																													
1	1	1	1	= Reserved																																																																																													
12	Receiver On	0 = Disable 1 = Enable																																																																																															
13,14,15	Not Used	—																																																																																															

Set Line Terminator - (*param9*)

This parameter establishes a set of line terminators for a device. A string can be specified to include any control characters from decimal ASCII 0 to 31. A read is terminated when the application user presses any one of the specified line terminators. The low byte of \$ZB returns the ASCII decimal value of the terminator. When the device is opened, `(RET)` and `(ESC)` are set as the default terminators

```
0 DEV:(.....:$(1,13))
```

This example specifies `(CTRL/A)` and `(RET)` as line terminators.

Set Application Interrupt Key - (*param10*)

This parameter establishes a set of application interrupt keys for the device. You can specify a string and include any control characters from 0 to 31. If an application user enters any of characters specified, an `<INRPT>` error is generated (but only if the characters are enabled by the `BREAK 1` command). The low byte of \$ZB contains the interrupt character. The default character is `(CTRL/C)` or the character specified at system generation. See Section 4.7.5 for more discussion of the application interrupt key.

```
0 DEV:(.....:$(1))
```

This example sets `(CTRL/A)` as the interrupt key.

Timeout

The timeout allows you to specify the length of time DSM-11 suspends execution if the requested device is not free. If DSM-11 cannot open the device during the specified period of time, it sets the special variable \$TEST to 0 and resumes execution. If DSM-11 can open the device during the specified period of time, it sets \$TEST to 1 and resumes execution.

The following example illustrates the OPEN command format for terminals.

```
0 5:(80:96:10:32:5::3*256+10)
```

In the example above, device 5 is assigned the following characteristics:

- DSM-11 automatically writes a carriage-return/line-feed immediately after the 80th character on each line
- Output Ring Buffer size is 96 bytes
- Input Field Length terminates the next read after 10 characters
- Input Ring Buffer size is 32 bytes

- Set Status sets the device as a CRT, and disables echo of characters that are typed on the terminal keyboard
- Clear Status does nothing
- \$X and \$Y are changed to the third line in the tenth column
- Line Parameter Register remains unaltered

USE Command

The format of the USE command for terminals is identical to that of OPEN. The USE command allows you to change terminal characteristics from the default values exactly as described with the OPEN command. In addition, USE makes the device the current device (\$IO) and sets the \$ZA special variable to the Device Status Word as described with OPEN.

READ Command

The READ command receives input data from the current device. For terminals, you can use the READ command as described in the *DSM-11 Language Reference Manual* with the following additions:

1. Field Length Specification

DSM-11 terminates a READ when one of the following conditions is true:

- A carriage return, `ESC` key, or line terminator specified in *param9* is typed
- A timeout on the read occurs
- The input data string exceeds the default maximum string size for the READ command (255 characters)

You can alter the maximum string size for the READ command by issuing a USE command with *param3* set; however, the new value remains in effect for one read only. You must respecify the USE command if you want it to be effective for a subsequent read. You can also use the READ command with a field length to do the same thing. See the *DSM-11 Language Reference Manual*. In either case, the low byte of \$ZB is returned with the decimal ASCII value of the terminator equal to 0 to indicate that the field length was exceeded.

2. Escape Processing

VT52 and VT100 terminals have, in addition to the `ESC` key, several other keys that generate an escape sequence. Each escape sequence consists of an escape character followed by an additional character.

When you set bit 6 of the Device Status Word to 0, escape processing is disabled. DSM-11 terminates the READ command as soon as you strike the `ESC` key or any other key that generates an escape sequence.

When you set bit 6 of the Device Status Word to 1, escape processing is enabled. DSM-11 performs the following actions when you strike any key that generates escape sequences.

- Waits for the character following `ESC` (ignoring a ?, which is used as padding in some sequences). The character following `ESC` is called the escape argument.
- Inserts the escape argument into the high byte of the \$ZB special variable, + 16 modulo 64.
- Terminates the read.
- In the case of a READ * command, returns the value of the variable as 0. Your routine can then examine \$ZB to determine which escape sequence you typed. If DSM-11 terminated the read because of some input other than an escape sequence, it sets the value of the high byte of \$ZB to 0.

NOTE

You can expand the number of escape sequences on the VT52 terminal by using the terminal in alternate keypad mode. To do this, transmit, using WRITE *, the decimal code 27 followed by code 61 (W *27,*61).

The following is a list of the escape keys on the keypad of a terminal:

Key	Value of \$ZB High Byte
Normal Mode	
Left blank	32
Center blank	33
Right blank	34
Up arrow	16
Down arrow	18
Right arrow	19
Left arrow	20

The following is a list of the escape keys on the keypad of a VT100 terminal:

Key Value of \$ZB High Byte
Normal Mode

Up arrow	17
Down arrow	18
Right arrow	19
Left arrow	20
PF1	32
PF2	33
PF3	34
PF4	35

The following list of escape keys applies to both the VT52 and the VT100.

Alternate Keypad Mode

All of the above (either VT52 or VT100) plus:

0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
ENTER	29
	62

- The ASCII value of the key that terminated the read is returned in the low byte of \$ZB. If the READ command terminated because the field length was reached, the low byte of \$ZB is 0. When escape processing is enabled, the high byte of \$ZB contains the escape argument, if one is present.

WRITE Command

WRITE sends data and/or control information to the current device. For details, refer to the *DSM-11 Language Reference Manual*.

Using the WRITE command, you can send special instructions terminal. The following is a list of some of the special functions for the VT52 or VT100. (For other terminals, consult the applicable user's manual.) Note that several functions require sending more than one code to the terminal.

Code Sequence	VT52 or VT100 Action
10	Line feed
27,66	Cursor down
27,73	Reverse line feed
27,63	Cursor up
27,67	Cursor right
8	Backspace (cursor left)
13	Carriage return
27,72	Cursor home
27,89, (Desired X-coordinate + 32), (Desired Y-coordinate + 32)	Direct cursor addressing
27,75	Erase to end-of-line
27,74	Erase to end-of-screen
7	Bell
27,88	Identify terminal type
27,61	Enter alternate keypad mode
27,62	Exit alternate keypad mode

ZUSE Command

The format and function of the ZUSE command are identical to those of the USE command, except that ownership of the device is not required. (That is, you do not have to issue an OPEN command before you use a device with ZUSE.) The device to which the ZUSE command is issued remains as the current device until another ZUSE or USE command is issued.

6.1.3 Terminal Error Conditions

The following is a listing of error conditions for terminals.

- If you attempt to issue an invalid parameter with either the OPEN, USE, or ZUSE commands, DSM-11 generates a <SYNTAX>, <MXNUM>, <MINIM>, or <PARn> error.
- If required ring buffer space is unavailable, DSM-11 can generate a <NOBUF> error when you issue READ and WRITE commands. Although DSM-11 allows you to specify ring buffer sizes during the SYSGEN procedure and as device assignment parameters, DSM-11 does not actually assign the ring buffer to a device until it is required. All ring buffers are returned to the system pool when a terminal is neither being used nor owned.
- When you have logged in or opened a remote terminal, DSM-11 monitors the telephone line to assure that a connection is present. If the connection is lost, DSM-11 generates a <DSCON> error. If you are in Programmer

Mode and the principal device disconnects, DSM-11 automatically logs you out, and any local data or routine information is lost. If you are running a routine and have \$ZTRAP set, DSM-11 executes the error routine.

- If you attempt to read from a terminal that your job does not own but is the current device, DSM-11 generates a <NODEV> error. This condition could occur as the result of a ZUSE command or as the result of a background job, initiated by the JOB command, attempting to read from its principal device (which it does not own).

6.2 Line Printer

The DSM-11 system supports any line printer that connects to the LP11 interface. This includes uppercase/lowercase line printers and line printers with 80 and 132 character-per-line capacity.

6.2.1 Line Printer Commands

The commands used for line printer I/O are:

Assignment	Output
OPEN	WRITE
CLOSE	ZPRINT
USE	ZWRITE
ZUSE	

6.2.2 Line Printer Functions And Characters

You can write the following decimal ASCII codes to a line printer with the WRITE * command to effect format control:

Code	Description
10	Line Feed
12	Form Feed (Top of Form)
13	Carriage return

The following example starts a new page on the line printer (device 3) and lists the current routine.

```
> 0 3 U 3 W *12 ZP C 3
```


In most cases, for formatting text within a DSM-11 WRITE statement, you should use the appropriate formatting characters, such as # (form feed) and ! (carriage return), for example:

```
>LBL3      W #,"TEXT",!, "MORE TEXT"
```

See the *DSM-11 Language Reference Manual* for more information on formatting characters.

6.2.3 Line Printer Error Conditions

When a line printer is the current device, \$ZA contains the line printer's Device Status Word. When an error occurs, DSM-11 sets bit 5 in \$ZA, if one or more of the following line printer conditions is true:

```
OFF LINE  
OUT OF PAPER  
YOKE OPEN  
POWER OFF
```

A line printer error causes the job sending the output to the printer to hang. For this reason, it is better to use the spooler for printing rather than print directly from an application. When the DSM-11 Caretaker background job is active, it prints line printer error messages on the Caretaker printer.

6.3 Magnetic Tape

You can use as many as four magnetic tape drives on a DSM-11 system; however, all drives except the TS11, TSV05, TU80, and the TU58 must attach to the same controller. With a TS11, TSV05, or TU80 you have one controller for each drive. You can have TS11 drives mixed with other drives. The TU58 cassette tape drive is handled by a separate device driver. (See Section 6.11 for a discussion of this device.)

Tape-labeling conventions, character sets (ASCII or EBCDIC), data formats, and physical block size are routine selectable. You can modify default values for these parameters with the ^MMD (Modify Magnetic Tape Default) system utility, see Section 10.7.5.

6.3.1 Magnetic Tape Device Numbers

The magnetic tape device numbers are as follows:

Number	Device
47	Magnetic tape Unit 0
48	Magnetic tape Unit 1
49	Magnetic tape Unit 2
50	Magnetic tape Unit 3

6.3.2 Magnetic Tape Commands

The magnetic tape commands are as follows:

Assignment	Input	Output
OPEN	ZLOAD	ZPRINT
CLOSE	READ	WRITE
USE		ZWRITE

OPEN Command

The format of the OPEN command for magnetic tape is:

```
OPEN{:postcond} DEV{:(param1:param2:param3)}{:timeout}
```

Format Switch - (*param1*)

When you use the OPEN command to establish device ownership, you can enter the optional parameters to override the default tape format characteristics established through the system utilities. If you specify parameters, they remain in effect until you close the unit. If you wish to change the parameters, you must first close the unit and then specify them the next time you open the unit. Each character of the optional string *param1* represents a format switch. The effect of each of these switches is described in Table 6-4.

Table 6-4: Magnetic Tape OPEN Switches

Switch Character	Meaning	Effect
A	ASCII	Selects ASCII character set.
B	Shared Buffer	Allows the tape unit to share the transfer buffer with VIEW (device 63). In this mode, no other OPEN parameters except C (continuous), T (tape mark), or digit (density) are allowed, and only WRITE commands are allowed.
C	Continuous	Allows your routine to continue processing while magnetic tape block transfer is in process. When you issue the next block transfer command, implicit or explicit, your routine may hang until the previous command completes. A side effect of this switch is that two buffers are required. If switch B is not specified, the OPEN command allocates two buffers automatically. If you use switch B to share the VIEW buffer, your routine must manage the transfer buffer by means of "U 63...". See Section 6.9. In Nonbuffer Mode, if you specify this switch, switches D and S are also in effect. See Section 6.3.6 for more information on the Continuous Mode.
D	DOS-11 Compatible	Uses DOS-11 labeling, the ASCII character set, and the stream data format.
E	EBCDIC	Translates ASCII characters to EBCDIC when written, and EBCDIC characters to ASCII when read. You can use this switch only if you selected the EBCDIC module during the SYSGEN procedure.
F	Fixed-Length Records Data Format	Assumes fixed-length logical records for input, and uses the stream data format for output. Thus, there is no automatic padding of record length on output. This switch requires the presence of an additional parameter on the OPEN command to specify record length.

Table 6-4: Magnetic Tape OPEN Switches (Cont.)

Switch Character	Meaning	Effect
L	Standard Labeling	Uses ANSI standard labels with the ASCII character set. Uses IBM standard EBCDIC labels with the EBCDIC character set.
S	Stream Data Format	During output operations, packs characters sequentially into the buffer. This format does not split strings across block boundaries; instead, it pads the buffer with null bytes. During input operations, this format treats LFs, VTs, and FFs as string delimiters, and ignores <code>RET</code> . See Section 6.3.5 for more information on stream data format.
T	Tape Mark Trap Inhibit	Inhibits trapping on tape mark condition. You are expected to check \$ZA for the tape mark condition.
U	Unlabeled	Does not provide labels. For files that require a tape mark at the end of the file, you must issue a <code>WRITE *3</code> after you write the file.
V	Variable-Length Records Data Format	This data format corresponds to the ANSI standard D format or the EBCDIC V format, depending upon whether you selected the ASCII or the EBCDIC character set. Each argument of a <code>WRITE</code> command corresponds to a logical record, which can be read as a single argument of a <code>READ</code> command.
3	Density	Specifies 800 bits/n (32 bits/mm) density. (Not available for TS11, TU80, or TSV05.)
4	Density	Specifies 1600 bits/n (64 bits/mm) density. (Not available for TS03 or TE10.)

DSM-11 does not allow all switch combinations. If you specify conflicting switches such as VF, DSM-11 recognizes only the acceptable switch or switches that you specified last in the combination. DSM-11 processes switches from left to right. Table 6-5 indicates which combinations you can use.

Table 6-5: Legal OPEN Switch Combinations

Switch combinations denoted by an "X" are permissible.

SWITCH	A	B	C	D	E	F	L	S	T	U	V	digit
A	X		X	X		X	X	X	X	X	X	X
B		X	X						X			X
C	X	X	X	X	X	X	X	X	X	X	X	X
D	X		X	X				X	X			X
E			X		X	X	X	X	X	X	X	X
F	X		X		X	X	X		X	X		X
L	X		X		X	X	X	X	X		X	X
S	X		X	X	X		X	X	X	X		X
T	X	X	X	X	X	X	X	X	X	X	X	X
U	X		X		X	X		X	X	X	X	X
V	X		X		X		X		X	X	X	X
digit	X	X	X	X	X	X	X	X	X	X	X	X

Logical Record Size - (*param2*)

The Logical Record Size parameter allows you to specify a fixed-length record size in bytes. If you are not using a fixed-length record format, you must use a value of 0 (zero) in this parameter. If you are using a fixed-length record format, you must use a value in the range of 1 to 255.

Physical Block Size - (*param3*)

The Physical Block Size parameter allows you to specify a physical block size in bytes. Its value can range from 14 to 8192 if not in Buffer Mode, or 14 to 32766 if in Buffer Mode. The value must be an even value; an odd value produces a magnetic tape error. In Buffer Mode, you cannot have read or write access to the tape; this mode is for block copying to and from the tape.

Timeout

The timeout allows you to specify the length of time DSM-11 suspends execution until a requested device is free. If DSM-11 cannot open the device during the specified period of time, it sets the special variable \$TEST to 0 and resumes execution. If DSM-11 can open the device during the specified time, it sets \$TEST to 1 and resumes execution.

The following example reserves magnetic tape unit 0 with the default format.

```
> 0 47
```

The following example reserves magnetic tape unit 0 and specifies the ANSI standard D format (labeled).

```
> 0 47: ("AVL")
```

The following example reserves unit 0 and specifies unlabeled EBCDIC with 80-character, fixed-length records and 240-byte blocks (3 records per block). If unit 0 is not available in 3 seconds, DSM-11 continues executing the routine.

```
> 0 47: ("EUF": 80: 240): 3
```

USE Command

Makes the magnetic tape the current device if you previously opened it. If you try to use a magnetic tape device without opening it, the system returns a <NOPEN> error.

CLOSE Command

Performs an implicit WRITE *9 (write an End-Of-File Label) if you are performing an output operation. That is, when you issue a CLOSE command, DSM-11 writes the contents of the buffer (if any). Also, if the magnetic tape unit is not on either the beginning or end of tape indicator, DSM-11 writes the appropriate label and two tape marks. DSM-11 then backspaces over the last tape mark. If the device is again opened, a new file may then be appended. If the device is not opened again, the file is properly terminated.

WRITE and ZPRINT Commands

The ZPRINT command writes the routine buffer to the magnetic tape that is the current device and the WRITE command writes a single logical record to the magnetic tape that is the current device.

WRITE * Command

You can use the control codes shown in Table 6-6 as arguments to the WRITE command to effect special tape functions. These control codes are used with the WRITE * syntax of the WRITE command.

Table 6-6: Magnetic Tape Control Codes Used with WRITE

Code	Meaning	Effect
1	Backspace	Backspaces one block.*
2	Forward Space	Spaces forward one block or magnetic tape mark. If the magnetic tape is at BOT (beginning of tape) skips the label before it performs the forward space.*
3	Write Tape Mark	Writes a magnetic tape mark on the tape.**
4	Write Block	Writes the current buffer. If the magnetic tape is at BOT, writes a volume label before writing the buffer.
5	Rewind	If the last operation was a write, performs an implicit "WRITE*9" and rewinds the magnetic tape.
6	Read Block	Either reads the next block into the buffer, or reads a magnetic tape mark. If the tape is at BOT, skips the volume label (appropriate for the media) and reads the next block.
7	Read Label	Either reads the next block into the buffer, or reads a magnetic tape mark. If the magnetic tape is at BOT, reads the first block, which may be the label.
8	Write Header Label	If you select the DOS-11 format, writes a DOS-11 label. If you select any other labeled format, writes the sequence "HDR1, HDR2, tape mark", preceded by a volume label if the tape is at the BOT. DSM-11 ignores this code for unlabeled formats.
9	Write EOF Label	If you select the DOS-11 or an unlabeled format, this code writes a tape mark. Otherwise, it writes the sequence "tape mark, EOF1, EOF2, tape mark, tape mark".**
10	Write \$ZA	This code updates \$ZA with the actual current tape status.

* The execution of this code clears the READ ONLY/WRITE ONLY switch. Subsequent magnetic tape I/O establishes the new mode for that switch.

** If the last operation was an output operation and data remains in the buffer, this code writes the contents of the buffer before making any other decisions or taking any other action.

ZLOAD and READ Commands

ZLOAD loads a routine from the magnetic tape that is the current device into your partition. A READ command reads the next logical record from your magnetic tape. If the tape is at BOT, and if there is a label, READ skips the label.

6.3.3 Magnetic Tape Operations

To write a single file to magnetic tape, a routine should:

- Issue an OPEN and USE for the unit
- Issue a WRITE *5 to rewind the magnetic tape (if it is not already at the beginning of the tape)
- Write the data
- Issue either a CLOSE for the unit or issue a WRITE *5 to rewind the tape

DSM-11 then writes a partially filled output buffer and performs the appropriate labeling. If the tape is at the BOT position, and if a tape-mark condition indicates the end of a file's data, DSM-11 skips the volume and header labels.

DSM-11 uses the \$ZA special variable to store status information for each I/O operation with various devices including magnetic tape. \$ZA represents the status of the unit at the completion of the last I/O operation, except when you use an OPEN command that establishes ownership of the unit. Table 6-7 lists the \$ZA bit assignments for magnetic tape.

Table 6-7: Magnetic Tape Device \$ZA Status Bit Assignments

Bit	Condition
0	Logical Error
1	Positioning in progress
2	Tape is write protected
3	Not used
4	Not used
5	Beginning of tape (BOT)
6	0 when tape unit does not exist, off line or powered down
7	Nonexistent memory*
8	Bad tape error*
9	Block length error
10	End of tape (EOT)
11	Bus grant late*
12	Parity error*
13	Cyclical redundancy*
14	Tape mark*
15	Error condition*

* If error trapping is set and these conditions are present when the operation completes, DSM-11 produces a <MTERR> trap.

DSM-11 allocates a buffer for the magnetic tape unit on the first OPEN command; *param3* of the OPEN command specifies the physical block size, which can range from 20 to 8192 bytes. (The default is 1024 bytes.) If you use the C switch, two identical buffers of the specified size are acquired.

DSM-11 uses the \$ZB special variable to store the actual number of bytes transferred during the last physical block read. Normally this is the same as the block size specified in *param3*. You can use the value stored in \$ZB to check for any blocks that are shorter than the expected size. DSM-11 reads short blocks successfully without issuing an error. In most operations, you do not need to check \$ZB; DSM-11 always processes short blocks successfully.

You deallocate the buffer with a CLOSE command. Part of the deallocation process is the writing of a partially filled buffer if the last I/O request was a WRITE or ZPRINT. Because DSM-11 automatically writes only full buffers on the magnetic tape, the last request in an output sequence must be a CLOSE or one of the special output operations. The special operations are WRITE *9 (write an EOF), WRITE *5 (rewind) or, in the case of a DOS-11 label format,

WRITE *3 (write a tape mark). Otherwise, the contents of the last buffer might be lost. This same sequence is automatically performed on a HALT.

You can also use the VIEW buffer for I/O transfers to magnetic tape. First you must open VIEW device 63. Then use the B switch as a modifying parameter when you open the magnetic tape device.

By using the read block or the write block operations (W *6, W *4), you can perform operations directly between magnetic tape and disk or between multiple tape units. DSM-11 does not process the internal data structure of the data blocks. You cannot perform logical record operations in this mode.

When you are using the B switch, you are allowed to use two parameters with the USE command. (This is the only case where you can have USE parameters with the magnetic tape device.) The format of the USE command when the tape has been opened with the B switch is:

```
USE DEV{:(param1:param2)}
```

where:

param1 is the physical block size of the transferred magnetic tape blocks. This number must be in the range of 14 to 32768 (default block size is 1024 bytes).

param2 is the transfer buffer offset location. This number must be a multiple of 64 and within the VIEW buffer (default buffer offset is 0).

Both of these parameters can be further restricted by the size of the VIEW buffer. It is not possible, for example, to write blocks larger than the buffer open for VIEW device 63. Also the transfer offset location must be within the current VIEW buffer segment. VIEW buffers are allocated in 1024 byte increments.

When you use the B switch, DSM-11 assigns you the same buffer as currently defined for the VIEW device. Unless the C switch is on, subsequent parameter modifications of the VIEW device with a USE command do not alter the base address of the buffer for magnetic tape operations. If you do use the C switch, the buffer address changes for each block transfer. Also, when you issue a USE command to a magnetic tape device in the B switch mode, you must specify any offsets with respect to the base address as defined when you issued the OPEN command.

6.3.4 Tape Labels And Multiple File Structures

There are four labeling options:

1. DOS-11 compatible label

A DOS label appears at the beginning of the magnetic tape (compressed label format) and a tape mark appears at the end of the file. The DOS file name is:

```
[1,1]MUMPS.001 < 233 >
```

2. ANSI standard label

The ANSI label used in DSM-11 is:

```
VOL1MUMPS1
```

The owner identification field contains:

```
D%B4444001001
```

Single File

```
VOL,HDR1,HDR2*...data...*EOF1,EOF2**
```

Multiple Files

```
VOL,HDR1,HDR2*...data...*EOF1,EOF2*HDR1,HDR2*...data...  
*EOF1,EOF2*HDR1...EOF2**
```

The HDR1 and HDR2 labels are:

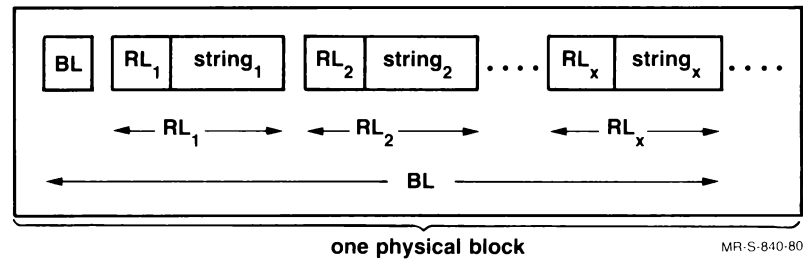
```
HDR1MUMPS.SRC and HDR2D0051200136
```

* Each asterisk represents a tape mark. VOL, HDR1, HDR2, EOF1, and EOF2 are each ANSI-specified 80-character blocks.

3. IBM standard EBCDIC label

With the exception of minor internal field differences and the EBCDIC character set, this labeling convention is the same as the ANSI standard. Figure 6-1 illustrates this labeling convention.

Figure 6-1: IBM Standard EBCDIC Label



4. Unlabeled

DSM-11 generates ANSI standard labels if the OPEN switches L and A are present (or are default switches). DSM-11 generates IBM standard labels if the L and E switches are present (or are default switches). The volume identifier is "MUMPS1," and the file identifier for every file on the tape is "MUMPS.SRC". The file sequence number of the first file is 1. DSM-11 increments the sequence number by one for each subsequent file on the tape.

DOS-11 compatible labels use the file name "MUMPS 001".

To write multiple labeled ANSI standard or IBM standard files, use the following sequence:

```
... write file(x-1) WRITE *9,*1,*8 write file(x) ...
```

Note that if nothing is written for file(x), a subsequent CLOSE or rewind does not properly close that file. A WRITE *9, however, properly closes that null file.

To read multiple labeled ANSI standard or IBM standard files, you should use the following sequence:

```
... read file(x-1), read the tape mark, WRITE *7,*7, read file(x) ...
```

6.3.5 Data Formats

You can select any of three data formats:

1. Stream

Stream data format refers to the data format and not to continuous operation of the tape drive (streaming). See Section 6.3.6 for more information on Continuous Mode. DSM-11 stores characters sequentially in blocks. During output operations, DSM-11 translates a carriage-

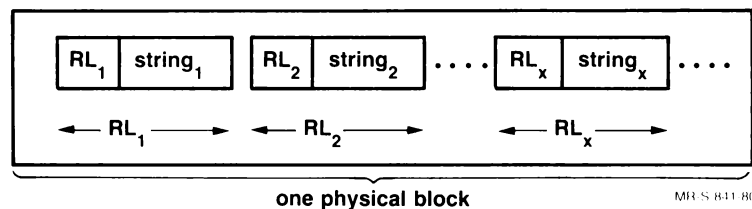
return/line-feed sequence to a line feed. If a line cannot fit into the remainder of a block, DSM-11 pads that block with NUL characters and writes the line as the first line of the following block.

During input operations, DSM-11 converts line feeds, form feeds, and vertical tabs to NUL characters, and discards carriage returns. DSM-11 also interprets a NUL character as the end of a block; input for that READ continues with the first character of the following block.

2. Variable Length

You can select the variable-length record format by using the OPEN switch V. Figure 6-2 illustrates this format.

Figure 6-2: Variable-Length Record Format



DSM-11 precedes each string by a 4-byte numeric character offset whose value is the byte length of the data string plus 4 for the offset length. If the next string plus offset cannot fit into the same block, DSM-11 places a circumflex (^) in the first character position of what would have been the next offset, and then places the string and its offset in the following block.

The OPEN switches E and V select the EBCDIC version of the variable-length record format. The EBCDIC version is the same as the variable-length format, except that every data block begins with a 4-byte block offset (two bytes for the offset, two bytes of 0 value). This block offset equals the length of all of the strings that are in the block and their offsets, plus 4 for the block offset length.

During output operations, DSM-11 treats each argument of the WRITE command as a separate record, and does not write the EOM character automatically.

3. Fixed Length

The fixed-length data format requires that you specify the record length (from 1 to 255 characters) in the OPEN command. DSM-11 continues reading characters until it reads the specified number, or until it encounters a NUL character. DSM-11 then skips the remaining characters in that block and reads the beginning of the next block.

During output operations, DSM-11 uses the same format as the stream format, but does not pad the record length. Thus, routines can write individual fields of a logical record at different points in a routine. The specification of block size should be an integer multiple of the record size.

6.3.6 Continuous Mode

Continuous Mode is chosen by the C switch on the OPEN command. This mode provides for continuous movement of the tape drive, which is sometimes referred to as streaming. This is different from the S switch on the OPEN command, which refers to a stream data format.

Continuous Mode avoids the usual stop and start operation of the magnetic tape drive. With Continuous Mode the tape drive can operate at speeds between 25 IPS (inches per second) and 100 IPS.

For streaming to occur, data must flow to the tape drive at a fast enough rate to keep the drive moving.

In Continuous Mode, DSM-11 allows a job to queue blocks to be written to a tape drive, and then allows the job to continue processing. This is asynchronous I/O, unlike synchronous I/O in which the job is suspended until the write process is complete.

DSM-11 accomplishes the Continuous Mode by using a double buffering technique. A data block can be read from the disk, and stored in a buffer for output to the tape. Another data block can be read from the disk to another buffer, while the first data block is being written out to tape. Continuous Mode is ideal for copying a disk to tape with continuous movement of the tape drive.

Using Continuous Mode requires that the VIEW buffer be owned by the job before opening the magnetic tape drive. Since double buffering is needed, the VIEW buffer must be opened with twice the size of a single magnetic tape block. For instance, when writing 1024 byte blocks, the VIEW buffer must be opened with a block size of 2048 (or 2).

Continuous Mode does not support any READ or WRITE data commands.

The following is an example of how the VIEW buffer can be used to write 4048-byte blocks to tape from disk. The VIEW buffer and magnetic tape pointers are managed by the output routine so that disk and tape operations alternate which segment of the VIEW buffer is being used for each.

In writing the output routine, you must minimize interpreter and CPU overhead between disk and tape operations since the CPU can potentially provide a bottleneck to getting tape operations started smoothly. In addition, other system activity can limit the throughput to the tape.

```

START      0 63:8                ;Get VIEW device with 8 1K byte
                                ;blocks (8096 blocks).
          0 47:"CB"              ;OPEN magnetic tape in Continuous Mode.
          S BLK=0,MAX=100        ;Set block number and maximum for loop.
LOOP      U 63:(1:4) V BLK:"DL0" ;Read 4K bytes of disk into the first
                                ;half of the VIEW buffer.
U         47:(4048:0) W *4       ;Write a 4K byte tape block starting
                                ;at byte 0 of the buffer.
          S BLK=BLK+4           ;Increment block number.
          U 63:(5:4) V BLK:"DL0" ;Read another 4K bytes of disk into
                                ;the second half of the VIEW buffer.
          U 47:(4048:4048) W *4  ;Write a 4K byte tape block starting
                                ;at byte 4048 of the VIEW buffer.
          I BLK=MAX 0           ;Quit if last block is read.
          S BLK=BLK+4 G LOOP     ;Figure next block number to read
                                ;from disk and loop

```

6.3.7 Magnetic Tape Error Conditions

DSM-11 reports the status of magnetic tape in the \$ZA special variable. That is, after any magnetic tape command but CLOSE, the bit pattern in \$ZA reflects the status of the magnetic tape. As shown in Table 6-7, \$ZA can reflect several possible conditions.

- The logical error bit (LER) is set in \$ZA only if a logical error is discovered when attempting to unpack (read) from a magnetic tape block.
- A Positioning-in-Progress Condition
- A Tape-is-Write-Protected Condition
- A Beginning-of-Tape Condition
- A tape-not-ready condition is caused by one of the following:
 - You have not loaded the tape properly.
 - You have not selected the unit properly.
 - You have not turned the power on.
 - You are in the process of positioning the tape.

- You have attempted to write to a WRITE-protected tape.

You can check item 5 before issuing the I/O request by examining \$ZA as shown in the examples at the end of this section.

- A Nonexistent-Memory Condition
- Bad magnetic tape and parity errors are probably caused by a physically bad magnetic tape
- A Record-Length error is a magnetic tape READ error. It occurs when you try to READ a block that is larger than you specified in the OPEN command.
- An EOF error terminates an input operation.

You should examine \$ZA after every tape operation for expected conditions (such as tape marks). The Magnetic Tape Status Check Utility (^%MTCHK) allows you to examine \$ZA. (See Section 7.5.6 for more information on ^%MTCHK.)

When you set the \$ZTRAP special variable at the same context level as the tape I/O, DSM-11 transfers control to the line and/or routine referenced in \$ZTRAP if bits 7, 8, 9, 11, 12, 13, or 15 are set in \$ZA. (See Table 6-7 for more information on \$ZA.) If Version 2 error processing is enabled, then transfer to the \$ZTRAP reference occurs regardless of which context level \$ZTRAP is set in. See Section 4.10 for more information on error processing.

DSM-11 also transfers control to the entry reference specified in \$ZTRAP if the tape is positioned on a tape mark when you are not writing to the tape. To avoid causing this trap; set the T switch (Tape Mark Trap Inhibit) in the optional OPEN parameters when you open the tape. (See Table 6-4 for more information.)

Any attempt to mix reading or writing operations produces a <MODER> error. This error is produced regardless of whether \$ZTRAP has been set or not.

An end-of-tape (EOT) condition is handled as follows: when reading from tape and an EOT is discovered an error is not returned to the routine. The EOT \$ZA bit is set, until a rewind command is given. when writing to tape, and an EOT is detected, the EOT \$ZA bit is set. If \$ZTRAP is set, then an <MTERR> error is returned to the routine. A rewind command resets the EOT bit in \$ZA. Your routine should terminate the file and write an EOF following the detection of an EOT.

The following example checks the magnetic tape for an error condition (bit 15). If it finds an error, it goes to the line labeled ERROR to process the error.

```
Q 47 U 47 G: $ZA \ 32768 ERROR
```


The following example checks bit 10 for an end-of-tape condition. If it finds such a condition, it goes to the line labeled ERROR to process the condition.

```
U 47 G:$ZA \ 1024#2 ERROR
```

6.4 Sequential Disk Processor

The Sequential Disk Processor (SDP) allows you to access the disk as a sequential-access or random-access I/O device.

SDP accesses both the space that is set aside for its use as well as space set aside for disk journaling; it cannot access any other disk space, such as the global data base. The SDP option, ^SDP, of the system utility package allocates disk space for SDP use, see Section 12.4. The journal allocate utility, ^JRNALL, allocates space for journaling. Section 16.3.4 describes this utility.

SDP transfers disk data in 1024 byte blocks, which is the standard DSM-11 block size. When you use SDP, you specify a DSM-11 block number and, optionally, a byte offset. After DSM-11 grants you use of a disk block, no other SDP user is permitted access to the block until the block is released. A block can become free if you access another block, or if you issue a CLOSE command.

The Sequential Disk Processor allows you to reference SDP disk space either sequentially, or in a random-access manner. When you use SDP to access SDP space, you can mix input and output commands. When you access a new block or close the SDP device, DSM-11 automatically rewrites modified buffers.

To read or write sequentially, you should specify parameters just once, in the OPEN or USE you issue to an SDP device. Each subsequent I/O command automatically begins at the byte following the previous I/O operation. To use the SDP area of your disk(s) as random-access space, issue a USE with new parameters each time you wish to direct your input or output to a new location in the SDP space.

When you use SDP to access journal space, you automatically start at the beginning (byte offset = 0) of a block. Moreover, you can only issue READ commands. Only one SDP device at a time can access journal space. If you attempt to open a second SDP device for journal reads, you will receive a <PROT> error. Note that although map 0 of disks, excluding the system disk, can be allocated to SDP space or journal space, only blocks 1 to 399 can be accessed because block 0 is used for the disk label.

6.4.1 SDP Device Numbers

DSM-11 reserves Device numbers 59 to 62 for use by the Sequential Disk Processor.

6.4.2 SDP Commands

The commands applicable to SDP operations are:

Assignment	Output	Input
OPEN	WRITE	READ
CLOSE	ZWRITE	ZLOAD
USE	ZPRINT	

OPEN and USE Commands

The format of the OPEN and USE commands for I/O operations on SDP space is:

OPEN/USE{:postcond} DEV{:(param1:param2:param3)}{:timeout}

where:

<i>DEV</i>	is an SDP unit number from 59 to 62
<i>param1</i>	is the byte within the block (0 to 1023)
<i>param2</i>	is the block number relative to the disk
<i>param3</i>	is the disk drive type and unit number
<i>timeout</i>	is the amount of time allotted for a successful OPEN

Refer to Section 6.4.5 for examples of OPEN and USE command usage in SDP I/O operations.

NOTE

You must know the maps allocated for SDP space to calculate *param2* correctly. Refer to Appendix D for more information about calculating the disk block number.

The format of the OPEN command for SDP operations on journal is:

```
OPEN{:postcond} DEV{:(param1:param2:param3)}{:timeout}
```

where:

DEV is an SDP unit number from 59 to 62.

param1 is the negative (signaling a READ ONLY journal area) of the number of map blocks in the journaling area.

param2 is the DSM-11 block number relative to the disk to begin reading from. (This allows you to begin reading from the middle of the journaling area.) The byte offset is automatically set to 0.

param3 is the disk drive type and unit number

timeout is the amount of time allotted for a successful OPEN.

CLOSE Command

When you close SDP and have modified the current block, DSM-11 automatically writes the block to the disk.

READ and ZLOAD Commands

When you issue a READ or ZLOAD command, SDP reads from the block beginning at the current byte position within the SDP block. SDP transfers data to the current user's partition line buffer until a logical EOM, or until the line buffer is full. If SDP does not detect an EOM or the line buffer is not full and the SDP buffer boundary is reached, SDP reads the next contiguous disk block and continues transfer from byte 0 of the new block until the following occurs:

- SDP detects a null byte in the first position of a block
- SDP reaches the end of a contiguous SDP space

WRITE, ZWRITE and ZPRINT Commands

You perform output to SDP using the DSM-11 syntax described in the *DSM-11 Language Reference Manual* .

When you issue an output command, transfer to disk might not take place immediately (because SDP operations are buffered). DSM-11 writes the contents of the buffer to the disk when:

- The buffer becomes full
- The device is closed
- A subsequent I/O command to this device references a different disk block

After SDP writes a block (when the output exceeds the size of its buffer), SDP reads the next contiguous disk block and continues transfer into the buffer at byte 0.

6.4.3 SDP Special Characteristics And Functions

SDP reports the disk block number of the block currently in the SDP buffer in the \$ZA special variable. SDP reports the byte offset into the current block in the \$ZB special variable.

During sequential I/O operations, SDP skips the DSM-11 map blocks (every 400th block beginning at block 399). An attempt to directly access a map block during an OPEN or USE command generates an error condition. SDP interprets null bytes as the end of a block and automatically skips to the next block. If SDP encounters a null byte in the initial position of a block, it reports "end of file" (\$ZA = -1).

6.4.4 SDP Error Conditions

When you OPEN or USE an SDP device, DSM-11 generates a <MXNUM> error if *param1* (byte within the block) is greater than 1023. If you open SDP with a negative block number to read journal space, and you attempt an output operation to the SDP device, you receive a <PROT> error.

Specifying a positive disk block number (*param2*) that is not part of SDP space or is a map block causes DSM-11 to set the \$ZA variable to -1. This error can occur on an OPEN or USE command or when a sequential operation overflows SDP or journal space. Note that sequential operations automatically skip map blocks. You should examine \$ZA after each OPEN, USE or I/O operation to determine the result of the last requested access. When you open or use an SDP device to access journal space (by specifying a negative block number), all subsequent commands you issue before the next OPEN, USE or CLOSE must be READ commands. Attempts to issue any output commands result in <SYNTAX> errors.

Using an inappropriate parameter in the OPEN statement can result in a <PARn> error.

6.4.5 Examples Of SDP OPEN And USE

The following example shows how to place several records in SDP space in sequential order, and retrieve them:

```
>S A=123,B=57897,C="Edit"
>O 59:(0:1600:"DL1")          ;byte 0, in block 1600
                               ;on RL02 unit 1

>S A=123,B=57897,C="Edit"
>ZW                            ;write the local symbol table
A=123
B=57897
C="Edit"
>U 59 W A,!,B,!,C,!          ;! delimits each record in SDP space
>KILL                          ;deletes contents of symbol table
>U 59:(0:1600:"DL1") R X,Y,Z  ;places X, Y, Z in symbol table
>ZW
X=123
Y=57897
Z="Edit"
```

The following example shows how to use SDP to transfer a routine from the routine directory of one UCI, to the routine directory of another UCI:

```
>O 59:(5:1621:"DL1") U 59 ZP
>ZL (Routine Name)
>O 59:(5:1621:"DL1") U 59 ZP  ;byte 5, in block 1621 on
                               ;RL02 unit 1
>C 59 HALT
```

After logging in under a different UCI:

```
>O 59:(5:1621:"DL1") U 59 ZP
>O 59:(5:1621:"DL1") U 59 ZL  ;places routine in routine buffer
>ZS (Routine Name)           ;puts routine in routine directory
```

6.4.6 Using SDP To Read Disk Journaling Space

When you want to access disk journaling space using SDP, open and use the SDP device with *param1* set to minus the number of maps in the journal space that you want to access. Because the disk journaling feature of DSM-11 writes records sequentially, DSM-11 must also read journal space sequentially to ensure that the records are recovered in correct sequence. Refer to Chapter 16 for a description of disk journaling.

6.5 In-Memory Job Communication

In-memory job communication (JOB COM) permits jobs to send information to other jobs without using the disk. Communication occurs through a series of pseudodevices that are used in pairs; even-numbered devices are "receivers" and odd-numbered devices are "transmitters".

To send information, a job opens and uses a transmitter, and then writes a message. Another job opens and uses the corresponding receiver and reads the message. Transmission is fully buffered; that is, a job can write messages even if the receiving job has not opened and is not using the corresponding receiver. Likewise, a job can read messages after they have been written even if the transmitter is no longer open. Furthermore, several jobs can OPEN a particular transmitter, print a message, and CLOSE the transmitter before earlier messages are read through the receiver. However, several jobs cannot OPEN a particular transmitter simultaneously.

An attempt to write characters when the buffer is full suspends the transmitting job until the receiver removes one or more characters. Similarly, an attempt to read characters when the buffer is empty temporarily suspends the input job.

The transmitter is write only, and the receiver is read only. An attempt to access a device improperly (WRITE to a receiver, for example) causes a <NODEV> error.

6.5.1 JOB COM Device Numbers

The JOB COM device numbers are:

Number	Device
224	Receiver for Unit 0
225	Transmitter for Unit 0
.	.
.	.
.	.
254	Receiver for Unit 15
255	Transmitter for Unit 15

6.5.2 JOBCOM Commands

The commands you can use for interjob communication are:

Assignment	Output	Input
OPEN	WRITE	READ
CLOSE	ZWRITE	ZLOAD
USE	ZPRINT	

WRITE and READ

Every argument of the WRITE command by the output job corresponds to an argument of the READ command by the input job. The data that is received is an image copy of the transmitted data. For example, execution of the following code results in A = "HELLO", B = "AGAIN", C = carriage-return/line-feed characters.

```
> O 225 U 225 W "HELLO", "AGAIN", ! O 224 U 224 R A, B, C
> ZW
A="HELLO"
B="AGAIN"
C=""
```

When using the JOBCOM device, you should always use a timed READ. Because it is difficult to synchronize the READ and WRITE activity between jobs, a timed READ command prevents your job from hanging indefinitely when there was no message sent to you.

ZPRINT and ZLOAD Commands

With these commands, it is possible for you to transfer routines between users. To send a routine and receive it elsewhere, execute the following:

```
> ZL ROU O 225 U 225 ZP
```

And then in the receiving job execute:

```
> O 224 U 224 ZL
```

6.6 DMC11

The DMC11 microprocessor provides high-speed communications between computers over a serial synchronous link. DDCMP communications protocol is implemented by firmware for reliable data transmission, high throughput and low processor overhead. (For further details on these concepts, refer to the *Terminals and Communications Handbook*.)

6.6.1 DMC11 Device Numbers

DMC11 device numbers in the range of 3 to 19 are assigned during SYSGEN.

6.6.2 DMC11 Commands

The DMC11 commands are as follows:

Assignment	Output	Input
OPEN	WRITE	READ
CLOSE	ZWRITE	ZLOAD
USE	ZPRINT	

You can perform data communication through the DMC11 in the following three modes:

- Message Mode - a message can be any DSM-11 expression. This mode is useful when there is a high degree of interaction between the sender and receiver.
- Block Mode - in this mode, the DMC11 transmits entire 1024 byte disk blocks. This mode is useful when you are transmitting a large volume of data.
- Buffer Mode - in this mode, DMC11 transmits the contents of the VIEW buffer to the receiving system's VIEW buffer.

.6.3 DMC11 Message Mode

You transmit DMC11 data using an output command. Input commands allow you to receive messages. A message can be the value of any DSM-11 expression with a maximum message length of 255 characters. The DMC11 device and its system handler automatically perform all necessary hand-shaking as part of each message transmission. The following example illustrates communication between processors using a DMC11 device:

System A:

```
> 0 4 U 4 W "HELLO"
```

System B:

```
> 0 5 U 5 R A:15
```

The READ command receives the message on System B. The timeout variable of 15 seconds prevents the job from hanging if nothing was transmitted, or the transmission is unsuccessful.

NOTE

You should always have both ends of the DMC11 link opened before attempting any data exchange.

6.6.4 DMC11 Block Mode

In Block Mode you send disk data from one system to another. To do this you must first open or use the DMC11 device on both systems using a Block Number parameter. The format of the command is:

```
OPEN or USE{:postcond} DEV{:(param1:param2)}
```

where:

DEV is a DMC11 device number in the range of 3 to 19.

param1 is the DSM-11 disk block number.

param2 is the disk type and unit number

The number you submit from the transmitting system is the first disk block DMC11 sends to the receiver. After each successful transmission of a block, DMC11 increases the DSM-11 disk-block number. The receiving system uses the number that is specified in the OPEN command as the initial disk block number. This disk block is used to store the data received. To perform DMC11 Block Mode operations, you need control codes as shown in Table 6-8.

Table 6-8: DMC11 Control codes Used with WRITE * for Block Mode

Code	Meaning	Effect
1	Block Mode On	Turn on Block Mode transmission
2	Block Mode Off	Turn off Block Mode transmission
4	Transmit Block	Send block and increment the DMC11 disk block number
5	Receive Block	Store the received block of data on disk and increment the disk block number

After you open or use the DMC with a block number parameter you can continue to use the Message Mode I/O commands. But, if you issue a W *1 command, the following rules apply:

- The system that first issues the W *1 command becomes the transmitter and is allowed to send disk blocks. The DMC driver sends a code of 1 to the other system. The code of 1 causes this system to enter Block Mode as the receiver. The receiver is only allowed to receive blocks, and store them on disk.
- When in Block Mode, the transmitter can issue W *4 commands to send a block starting with the DSM-11 disk block number given in the last OPEN or USE command.
- The receiver can only issue W *5 commands to receive blocks and store them on disk starting at the disk block number indicated in its last OPEN or USE command.
- The transmitter can stop Block Mode transmissions by issuing W *2 command. This causes the transmitter to return to Message Mode. The DMC driver also sends a 2 to the receiving system to indicate that it should return to Message Mode.
- When both systems have returned to Message Mode, normal READ/WRITE operations are again permitted until Block Mode is reactivated. In this case, the next sequential block is sent unless you specified a new block number with a USE command.

6.6.5 DMC11 Buffer Mode

In Buffer Mode, you transmit the contents of the VIEW buffer. First, you must open the VIEW device (device 63). You must also open or use the DMC11 device with a Buffer Mode parameter before you can perform Buffer Mode I/O operations. You must open both these devices on both systems connected to the DMC11. The format of the command is:

OPEN or USE *DEV*{:*param1*}

where:

DEV is a DMC11 device number in the range of 3 to 19.

param1 must evaluate to the character "B".

For example:

```
> USE 4:"B"
```

The transmitting system transmits the entire contents of the VIEW buffer. The VIEW buffer must be only one block in length. The receiving system stores this block of data in its VIEW buffer. The VIEW buffer must be the same size (one block) on both systems. When using Buffer Mode, you must be careful not to modify the VIEW buffer until \$ZA bit 5 is reset to 0. When \$ZA bit 5 is 1, a transmission is still pending. (See Section 6.6.7.) This also means that you must not issue W * commands until the \$ZA bit 5 is 0, since the VIEW buffer is also used to send these commands.

To effect Buffer Mode I/O operations, use the control codes listed in Table 6-9.

Table 6-9: DMC11 Control Codes Used with WRITE for Buffer Mode

Code	Meaning	Effect
1	Buffer Mode On	Turn on Buffer Mode
2	Buffer Mode Off	Turn off Buffer Mode
4	Transmit Buffer	Send VIEW buffer to receiving system
5	Receive Buffer	Receive into VIEW buffer

In Buffer Mode, the W * control codes operate in a way that is similar to Block Mode. The difference is that the sender must overtly fill the VIEW buffer, and the receiver must overtly empty it.

Note that the VIEW buffer can also be simultaneously shared with a magnetic tape device or another DMC device. This allows you to transfer magnetic tape data directly between DSM-11 systems through the DMC11 device.

NOTE

You must close the DMC11 device that is sharing the VIEW buffer before you can close the VIEW device.

6.6.6 Using DMC On A Switched Telephone Network

The following applies when using DMC on a switched telephone network:

- When you select DMC devices at SYSGEN, you can specify whether a given device is to operate half-duplex. If you select half-duplex operation, you must also specify whether the device is to be the primary or secondary station. When two DMC devices are connected in half-duplex mode, one must be the primary station, and the other, the secondary station.
- When you issue the CLOSE command on a DMC device, the DMC driver drops the Data Terminal Ready signal (DTR) for one second to initiate telephone disconnection.

6.6.7 DMC11 Error Conditions

The system variable \$ZA reports error conditions. Table 6-10 lists the \$ZA bit assignments for DMC11.

Table 6-10: DMC 11 Device \$ZA Status Bit Assignments

Bit	Condition	Description
0	Data Check	Failure after seven retries. An indication that either the communication channel is defective, or the other end of the channel failed to open. This condition is nonfatal.
1	Time Out	No response within 21 seconds. An indication that the communication channel is defective or there is a power failure at the other end. This condition is nonfatal.
2	Overrun	DMC11 received a message from the remote end but no buffer was assigned. This condition is nonfatal.
3	Maintenance	DMC11 has entered the maintenance state. This condition is nonfatal.
4	Lost Data	DMC11 received a message that was longer than the supplied buffer. This condition is fatal.
5	Transmit Pending	DMC11 has not yet completed transmission.
6	Disconnect	An on/off transition of the modem ready lead (remote operation only). This condition is nonfatal.
7	DDCMP Start	Remote DMC11 has initialized its end. This condition is fatal.
8	Nonexistent Memory	An invalid buffer address has been given to the DMC11 or memory is defective. This condition is fatal.
9	Procedure Error	An illegal DMC11 operation was attempted. This condition is fatal.

Table 6-10: DMC 11 Device \$ZA Status Bit Assignments (Cont.)

Bit	Condition	Description
10	Not Used	
11	Block or Buffer Mode On	DMC is being used in Block or Buffer Mode
12	Transmission Error	Last message could not be transmitted. This condition is nonfatal.
13	DMC11 Flag	Indicates that this is a DMC11 device. Always set for DMC11 devices.
14	Abort	System aborts DMC11 operations. This condition is fatal.
15	Fatal	A fatal condition has occurred (bits 4, 7, 8, 9, or 14).

The bit values in \$ZA are preserved until another READ or WRITE command is issued to the DMC11 device. If you attempt an illegal I/O operation while using a DMC11 device, you receive the <DMCER> error code. The following are examples of illegal I/O operations that produce this error:

- Using illegal parameters with OPEN or USE.
- Trying to issue a W *4 from the receiving system.
- Receiving a data block of a different size from the opened VIEW device.

You can try the last I/O operation again if the condition is not fatal. A fatal condition may require that you open the device again. Continuing error conditions may indicate a failure in the DMC11 hardware. Refer to the *Terminals and Communications Handbook* for additional detail on Status Register 6 (SELG) after a Control Out transfer is received.

6.7 Spooling Device

The Spooling Device is a file-structured mechanism used for temporary storage of information on a disk. Typically, spooling is used to store files temporarily before they are printed on a printer. This output process is referred to as despooling.

The advantage of spooling for a multiuser system is that users can print information (globals, routines, routine output) by means of the spooler without having to own the print device (with an OPEN command) each time they wish to print something. Several users can simultaneously access the Spooling Device, while only one user at a time can own a printer.

DSM-11 has the capability for two types of spooling:

- Transparent spooling - despooling is handled by the DSM-11 `^DESPOOL` routine. All files handled by this routine are sent to one default output spooling device, such as device number 3, a printer. The files are not sent directly to the device, but are stored temporarily on disk as spooled files. This is all transparent to the user because he does not deal with spooling, despooling, or destination codes.
- Explicit spooling - you can write your own despooling routine to take advantage of the DSM-11 capability of providing destination codes for despooling. Despooling can be done to several destinations. To take advantage of this capability, you should examine the `^DESPOOL` routine and use it as a model to customize the spooling facilities at your particular installation.

With explicit spooling you can specify:

- The lowest numbered spool file (the default value)
- A destination code (such as a printer other than the default output spooling device)
- A spooling file number

Spooling creates numbered sequential files within a preallocated area of disk space. There can be up to 253 separate files within spool space at any one time, numbered 1 through 253. You first use the `^SPL` utility to allocate a contiguous space of 400-block maps to spool space, then at start-up (or later with the spooling utilities, see Section 12.5) you enable the spool facility. Your application routines can then create spool files as necessary. The resulting spool files can later be read by a despooling routine and output to a line printer, magnetic tape, or other device.

Although there can be up to 253 spool files existing, there is room in the system tables for only 16 files to be actually opened at any particular time.

6.7.1 Spooling Device Numbers

The spooling device uses device number 2 exclusively.

6.7.2 Spooling Device Commands

The spooling device commands are:

Assignment	Input	Output
OPEN	ZLOAD	ZPRINT
CLOSE	READ	WRITE
USE		ZWRITE

OPEN Command

The forms of the OPEN command are as follows:

TRANSPARENT SPOOLING:

OPEN *DEV* This form creates a spool file whenever the *DEV* matches the device number you specified as the default output spool device.

EXPLICIT SPOOLING:

OPEN 2: This form creates a spool file with the default spool device as the destination. If no default spool device has been declared, a <NODEV> error results. Use this form only if you specify a default output spool device.

OPEN 2:*dest code* Creates a new spool file and opens file for reading. You should include the optional destination code (a number from 1 to 255) if your system has more than one possible destination for spool files.

DESPOOLING:

OPEN 2:512 This form opens the lowest numbered already-existing spool file, regardless of its destination code. (This is the form used by ^DESPOOL.)

- OPEN 2:512 + *dest code* Opens the lowest numbered spool with the specified destination code. This form allows you to write a more sophisticated despooler that prints only those files destined for a particular device.
- OPEN 2:256 + *file no.* This is a specialized form that specific files to be opened.

The code, *dest code*, is an identifying code used to classify the file. The despooling routine uses this code to identify the intended output device. This code can also be used to indicate a despooling routine number or a device class where this file is to be sent. There is no inherent restriction on the use of this code. Note that the *dest code* parameter is for file retrieval purposes only.

The DSM-11 special variable \$ZA holds the file number in the low byte, and the destination code in the high byte.

USE Command

The forms of the USE command are as follows:

- USE 2 You should select this form only if the current job does not open more than one spool file at a time, or if the most recently used or opened file is used. If more than one file is opened, the system will USE the lowest numbered spool file that is open. You should use the next form of the USE command when more than one spool file is open.
- USE 2:*file no.* Makes the specified file the current device. If you are writing to two more spool files concurrently, you must supply the file number when you issue the USE command. The special variable \$ZA equals -1 if the tape has been read beyond the end-of-file.

CLOSE Command

The forms of the CLOSE command are as follows:

- CLOSE 2 Close the currently opened file

CLOSE 2:256	Close and erase the currently used file
CLOSE 2: <i>file no.</i>	Close the file indicated by <i>file no.</i> You must close the file to properly terminate the file. You must include the file number if you have more than one spool file open.
CLOSE 2: <i>file no.</i> + 256	Closes the file indicated by file <i>file no.</i> If + 256 is included with the command, it will delete the file and return its disk blocks to the free-space list. If no file number is specified, the lowest numbered file opened by the job is closed (and deleted if "C 2:256").

The spool file supports the \$X and \$Y variables so that horizontal tabulation (?n) support is possible. In addition, the byte output (the WRITE * format of the WRITE command) is supported. This allows special control codes to be passed to the despooling routine to do special forms requests, error processing, and so forth.

The range of these ASCII codes is from 0 to 127. A code of 0 writes a null string that causes the system variable \$ZA to return a -1.

The following example shows how to process a file if more than one file is currently opened.

```
> OPEN 2:3 SET FILE=$ZA#256
> USE 2:FILE
```

(do output)

```
>>CLOSE 2:FILE
```

6.7.3 Transparent Spooling

Transparent spooling is a special feature that allows a designated device, called the default spool device, to be opened concurrently by many jobs. Whenever an OPEN command is issued for this device, a spool file is created, transparently to the user of the device. This feature allows many users access to a single line printer concurrently, while a despooler job actually performs the printing in the background. You can mix use of a transparent spooler and explicit spooling on the same system with complete freedom. Transparent spooling uses the same spool space as, and shares the file numbers with, explicit spooling.

You can use a set of DSM-11 system utilities for spooling to allocate spool space and to perform other spooling functions. You can access these utilities through the SYSTEM UTILITIES menu (^SYS), or by typing:

```
DD ^SPL
```

These utilities are discussed in Section 12.5.

You can enable transparent spooling by the following steps (using ^SPL):

1. Allocate spool space
2. Designate a default spool device
3. Enable spooling
4. Start the despooling job

If you have already performed steps 1 and 2, you can specify in the System Start-up routine (^STU), described in Section 11.1, that steps 3 and 4 be performed automatically each time the system is started up. (Steps 1,3, and 4 are also required for explicit spooling.)

With transparent spooling enabled, each sequence of opening, using, writing, and closing the default spool device creates a numbered spool file with its destination code set to zero. These files, once created, are indistinguishable from those spool files that are created explicitly (but with no destination code).

6.7.4 Explicit Spooling

Despooling (by ^DESPOOL) is not actually a system feature, but is performed by a normal DSM-11 job.

A special flag set in the job status word of the despooler (^DESPOOL) allows ^DESPOOL to open devices other than the default output spool device. The no-default-spooling flag is bit 11 of the job status word, which is word 2 of each job.

If your system requires more sophisticated despooling, with more than one destination code, for instance, you must modify ^DESPOOL to fit your needs.

You can explicitly create a numbered spool file through device 2, (a pseudodevice for spooling). Unlike other devices, device 2 can be concurrently opened by more than one job (and thus by more than one user).

Each concurrent opening device 2 must access a different numbered file within spool space. The file number of the newly created spool file is assigned to the DSM-11 system, and you can find it in the low byte of \$ZA after you "USE 2". However, you usually do not need this number unless you plan to create two or more spool files concurrently from the same job.

If you wish to use numbered files, the routine you use to write the file can create global nodes named logically, with the spool file number (from the low byte of \$ZA) as the data value. Another specialized despooler routine could then discover the spool file number by accessing the global node.

You can use several methods of opening a spooling file using explicit spooling. (See Section 6.7.2 for the syntax of the different forms of the OPEN command.)

Regardless of the method of opening the file, your despooling routine should issue a USE 2:*file no.* command (including a file number only if more than one file is open). READ commands can then read successive strings starting at the beginning of the spool file. Your despooling routine should examine \$ZA after each READ command; \$ZA is negative when the end of this file has been passed. The spool file can be deleted by appending the code number 256 to the CLOSE command.

6.7.5 Spooling Device Error Conditions

When you open a file, \$ZA returns the following values:

If successful:	dest#*256 + file#
If unsuccessful:	negative error code
	-1 EOF on READ
	-2 No file exists
	-3 Open File Table is full, try again later
	-4 File structure is corrupted (reinitialize)
	-5 Spool space is full.

6.7.6 Spool File Structure

Each spool file is a linked list of disk blocks. The available (unused) space is also a linked list of disk blocks. Together, these blocks comprise all of the blocks set aside as "spool space" in your system. The first block in your spool space holds the "Spool Directory Block."

Figure 6-3 shows the general layout of the spool file structure, and Figure 6-4 shows the layout of the Spool Directory Block.

Figure 6-3: Spool File Structure

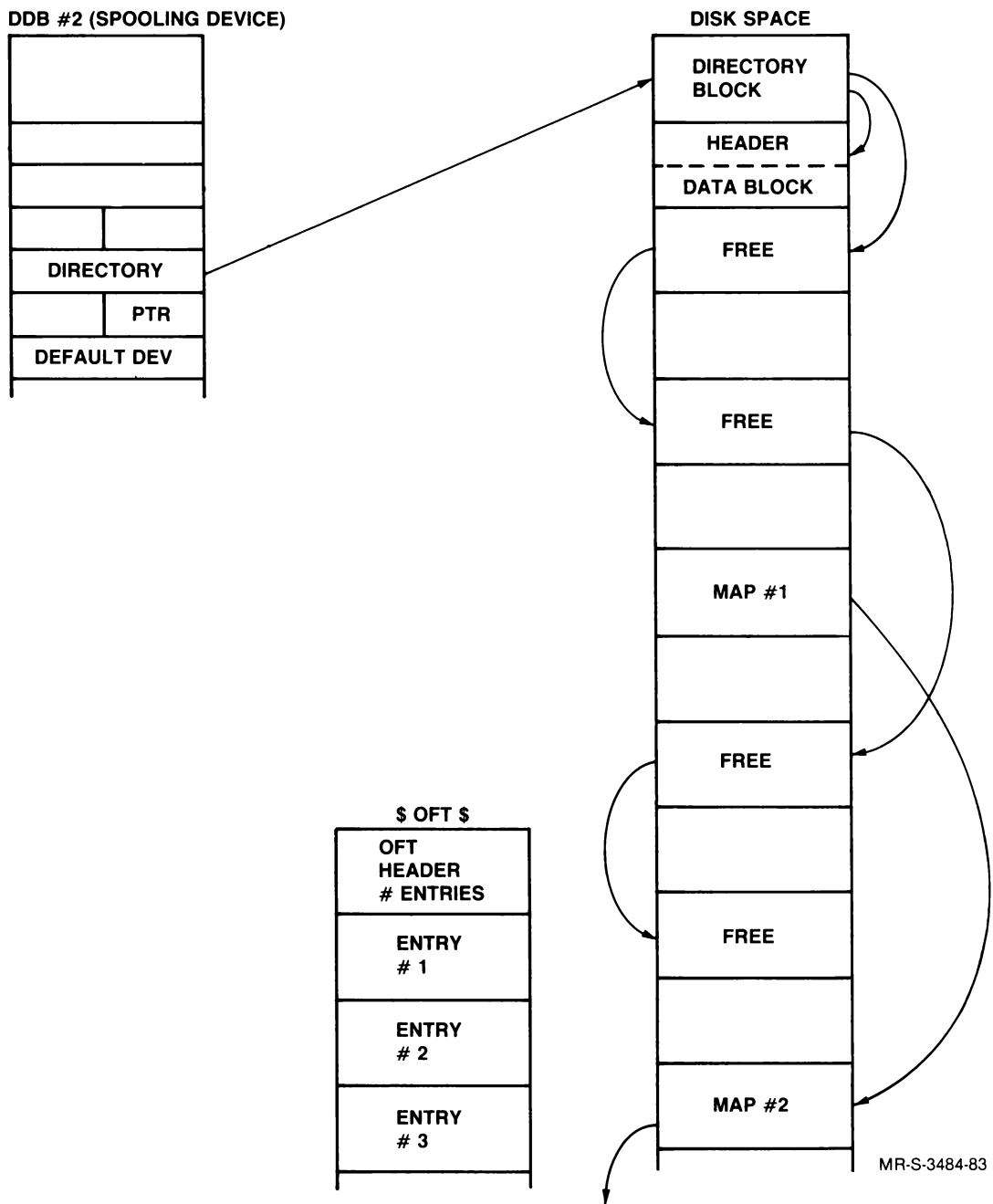
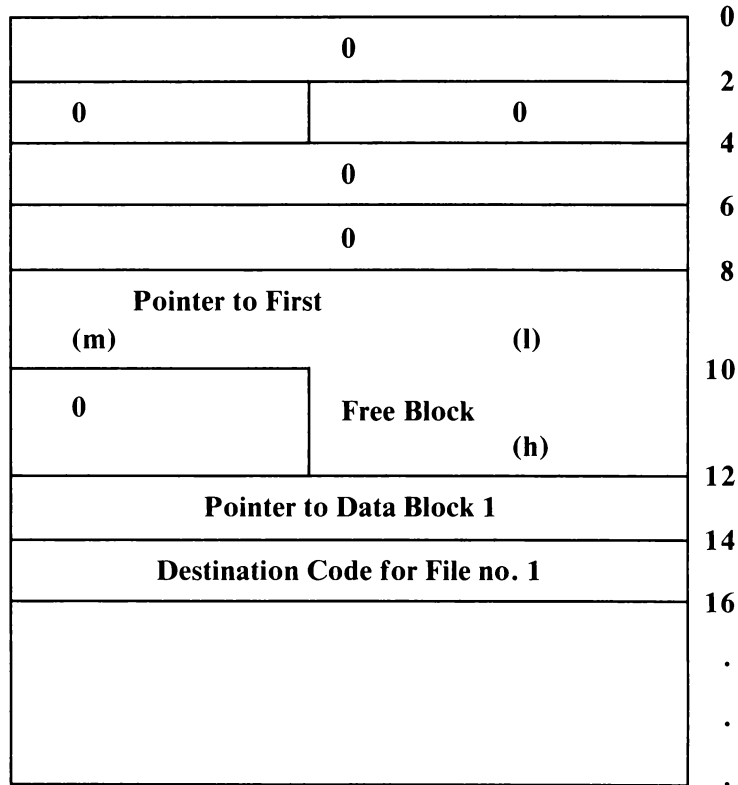


Figure 6-4: Structure of Spool Directory Block

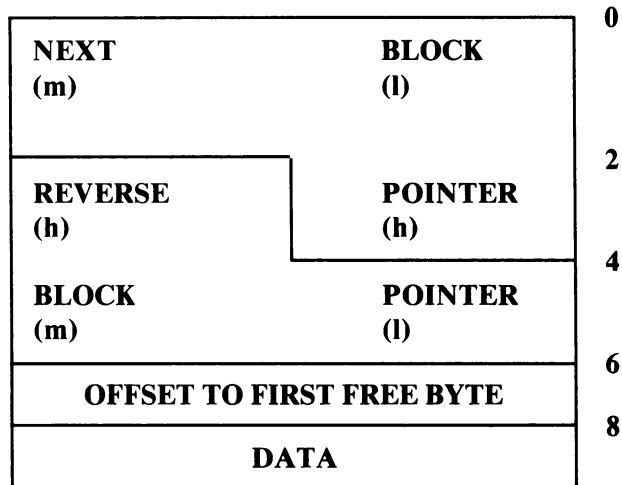


MR-S-3485-83

The directory block cannot be continued. You can use a maximum of 253 files.

The directory block contains pointers to the first free block as well as the first data block for each file (as shown in Figure 6-4). Each spool data block begins with a header as shown in Figure 6-5.

Figure 6-5: Header of Spool Data Block



MR-S-3486-83

The in-memory structures that support spooling are:

- SYSTAB + 230 holds the device number of the default spool device.
- A DDB for device number 2 holds information such as the Spool Directory Block number.
- The OPEN file table (OFT) describes up to 16 concurrently open spool files.

Figure 6-6: Open File Table Entry Format

NUMBER OF ENTRIES IN OFT		0
JOB INDEX	FILE #	2
\$ZA (ERROR REGISTER)		4
CURRENT (m)	BLOCK (l)	6
DESTINATION DEVICE CODE	NUMBER (h)	8
CURRENT BUFFER ADDRESS		10
OFFSET INTO CURRENT BLOCK		12
\$Y	\$X	14

MR-S-3487-83

NOTE

The designations m, l, and h indicate medium, low, and high bytes in the entry.

The OFT entry is the basic information block used by the DSM-11 system to keep track of all opened files in the system. A file is open if it appears in this memory-resident table. Thus, if the system requires rebooting, no files are open because the table is initialized when the system is booted.

All READ and WRITE commands use this table entry. The OPEN command sets up the table entry. The USE command points to it.

If the value of the first two bytes of the OFT entry is -1, then the entry is free. There must be one OFT entry for each active spool file.

A file called the free-pool file holds all unused spool. This file is actually file number 0, but its use is restricted to the system.

6.8 Routine Interlocks

Device numbers 20 to 46 and 200 to 223 are Routine Interlocks. These are logical devices used for communication between routines. Through the OPEN and CLOSE commands, DSM-11 routines can signal one another. The significance of any interlock being owned by a routine is established by a user's conventions. The assignment of these dummy devices has no significance to the DSM-11 operating system. In an application, a Routine Interlock is generally assigned to a function that must not be interrupted by other users when it is operating.

6.8.1 Routine Interlock Device Numbers

The Routine Interlock device numbers are as follows:

Number	Device
20	Routine Interlock #1
.	.
.	.
.	.
46	Routine Interlock #27
200	Routine Interlock #28
.	.
.	.
.	.
223	Routine Interlock #51

6.8.2 Routine Interlock Commands

The commands applicable to Routine Interlock operations are:

Assignment

OPEN
CLOSE

6.9 VIEW Device

The VIEW device, number 63, allows you to modify and examine disk data. You can achieve this by using the VIEW command.

NOTE

During system generation, the System Manager can choose to restrict ownership of the VIEW device to the manager's UCI and the library utilities.

When you own device 63, you own a buffer into which you can read write disk data. You can also modify this buffer using VIEW mode 0 (refer to the *DSM-11 Language Reference Manual*.) You can also share this buffer with magnetic tape devices and a DMC11 device.

The commands applicable to the disk and memory VIEW device are:

Assignment	Input/Output
------------	--------------

OPEN	VIEW
CLOSE	\$VIEW
USE	

OPEN Command

The format of the OPEN command for the disk and memory VIEW device is:

```
OPEN{:postcond} 63{(:param1:param2:param3:param4)}{:timeout}
```

Buffer Size - (*param1*)

You must specify the Buffer Size in blocks (multiples of 1024 bytes). The default is one block. The size of the VIEW buffer is limited only by the total number of blocks in the disk-tape buffer pool, minus 4.

Access Start - (*param2*)

You can specify the start and extent of the current area of the VIEW buffer in blocks (multiples of 1024 bytes.) The current access area applies to both disk transfers and logical addresses for VIEW and \$VIEW mode 0 references. If you specify an Access Start of null, you receive a default of 1. This indicates that you will start with the first VIEW buffer.

Access Extent - (*param3*)

You can specify the Access Extent in blocks (multiples of 1024 bytes). The Access Extent parameter defaults to *param1-param2* + 1.

View Switches - (*param4*)

The possible VIEW switches are described in Table 6-11.

Table 6-11: VIEW Switches

Switch Character	Effect
Z Enable block 0 WRITE	<p>Allows VIEW to write block 0 of any disk. To protect the DSM-11 label and the bad-block list, writing block 0 of any disk is normally illegal.</p> <p style="text-align: center;">NOTE</p> <p>There is a particular problem in writing block 0 of each disk, because the DSM-11 language converts -0 to 0. As a special convention, therefore, DSM-11 recognizes V -16777216 as a command to write block 0.</p>
F Format	<p>Causes the disk driver to change all WRITE commands through VIEW to "format" commands. Subsequent VIEW commands will reformat disk blocks and destroy any data stored on those blocks.</p>
T Test	<p>Causes disk errors generated by VIEW commands not to be retried and not to cause a <DKHER>. You can examine the \$ZA special variable to determine whether the disk transfer was successful. When device 63 is the current device, bit 6 of \$ZA is 1 if there was an error, or 0 if there was not.</p>
P Protection	<p>Prevents global access to the block that is currently in the VIEW buffer. Routines that require global access to the protected block are placed in a "hung" state until either a different block is read into the VIEW buffer, or the protection switch is turned off. If the owner of device 63 attempts a global access to a protected block, DSM-11 generates a <BLPRT> error. Note that protection generates a <SYNTAX> error for a VIEW buffer size other than 1. This switch works only with the "volume set" syntax of the VIEW command, such as the following:</p> <p>VIEW 2145:"S0"</p> <p>For more information on this syntax, see the discussion of the VIEW command in the <i>DSM-11 Language Reference Manual</i>.</p>

Table 6-11: VIEW Switches (Cont.)

Switch Character	Meaning Effect
C Clear Switches	Clears any other VIEW switches. VIEW processes switches from left to right so that U 63:(:::"CP") clears all switches and then sets the protection switch. The C switch does not clear the \$ZA error flag, bit 6. The \$ZA error flag is only cleared by a successful VIEW disk transfer.
V Verify	Causes the disk driver to change all READ commands to "write check" commands. This has the effect of comparing the VIEW buffer with the specified disk blocks. If you set both the T and V switches, DSM-11 reports errors in the \$ZA special variable. If you do not set the T switch, DSM-11 generates a <DKHER> on encountering an error.

Timeout

The timeout specifies the amount of time allotted for a successful OPEN. For example:

```
U 63:(10:5:6:"CP"):2
```

This command OPENS the VIEW device with 10 buffers. This command also enables access to buffers 5 to 10, clears all switches, and sets protection. If DSM-11 cannot open device 63 within 2 seconds, it terminates the command, sets the \$TEST special variable, and resumes execution.

USE Command

The format of the USE command for the VIEW device is:

```
USE{:postcond} 63{:param2:param3:param4}
```

You cannot change *param1* (buffer size) with USE 63. (*param2*, *param3*, and *param4* are as described with the OPEN command.)

VIEW Command and \$VIEW Function

When you access the VIEW buffer with VIEW or \$VIEW, the addresses you specify are relative to the access area enabled by the most recent OPEN or USE command. For example, if you issue the OPEN command that is shown above, the command V 4:0:\$V(4,0) + 1 will increase by 1 the 3rd word of the 5th block (address 4100) of the VIEW buffer. You would not be able to access addresses in blocks 1-4 because they are not in the current access area.

Refer to the *DSM-11 Language Reference Manual* for the specific forms of the VIEW command. Note that disk transfers are limited to 127 consecutive buffers. Also VIEW and \$VIEW (mode 0) allow access to all possible addresses within the VIEW buffer.

6.10 RX02 Diskette Drive

The RX02 diskette device is handled in DSM-11 by a separate loadable driver. The driver can be loaded at system start-up, or loaded and unloaded while the system is operating by using the ^LOAD and ^UNLOAD utilities. See Chapter 10 for information on how this driver is configured during system generation. See Section 12.8 for a description of the ^LOAD and ^UNLOAD utilities.

The RX02 driver is functionally like an SDP device. See Section 6.4 for a discussion of the SDP device.

Block numbers are addressable, and each physical unit is treated as a single logical unit. The RX drive can be used in single- or double-density mode. The disk capacity/drive is 500 1024-byte blocks for double density and 250 for single density. The variables, \$X and \$Y, are reset on any new block read or write by means of the USE command.

6.10.1 RX02 Commands

The commands applicable to the RX02 operation are:

Assignment	Output	Input
OPEN	WRITE	READ
CLOSE	ZWRITE	ZLOAD
USE	ZPRINT	

OPEN and USE Commands

The format of the OPEN command for using a RX02 is:

```
OPEN DEV{:(param1:param2:param3:param4)}{:timeout}
```

The format for the USE command is:

```
USE: {(param1:param2)}
```

Device Numbers - (*DEV*)

The device number can be between 51 and 58. These numbers are assigned at system generation.

Byte Offset - (*param1*)

Indicates the byte position within the block.

Block Number - (*param2*)

Indicates the block number in the diskette.

Switches - (*param3*)

These switches include block format switches and I/O mode switches as indicated:

Table 6-12: RX02 OPEN Block Format Switches

Switch	Meaning	Effect
A	ASCII	Selects ASCII character set
E	EBCDIC	Selects EBCDIC character set
V	Variable	Variable-length records
S	Stream	Stream format
F	Fixed	Fixed-length record format

Table 6-13: RX02 OPEN I/O Mode Switches

Switch	Meaning	Effect
1	Single-Density Mode	RX01 compatible
2	Double-Density Mode	RX02 compatible
B	Buffer Mode	Shared view buffer
C	Control Mode	Only block READ or WRITE commands
R	Read Mode	WRITE protected
M	Mixed Mode	Both READs and WRITEs can be executed
W	Write Only	No implicit READs after block WRITE

When you use the OPEN command, the specified block is read and positioned at the specified byte. If you do not specify a block, nothing is read. If you specify Buffer Mode, a block is not read; but the buffer is positioned. The shared buffer can not exceed one block in size.

Record Size - (*param4*)

The record size parameter is required only for fixed-length records.

6.10.2 Mixed Mode

If you select Byte Position and Block Number parameters, then the specified block number is read; and the buffer pointer is positioned as specified by the Byte parameter.

In Mixed or Write Modes, the following WRITE * command applies:

W *9 = write a logical end-of-file at current position

Mixed Mode is the default I/O mode. It allows you to read and write to the diskette in exactly the same manner that you use with SDP. Before you issue the READ command, you must first specify the block and byte to start reading by means of the OPEN/USE command. When all records are read from the current block, the next sequential block is automatically read by the driver. On writing, your routine selects the byte and block to begin writing (by means of OPEN and USE commands), then your routine issues DSM-11 WRITE commands.

NOTE

Just like SDP, when a block is filled, it is written. Then, the next sequential block is read, and the byte pointer is set to 0. If writes are continued, the data in the current block is overwritten.

This allows true random access and update, in the same manner SDP.

6.10.3 Read Mode

Read Mode logically protects a diskette from being written on. Any attempt to write to a diskette which is opened with the Read Mode switch produces a <MODER> (mode) error.

6.10.4 Write Mode

Write Mode provides greater efficiency when writing to a diskette. When you select this mode, the driver does not read the next sequential block after writing. Consequently, Write Mode should not be used in situations where you require random access for mixed reading and writing. It does allow explicit reading.

6.10.5 Control Mode

If you specify the Byte Offset (*param1*) and Block Number (*param2*) parameters, then the parameters are stored as the current pointers. Subsequent WRITE * commands use these stored pointers as parameters.

When you are in Control Mode, you can use any of the following WRITE * commands:

W *4 = write buffer to the block number specified in USE
W *6 = read block into buffer from the block number specified in USE
W *7 = initialize diskette to current density setting

Control Mode allows Block Mode I/O with any of the special WRITE * commands above. Note: Any attempt to use the WRITE * while in Read or Write Mode causes a <MODER> error.

New diskettes must be initialized before using them. The following example shows how to initialize a diskette:

```
> O 52:(::"C2")  
> U 52 W *7
```

6.10.6 View Buffer Mode

View Buffer Mode allows sharing of the device 63 view buffer. This is especially useful for copying block-by-block from one unit to another unit or device which has been opened to share the view buffer.

6.10.7 Examples Of The OPEN And USE Commands

The following examples show how to use the OPEN command:

```
O 54:(0:0) ; Opens unit double-density, stream, ASCII, reads  
; block 0 and points block pointer to byte 0  
  
O 54:(::"EVR1") ; Opens unit single-density, variable-length records,  
; EBCDIC character set, read only - a block is not  
; read, any attempt to read data by means of a READ
```



```

; command returns a -1 in $ZA until a block is selected
; using the USE command. USE 54:(0:0) reads
; a block.

```

```

0 54:(:"C1") ; Opens unit single-density in Control Mode.

```

The following example shows how to copy a double-density diskette.

```

0 63 ; Opens the VIEW buffer and acquires and allocates
; a buffer for sharing between devices (only
; a buffer size of 1).

```

```

0 55:(:"BC2R") ; open unit 55 control, Buffer Mode,
; double density, read only

```

```

0 56:(:"BC2") ; open unit 56 control, Buffer Mode,
; double density

```

At this point, devices 55 and 56 are both using the same buffer (view buffer) for reading and writing.

```

F I=1:1:500 U 55:(:I) W *6 U 56:(:I) W *4
; copy all 500 blocks from unit 55 to 56

```

6.10.8 Error Codes

Table 6-14: RX02 Error Codes

Code	\$ZA	Explanation
INVBLK	-1	Attempt to access invalid block number
TIMERR	-5	Timeout during I/O operation
NOBUF	-6	Unable to allocate transfer buffer
CRC	-7	Cyclic redundancy check error
DENSER	-8	Diskette is wrong density
HRDERR	-9	Unknown hardware error
GENERR	-10	Hardware error - no volume mounted or other error
ABORT	-11	I/O aborted by MUMPS error
SOFERR	-12	Software error (word count overflow)
ACLO	-13	RX power supply failure
NEWDEN	-14	Disk initialized to new density and current block is invalid
EOF	-26	Logical end-of-file
ENDVOL	-56	Physical end-of-volume

The \$ZB variable returns the current byte pointer in the block.

A <VWERR> is issued when:

- The RX02 is opened in Buffer Mode, and the view buffer was opened with more than one 1024-byte cache block for sharing.
- An attempt is made to open the RX02 in Buffer Mode before opening device 63.
- The view buffer transfer address is not pointing to the start of the block.
- Device 63 is closed before the devices sharing the view buffer are closed.

A <NOEBC> is issued when attempting to open the RX02 for EBCDIC character set, but EBCDIC support is not configured into the running system. This support must be configured during system generation, see Section 10.4.7.

A <MODER> error is issued when attempting to execute a command that is not allowed in the current mode.

6.11 TU58 Magnetic Tape

The TU58 magnetic tape device is handled in DSM-11 by a loadable driver. The driver can be loaded at system start-up, or unloaded and loaded while the system is operating by the ^LOAD and ^UNLOAD utilities. See Chapter 10 for information on how this driver is configured during system generation. See Section 12.8 for a description of the ^LOAD and ^UNLOAD utilities.

To the user, the TU58 driver is functionally like an SDP device. See Section 6.4 for a discussion of the SDP device. TU58 block numbers are addressable, and each of the two units are treated as separate devices with 256(10), 1024(10) byte blocks.

6.11.1 TU58 Commands

The commands applicable to the TU58 operation are:

Assignment	Output	Input
OPEN	WRITE	READ
CLOSE	ZWRITE	ZLOAD
USE	PRINT	

OPEN and USE Commands

The format of the OPEN command for using a TU58 is:

OPEN *DEV*:{(*param1:param2:param3:param4*)} {:*timeout*}

The format for the USE command is:

USE:{(*param1:param2*)}

Device Numbers - (*DEV*)

The device number can be between 51 and 58. These numbers are assigned at system generation.

Byte Offset - (*param1*)

Indicates the offset position of the byte in the block.

Block Number - (*param2*)

Indicates the block number to be used.

Switches - (*param3*)

These switches include block format switches and I/O mode switches as indicated:

Table 6-15: TU58 OPEN Block Format Switches

Switch	Meaning	Effect
A	ASCII	Selects ASCII character set
E	EBCDIC	Selects EBCDIC character set
V	Variable	Variable-length records
S	Stream	Stream format
F	Fixed	Fixed-length record format

Table 6-16: TU58 OPEN I/O Mode Switches

Switch	Meaning	Effect
B	Buffer Mode	Shared view buffer
C	Control Mode	Only block READ or WRITE commands
R	Read Mode	WRITE protected
M	Mixed Mode	Both READs and WRITEs can be executed
W	Write Mode	No implicit READs after block WRITE

When you give the OPEN command, the specified block is read and positioned at the specified byte. If a block is not specified, nothing is read. If Buffer Mode is specified, a block is not read, but the buffer is positioned. The shared buffer must not exceed one block in size.

Record Size - (*param4*)

The record size parameter is required only for fixed-length records.

6.11.2 Mixed Mode

If you select the Byte Position and Block Number parameters, then the specified block number is read, and the buffer pointer is positioned as specified by the Byte parameter.

In Mixed or Write Modes, the following WRITE * command applies:

W *9 = write a logical end-of-file at current position

The Mixed Mode is the default I/O mode. It allows you to read and write to the tape in exactly the same manner that you use SDP. Before you issue the READ command, you must first specify the block and byte to start reading by means of the OPEN/USE command. When all records are read from the current block, the next sequential block is automatically read by the driver. On writing, your routine selects the byte and block to begin writing (by means of the OPEN or USE command), then your routine issues DSM-11 WRITE commands.

NOTE

Just like SDP, when a block is filled, it is written. Then, the next sequential block is read, and the byte pointer is set to 0. If writes are continued, the data in the current block is overwritten.

This allows true random access and update, in the same manner SDP.

6.11.3 Read Mode

Read Mode logically protects a cassette from being written on. Any attempt to write to a diskette which is opened with the Read Mode switch produces a <MODER> (mode) error.

6.11.4 Write Mode

Write Mode provides greater efficiency for writing to a cassette. When you select this mode, the driver will not read the next sequential block after writing. Consequently, you should not use Write Mode in situations where you require random access for mixed reading and writing. Write Mode does allow explicit reading.

6.11.5 Control Mode

If you select the Byte Offset (*param1*) and Block Number (*param2*) parameters, then those parameters are stored as the current pointers. Subsequent WRITE * commands use these stored pointers as parameters.

When you are in Control Mode, you can use any of the following WRITE * commands:

W *4 = write buffer to the block number specified in USE
W *6 = read buffer from block number specified in USE

Control Mode allows block mode I/O with any of the special WRITE * commands above. Note: Any attempt to use the WRITE * while in Read or Write Mode, causes a <MODER> error.

6.11.6 View Buffer Mode

View Buffer Mode allows sharing of the device 63 view buffer. This is especially useful for copying block-by-block from one unit to another unit or device which has been opened to share the view buffer.

6.11.7 Examples Of The OPEN And USE Commands

The following examples show how to use the OPEN command:

```
0 52:(0:0)      ; Opens unit double-density, stream, ASCII, reads
                ; block 0 and points block pointer to byte 0
0 52:(:"EVR1")  ; Opens unit single-density, variable-length records,
                ; EBCDIC character set, read only - a block is not
                ; read, any attempt to read data by means of a READ
                ; command returns a -1 in $ZA until a block is selected
                ; using the USE command. USE 52:(0:0) reads
                ; a block.
```

```
Q 52:(:"C1") ; Opens unit single-density in Control Mode.
```

The following examples show how to copy a cassette.

```
Q 63 ; Opens the view buffer and acquires a buffer
; for sharing between devices
; (only a buffer size of 1)
```

```
Q 53:(:"BCR") ; Opens unit 53 control, Buffer Mode,
; read only
```

```
Q 54:(:"BC") ; Open unit 54 control, Buffer Mode
```

At this point, devices 53 and 54 are both using the same buffer (view buffer) for reading and writing.

```
F I=1:1:256 U 53:(:I) W *6 U 54:(:I) W *4
; copy all 256 blocks from unit 53 to 54
```

6.11.8 TU58 Error Codes

The following error codes are returned in \$ZA. An asterisk (*) indicate errors returned from the TU58.

Table 6-17: TU58 Error Codes

Code	\$ZA	Explanation
INVLK	-1	Attempt to access invalid block number
BDTEST	-3	Failed self-test*
PARTOP	-4	Partial operation (end of medium)*
TIMERR	-5	Timeout during I/O operation
NOUNIT	-9	Bad unit number*
NOTAPE	-10	Cartridge not loaded*
ABORT	-11	I/O aborted by MUMPS error
PROTECT	-12	Write protected*
INIERR	-13	Unable to initialize device
DATERR	-18	Data error*
EOF	-26	Logical end-of-file
SEEKER	-33	Block error (block not found)*
STOPMO	-34	Motor stopped*
BADCOD	-49	Bad opcode*

Table 6-17: TU58 Error Codes (Cont.)

Code	\$ZA	Explanation
ENDVOL	-56	Physical end of volume*; all recoverable errors must start here
HARDER	-60	Framing or data overrun error
LGERR	-61	Logical or protocol error
CHKSMR	-62	Checksum error

The \$ZB variable returns the current byte pointer within the block.

A <VWERR> is issued when:

- The TU58 is opened in Buffer Mode and device 63 was opened with a buffer size greater than one 1024-byte block.
- An attempt was made to open the TU58 in Buffer Mode before opening device 63.
- The view buffer transfer address is not pointing to the start of the block.
- Device 63 is closed before the devices sharing the view buffer are closed.

A <NOEBC> is issued when attempting to open the TU58 for the EBCDIC character set, but EBCDIC support is not configured into the running system. This support must be included during system generation, see Section 10.4.7.

A <MODER> error is issued when attempting to execute a command that is not allowed in the designated mode.

Chapter 7

Using The DSM-11 Library Utilities

This chapter describes the DSM-11 library utilities.

7.1 Introduction to the Library Utilities

The DSM-11 library utility routines perform a variety of services. These include:

- Editors
- Global related
- Routine related
- Miscellaneous

Chapter 12 describes utilities that aid in the operation and maintenance of the system. Section 4.6 gives an overview of the editor utilities, and Appendix C describes two of the editors in more detail. The later parts of this chapter summarize the other categories of library utilities. You can use the utility routines directly in Programmer Mode, or in your user-written routines.

The library utilities include an interactive library utility package that provides on-line help information and access to the utilities through a series of menus. A listing of these menus along with help information can be printed by using the utility `^%MENLIS`. (See Section 7.4.2.) When you invoke the package, it

displays utility options on your terminal; and you select those options required to do a given task.

All library utility routines are stored in the manager's UCI (UCI 1), and are listed in the routine directory of the manager's UCI. These routines are accessible to all DSM-11 users on a run-only basis. Privileged users, (that is, users with access to the manager's UCI and the system PAC), can modify library utilities.

7.1.1 Running The Library Utilities

You have three ways in which you can initiate utility routines:

1. In Programmer Mode, type:

```
> D ^ZUTL
```

This command invokes the utility package menu-driver that allows you to select from the displayed options. To access the library utilities menu directly, type:

```
> D ^ZLIB
```

2. Type:

```
> D ^ZName
```

where [^]%Name is the name of a library utility. The routine name for each utility in the package is displayed following its entry in the menu.

3. Log in DSM-11 in Run Mode, using the routine name of a utility or the library package menu-driver (%UTL), for example:

```
DSM-11 Version n.n Device #64 UCI: MGR:ZUTL
```

This log-in procedure loads and starts the utility package menu-driver in the System Manager's UCI, MGR.

To load a library utility without executing it, type:

```
> ZLOAD ZName
```

Only users in the manager's UCI can ZSAVE a library utility routine.

7.1.2 Library Utility Conventions

The following conventions apply to the library utility package:

1. If you are unsure of the proper response to a question, type a question mark (?). This gives you a list of valid responses to a question.
2. The menu-driver displays a number followed by an option. You can enter either the number of the option you want to run, or enough characters of the option name to distinguish it from other options.
3. Unless otherwise specified, terminate all input with a carriage return.
4. Default conditions are displayed between left and right angle brackets (< >) as part of the question. A carriage return in answer to a prompt with a default means that the default will be used as the input for that prompt.
5. To exit or quit a routine instead of answering a prompt, use a carriage return if the prompt has no default value. You can also exit and return to the previous question or menu by typing an up arrow, (^).
6. Unless otherwise specified, any number you enter is considered to be decimal; numeric values returned by routines are decimal, unless they are preceded by a #, which means the number is octal.
7. You can respond to all prompts for an input or output device with either a device mnemonic or a number (if it represents a valid device in the system). Device 0 is always the terminal on which you logged in.

7.2 Global Utilities

The following utilities can be accessed from the LIBRARY UTILITIES (^%LIB) menu.

Utility	Routine Name
Management	^%GLOMAN
Copy	^GC
Directory	^%GD
Efficiency	^%GE
List	^%G

Restore	^%GTI
Save	^%GTO
Selector	^%GSEL
Subscript Filter	^%FIND
Extended Global Directory	^%EGD

Two other utilities for saving and restoring globals, ^%GS and ^%GR, are described in Sections 7.4.6 and 7.4.7.

The Global Editor (^%GEDIT) is described with the other DSM-11 editors under Section 4.6.4.

7.2.1 Global Management (^%GLOMAN)

Lists the journaling, collating, and protection information for a specified global. You can also change these characteristics, and change the data growth area.

7.2.2 Global Copy (^%GC)

Transfers globals or parts of globals from one UCI to another.

7.2.3 Global Directory (^%GD)

Lists the names of all globals in your current global directory.

7.2.4 Global Efficiency (^%GE)

Shows how much of each block (as a percentage) is being filled with data for a given global; actually reflects the efficiency of the DSM-11 operating system in using disk block space.

7.2.5 Global List (^%G)

Displays both the contents and logical structure of a global or set of globals.

7.2.6 Global Restore (^%GTI)

Reads globals that were stored with global save (^%GTO) from a sequential storage medium such as the Sequential Disk Processor (SDP) or magnetic tape. The global can be renamed before being restored.

7.2.7 Global Save (^%GTO)

Outputs the specified globals onto a sequential medium such as the SDP, magnetic tape, a printer, or a terminal. Globals saved with ^%GTO can be restored with ^%GTI.

NOTE

You should use ^%GTO and ^%GTI for global save and restore rather than the earlier version of these routines, ^%GS and ^%GR. The earlier versions do not handle all formats correctly. Note also that ^%GS should always be used with ^%GR, and ^%GTO with ^%GTI.

7.2.8 Global Selector (^%GSEL)

Selects a list of global names.

7.2.9 Global Subscript Filter (^%FIND)

Finds all global subscripts which contain specified characters.

7.2.10 Extended Global Directory (^%EGD)

Produces a list of all globals in the current UCI including information about collating, journaling, protection, and disk growth areas.

7.3 Routine Utilities

These routines are useful for manipulating or reporting on routines. They can be accessed from the LIBRARY UTILITIES (^%LIB) menu.

Utility	Routine Name
Compare	^%RCMP
Copy	^%RCOPY
Directory	^%RD
List First Line	^%FL
Restore	^%RR
Save	^%RS
Search	^%RSE
Summaries	^%SUM
Change Every	^%RCE
Selector	^%RSEL
Cross Reference	^%CRF
Extended Routine Directory	^%ERD

These utilities are described in the following sections.

7.3.1 Routine Compare (^%RCMP)

Compares two DSM-11 routines and prints out any lines with any differences.

7.3.2 Routine Copy (^%RCOPY)

Copies segments of DSM-11 code from one routine to another. This routine can also append several routines into one routine. To copy one line of code only, enter the same line reference (line label or line label plus offset) for both the *From Line* and the *Through Line* prompts.

7.3.3 Routine Directory (^%RD)

Displays the names of all DSM-11 routines stored in the current UCI.

7.3.4 First Line List (^%FL)

Displays the first lines of a selected list of routines. This is most useful if you make the first line of each routine the name of the routine followed by a comment describing the routine.

7.3.5 Routine Restore (^%RR)

Restores routines that were saved to a file on a sequential medium using Routine Save (^%RS). A routine can be renamed as it is restored.

7.3.6 Routine Save (^%RS)

Saves selected DSM-11 routines onto a sequential medium or lists them on a terminal or line printer.

7.3.7 Routine Search (^%RSE)

Searches a DSM routine or set of routines for the occurrence of a string. This utility prints any line that contains the string, as well as a reference to the routine in which the line occurred. If Routine Search is unable to find a routine that you have properly saved (through ZSAVE), then answer YES to the prompt about refreshing the directory in ^UTILITY.

7.3.8 Routine Summaries (^%SUM)

Compiles and prints documentation for program packages.

7.3.9 Routine Change Every (^%RCE)

Searches a selected group of routines for a specified string and replaces the string with a specified replacement string.

7.3.10 Routine Selector (^%RSEL)

Selects a list of routine names.

7.3.11 Routine Cross Reference (^%CRF)

Provides a cross reference of all routine calls within a specified set of routines.

7.3.12 Extended Routine Directory (^%ERD)

Produces a list of routines in the current directory, including time and date of the last ZSAVE, size in bytes, and starting block number of each routine.

7.4 Miscellaneous Utilities

These utility routines perform tasks in a variety of areas. They generally do not produce interactive dialog. The following utility routines can be accessed from the LIBRARY UTILITIES menu (^%LIB) and are described later in this section.

Utility	Routine Name
Decimal/Octal Conversion	^%DOC
List ^MENU Global	^%MENLIS
Load DSM-11 Editor	LOAD^%ED
Show Current Date	^%D
Show Current Time	^%T
Version 1 Global Restore	^%GR
Version 1 Global Save	^%GS
I/O Device Selector	^%IOS
Header Formatter	^%HDR
Menu Manager	^%MENU
Modem Autodialer	^%DIAL

7.4.1 Decimal/Octal Conversion (^%DOC)

Is an interactive utility that converts a number from octal to decimal or decimal to octal. The utility prompts for a number as input. If the number has a # preceding it, then the utility assumes that the number is octal.

7.4.2 List ^MENU Global (^%MENLIS)

Lists the contents of the MENU global. This is a summary listing of the menus used in the utility package, along with help information on each utility. This utility prompts for a device number (such as for a printer) to use in listing the information. The default device is the terminal that you are logged in on.

7.4.3 Load DSM-11 Editor (LOAD^%ED)

Loads the ^% editor (the DSM-11 editor). You do not normally need to use this utility unless the ^% editor was killed accidentally.

7.4.4 Date (^%D)

Prints the current date in the form:

Day-Month-Year

For example:

17-JAN-81

7.4.5 Time (^%T)

Prints out the current time in the format:

HH:MM:SS

where:

HH is a two-digit representation of the hour

MM is a two-digit representation of the minute

SS is a two-digit representation of the second

For example:

10:35:02

7.4.6 Version 1 Global Restore (^%GR)

Restores a global from a sequential medium such as SDP or magnetic tape. It is intended for use with globals stored with ^%GS. You can restore some, all, or none of the globals in the specified file; the global can also be renamed before being restored.

7.4.7 Version 1 Global Save (^%GS)

Saves the specified global or global nodes onto a sequential medium such as the SDP, a printer, a terminal, or magnetic tape. This utility is intended for use with ^%GR.

NOTE

These two global routines, ^%GS and ^%GR, are the earlier versions of the global save and restore routines, and do not handle all formats properly. It is recommended that ^%GTO and ^%GTI be used. Note also that ^%GS should always be used with ^%GR, and ^%GTO with ^%GTI.

7.4.8 I/O Device Selector (^%IOS)

Obtains an I/O device number or name.

7.4.9 Header Formatter (^%HDR)

Produces a formatted header for printer output.

7.4.10 Menu Manager (^%MENU)

Provides for simple menu creation using a global data dictionary. Includes facilities for recursive menu calls and help text.

7.4.11 Modem Autodialer (^%DIAL)

Controls and dials an autodialing modem.

7.5 Other Miscellaneous Utilities

The following utilities are not interactive. You can call these utilities from other routines to produce the result described.

Utility	Routine Name
Date and Time	^%H
Decimal to Octal Conversion	^%DO
Octal to Decimal Conversion	^%OD
Cursor Control	^%CURSOR
Get UCI	^%GUCI
Magnetic Tape Status Check	^%MTCHK

7.5.1 Date And Time (^%H)

Sets the date and time in \$HOROLOG. ^%H does not produce an interactive session. This subroutine also has some additional entry points to provide for date and time text formatting.

7.5.2 In-line Decimal To Octal Conversion (^%DO)

^%DO converts any positive decimal integer to its octal equivalent. This utility is not designed for interactive use; it is meant to be used in DSM-11 application routines.

To use ^%DO, first set the variable %DO to a decimal value. Next, call the utility ^%DO, which converts the decimal value to octal. Finally, write the new value of the variable %DO. For any input that is not a valid decimal number, ^%DO returns "B" (for bad entry). If the input was valid, %DO contains the converted number. Consider the following example using ^%DO.

```
GETNUM   R !, " DEC NUMBER: ", %DO
         D ^%DO I %DO="B" G GETNUM
         W %DO
```

This routine requests that you enter a decimal number. If the number entered is not decimal, GETNUM prints an error message, and prompts for another number. If the number is valid, GETNUM writes it in octal to the current I/O Device.

7.5.3 In-Line Octal To Decimal Conversion (^%OD)

^%OD converts any positive octal integer to its decimal equivalent. The first step is to set the variable %OD to an octal value. Next, call the ^%OD utility, which makes the conversion to decimal. Then write the new value of the %OD variable. For any nonpositive octal integer that you enter, ^%OD returns a "B" (for bad entry). If the input was good, %OD contains the converted number.

This utility is not designed for interactive use. It is designed for use in DSM-11 application routines.

7.5.4 Cursor Control (^%CURSOR)

Returns a set of variables that you can use for cursor control. First, ^%CURSOR interrogates the terminal to determine its type. Then, ^%CURSOR gives the set of variables the values of character strings appropriate to that type of terminal. To activate a particular cursor-control function, use the ^%CURSOR-created variable that represents that function as an argument of the WRITE command.

7.5.5 Get UCI (^%GUCI)

Returns your UCI name and number. ^%GUCI is not an interactive subroutine. When you call ^%GUCI, it returns values in two % variables: in %UCI, it returns the name of your UCI, and in %UCN, the number of your UCI (the manager's account being number 1)

7.5.6 Magnetic Tape Status Check (^%MTCHK)

^%MTCHK assists you in determining the status of a magnetic tape device. You can use ^%MTCHK in two ways.

1. Make the magnetic tape the current device and execute %MTCHK:

```
>U dev D ^%MTCHK
```

where:

dev is the magnetic tape device

^%MTCHK examines the \$ZA special variable and writes a one-line interpretation of each bit that is set in \$ZA.

The following example gains ownership of magnetic tape unit 47 and makes it the current device. Then, it uses ^%MTCHK to write the status of magnetic tape unit 47.

```
>O 47 U 47 D ^%MTCHK
```

```
WRITE LOCKED
```

2. Execute %MTCHK starting at the line labeled %SET:

```
>D %SET^%MTCHK
```

%MTCHK sets a series of variables to strings that you can evaluate at any time after a magnetic tape operation for a Boolean true/false condition. The following is a list of the variables that ^%MTCHK sets and their meaning if evaluated as true (1).

%MTBOT Magnetic tape is positioned at beginning of tape

%MTEOT Magnetic tape is positioned at end of tape

%MTERR One or more significant error bits are set in \$ZA

%MTLER Logical error. Illegal interrupt, illegal command, or unrecoverable tape error occurred

%MTON Magnetic tape unit is ready

%MTPIP Magnetic tape is rewinding

%MTTMK Magnetic tape is positioned on tape mark

%MTTYP Specifies magnetic tape type:

10 = TU10, or TS03

11 = TS11, TSV05, TU80

16 = TE16, TU16, TU45, or TU77

%MTWLK Magnetic tape is write-protected

The following example shows how ^%MTCHK sets the magnetic tape variables.

```
>D %SET^%MTCHK
>ZW
%MTBOT="$ZA\32#2"
%MTEOT="$ZA\1024#2"
%MTERR="$ZA\32768#2"
%MTLER="$ZA#2"
%MTON="$ZA\64#2"
%MTPIP="$ZA\2#2"
%MTTMK="$ZA\16384#2"
%MTTYP="16"
%MTWLK="$ZA\4#2"
```

You can then use the IF command to check the status of these variables. The magnetic tape device must be the current device when you make the check. For example, the following line checks for an end-of-tape condition.

```
U 47 I @%MTEOT G EOT
```


Chapter 8

Globals

This chapter describes the implementation of global arrays under DSM-11. It provides an overview of global concepts, and describes the logical structure of globals. This chapter also describes global access across UCIs and volume sets, and Distributed Data Processing (DDP), which permits access to globals on other systems.

8.1 Global Concepts

DSM-11 incorporates a hierarchical approach to data base organization. This is done through the use of global arrays. A global array, or simply *global*, is a tree-structured system of nodes stored on disk. Thus, a global can be thought of as a file.

The DSM-11 data base supervisor handles the physical and logical allocation of disk storage for globals. The data base supervisor is called the global module. Unlike data base systems that treat disk space like a sequential I/O device, DSM-11 treats disk space as an extension of memory. The system maps the structure and content of globals created in your partition into the physical storage medium of the system in 1024-byte blocks. Because space is not preallocated for globals, the need for dimensioning is eliminated.

The system handles disk allocations for globals transparently. Thus, you do not have to concern yourself with the physical structure of your files when designing a data base application. You need only concern yourself with the data relationships that comprise the logical design of your files.

8.1.1 Global Variables

To create a global, you define a *global variable* in a routine or command line. You define a global variable by setting it equal to a data value. The naming conventions for global variables are the same as for local variables, except that a circumflex (^) or up arrow precedes the name. The following are examples of global variables:

```
^Z  
^NAME("LAST", "FIRST")  
^IDNUM(1,3,4)
```

The *DSM-11 Language Reference Manual* describes the SET command and its use with global variables in more detail. Section 8.2 describes a special extended global reference syntax used for inter-and intra-processor communication.

Global variables can be either simple or subscripted. If a global variable is subscripted, it can have up to 63 characters per subscript. Subscripts can be either of the following:

- Numeric
- String

The system places global variables in the global hierarchy according to their collating sequence. A global can be collated in either of two collating sequences:

- Numeric collating sequence
- ASCII collating sequence

In numeric sequence, first numeric subscripts are collated in numeric order. (The numbers in numeric subscripts must be canonic: numbers with no leading or trailing zeros.) Second, nonnumeric subscripts are collated in the order of their ASCII characters. In a numeric collating sequence, you can mix strings and integers. The following is an example of a global array with mixed subscripts:

```
BETA(1,1)  
BETA(1,1,1)  
BETA(1,2,2)  
BETA(1,3.14159)  
BETA("FIRST", "SECOND", "THIRD")
```

In ASCII sequence, all subscripts (numeric and nonnumeric) are collated in order of their ASCII characters. The example below shows the array A collated in ASCII order:

```
A("01")
A(1)
A(10)
A(2)
A(2.5)
A(20)
A(3)
A("B")
```

Global variables collate in numeric order by default. The collating order for a global can be changed if:

```
$DATA(Global Ref) = 1
```

The Global Management library utility (^%GLOMAN) allows you to change collating sequence for a global.

8.1.2 Sparse Arrays

When global variables are subscripted, the resulting arrays are called *sparse arrays*. In a sparse array, elements only exist for those nodes that actually contain data or have descendants. Descendants are elements of an array that exist in a logical path down the array. For example, array element $\wedge B(1)$ can have descendants $\wedge B(1,5)$ and $\wedge B(1,7)$. However, the element $\wedge B(2,4)$ belongs to a line of descent coming from $\wedge B(2)$.

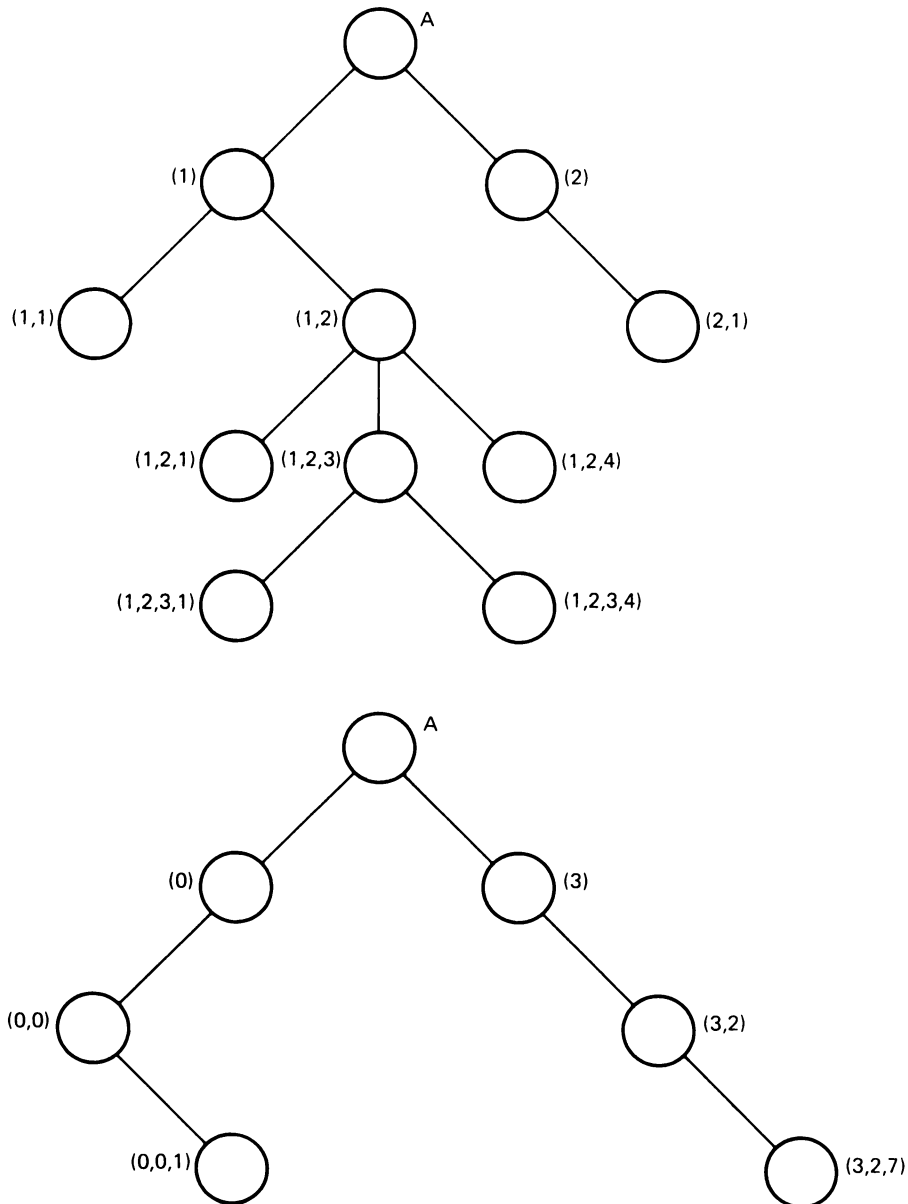
One advantage of sparse arrays is that they permit you to assign a value to a node even if no other nodes exist in a logical path to the node you want to define. The system automatically creates a path to a node. For instance, if you want to define $\wedge A$ and $\wedge A(1,2,3,4)$, you need not define $\wedge A(1)$, $\wedge A(1,2)$, and $\wedge A(1,2,3)$ first. The system creates a logical path directly between $\wedge A$ and $\wedge A(1,2,3,4)$.

An additional advantage of sparse arrays is that they conserve storage space. Unlike data base systems that require a declaration of the maximum size of an array to preallocate space for it, DSM-11 allocates storage space dynamically. The system also recovers disk space dynamically when data is no longer needed.

8.1.3 Global Data Structures

Global arrays can be represented logically as upside-down trees. DSM-11 uses a tree structure known as a balanced, multiway tree, often referred to simply as a balanced tree. The balanced tree provides an efficient way to access the individual pieces of information associated with a global. Figure 8-1 illustrates the structure of a typical balanced tree.

Figure 8-1: Typical Balanced Tree



MR-S-852-80

The individual pieces of information associated with a global are called *data records*. DSM-11 treats all data records as ASCII strings - how you interpret a string depends on how it is used.

A data record can be of variable length. The maximum length of a data record is 255 characters. However, it is not necessary to predefine the maximum length of a data record. Each node of a global array can contain data that is different in size and content from other nodes in the array. Thus, one node can contain a short record while another node of the same global contains a concatenated string of information. For example:

`^X(156342,1) = Name`

`^X(156342,2) = Street Number__Street Name__Town Name__Zip Code`

Where 156342 represents an identifying number. The following example shows an alternate way to store the same information:

`^X("Last Name", "First Name", "Initial", 1) = Street Number__Street Name__Town Name__Zip Code`

`^X("Last Name", "First Name", "Initial", 2) = Data`

8.1.4 Programming Strategy

Global key and data record structure are linked directly to your application design. While it is not the purpose of this document to recommend techniques for designing a data base, the following programming considerations should be noted:

The subscripts you choose have a significant effect on the rate at which your globals grow, affecting both total storage required and data access efficiency.

For example, if you put data in your subscripts, such as a name or ID number, it conserves overall storage space because of key compression.

If practiced consistently, this could also improve data access time. Suppose, for example, you were writing a routine to store and retrieve demographic data. You could create a number of globals to store the data you required, such as:

`^DDN -- Name`

`^DDA -- Address`

`^DDT -- Telephone number`

Or, you could create one global, and assign meaningful subscripts to indicate the type of data each node contains, as in the following examples:

[^]DD(NAM) -- Name
[^]DD(ADR) -- Address
[^]DD(TN) -- Telephone number

The latter approach places only one entry in your global directory; this reduces the time it takes to sequentially search the directory each time you want to access a given record.

Thus, the use of few, large globals is typically more efficient than the use of many smaller globals. You save storage space because of key compression and decrease the time it takes the system to scan the global directory for the required entry. Also, if you have fewer globals in your data base there is a better chance of having a needed block in cache.

8.2 Accessing Globals Outside Your UCI With Extended Global References

You can access globals external to the UCI you are currently working in. These include globals from:

- Another UCI within the same volume set
- Another system through Distributed Data Processing
- Another data base volume set within the same system

You can access external globals with the extended global reference syntax described in the following sections. You can also use the UCI Translation Table, described in Section 8.2.4.

In the discussions of the extended syntax in Sections 8.2.1, 8.2.2, and 8.2.3:

UCI is the name of the UCI that contains the global you want to access.

GLOBAL REFERENCE is the name of the global that you access. The reference can be:

- To a simple global variable
- To a subscripted global variable
- A naked reference
- An indirect reference

If the UCI that you refer to is undefined you get a <NOUCI> error. If the remote system or volume set that you refer to is undefined, you get a <NOSYS> error.

8.2.1 UCI-to-UCI Global Access Within The Same Volume Set

The simplest form of the extended global reference syntax allows you to access globals in other UCIs which are part of the same volume set as your own UCI. To access a global in another UCI, you reference it in a command or routine line using the following syntax:

```
^["UCI"]GLOBAL REFERENCE
```

The cross-UCI syntax allows you to reference any global to which you have access privileges. If you try to access a protected global, the system returns a <PROT> error.

DSM-11 returns all related programming and operating system error messages when errors occur as a result of a global access across UCIs. You can use the entire DSM-11 command set when using the extended global syntax to access globals across UCIs.

Examples:

The following example writes the value of [^]RRR(1) in the UCI JPB to the current device.

```
> W ^["JPB"]RRR(1)
```

In the following example, the system returns a <PROT> error because the user tried to access a global to which he did not have WRITE privilege.

```
> S ^["MGR"]A(22)=11996
<PROT>
```

In the following example, the user is allowed to READ from a library global, but not WRITE to it.

```
> S A=^ZDATE(T)
> S ^ZDATE(T)=A
<PROT>
```

8.2.2 UCI-to-UCI Global Access Across Volume Sets

DSM-11 refers to mounted volume sets as "volume sets." Each mounted volume set has both a 3-character name and a position number in the volume set table. The system disk is volume one of the default volume set (Volume Set 0), whose name was assigned when DSM-11 was installed on the disk. You can access a global from other volume sets in addition to the one you are currently working on. These volume sets must be on the same system that you are logged in on. To access globals in other volume sets, you use the following extended global syntax in a routine or command line:

```
^["UCI","VOL"]GLOBAL REFERENCE
```

where:

VOL is the name of the volume set (on the same system) where the global resides. The name "VOL" is established when the structure is mounted, and is a 3-character uppercase alphabetic. Generally, the volume set name is read from the label block of the first volume of the set. When the volume set is mounted (by using the System Start-up utility, [^]STU, or the Disk Mount utility, [^]MOUNT), however, its name may be temporarily changed if it matches the name of another already mounted volume set.

8.2.3 Distributed Data Processing

Distributed Data Processing (DDP) is a feature of DSM-11 that permits access to globals on DSM-11 systems other than the one on which you logged in. DDP limits access to systems that are directly interconnected. This means that communication using DDP is point-to-point exclusively. In order to use the DDP facility, you have to include it in the system configuration during SYSGEN.

DDP communication is performed through the DMC11 synchronous communications controller. The DMC11 controller can also be used for data transfer with MUMPS READ and WRITE commands as described in Section 6.6. However, it cannot serve both purposes concurrently. A DMC11 in a particular configuration must be devoted exclusively to one function or the other.

To access globals on other systems, use the following extended global syntax in a routine or command line:

```
^["UCI","SYS"]GLOBAL REFERENCE
```


where:

SYS is the name of the system where the global is stored. (This is actually a logical label associated with a particular DMC11 line). SYS is established at SYSGEN, and is a 3-character uppercase alphabetic. Each DMC11 line can be assigned up to four different names, each name corresponding to one of the four possible mounted structures on the remote system.

Examples:

This example sets the variable B equal to the value of ^X(1,2) in the UCI MGR, on the system COS.

```
> S B=^[ "MGR", "COS" ]X(1,2)
```

Assuming the previous example was the last global reference (in a routine or command line), the following example is a naked reference that sets B(1) equal to the value of ^X(1,3) in UCI MGR, on the system COS.

```
> S B(1)=^(3)
```

8.2.4 Accessing A Global In Another UCI Through The UCI Translation Table

You can use a global outside of your current UCI by using the UCI Translation Table. When a global has been entered properly in this table, any reference to it is a reference to that global in a remote UCI, volume set, and/or system. To make an entry in the UCI Translation Table, you use ^UCITRAN from the System Manager's UCI, see Chapter 17.

Chapter 9

Optimizing DSM-11 Applications

This chapter describes procedures that decrease software overhead in DSM-11 code, and offers some guidelines for structuring DSM-11 applications.

9.1 Optimizing Routine Interpretation

There are several programming practices you can use to ensure that your DSM-11 routines are interpreted as efficiently as possible:

- Using a direct, top-down execution path (except for routines called very frequently by line label). See Section 9.1.1 for more information.
- Placing frequently called routines first, when using line labels to call routines. See Section 9.1.2 for more information.
- Using block-structured programming (the argumentless DO command), which executes faster than calling routines with the DO command (DO with an argument) because no label search is required. See the *DSM-11 Language Reference Manual* for more information on block-structured programming.
- Using the shortest interpretation path. Indirection is inefficient because it is a hidden addition to the interpretation path, especially when it requires a symbol look up. In general, indirection and XECUTEs should be used sparingly. See Section 9.1.6 for more information.
- Checking for the expected pattern first, when using pattern matching. See Section 9.1.5 for more information.

- Avoiding postconditionals on DO, GOTO, and XECUTE arguments, because they require more processing than postconditionals on the command itself. The interpreter must evaluate the argument even though it can not be used if the postconditional is false.
- Using increments of 1 with FOR loops, because these loops are optimized for this increment.
- Using short line labels and variable names. See Section 9.1.3 for more information.
- Abbreviating DSM-11 commands. See Section 9.1.4 for more information.
- Grouping accesses to the same global to take advantage of caching.

These practices decrease software overhead by using DSM-11 searching, storing and parsing mechanisms to the best advantage. Incorporating them in a logical, consistent manner ensures that your routines execute with maximum speed and efficiency.

9.1.1 Using A Direct, Top-Down Routine Execution Path

You should always try to write routines that have the shortest possible execution path. This applies to single, closed routines that do not make subroutine calls or to routines that make a number of subroutine calls. A short direct execution path optimizes routine interpretation and therefore decreases execution time.

One way to keep an execution path short, is to execute routine lines in a top-down fashion, avoiding the excessive use of GOTO and DO commands.

One way to accomplish this is by using block-structured programming, which promotes a clean top-down flow to the routine. Block-structured programming is discussed in detail in the *DSM-11 Language Reference Guide*.

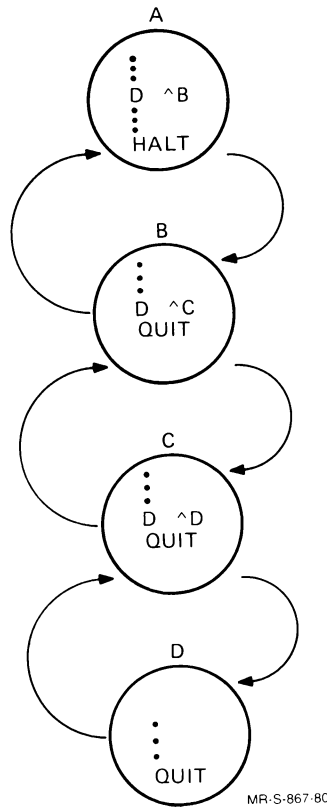
The DO command provides a general subroutine calling function. It should only be used when you want to call a closed routine. A closed routine executes to completion, then either HALTs or QUITs. If the routine is a subroutine, a QUIT transfers control back to the calling routine.

Avoid nesting DO commands in subroutines; that is, calling subroutines from other subroutines. Nesting DO commands can lead to an increase in software overhead.

This increase in overhead arises because DO commands return control to the calling routine, and not necessarily to the first routine in a routine hierarchy.

Figure 9-1 shows the control path of a DSM-11 application where a routine calls a series of subroutines using nested DO commands.

Figure 9-1: Routine Execution Path Using Nested Do Commands



In Figure 9-1, program "D" encounters a QUIT at its highest nesting level, and returns control to the command immediately following DO ^D in program "C". Since this command is QUIT, it returns control to the command immediately following DO ^C in program "B". This command is also QUIT, so it returns control to the command immediately following DO ^B in program "A".

Note that you *must* sequentially backtrack through each subroutine before returning to the body of the first program in the series. This increases execution time; it also makes routine logic hard to follow. When routine logic is confusing, it is harder to maintain the application where the routine is used.

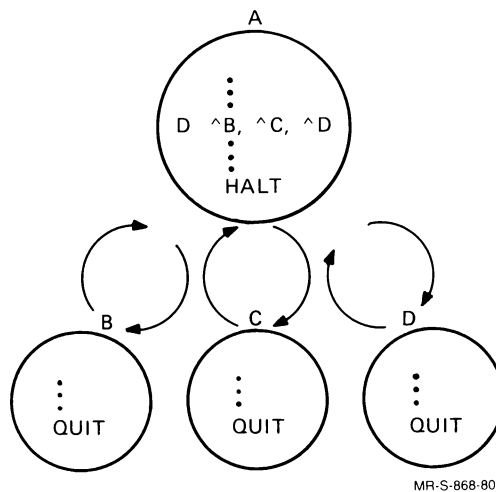
If you use a direct, top-down routine execution path, you can decrease the sequential backtracking associated with nested DO commands. This practice also makes routine maintenance easier and error recovery simpler. Suppose, for example that you rewrite program "A" in Figure 9-1 to include all subroutine calls. Program "A" will include a code segment such as:

LABLA D ^B, ^C, ^D

HALT

As a result, control only passes between one subroutine and the control program "A". Figure 9-2 illustrates the control path of a DSM-11 application that uses top-down structure. In Figure 9-2, the subroutine call sequence is the same as the one shown in Figure 9-1, that is, $A \Rightarrow B \Rightarrow C \Rightarrow D \Rightarrow A$. However, there is no sequential backtracking through routines.

Figure 9-2: Routine Execution Path Using Top-Down Structure



If a routine calls a number of subroutines, it should include an error-processing routine that prevents the program from aborting. This can be done by setting \$ZTRAP to an error-processing routine in the routine or in the subroutines. If \$ZTRAP is set in a subroutine and handles the error, the subroutine can continue to execute. Refer to Section 4.10 for a further discussion of error processing routines and error processing in a situation of nested DO and XECUTE commands.

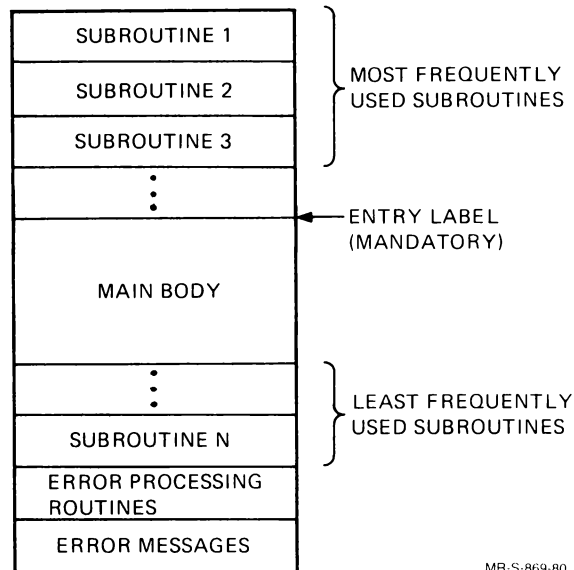
9.1.2 Optimizing Routines That Call Subroutines With Line Labels

The following practices optimize routines that call subroutines by a line label with GOTO and DO commands. If your routine uses the argumentless DO command, do not follow these practices. See the *DSM-11 Language Reference Manual* for more discussion of block-structured programming with the argumentless DO command.

DSM-11 always searches for line labels from the top of a routine. Therefore, when you write a routine that calls subroutines by line label, you should:

1. Place subroutines before and after the main body of the total routine structure (see Figure 9-3).
2. Place the most frequently used subroutines at the top of the routine structure.
3. Place infrequently used subroutines, error processing routines, and error messages after the main body of the total routine structure.

Figure 9-3: Optimum Code Structure of a Routine and Its Components



9.1.3 Using Short Line Labels And Variable Names

You can improve interpreter efficiency by selecting line label and variable names carefully. Try to choose a line label or variable name on the basis of a compromise between its readability and its length.

Line labels should be long enough to have meaning relative to the code segments they preface; variables should be long enough to have meaning relative to the type of information they contain.

However, line labels and variables should be short enough to minimize search time whenever the labels or variables are referenced. Suppose, for example, the following line label prefaced code that XECUTEd an expression:

```
XECUTNOW
```

The label XECUTNOW is legal since it is less than nine characters long. It might be just as meaningful, however, if it were truncated to:

```
XNOW
```

Moreover, this truncated label would save partition space and decrease search time when referenced in some other portion of the routine.

Section 9.1.5 describes some additional practices you can use to optimize local variable names.

9.1.4 Abbreviating DSM-11 Commands

You can abbreviate any DSM-11 command to its shortest unique form. Command abbreviations make DSM-11 code more efficient for the following reasons:

1. DSM-11 interprets command abbreviations faster than their full spellings.
2. Command abbreviations save partition space.

9.1.5 Optimizing Pattern Matching

The binary Pattern Match operator (?) checks that the pattern to its right is correctly satisfied by the string to its left. Patterns are specified by a sequence of special characters and/or string literals, each of which can be preceded by an integer constant or the period character (.). (The *DSM-11 Language Reference Manual* describes the pattern match codes in detail.)

The pattern match operator is generally used to verify the occurrence of a particular kind of response after a READ command. For example:

```
R "INPUT> ",X
```

where:

X is the input whose pattern needs to be determined.

Pattern matching is a slow operation for the DSM-11 interpreter. Pattern matching invokes a series of tests to determine if a match has or has not occurred. For instance, the interpreter may have to determine string length and character type (of which there are many). Therefore, to improve interpreter efficiency when using the pattern match operator, observe the following rule:

- When pattern matching after a READ command, check for the *expected* pattern before checking for other possible patterns.

In the following example, a routine expects a pattern consisting of 1 uppercase alphabetic character followed by 1 numeric character. The IF statement checks for this pattern using the pattern match operator. If a user enters the expected pattern, the routine immediately processes the request. If the input does not match the expected pattern, a number of tests are made to determine if the input matches other acceptable responses. If the input does not match any acceptable response, the routine prints an error message and returns to TAG1 to reinitiate the REQUEST sequence:

```
TAG1 R "REQUEST> ",I Q:I=""  
I I'?1U1N G TAGQ:I="BACK",HELP:I="HELP" G IR
```

(Process request)

```
G TAG1 ;GET NEXT RESPONSE  
  
IR W !,"INCORRECT RESPONSE - ENTER 'HELP' FOR MORE INFO" G TAG1
```

9.1.6 Optimizing Xecute And Indirection Usage

The XECUTE command and INDIRECTION (@) operator are powerful coding tools. However, they should be used sparingly, and in a calculated manner because their excessive use involves a trade-off between compactness of expression and speed of execution.

XECUTE and INDIRECTION are reservoirs of computational power:

1. XECUTE allows you to execute an expression as a subroutine.
2. INDIRECTION allows you to execute data values as DSM-11 code.

Nonetheless, there are drawbacks to using XECUTE and INDIRECTION indiscriminately. First, the DSM-11 interpreter requires a lot of time to resolve statements that use these commands.

The excessive use of XECUTE statements also makes a DSM-11 routine hard to maintain if it requires debugging, patching, or enhancing. XECUTE statements generally transfer control to other segments of code. This impedes top-down execution, confuses routine logic, and makes tracing a routine's execution path more difficult.

The indiscriminate use of INDIRECTION causes similar problems. First, extensive use of INDIRECTION causes the interpreter to rapidly search for, and insert indirect text. This can be time inefficient, particularly if you nest the

INDIRECTION operator, that is, if you execute indirect expressions defined in terms of other indirect expressions. For example:

```
>TAG   S COND="IF IND=$F(@NODE,@POS)"  
        I @COND W "FOUND"  
        E   W "NOT FOUND"
```

Nested INDIRECTION can also cause serious maintenance problems. The extensive use of INDIRECTION and nested INDIRECTION makes it almost impossible to trace a routine's execution path. This makes it very difficult for anyone other than the original programmer to update or enhance the routine, should the need arise.

9.2 Standardizing Application Structure

DSM-11 applications are typically modular. That is, they are constructed in segments. In a modular application, each module performs a separate, though related, task.

A module generally consists of a group of related routines. A group of related routines is called a *program*. A group of related programs is called a *package*; packages make up an application system.

The following sections describe some measures that you can use to standardize the structure of a modular application. Particular attention is given to interactive applications.

9.2.1 Using Programming Conventions

Programming conventions are essential to the orderly definition of any programming project. The use of programming conventions is especially important when a project is broken into segments and assignments are given to individuals within a project.

If a group of developers adhere to a well defined set of conventions during the programming phase of a project, it ultimately:

- Makes maintenance of a group of related routines easier
- Enables the individual segments of a routine, program, or routine package to come together in an orderly way for the final test phases of the project
- Makes future enhancements easier
- Makes a group of related routines perform as a unit in the eyes of the user

The following sections describe some suggested programming conventions. They can provide a starting point for the complete definition of conventions for a modular programming project. No attempt has been made to cover all possible conventions, since programming conventions inevitably vary from application to application.

9.2.2 Programming Conventions

Each routine of a DSM-11 program or routine package should adhere to a set of routine programming conventions in the following areas:

- The contents of the first routine line
- Comments
- Line labels
- The contents of the last routine line

For maintenance purposes, the first line of every routine in the system should be a comment that includes:

- The name of the routine
- The name of the program, package, and system that uses the routine
- The date when work began on the routine
- A brief, but informative description of the contents of the routine
- The name of the programmer

The first line convention summarizes the lineage of a routine. There are two additional routine coding conventions that you should use. Comments are the last element in a routine line. Comments are prefixed by a semicolon (;). For example:

```
label(TAB)code ;display demographic data
```

You should avoid the excessive use of comments. Comments are not interpreted, but they use as much partition space as their ASCII characters require. So you should only use comments to highlight something that is not obvious. When you must comment on your DSM-11 code, make it brief, yet informative.

9.2.3 Variable Naming Conventions

The first character of a DSM-11 variable must be an uppercase alphabetic character or the percent character (%). You should restrict the use of global "%" variables to general library routines or utilities. Doing this allows these variables to pass information between packages. Only the originating package should have modify and KILL privileges.

Global variables should begin with the same letter as the routine names with which they are associated. For example, the following globals are associated with a billing program:

```
^BAC  
^BF1  
^BF2
```

You should reserve global names of one character for use as scratch; that is, to store temporary information. One-character globals should also adhere to the first letter assignment rule for each package. This makes it easy to identify the programs with which the scratch globals are associated.

9.2.4 Terminal Conventions

Data base systems generally have routines that allow users to interact with the computer. This type of routine is called an *interactive routine*. The user perceives an interactive routine as a series of operations involving a:

- Query

followed by a:

- Response

Typically, a user enters or edits data during an interactive session, or he can use the computer to acquire information stored in the data base.

An efficient way to structure an interactive package is to provide access to its components through menus. In a menu-driven package, the system displays a group of options and the user selects those options that are appropriate to do a given task. This kind of interface makes an application appear to function as a unit in the eyes of its users.

NOTE

For an example of a menu-driven routine, you can examine the menu-driver of the DSM-11 utility package: ^%UTL.

The following paragraphs describe additional screen format techniques for the query portion of an interactive package. Later paragraphs describe input conventions for initiating common interactive routine procedures and conventions for responding to invalid user input. The query conventions apply to screen displays that roll. They do not apply to formatted block mode screen displays.

- Begin all query text on the left side of the terminal screen, allowing response text to expand toward the right. This provides a reading environment that users find familiar and easy to follow.

Also, a standard character, such as the right angle bracket (>) should be appended to the end of all query text. This character highlights the fact that a response is required of the user.

Place this character immediately following the query if the response is expected to fit on the same line. For example:

```
DATE> 7-JUL-80. RET
```

If the response is expected to exceed the screen width, place a line-feed carriage return in your DSM-11 code prior to the highlighting character. For example:

```
W "ENTER COMMENT"  
R !, "> ", T
```

This produces the following output:

```
ENTER COMMENT  
>Patient is responding to treatment. RET
```

that enables the user to enter substantially more text on the line.

- Interactive routines often require users to edit information in the data base. Therefore, the current data associated with a query should be displayed. You might place existing data between a set of right and left angle brackets immediately following the query prompt. For instance:

```
TELEPHONE NUMBER <555-2948>
```

The user should be able to update the information by simply typing new data after the trailing right angle bracket.

Display default values for particular queries in the same manner. In the following example, a display of the data includes a default fee.

```
FEE <12.00>
```

A successful interactive package addresses the needs of its users. This means that it must:

1. Handle invalid responses to queries
2. Provide on-line documentation
3. Enable a user to negate a response already entered
4. Enable a user to scroll through contiguous queries
5. Enable a user to leave a routine at any time except at critical points, and return to the menu of the program

The following paragraphs describe conventions that address each of these areas.

- Handle all input errors by displaying a standard message. For instance:

```
INCORRECT RESPONSE ENTER "?" FOR HELP
```

- Since application users occasionally respond to routine queries incorrectly, interactive packages should include a help facility. A HELP facility provides users with information about a query or group of queries. Display HELP text if a user responds to a query with a standard help convention, such as a question mark (?), or HELP. This text should be brief and to the point. The following is a good example of concise, yet meaningful HELP text:

```
CLASSIFICATION > ?
```

```
ENTER ONE OF THE FOLLOWING NUMBERS,  
ABBREVIATIONS, OR FULL NAMES:
```

```
001  SM   SYSTEM MANAGER  
099  PR   PROGRAMMER  
301  CP   CLERICAL PERSONNEL  
100  PHY  PHYSICIAN
```

HELP text can be embedded in DSM-11 code or placed in globals. A more efficient alternative, however, is to create one or more subroutines containing HELP text that are called whenever a user requests information.

In addition to HELP text, provide an input convention that lists related entries, if they exist. An appropriate symbol is:

`^L`

or:

`^LIST`

Obviously, this response is only valid at particular queries.

- **HELP** facilities reduce the number of times a user tries to enter invalid text. However, they do not limit the number of times a user enters acceptable, though unwanted text. Therefore, you should employ a terminal convention that deletes unwanted text. The minus sign (-) is a useful symbol for this procedure. In the following example, a user deletes a person's age, then the query is repeated:

```
AGE <52>-
```

```
AGE >
```

- It is often necessary for a user to scroll through contiguous queries. The circumflex, or "up arrow" (^) is particularly appropriate for indicating that you want to go to the previous query, because users can intuitively relate this symbol to the task it performs.

However, you should not have an unconditional proceed-to-next-query convention, because some queries may require a response from the user. Nonetheless, a null string ("") response could provide this capability under the right circumstances. You enter a null string by typing a carriage return without entering text.

If a query has a default value, or if data has already been entered, a null response could leave the current data unaltered; then the routine should advance to the next query. If neither a default value nor previously entered data exists, and if no response is required, a null response should return a user to the previous query (just as an ^would). The null response could also be taken as an invalid entry if a query requires a response.

- A menu-driven package must provide an input convention that returns a user to the menu of the program. For example:

`^Q`

or:

`^QUIT`

Treat **READ** commands that have timed out the same as the ^**QUIT** convention. This way, your application automatically backtracks toward the outermost menu. At the outermost **MENU**, include an autologout procedure to log the unused partition off the system.

Part 4: Managing the DSM-11 System

Chapter 10

Generating DSM-11

This chapter describes the DSM-11 SYSGEN procedure.

10.1 The SYSGEN Concept

SYSGEN refers to system generation and is the procedure you use to create a multiple-user DSM-11 system. SYSGEN is an interactive session during which you define the customized hardware and software configuration for your system.

SYSGEN provides you with two approaches for setting up a configuration:

- Autoconfiguration - you can follow this procedure during system installation; most options and choices are based on default parameters.
- Manual configuration - you can make choices about parameters and system features.

The products of manual configuration SYSGEN sessions are DSM-11 systems that you have selectively configured to meet your operating needs, and a system global that contains your system's configuration parameters. If you so desire, you can perform multiple SYSGENs. You can even perform a SYSGEN during DSM-11 system operation.

Before performing a SYSGEN for a manual configuration, you should have a list of hardware interrupt Vector and Control Status Register (CSR) addresses of all peripherals to be included in your configuration. This information can

be obtained from your Digital field service representative or from your *Site Management Guide*.

The Vector and CSR addresses are also provided when doing an autoconfiguration (this is done at first installation) or running the ^CONFIG utility. See Section 10.6 for more information.

You can run an autoconfiguration for SYSGEN provided that:

- You are running the Baseline System.
- You wish to define a new configuration (rather than edit an existing configuration).

Section 10.6 provides an example of an autoconfiguration run after the first installation.

You should also have some knowledge of common Digital hardware and software concepts. The *PDP-11 Processor Handbook*, the *Peripheral Handbook*, and the *Terminals and Communications Handbook* contain descriptions of the hardware used with your DSM-11 system. Refer also to your PDP-11 processor user's, system installation, or system operating guide.

10.2 System Definition Utilities

The SYSGEN procedure automatically calls two utilities to complete the SYSGEN session:

Utility	Routine Name
System Generation	^SYSGEN
Modify Basic System Parameters	^MBP

The Modify Basic System Parameters utility is called as Parts 11 and 12 of the system generation process. Several additional utilities discussed in this chapter add or modify system features or parameters.

Utility	Routine Name
Terminal Device Characteristics	^MUX
Add UCIs	^UCIADD
UCI Name Change	^UCIEDIT
UCI List	^UCILI
Magtape Default Mode	^MMD

Routine Mapping	^RMAP
Tied Terminal Table	^TTTG
UCI Translation Table	^UCITRAN
Autoconfiguration	^CONFIG

These utilities are discussed in Section 10.7, except for ^CONFIG which is discussed in Section 10.6.

The utilities discussed in this chapter can also be accessed through the SYSTEM UTILITIES menu (^SYS), or by typing:

```
>DO ^SYSDEF
```

10.3 The SYSGEN Procedure

When you initiate SYSGEN, you automatically start an interactive dialogue that covers all aspects of DSM-11 in a step-by-step manner. Some of the questions that SYSGEN asks are interdependent; thus, SYSGEN might ask additional questions as a result of the answer to a previous question.

If you are uncertain how to respond to a particular question, type a ?, and SYSGEN will display the acceptable responses.

You can also that request SYSGEN provide help information throughout the SYSGEN session without having to type a ? at each question. You can do this by answering YES to a question on "extended help" which is one of the first questions that you are asked.

10.4 The SYSGEN Session

The SYSGEN procedure is divided into sets of questions on a common subject. Within each set of questions, there are individual SYSGEN prompts and descriptions of their meaning and usage. All responses can be terminated with a carriage return. Typing an up arrow (^) causes SYSGEN to return to the preceding question. When you are ready to begin the SYSGEN procedure, type the following command:

```
>D ^SYSGEN
```

The twelve parts of the SYSGEN session are:

1. SYSGEN Configuration
2. Disk Information

3. System Devices
4. DMC11s Configuration
5. Software Configuration
6. Assign Device Numbers
7. Software Options
8. Memory Buffer Allocation
9. System Data Structures
10. Job Partition Definition
11. Data Base Parameters
12. Basic System Parameters

The following sections describe the DSM-11 SYSGEN options. The following sections also describe some of the planning you must perform to determine an optimum configuration for your installation. An actual example of a SYSGEN session is provided in Section 10.5.

10.4.1 SYSGEN Configuration

A configuration is a data base file created by the system generation procedure. The configuration completely describes the hardware devices present on your computer system, and the software options that you request.

This part of SYSGEN is a short series of questions asking you such things as whether you want to modify an existing configuration or start a new one.

You can create as many configurations as you like. When you boot the DSM-11 system, you are asked which configuration you want to run.

10.4.2 Disk Information

These questions allow you to specify disk types and units for your system. DSM-11 permits disk types to be mixed on the same system. You must list all disk drives in this part of SYSGEN except for RX02 floppy diskette drives, which are handled in the next part of SYSGEN.

10.4.3 System Devices

This part asks you to specify the system devices that are attached to the system. This includes such devices as magnetic tape drives, line printers, and peripheral controllers to which terminal-type devices are connected. Disk drives are not included in this section except for the RX02 floppy diskette drive.

DSM-11 supports both single-line and multiplexer terminal device controllers. DSM-11 can support a maximum of 17 single-line devices (3 to 19), in addition to device 1, which is the console terminal. When specifying a single-line device, you must specify in octal the Vector and the Control Status Register (CSR) addresses for the device. You must also specify the CSR address and the Vector address for each DMC11 device that is connected to your system.

DSM-11 can support any combination of DH11 or DZ11 multiplexers (devices 64 to 191) as long as the total number of lines does not exceed 128. If your system has multiplexers, you must specify in octal the Vector and CSR address for each multiplexer.

10.4.4 DMC11s Configuration

DMC11 controllers are used for communications with other systems, and thus are involved in Distributed Data Processing (DDP). In this part you answer questions about whether the DMC11 line is half or full duplex, and whether you are using a DMC11 for DDP.

HALF OR FULL DUPLEX

Whether a DMC11 is half or full duplex is related to the data connection between the DMC11 and its remote DMC11. Generally, DMCs linked by means of a coaxial cable will be running full duplex. DMCs linked by means of a telephone line may be run either full or half duplex. If you do not know how your DMC11 is configured, contact your Digital field service representative.

PRIMARY DMC

In half-duplex operation, one side of the DMC11-to-DMC11 connection must be designated as the primary.

DISTRIBUTED DATA PROCESSING

You must indicate whether you want one or more of your DMCs to be used for Distributed Data Processing (DDP).

DDP is a facility that allows access to global arrays on a remote DSM-11 system by means of the DMC11 communications device. A routine can request data from the remote system using an "extended" global reference syntax. See Section 8.2.3 for information on this syntax for DDP.

DMCs can be used for DDP communications or as single-line CPU-to-CPU communications devices.

Systems interconnected by means of DDP must have an identifying code which is used in the extended global reference to access data. This code must be a 3-character uppercase alphabetic code.

Any DMC controllers that you do not assign to DDP are assigned as single-line communications devices.

10.4.5 Software Configuration

In this part, you indicate whether you want to have the standard software options or choose your own software options. In part 7 of SYSGEN, Software Options, you actually get to choose those options.

The standard system includes:

- Journaling
- Interjob communications
- Sequential Disk Processor (SDP)
- Mountable data base volume sets
- UCI Translation Table
- Default sizes for all system data structures
- Default values for basic system software parameters

The standard system does not include:

- Mapped routines
- Executive Debugging Tool (XDT)
- Spooling

The standard system may or may not include:

- EBCDIC support
- Loadable driver support

The EBCDIC support is included if you have magnetic tape drives, RX02 drives, or Binary Synchronous Communications (BISYNC) included with your system. The loadable driver support is provided if you have a RX02, TU58, DUV11, or DUP11 device connected to your system. The DUV11 and DUP11 devices are communications devices used by the BISYNC driver.

The amount of memory each of these options occupies is dependent on the size of the option module and any selectable buffer sizes associated with the module.

Some hardware configurations cannot handle all options because of memory constraints on system executive size. In this case, you can select those options that are most important to you. You can also design configurations with different choices of options. For a particular purpose or situation, you can run a specific DSM-11 configuration. You can also set up configurations to maximize the amount of space in memory by minimizing the number of options.

10.4.6 Assign Device Numbers

SYSGEN automatically assigns device numbers for single-line communications devices (DUV11 or DUP11) and the TU58. The DUV11 and DUP11 are used by the BISYNC driver for communications with other computers. In this part, you can change these device numbers (within certain limits). Refer to the SYSGEN dialogue and Sections 5.2 and 5.3 for more information on DSM-11 device numbers.

10.4.7 Software Options

These questions allow you to determine which software modules are to be included in your system. You should respond with either Y or N for each option. If you type a carriage return, you receive the default value.

Remember that each software option takes up space in memory.

SEQUENTIAL DISK PROCESSOR

The Sequential Disk Processor (SDP) allows you to store and retrieve data from sequential storage on disk. See Section 6.4 for more information on the SDP.

JOURNALING

Journaling provides a record of all data base writes. This record is useful in case of a system failure. Journaling, described in Chapter 16, can be used with backing up disks, described in Chapter 11, to provide for total data base recovery after a failure.

The journal procedure requires 1K byte buffers in memory to hold records of each data base WRITE transaction. When a buffer is full, the journal job marks the buffer to be written to the selected output device. If more than one buffer has been allocated to journaling, the journal procedure immediately begins filling the next free allocated buffer with data base transactions. If no allocated buffers are free, jobs attempting to journal transactions are put into a wait queue until a buffer becomes free.

Allocating more than one buffer to journaling can increase journaling efficiency by "streaming" blocks to the output device. However, the block buffers are allocated directly from the system disk buffer cache, so that memory space and disk efficiency must be traded against journaling requirements. Buffer allocation is also discussed in Section 15.2.

INTERJOB COMMUNICATIONS

Interjob communications (JOB COM) allows two DSM-11 jobs to communicate by means of assigned JOB COM channels using MUMPS READ and WRITE commands.

Each JOB COM channel is a pair of pseudodevices that communicate by means of a single ring buffer. There are 32 device numbers (224-255) reserved for a total of 16 channels. Each pair consists of an even-numbered device (the receiver), and the following odd-numbered device (the transmitter).

Each JOB COM channel requires a ring buffer of 2 to 255 bytes. Ring buffers are intermediate storage buffers for data being transferred from one job to another. When the ring buffer is filled, the job transmitting the data must be placed in a wait queue until the receiving job has removed some of the data. While it may be desirable to increase communications speed by increasing ring buffer size, this must be traded against memory space considerations.

EBCDIC TRANSLATION TABLE

The EBCDIC translation table provides translation of ASCII characters to EBCDIC characters and ASCII to EBCDIC conversion. Magnetic tape, RX02, and TU58 can use the table. The Binary Synchronous Communications (BISYNC) driver requires this table.

LOADABLE DRIVERS

Loadable drivers are device drivers that can be loaded or unloaded dynamically while the system is running. You are asked whether or not to provide space for loadable drivers in your system. You need this space if you are going to use the DSM-11 loadable drivers, or if you have loadable device drivers that you write yourself.

The TU58 cassette tape, RX02 diskette, and BISYNC drivers are all provided as loadable drivers. The BISYNC driver is used with either a DUP11 or DUV11 communications device. See the *DSM-11 BISYNC Programmer's Guide* for more information on the BISYNC driver. If you plan to use these drivers, you must select this option.

The approximate space needed for each driver is 2K bytes for the TU58 driver, 2K bytes for the RX02 driver, and 4K bytes for the BISYNC driver. This space requirement can increase if the drivers are modified.

The space set aside for loadable drivers can be increased or decreased depending on the requirements of the particular installation. If system memory space is limited, and several loadable drivers have been selected, this space can be reduced to free up system memory. Reducing loadable driver space may limit your ability to load all required loadable drivers in memory at the same time. You will, however, be able to load a driver, use the device, unload the driver, and load another driver.

You may want to increase loadable driver space if some special requirement exists for memory space within the system image.

These drivers can be loaded or unloaded with the `^LOAD` or `^UNLOAD` utilities (described in Section 12.8), or automatically installed at start-up. You must, however, reserve space for these drivers during the SYSGEN session (before loading them).

EXECUTIVE DEBUGGING TOOL

The Executive Debugging Tool (XDT) allows you to use the console terminal to interrupt the system, directly access any location in memory, and set breakpoints within the DSM-11 executive. XDT can also show machine registers, stacks, and status locations. For more information on XDT, see the *DSM-11 XDT Reference Manual*

MAPPED ROUTINES

Mapped routines allow you to lock a set of DSM-11 routines into memory thereby reducing the system overhead required to load a routine into a user's partition when the routine is called. This can greatly increase system performance when application users are sharing the same routines. However, you must reserve memory space in which to load these mapped routines. Each routine must be no larger than 8K bytes. For more information on mapped routines, see Section 15.1. See Section 10.7.6 for discussion of the Routine Mapping utility, `^RMAP`.

UCI TRANSLATION TABLE

The UCI Translation Table provides a flexible way to logically change the UCI and system names for one or more specific global arrays. You can make a Translation Table entry that assigns a new UCI and system name to any

global. The system makes each reference to that global through the Translation Table using the new UCI and system name. The UCI Translation Table utility, [^]UCITRAN is discussed in Chapter 17.

MOUNTABLE VOLUME SETS

DSM-11 allows up to four mountable data volume sets to be resident on the system at any one time. These volume sets are complete and independent data bases. A volume set can consist of one or more disk packs, and can have one or more UCIs. At least one volume set, the system volume set, is mounted on a DSM-11 system. See Sections 13.2, 13.3, and 14.7 for more information on volume sets.

10.4.8 Memory Buffer Allocation

These questions allow you to allocate memory to ring buffers and to disk-tape buffers. For additional information on buffer allocation, see Section 15.2.

RING BUFFERS

All terminal-type devices (1,3 to 19, 64 to 191) and job communication devices require ring buffers. These buffers are taken from the ring buffer pool and are allocated on a first come, first served basis. Most devices require two ring buffers, one for input and one for output. Ring buffer size for DSM-11 cannot be less than 32 bytes or no greater than 255 bytes.

When a ring buffer is filled, the job transmitting the data must be placed in a wait queue until the terminal has received some of the data. While it may be desirable to maximize a job's I/O throughput by increasing ring buffer size, this must be traded against memory space considerations.

The default ring buffer size is the size of the buffer acquired when a terminal is first opened. A different buffer size can be selected at open time by means of an optional OPEN parameter. Memory will be most efficiently used if the buffer size is specified in multiples of 32 bytes.

The ring buffer pool is the memory space from which terminal and JOBCOM ring buffers are allocated when a device is opened or logged in to. Space is allocated in bytes, and the ring buffer pool must hold at least two default-sized ring buffers. Your system requires a total of two ring buffers for each active terminal and for each active JOBCOM channel. This is the default presented by SYSGEN.

DISK-TAPE BUFFERS

The disk-tape cache blocks are 1K byte block buffers used for disk, tape, journaling, VIEW, DMC block mode, TU58, RX02, BISYNC, and all SDP

disk operations. These buffers reside in a common buffer pool. The nonallocated disk-tape buffers are used by the data base handler for storage of the memory-resident portions of the data base.

You must allow for seven cache blocks plus the number of buffers selected for journaling. You can have as many as 200 disk-tape buffers.

10.4.9 System Data Structures

This part includes questions on allocating space for the DSM-11 Lock Table, mountable data base volume sets, and mapped routines.

LOCK TABLE

The DSM-11 Lock Table maintains a list of all local and global variables that have been locked by the **LOCK** and **ZALLOCATE** commands. Refer to Section 14.4 for additional information about the DSM-11 Lock Table.

The size of the DSM-11 Lock Table cannot be less than 64 bytes or greater than 8192 bytes, and must be a multiple of 64.

A job requesting a **LOCK** must wait for space in the Lock Table if the Lock Table is full. When you allocate memory to the Lock Table, you must make a tradeoff between remaining memory space and the number of data elements you expect to be concurrently locked.

MOUNTABLE VOLUME SETS

In DSM-11, a volume set is a single or multiple disk set that appears to the system as a single logical data base.

Your system is allowed up to four concurrently mounted data base volume sets. Each data base volume set contains disk volumes and has its own UCI table. Volume Set 0, of which the system disk is the first volume, remains mounted at all times. Up to three additional volumes or volume sets, however, can be mounted and dismounted as volume sets 1 through 3. You can specify that space be reserved for one, two or three additional UCI tables. The only reason for not specifying the maximum is to save memory (1024 bytes/UCI table), at the cost of limiting the number of concurrently mounted data base volume sets.

Notice that this limitation does not apply to the mounting of disks for **VIEW**-only purposes, which is required, for example, to prepare new volumes for backup. Disks mounted for **VIEW**-only are not data base volume sets, and no global or routine access is allowed to or from those disks.

Refer to Sections 13.2, 13.3, and 14.7 for more information on volume sets.

MAPPED ROUTINES

The mapped routine option is a facility for providing a memory cache for frequently used routines. These routines then reside in common system memory rather than in each user's partition.

This serves to reduce the system overhead because a mapped routine is not loaded into a user's partition when the routine is called. This can greatly increase system performance in situations in which application users are sharing the same routines. However, you must make a tradeoff between space allocated to mapped routines and space required for job partitions. Mapped routines are discussed in more detail in Section 15.1.

10.4.10 Job Partition Definition

These questions allow you to allocate memory for partitions. Chapter 14 contains additional information concerning DSM-11 partitions. In DSM-11, partitions are allocated in increments of 1K bytes. Partition size can range from a minimum of 1 increment to a maximum of 16 increments. During this session, you must specify the default partition size that is to be used on log-in and JOB commands. (Note that for the JOB command, the partition size for jobs started with the JOB command is specified in one-half K byte increments.) You must also specify the number and sizes (in bytes) of all fixed partitions that are to be included in your system. The maximum number of partitions you can specify is 63. When you log in, you can override the default partition size.

If there is memory remaining at the end of this session, DSM-11 automatically assigns it to the dynamic partition pool. From the dynamic partition pool, you can create partitions of any allowable size when you log in the system. The largest allowable partition size you can specify is 16384 bytes.

Generally, you should allocate as much space to the partition pool as possible. Specific partitions can be useful in cases where a background or batch job always requires a particular partition size different from the default.

10.4.11 Data Base Parameters (^MBP)

These questions and those in Part 12 of SYSGEN (Basic System Parameters) cover subject areas that are also presented in the Modify Basic Parameters utility (^MBP). This part (Part 11) asks basic questions about data base parameters (for disks) such as write-checking for disks, disk space reserve, and default global characteristics.

DISK WRITE CHECK

Enabling write check causes DSM-11 to read every block it writes to disk immediately following the write. This is done to see if the block was written correctly. The system attempts to correct any errors detected. The system informs the System Manager if there is an error. Selecting this feature gives you immediate information on the status of disk writes; however, it does increase disk I/O overhead.

DISK BLOCKS FOR DISK RESERVE

Allocation of disk blocks to global arrays can occur during any global SET command. If the disk is entirely full, attempts to allocate a block during the SET operation will fail, and the SET may not complete, possibly causing a degradation of the data base.

To avoid this problem, a certain number of disk blocks on the disk volume set should be held in reserve. When all disk blocks in the volume set have been allocated, a block is allocated from the reserve; the SET operation is completed; and <DKRES> error is given to the job which caused the SET. The System Manager must then provide additional disk space.

A maximum of 399 blocks can be reserved per disk volume set. This systemwide value is used only on disk volume sets greater than two megabytes. A fixed value of 10 blocks is assigned to smaller volume sets.

DEFAULT GLOBAL CHARACTERISTICS

System default global characteristics are:

8-Bit Subscripts:	Yes
Journaling:	Yes
Collating Sequence:	Numeric

Default global characteristics are applied to all newly created globals on a systemwide basis. While these are the default characteristics, globals with different characteristics can still be created using the Global Management utility, ^%GLOMAN.

10.4.12 Basic System Parameters (^MBP)

These questions cover a variety of subject areas that are also presented in the Modify Basic System Parameters utility (^MBP). They are included as Part 11 and Part 12 of the system generation process. After calling ^SYSGEN, you do not have to call ^MBP; ^SYSGEN automatically calls ^MBP. You can also call ^MBP after you have established an initial configuration. You then use

this utility to modify the basic system parameters independently of the system generation process. Specifically, this section asks questions concerning:

- VIEW Mode restriction
- ZUSE restriction
- Log-in echo
- Default application interrupt key
- Default programmer abort key
- Power fail restart delay
- Disconnect delay
- Number of significant digits in division computations
- Power line frequency
- Programmer access code

VIEW PROTECTION

VIEW protection allows the use of the VIEW command only from the System Manager account and LIBRARY utilities (routines with names that begin with the % character). This is generally a good idea since the VIEW command should be used carefully. You can use the VIEW command to modify disk blocks or memory. See the *DSM-11 Language Reference Manual* for more information on the VIEW command.

ZUSE PROTECTION

ZUSE protection allows the use of the ZUSE command only from the System Manager account and LIBRARY utilities. The ZUSE command can be used to write messages to terminals other than the one you are logged in on. This command can be abused or can be a nuisance if there are a large number of users on the system. See the *DSM-11 Language Reference Manual* for more information on the ZUSE command.

LOG-IN SEQUENCE ECHO

The log-in sequence echo causes all characters typed during login to be echoed on the terminal. You can improve the security of your system by specifying that the echo be disabled during the log-in sequence. You must specify disabling, because the default is to enable echoing during the log-in sequence.

DEFAULT APPLICATION INTERRUPT KEY

The application interrupt key, when enabled, causes an <INRPT> error to occur. The error can then be trapped with an error processor and handled as a special event. See Section 4.7 for more discussion of the application interrupt key.

When a terminal device is first opened, it is given the systemwide default application interrupt key. An OPEN command parameter allows you to define one or more interrupt keys for that specific terminal.

The application interrupt key can also be used to log in a nonautobaused terminal. The return key can be used to log in any terminal.

The key must be a control character with a decimal value between 0 and 31. The default value of 3 gives an application interrupt key of `CTRL/C`.

DEFAULT PROGRAMMER ABORT KEY

The programmer abort key is available only to programmers logged in to DSM-11 in Programmer Mode. When pressed, the key causes an <ABORT> error to occur. Any error-processing routines are bypassed, and the execution of all routines are terminated. See Section 4.7 for more information on the programmer abort key.

The key must be a control character with a decimal value between 0 and 31. The default value of 25 gives an abort key of `CTRL/Y`.

SYSTEM RESTART AFTER A POWER FAILURE

DSM-11 restarts automatically when power is restored after a power failure, if memory is still intact. But if the power failure is more than momentary, main memory is erased unless it is "nonvolatile" memory. Often machines without nonvolatile memory can recover because the power failure did not destroy memory. The time delay is necessary to allow disk drives to become ready. If you wish to restart under operator control, set the time to 0. Otherwise, enter a time interval up to 500 seconds. Note that, if power-fail restart occurs while magnetic tape journaling is in progress, operator intervention is demanded.

TELEPHONE DISCONNECT DELAY

The telephone disconnect delay allows a time delay between logout and telephone line disconnect. This is very useful in dial-up environments where it is necessary to log out one UCI and log in another. If the delay is set to 0, DSM-11 disconnects the telephone line immediately following logout.

SIGNIFIGANT DIGITS IN DIVISION COMPUTATIONS

The number of significant digits returned from a division operation affects both accuracy and speed of computation. Increasing the number of significant digits increases the degree of accuracy, but also requires more processor time.

DSM-11 supports computations which yield between 15 and 31 significant digits.

LINE FREQUENCY

The frequency of the computer's power source is either 50 HZ or 60 HZ. The power line frequency directly affects the timing interval (or time between ticks) of the DSM-11 internal clock. At 60 HZ, one tick is equal to 16.6 milliseconds. At 50 HZ, one tick is equal to 20 milliseconds.

Because DSM-11's time-sharing interval (standard time slice) is based on ticks, the power line frequency directly affects the amount of CPU time allotted to each active job (terminal user or routine) in the system. To specify a 50 HZ power line frequency, type N. To select a power line frequency of 60 HZ, type Y. The default is 60 HZ.

PROGRAMMER ACCESS CODE (PAC)

Enter the 3-character password that lets users enter Programmer Mode. Use of control characters (those that have an ASCII value less than 32), while allowable, could confuse the system or have an undesired result because of the special functions these characters can have. See Section 3.1.2 for more information on these special characters. See Section 3.2.2 for more information on the PAC.

10.5 Example SYSGEN

Answers to each SYSGEN question must be followed with a carriage return. In general the carriage returns are not indicated in this example. A carriage return alone causes the default answer to be used.

On any question asked during SYSGEN, you can enter:

^ to return to the previous question
? for additional help

If a value appears between angle brackets, < >, it is a default value. Typing a **RET** causes the default value to be used as the answer to the question.

System generation for DIGITAL Standard MUMPS

Type ? for HELP at any time

PART 1: SYSGEN

See Section 10.4.1 for more information on this part of SYSGEN.

1.1 Would you like extended help [Y OR N] ? <N> N

1.2 Enter the configuration identifier <11/34> *

Type * to see a list of existing configurations.

Defined configurations are:

```
11/34
AUTO
NEW
TEST
XXX
```

Type the name of the configuration that you wish to edit or create. When you type in a name that is not already defined, you create a new configuration. Your answers to the SYSGEN questions become the values of this configuration.

To modify an existing configuration, simply enter the name of that configuration. The default answers to the SYSGEN questions are the existing values for that configuration. By typing in a different answer to the question, you modify that value in the configuration.

The name must not exceed 12 characters and must not be the value 0. Note that names that are the same except for differences in uppercase or lowercase are treated as separate configurations. Use the configuration name whenever you wish to refer to this configuration.

1.2 Enter the configuration identifier <11/34> SST

1.5 Enter the processor type > 11/34

Type in the type of processor for this configuration. The answer should be in the form:

```
11/XX          if the processor is a PDP-11/XX
MICRO         if the processor is a MICRO/PDP-11
```

Processors supported include all PDP-11s from the 11/23 to the 11/70, and the MICRO/PDP-11.

This machine has 248K bytes of memory

1.6 How many K bytes do you wish this configuration to have ? <248>

Press **(RET)** to accept the default value for the amount of memory for this configuration.

Type the value in K bytes, if you wish to specify a different value for the memory size. The value must be at least 128K bytes, but not more than 4088K bytes (4 megabytes), and must also be a multiple of 8. The value cannot be larger than the amount of memory physically available on your system.

PART 2: DISK INFORMATION

See Section 10.4.2 for more information on this part of SYSGEN.

Please enter all the disk drives in your system using the form:

DRIVETYPE=NUMBER

Press RETURN to end the list.

2.1 DISK DRIVE >*

Type "*" to get a list of supported drives.

The following disk drives are supported:

RA60	RA80	RA81	RD51
RK05	RK06	RK07	RL01
RL02	RM02	RM03	RM05
RM80	RP04	RP05	RP06
RX50			

2.1 DISK DRIVE >RL01=1

2.1 DISK DRIVE >RL02=2

2.1 DISK DRIVE >

Type in all of the disk drives (in question 2.1) on the system you are configuring. The answers shown in this example are for a system with one RL01 and two RL02s.

To delete a drive from the list type:

DRIVETYPE=0

Type "L" to get a list of devices already entered.

PART 3: SYSTEM DEVICES

See Section 10.4.3 for more information on this part of SYSGEN.

3.1 How many magnetic tape units are there (Max = 4) ? >1

Type the number of magnetic tape drives that you have on the system.

3.2 What type is magnetic tape unit 0 ? >TS11

Type the type of magnetic tape drive. The supported types are:

TU80 TS03 TS11 TSV05 TU10
TE10 TE16 TU16 TU45 TU77

The following rules apply when configuring tape drives:

- If unit 0 is TS11/TU80/TSV05, all units must be TS11/TU80/TSV05.
- If unit 0 is TS03/TU10, other units can be TU10/TS03, or TS11/TU80/TSV05.
- If unit 0 is TE16/TU16/TU45/TU77, other units can be TE16/TU16/TU45/TU77 or TS11/TU80/TSV05.

Enter the VECTOR address, in OCTAL, for Tape Unit 0 >240

Enter the CSR address, in OCTAL, for Tape Unit 0 <172520>

Type in the Vector address for the magnetic tape unit. The Vector address is an octal number in the range 0 to 770 and is generally a multiple of 10.

Type in the CSR address for the magnetic tape unit. The Control and Status Register (CSR) address is an octal number in the range of 160000 to 177540.

If you do not know the Vector or CSR address of this hardware device, contact your DIGITAL field service representative or use the autoconfiguration option of SYSGEN from the Baseline System. Do not use the numbers given in this example; they may not be correct for your system. See Section 10.1 for more information on obtaining Vector and CSR addresses, and Section 10.6 for an example that provides these addresses.

3.3 How many LP11s are there (max = 8) ? <0>1

Type in the number of LP11 line-printer controllers in this configuration.

Enter the VECTOR address, in OCTAL, for LP11 controller 1 >260

Enter the CSR address, in OCTAL, for LP11 controller 1 >161200

3.4 How many DL11s are there (max = 17) ? <0>1

Type the number of DL11 single-line asynchronous controllers in this configuration. Do not include the console DL.

Enter the VECTOR address, in OCTAL, for DL11 controller 1 >310

Enter the CSR address, in OCTAL, for DL11 controller 1 >176510

3.5 How many DMC11s are there (max = 4) ? <0>2

Type the number of DMC11 synchronous controllers in this configuration. Be sure to enter all DMCs regardless of their use in the system.

Enter the VECTOR address, in OCTAL, for DMC11 controller 1 > 340
Enter the CSR address, in OCTAL, for DMC11 controller 1 > 160110

Enter the VECTOR address, in OCTAL, for DMC11 controller 2 > 350
Enter the CSR address, in OCTAL, for DMC11 controller 2 > 160120

3.6 How many DH11s are there (max = 8) ? <0>

Type the number of DH11 16-line asynchronous multiplexers in this configuration.

Note that not all questions are asked in this part of SYSGEN; for example, question 3.7 is not asked. These questions are for devices that are not physically present in this particular system.

3.8 How many DZ11s are there (max = 16) ? <0> 1

Type the number of DZ11 8-line asynchronous multiplexers in this configuration.

Enter the VECTOR address, in OCTAL, for DZ11 controller 1 > 360
Enter the CSR address, in OCTAL, for DZ11 controller 1 > 160140

3.9 How many DZV11s are there (max = 32) ? <0> 0

3.10 How many RX02s are there (max = 1) ? <0> 1

Type the number of RX02 dual-density diskette controllers in this configuration. (Note that one controller supports two diskette drives.)

Enter the VECTOR address, in OCTAL, for RX02 controller 1 > 264
Enter the CSR address, in OCTAL, for RX02 controller 1 > 177170

3.11 How many TU58s are there (max = 1) ? <0> 1

Enter the number of TU58 cassette tape controllers in this configuration. (Note that one controller supports two cassette drives.)

Enter the VECTOR address, in OCTAL, for TU58 controller 1 > 300
Enter the CSR address, in OCTAL, for TU58 controller 1 > 176500

3.13 How many DUP11s are there (max = 4)? <0> 0

PART 4: CONFIGURE DMC11s

See Section 10.4.4 for more information on this part of SYSGEN.

4.1 Is DMC controller 1 HALF DUPLEX [Y OR N] ? <N> Y

Indicate whether the DMC11 is to be half or full duplex.

4.1 Is DMC controller 2 HALF-DUPLEX [Y OR N] ? <N> Y

4.3 How many of your DMC11s do you wish to use for DDP [Y OR N] ? > 2

At this point you indicate the number of DMC11s you want to use for DDP. If you have more than one DMC11, questions 4.4 through 4.6 are repeated for each DMC11 line.

4.4 Which DMC-11 controller do you wish to use for DDP line 1 > 1

4.5 Enter the 3-letter code for DDP line 1 > AAA,BBB,CCC,DDD

The code must be unique for each line since the code is used by DSM-11 routines to specify that line for extended global references. The codes can also be used in the UCI Translation Table. The code must contain 3 uppercase alphabetic characters. These codes should refer to volume sets that reside on another PDP-11 system connected by DMC11s. There can be as many as 4 codes, referring to the 4 volume sets that can be mounted on a remote system.

4.6 Will DDP line 1 be connected to a Version 2 system [Y OR N] ? <N>

If the line is to be connected to a Version 2 system, you should answer this question Y to insure compatibility between the Version 3 and Version 2 systems.

4.4 Which DMC-11 controller do you wish to use for DDP line 2 > 2

4.5 Enter the 3-letter code for DDP line 2 > EEE,FFF,GGG,HHH

4.6 Will DDP line 2 be connected to a Version 2 system [Y OR N] ? <N>

PART 5: SOFTWARE CONFIGURATION

You can choose to take the standard software options which greatly reduce the number of questions asked to configure this system. This "standard" system is the one given during autoconfiguration. See Section 10.4.5 for more information on this part of SYSGEN.

Answer Y if you wish to have the standard system.

If you answer N to this question you are asked specific questions for each option in Part 7 of SYSGEN.

5.1 Do you wish to use the STANDARD SOFTWARE OPTIONS [Y OR N] ? <Y> N

PART 6: ASSIGN DEVICE NUMBERS

Single-line devices are numbered from 3 to 19 in DSM-11. The DL11, LP11, and DMC11 (when used as a single-line device, not for DDP) are all considered single-line devices. See Section 10.4.6 for more information on this part.

The following single-line device assignments have been made:

Device Number	Controller-Number
3	LP11-1
4	DL11-1
5	DMC11-2

If you would like to make your own device assignments to these controllers, answer Y in the next question. You are then asked questions giving you the opportunity to reassign the device number for each of these devices. Otherwise, these default assignments remain.

6.1 Do you wish to edit these assignments [Y OR N] ? <N> N

The following LOADABLE DRIVER device assignments have been made:

Device Number	Controller-Number	Unit
51	RX02-1	0
52	RX02-1	1
53	TU58-1	0
54	TU58-1	1

The loadable drivers are the TU58, RX02, and BISYNC drivers. The BISYNC driver uses either the DUP11 or DUV11 communications device. DSM-11 device numbers for the related hardware devices (RX02, TU58, DUV11, and DUP11) can be assigned values between 51 and 58. SYSGEN automatically makes these assignments; you can then edit them to change the numbers (as long as they remain between 51 and 58). For more information on the BISYNC driver, see the *DSM-11 BISYNC Programmer's Guide*.

Do you wish to edit these assignments [Y or N] ? <N> N

PART 7: SOFTWARE OPTIONS

See Section 10.4.7 for more information.

7.1 Include support for SEQUENTIAL DISK PROCESSOR [Y OR N] ? <Y> Y

7.2 Include support for JOURNAL [Y OR N] ? <Y> Y

The journal procedure requires 1K byte buffers in memory to hold records of each data base WRITE transaction.

Enter the number of DISK-TAPE buffers to allocate to JOURNAL? <2>

7.3 Include support for SPOOLING [Y OR N] ? <Y>N

7.4 Include support for INTERJOB COMMUNICATIONS [Y OR N] ? <Y>Y

Interjob communications (JOB COM) allows two DSM-11 jobs to communicate by means of assigned JOB COM channels using MUMPS READ and WRITE commands.

How many communication channels do you want ? <16>

Each JOB COM channel requires a ring buffer of 2 to 255 bytes.

Enter the default size you would like for JOB COM ring buffers. Ring buffers use memory most efficiently if specified in multiples of 64 bytes.

Enter the size, in bytes, for each RING BUFFER channel ? <64>

7.5 Include support for EBCDIC-ASCII TRANSLATION TABLES [Y OR N] ? <Y>Y

7.6 Include support for LOADABLE or USER DRIVER SPACE [Y OR N] ? <N>Y

Enter the number of BYTES to allocate for LOADABLE DRIVERS ? <0>5000

A typical amount of space to allocate is 5000 bytes. This insures that there is enough space for any one driver to be loaded at one time. If you choose the standard software options (see Part 5), then the amount of space needed is automatically calculated, assuming you have the correct hardware devices connected to your system for the loadable drivers.

7.7 Include support for EXECUTIVE DEBUGGING TOOL [Y OR N] ? <N>N

7.8 Include support for MAPPED ROUTINES [Y OR N] ? <N>Y

7.9 Include support for UCI TRANSLATION TABLES [Y OR N] ? <Y>N

7.10 Include support for MOUNTABLE DATA BASE VOLUME SETS [Y OR N] ? <Y>Y

Total System Exec size: 67.45K Bytes

PART 8: MEMORY BUFFER ALLOCATION

Space remaining for Buffers and Partitions: 179.30K Bytes

Maximum space that can be used for Buffers: 169.30K bytes

See Section 10.4.8 for more information on this part of SYSGEN.

8.1 Enter the default terminal RING BUFFER size <64>

Ring buffer sizes can not be less than 32 bytes or more than 255 bytes.

8.2 Enter the number of bytes to allocate to RING BUFFER space (1280)

Your system requires a total of two ring buffers for each active terminal and for each active JOBCOM channel.

Space remaining for Buffers and Partitions: 177.99K Bytes

Maximum space that can be used for Buffers: 168.00K bytes

8.3 Enter the number of DISK-TAPE cache blocks to allocate (89)

The disk-tape cache blocks are 1K byte block buffers. You must allow for seven cache blocks plus the number of buffers selected for journaling. See Section 10.4.7 or 15.2 for more information on allocating buffer space. There is a theoretical maximum of 200K bytes for these buffers, but the actual amount of space is limited by system constraints and cannot be larger than the number immediately preceding this question (168K bytes in this example).

PART 9: SYSTEM DATA STRUCTURES

Space allocated for DISK-MAP and BAD BLOCK TABLE: 704 Bytes

Space remaining for data structures + partitions: 88.30K Bytes

See Section 10.4.9 for more information.

9.1 Enter the number of bytes to allocate to the LOCK TABLE (512)

9.2 Enter the number of ADDITIONAL mountable DATA BASE VOLUME SETS (3) 1

Your system is allowed up to four concurrently mounted data base volume sets. Volume Set 0, of which the system disk is the first volume, remains mounted at all times. You can specify that space be reserved for one, two, or three additional UCI tables.

9.4 Enter the number of bytes to allocate to MAPPED ROUTINES) 4000

Space adjusted to 4032 Bytes

PART 10: JOB PARTITION DEFINITION

See Section 10.4.10 for more information.

PARTITIONS are allocated in 1024-byte increments.

The following PARTITIONS have been defined:

JOURNAL system job 1KB

GARBAGE COLLECTOR system job 1KB

Job 1 (to guarantee one 8K byte PARTITION) 8KB

Space remaining for PARTITION allocation: 71.87K bytes

Default-sized partitions will be used for logins and JOB commands that do not specify partition size. The size of a partition is expressed in 1 K byte increments.

10.1 Enter the default PARTITION SIZE in 1KB increments (8)

Type in the numbers for partition size and number of fixed partitions in the table printed out by SYSGEN.

10.2 Enter fixed PARTITION sizes (in increments) and the number of each. A carriage return in the size field terminates the session.

PARTITION size	number of each
-----	-----
16	1
10	1
RET	

Space remaining for PARTITION allocation: 45.24K bytes

The remainder of memory is assigned to the DYNAMIC PARTITION POOL

PART 11: DATA BASE PARAMETERS

See Section 10.4.11 for more information.

11.1 Enable WRITE CHECK after every DISK WRITE [Y OR N] ? (N)

System default global characteristics are:

8-Bit subscripts:	Yes
Journaling:	Yes
Collating sequence:	Numeric

11.2 Change the DEFAULT GLOBAL CHARACTERISTICS [Y OR N] ? (N)

PART 12: BASIC SYSTEM PARAMETERS

The questions in this part are those asked by the Modify Basic System Parameters utility, ^MBP. More information on these questions is given in Section 10.4.12.

12.0 Enter the UDA disk units, separated by commas, that you wish to be DUAL-PORTED (NONE)

Some disk units (such as the RA80) can be connected to two computers through dual porting. Answer YES to this question only if you wish to enable both computers to have simultaneous access to the disk (dynamic dual

porting). Simultaneous access can cause data base problems and increase disk access time.

These disk drives can be used to provide for a manual switchover of the disk from one system to the other. This switchover is enabled by pressing buttons on the disk drive and is referred to as "static dual porting." In this case, take the default answer to this question, which is NONE.

12.1 Restrict use of the VIEW BUFFER [Y OR N] ? <Y>Y

12.2 Restrict the use of the ZUSE command [Y OR N] ? <Y>Y

12.3 Echo the LOG-IN SEQUENCE [Y OR N] ? <Y>

12.4 Enter the ASCII DECIMAL value of the default APPLICATION INTERRUPT KEY
<3>

The key must be a control character with a decimal value between 0 and 31. The default value of 3 gives an application interrupt key of CTRL/C.

12.5 Enter the ASCII DECIMAL value of the default PROGRAMMER ABORT KEY <25>

The key must be a control character with a decimal value between 0 and 31. The default value of 25 gives an abort key of CTRL/Y.

12.6 Enter the number of seconds to delay SYSTEM RESTART after a POWER FAILURE
<40>

12.7 Enter the number of seconds to delay TELEPHONE DISCONNECT after LOGGING
OUT of a MODEM CONTROLLED LINE <15>

12.8 Enter the number of SIGNIFICANT DIGITS to include in DIVISION computa-
tions <20>

DSM-11 supports computations which yield between 15 and 31 significant digits.

12.9 Is the LINE FREQUENCY 60 HZ [Y OR N] ? <Y>

The frequency of the computer's power source is either 50 HZ or 60 HZ. In the United States it is 60 HZ.

12.10 Enter the 3-character Programmer Access Code (PAC) >>>>

Please enter your initials >JSS

Enter your initials. The response must be uppercase alphabetic characters, 2-22 characters in length.

Enter comment (max. 200 chars.) >EXAMPLE SYSGEN

You can enter a comment that gives appropriate information about this configuration. The comment is saved in the ^SYS global (along with the other information for the configuration).

The system global ^SYS has been built by SYSGEN.
^SYS is a reserved global and should not be altered.

10.6 Example Autoconfiguration

DSM-11 can be installed as an autoconfiguration. This can be done when you first install your system from the distribution disk to the system disk. An example of autoconfiguration during installation is given in Chapter 1.

The example that follows shows how to produce an autoconfiguration after installation has been completed. This can be useful, for example, to obtain the CSR and Vector addresses of hardware devices. If you want the CSR and Vector addresses, you can obtain them by calling the Autoconfiguration utility, ^CONFIG from the Baseline System. Note that ^CONFIG provides these addresses and does not complete the system generation process.

This example begins with booting the system from the system disk.

EXAMPLE:

```
28
Start ? DL1
```

```
Booting DSM-11...
```

Note that the booting procedure can be different for different PDP-11 processors.

```
DSM-11 Version 3
Now running the Baseline System.
```

```
Please enter today's date <DD-MMM-YY> 3-Jun-83
Is today Friday ? <Y>
Please enter time [HH:MM:SS] > 12:15:00
Is this 12:15 PM in the afternoon ? <Y>
```

```
Start up the default system (TEST10) [Y/N] ? <Y> N
Remain in Baseline System [Y/N] ? <N> Y
```

```
>DD ^SYSGEN
```

If you want to obtain only the CSR and Vector addresses (without running SYSGEN), type ^CONFIG instead of ^SYSGEN at this point.

```
System generation for DIGITAL Standard MUMPS
```

Type ? for HELP at any time.

PART 1: SYSGEN

1.1 Would you like extended help [Y or N] ? <N>

1.2 Enter the configuration identifier <TEST10> TEST12

1.3 Do you wish to autoconfigure the current system [Y or N] ? <Y>

SYSGEN continues and produces an autoconfigured system as in the example during installation that is shown in Chapter 1. This autoconfigured system includes the Vector and CSR addresses for hardware devices. At the end of SYSGEN, you return to the Programmer Mode prompt (>); however, you are still in the Baseline System. To leave the Baseline System, proceed as follows:

>D0 ^STU

Please enter today's date <3-Jun-83>

Is today Friday ? <Y>

Please enter time [HH:MM:SS] > 12:30:00

Is this 12:30 PM in the Afternoon? <Y>

Start up the default system (TEST10) [Y/N] ? <Y>

Reconfiguring memory...

Memory reconfigured

Now mounting volume set SYS in table slot S0

Volume 1 on DL1 has	10000 blocks	8937 available.
---------------------	--------------	-----------------

Total in volume set:	10000 blocks	8937 available.
----------------------	--------------	-----------------

Reloading the UCI TRANSLATION TABLE

Building terminal control blocks...

Caretaker is now running as job number 2

DSM-11 Version 3 TEST10 is now up and running!

Exit

(RET)

DSM-11 Version 3 Device #1 UCI: MGR:XXX (RET)

>

The system is now in Programmer Mode with the configuration, TEST10, running.

10.7 Modify System Parameters

You can define basic system parameters by using the SYSTEM DEFINITION (SYSDEF) option of the SYSTEM UTILITIES MENU (^SYS), or (after you have logged in your configuration) by typing:

```
> DO ^SYSDEF
```

10.7.1 Device Characteristics (^MUX)

These questions allow you to establish and modify parameters for terminal devices (1,3 to 19, 64 to 191). The dialogue in this section is in tabular form. All responses are entered in a left-to-right, column-by-column manner. An example of ^MUX is given at the end of this section.

The characteristics that are covered are:

DEVICE NUMBER

Enter the terminal number (nn) for the parameter you want to modify. The number you enter must be 1, 3 to 19, or 64 to 191. You can also specify identical characteristics for a group of terminals by specifying the starting and ending device numbers separated by colons (:). To terminate this question, simply type a carriage return in response to the device number prompt.

PARITY

This question is applicable to DZ11 and DH11 multiplexer devices only. This question allows you to select I/O parity generation and checking. If you want a multiplexer device to have even parity, type an E. If you want a multiplexer device to have odd parity, type an O. If you do not want parity checking, type N or a carriage return.

CRT

If the device is a video terminal, type Y; otherwise, type a carriage return.

AUTOBAUD

If you want autobauding for the terminal, type Y; if not, type N. If you select autobauding, you do not need to select a receiver speed or transmitter speed. The default speed is set to 4800 baud for the detecting speed. If you select autobauding, you log in this terminal by typing two carriage returns (see

Section 3.2.8). If you do not select autobauding, you log in by typing a carriage return or `CTRL/C`.

RECEIVER SPEED

This question is applicable to DZ11 or DH11 multiplexer devices only. Enter a number that represents the baud rate for input from the specified multiplexer. A baud rate of 0 (DH11s only) disconnects a terminal's receiver from the multiplexer, and is a method of disabling currently nonexistent or broken terminals. For DZ11 multiplexers, receiver speed will be assigned to the transmitter as well.

TRANSMITTER SPEED

This question is applicable to DH11 multiplexers only, and allows you to specify an output baud rate that is different from the input baud rate.

MODEM CONTROL

If a terminal is to interface with DSM-11 through a modem, type Y; otherwise, type a carriage return.

ZUSE

If you want messages sent by the ZUSE command to be received and displayed by this terminal, type Y; otherwise, type N.

OUTPUT ONLY

If a terminal is to perform output operations only, type Y; otherwise, type two carriage returns. Note that input is ignored from an output-only device, except for XON and XOFF.

LOGIN

If you type N to this question, users will not be able to log in on the terminal. A Y response to this question enables login either by typing a carriage return, if autobauding is in effect, or by typing a carriage return or `CTRL/C`.

STALL COUNT

Some terminals, such as VT100, VT50, VT52 and VT55 video display terminals, have zero stall counts. In these cases, you should enter a carriage return to advance to the next column.

When certain terminals operate at data rates of 300 baud or greater, a few of their functions, such as carriage return, form feed, and line feed, cannot be performed within the interval allotted to the other characters and functions. This can cause character displacement or total loss for characters that immediately follow one of these functions. To prevent this from occurring, DSM-11

can wait a specified number of character times after one of these functions before sending out the next character.

Suggested stall counts for Digital terminals are:

Serial	VT05 (300 baud)	= 1
	VT05 (600 baud)	= 4
	VT05 (1200 baud)	= 8
	VT05 (2400 baud)	= 20
	LA36 (300 baud)	= 10

NOTE:

Loss of characters at the beginning of a display line can be corrected by an upward adjustment of the stall count for that terminal.

TAB CONTROL

DSM-11 assumes that a tab stop is equivalent to eight character spaces. If you press the TAB key and the cursor does not move eight character spaces, you should respond with N to this question. This causes DSM-11 to convert tab characters to spaces during output operations. The default for this question is Y, indicating that the terminal can properly handle tab characters.

LOWERCASE

If you want a terminal keyboard to have a lowercase character set as well as an uppercase character set, type Y. If you type N, DSM-11 converts all lowercase input characters to uppercase characters.

OUTPUT MARGIN

This question allows you to specify the right margin at which DSM-11 will automatically generate a carriage-return/line-feed. If you do not want DSM-11 to generate an automatic carriage-return/line-feed, you should type a zero (0). If you do want DSM-11 to generate an automatic carriage-return/line-feed, enter a number (such as 80) between 10 and 255 inclusive.

ROUTINE NUMBER

If you intend to tie a terminal, you must enter a response to this prompt. You must specify a routine number to which a terminal is to be tied. The routine number that you specify must be a decimal value from 1 to 7. If you do not want to tie a terminal, enter zero (0) as the routine number. See Section 10.7.6 for information on a related routine, Tied Terminal Table Generation, ^TTTG.

EDIT COMMENT

If you respond with a Y to this question, DSM-11 allows you to enter a comment that is stored with the terminal characteristics. The comment will be printed as part of the System Status Report (^STA).

EXAMPLE:

The following is an example of ^MUX:

```
> D0 ^MUX
```

```
Enter the name of the configuration you wish to alter <TEST10>
```

```
Configuration "TEST10" is running now. If you continue,  
both memory and disk will be modified.  
Are you sure you want to proceed? <NO> Y
```

```
*****Terminal Device Characteristics *****
```

```
Enter device number, or range of device numbers (NN:NN). Enter carriage return  
when done.
```

Device Number	Parity	CRT	Auto Baud	Rcvr Xmit Spd Spd	Modem Cntrl	ZUSE	Output only	Login	Stall Count	Tab	Lower Case	Output Margin	Rtn Num	Edit Comment
66	N	Y	Y	4800 4800	Y	Y	N	Y	0	Y	Y	80	0	N

```
RET
```

After you type in the device number (in this case 66), the utility provides the present values of the device characteristics. The utility repeats the device number, and you can then provide new values at each column position. By pressing a carriage return at a given column position, you take the current value. By pressing a carriage return in the device number column, you terminate the utility. After you provide a value for each column in the line, and press a carriage return in the device number column, the utility continues as follows:

```
Please enter your initials> JSS
```

```
Enter comment (max. 200 chars.) > TEST FOR DOCUMENTATION JUNE 3
```

10.7.2 Add UCIs (^UCIADD)

These questions allow you to modify, add, or edit UCI codes. You can specify a maximum of 30 UCIs. The dialogue in this section is in tabular form. You enter your responses in a left-to-right, column-by-column manner. By pressing a carriage return after entering the new UCI, you receive the default values for each column.

EXAMPLE:

Inspect/Add UCI data for which volume set? (S0)

This question is asked only if there is more than one volume set mounted on the system. Note that you cannot add a UCI to a volume set that is not mounted on the system.

*****Inspect/Add UCIs for Volume Set S0*****

Enter disk data as DDU:M:BL

Where DD = disk type
U = unit no.
M = map no.
BL = block no. within map

or hit carriage return to accept default values.

For "UCI name" enter 3 alphabetic characters, or hit (CR) if done adding UCIs.

UCI# NAME	GLOBAL DIRECTORY	NEW GLOBALS POINTER AREA	ROUTINE DIRECTORY	NEW ROUTINE GROWTH AREA	NEW GLOBALS DATA AREA
1 MGR	DL0:0:95 95:S0	DL0:0:0 0:S0	DL0:0:96 96:S0	DL0:0:0 0:S0	DL0:0:00 0:S0
2 AAA	DL0:20:202 8202:S0	DL0:20:0 8000:S0	DL0:20:203 8203:S0	DL0:20:0 8000:S0	DL0:20:0 8000:S0

RET

If you do not want to enter a new UCI name, type a carriage return to terminate the question. If you type a carriage return anywhere other than under the name column, you receive the default value. If you want to allocate a directory on the system disk, type a carriage return. Type an up arrow after the next question to exit from the utility.

Inspect/Add UCI data for which volume set? (S0) ^

10.7.3 UCI Edit (^UCIEDIT)

This utility has a simple dialogue that allows you to change the name of a UCI. You can also change the default library UCI for a specified UCI. The library UCI contains utilities that can be used by several UCIs; an example is the library utilities contained in the manager's UCI.

EXAMPLE

An example of ^UCIEDIT follows:

```
>D ^UCIEDIT

Enter one of the following:
1. Edit UCI name
2. Edit library UCIs
Enter option (1 or 2) > ? 1

Enter UCI name > AAA

New name ? > BBB

Enter one of the following:
1. Edit UCI name
2. Edit library UCIs
Enter option (1 or 2) > ? 2

Enter UCI name > DDD

Library UCI ? <1> 2
```

This example changes the name of UCI AAA to BBB. The example also changes the library UCI for UCI DDD from UCI 1 to UCI 2.

10.7.4 UCI List (^UCILI)

This routine produces a list of all loaded and accessible UCIs on the current configuration.

10.7.5 Magnetic Tape Default Mode (^MMD)

These questions allow you to establish and modify the magnetic tape format for your system. Section 6.3 contains additional information about magnetic tape formats.

You can specify one of two magnetic tape formats: DOS-11 compatible (ASCII characters, DOS labeling, and stream data format), or any combination of the following:

- Character set ASCII or EBCDIC
- Labeling standard labeling or unlabeled
- Format stream data or variable-length records

The magnetic tape format mode can be temporarily modified by any user with the OPEN command.

EXAMPLE:

```
> DD ^MMD
```

```
Enter the name of the configuration that you want to modify <TEST10>
```

```
Configuration "TEST10" is running now. If you continue,  
both memory and disk will be modified.
```

```
Are you sure you want to proceed? <NO> Y
```

```
Current magtape default mode is:
```

```
DOS compatible, 1600 BPI
```

```
Modify magtape default mode <N> Y
```

```
DOS-11 compatible format (Y or N) <Y> N
```

If you choose DOS-11 compatible format then the other characteristics for magnetic tape are automatically assigned.

```
ASCII or EBCDIC (type A or E) <A> E
```

```
Labeled or Unlabeled (type L or U) <L> U
```

```
Stream, Fixed, or Variable record format (type S, F, or V) <S> S
```

```
Please enter your initials > JSS
```

```
Enter comment (max. 200 char.) > TEST FOR DOC
```

10.7.6 Routine Mapping (^RMAP)

Routine mapping is a way of placing specified routines in memory. System overhead is reduced because the routines reside in memory and are not loaded into the user's partition when they are called. See Section 15.1 for more information.

Space for mapped routines must be reserved during Part 6 of SYSGEN.

When you call up the ^RMAP option of the ^SYSDEF menu, you get the ^RMAP menu of routine-mapping routines:

1. DISABLE/ENABLE ROUTINE MAP (^RMDIS)
Disables or enables routine mapping.
2. DISPLAY LOADED ROUTINE SETS (^RMSHO)
Displays information about loaded routine sets.
3. LOAD A ROUTINE SET (^RMLOAD)
Loads a defined routine set into memory.
4. ROUTINE SET MANAGEMENT (^RMBLD)
Allows you to create, edit, or delete a routine set.

You should use the following procedure to map a set of routines onto memory:

1. Determine a set of routines to be grouped together into a routine set that is to be mapped. Each set is associated with an individual UCI and volume set. Determine the approximate total size of all the routine sets that you want to map.
2. Reserve space during SYSGEN for routine mapping based on the total amount of space for all the routine sets that you want to map.
3. Use the Routine Set Management utility (^RMBLD) to name and set up a list of routines to be included in a routine set. Set up the routine set carefully since you cannot unload the routine set with a utility once it is loaded. You must reboot the system to unload a routine set. You can disable access to an individual routine or to a routine set; however, you cannot reenable access to an individual routine.
4. Use the Routine Set Loading utility (^RMLOAD) to actually load your routine set.
5. Use the Display Loaded Routine Sets utility (^RMSHO) to see what routine sets have been loaded.
6. Use the Disable/Enable Routine Mapping utility (^RMDIS) to disable access to a routine or routine set. You can reenable access to a routine set, but not to an individual routine.

You cannot unload a routine set with a utility (you must reboot the system) because there is no complete way to judge whether an individual routine is being used at a given time. If you unload a routine set while someone is using the routine, you can cause a wide variety of problems.

10.7.6.1 Disable/Enable Routine Mapping Utility (^RMDIS) — The following is an example of the Disable/Enable Routine Mapping utility, option 1 of ^RMAP.

EXAMPLE

```
Enable/Disable Mapped routines and sets

Enter the ROUTINE SET name > SSTEEST

Routine set, SSTEEST, exists for for JSS, SYS - currently enabled

DISABLE the entire ROUTINE SET [Y or N] ? > Y

SSTEEST disabled
```

The previous example disables the routine set SSTEEST. If you want to disable one routine only you would answer N to the question about disabling the entire routine set. The utility then asks you for the name of the routine in the routine set that you want disabled. Remember that if you disable a routine you cannot reenable it; you can reenable a routine set after it has been enabled. The following example shows how to reenable a routine set.

EXAMPLE

```
Enable/Disable Mapped routines and sets

Enter the ROUTINE SET name > SSTEEST

Routine set, SSTEEST, exists for for JSS, SYS - currently disabled

ENABLE Mapping of this ROUTINE SET [Y or N] ? > Y

Routine Set for JSS,SYS is enabled.
```

10.7.6.2 Display Loaded Routine Sets Utility (^RMSHO) — The following is an example of the Display Loaded Routine Sets utility, option 2 of the ^RMAP menu:

EXAMPLE

```
The following routines are mapped into memory:

Routines Mapped for UCI and Volume Set: DEB,SYS

CONTROL  CPUTIM  DELTIM11  GSTATS  JSTART  LABELS
TEST     TIMCPU  TIME1     XTEST

Routines Mapped for UCI and Volume Set: BOB,SYS

TEST  TEST2  XYZ  XYZZY  ZZZ
```

This routine lists all routine sets that are mapped onto memory. In this case there is the one set that was loaded in the preceding example, and another set that was loaded at an earlier time. Type a carriage return to exit from this routine and return to the ^RMAP menu.

10.7.6.3 Routine Set Loading Utility (^RMLOAD) —The following is an example of the Routine Set Loading utility, option 3 of the ^RMAP menu:

EXAMPLE

Load a Routine Set Into Memory

Enter the ROUTINE SET name > TEST

Loading Mapped Routine set: TEST

CONTROL	CPUTIM	DELTIM11	GSTATS	JSTART	LABELS
TEST	TIMCPU	TIME1	XTEST		

Type a carriage return to exit from this routine and return to the ^RMAP menu.

10.7.6.4 Routine Set Management Utility (^RMBLD) —The following is an example of the Routine Set Management utility, option 4 of ^RMAP:

EXAMPLE

Create, Edit, or Delete a Routine Set

Enter the ROUTINE SET name > TEST

This name is used in all future references to specify this routine set. The name can not exceed 10 characters.

Enter the UCI name in which the routines reside <MGR> DEB

Enter the VOLUME SET name in which the routines reside <SYS>

Enter routine names to add or delete

Routine(s) ? > *

By typing an asterisk (*) in this example, all routines in DEB have been chosen. You can answer this question with:

- A routine name
- nam* for all routines beginning with the letters nam
- nam1 to nam2 for all routines in the range from nam1 to nam2 inclusive

- * to select all routines
- - followed by any of the preceding to deselect routines which have been selected
- ^L to list the selected routines
- ^D to list all routines in the reference UCI
- ^ to terminate without selection
- ^R to repeat the previous set of routines

Routine(s) ? > ^L

```
ALL   CONTROL  CPUTIM  DELTIM11  GSTATS  JSTART  LABELS
TEST  TIMCPU   TIME1   TIME2     XTEST
```

This is a list of all of the selected routines. In the next question, the user deletes two of the routines.

Routine(s) ? > -ALL

Routine(s) ? > -TIME2

Routine(s) ? > (RET)

10 routines included in MAPPED ROUTINE SET: TEST

Enter the ROUTINE SET name)

You can create another routine set, or type a (RET) to return to the ^RMAP menu.

10.7.7 Tied Terminal Table Generation Questions (^TTTG)

In DSM-11, you can link or "tie" a terminal directly to a specified routine. When you tie a terminal, you restrict that terminal to running routines that are directly called by the original routine. Note that you can specify a routine for multiple terminals, but you can only tie a terminal to a single routine. You can specify a total of seven routines.

In order to set up a tied terminal, you must first establish a routine that runs when a user logs in that terminal. The name of this routine must be two

characters in length. You then establish a table of tied terminal routines by using ^TTTG.

You do not actually assign terminals to routines with ^TTTG, you make this assignment by using the Device Characteristics utility, ^MUX. In the column for routine numbers you can enter a routine number that corresponds to a routine in the Tied Terminal Table established by ^TTTG. For more information on the Device Characteristics utility, ^MUX, see Section 10.7.1.

EXAMPLE:

```
> D ^TTTG
```

```
Enter the name of the configuration you wish to alter <11/44> ?
```

```
Configuration "11/44" is running now. If you continue,  
both memory and disk will be modified.  
Are you sure you want to proceed? <NO> Y
```

```
*****Tied Terminal Table Generation*****
```

```
Tied terminal table for configuration "11/44" :
```

ROUTINE #	VOLUME SET NUMBER	UCI #	PARTITION SIZE (1 KB INCREMENTS)	ROUTINE NAME
-----	-----	----	-----	-----
1	S0	1	8	TT

```
RET
```

If there are already tied terminal routines, the utility prints them out as a table. Note that you must establish a partition size that will be assigned at log-in time for the tied terminal routines. If you wish to see the table alone, run the utility, ^TTTSHO.

```
Terminate by responding with <CR> to routine number question
```

```
Create or edit a table entry:
```

```
Routine Number> 2  
Volume Set Number> S0  
UCI Number> 1
```

You can indicate a routine in a different volume set from the one that you are currently in. For example, S1 indicates Volume Set 1. The default is for Volume Set 0, which is listed as S0.

```
Partition Size>16  
2 character routine name> XX
```

The partition size is in 1K-byte increments. The routine name must be a 2-character routine name.

Routine Number> RET

Please enter your initials > BAS

Enter comment (max. 200 chars.)> TEST RUN

10.7.8 UCI Translation Table (^UCITRAN)

The UCI Translation Table is a means of accessing a global in another volume set or system. For more information on this table and using the ^UCITRAN utility, see Chapter 17.

Chapter 11

Operating and Maintaining DSM-11

This chapter describes the procedures for starting, restarting, backing up, and shutting down a DSM-11 system. The system utilities discussed in this section are:

Utility	Routine Name
System Start-up	^STU
Define Start-up Parameters	^STUBLD
Disk Backup	^BACKUP
Disk Restore	^REST
Autopatch	^AUPAT
System Shutdown	^SSD

11.1 System Start-up (^STU and ^STUBLD)

When you boot your DSM-11 system, the System Start-up routine, ^STU, automatically begins execution. The purpose of this routine is to:

- Establish the system date and time
- Enable the hardware and software features of a defined configuration

- Mount all of the disks in your data base
- Enable the start-up of DSM-11 background jobs

Before starting your DSM-11 system, you must always enter the date and time. The format for both the date and time is displayed on the console terminal. After you have entered the date and time, DSM-11 asks you whether you want to run the default system. Typically, you start up DSM-11 in default mode using the start-up parameters that you have already defined.

If you choose not to run the default configuration, the System Start-up routine asks you a series of questions that define the run-time environment of the configuration. During the start-up session, you can choose to have your responses become the new default start-up parameters. Note that you are allowed to specify one set of start-up parameters for each DSM-11 SYSGEN configuration.

You can establish or define a set of default start-up parameters (to be used during the next system startup) once the system has been started. This can be done by using the Define Start-up Parameters utility, `^STUBLD`.

An example of `^STUBLD` follows:

```
> DO ^STUBLD
```

Begin defining a new startup command file.

```
Configuration ? (TEST)
Apply patches to memory [Y/N] ? (N)
```

Answer Y to apply patches on the system disk to memory. See Section 11.3 for more information on patches.

```
Start up the Journal [Y/N] ? (N)
```

Answer Y to start up journaling. See Chapter 16 for more information on journaling.

```
Enable the Spool device (device 2) [Y/N] ? (N)
```

The spooling device stores output temporarily before printing. See Section 6.7 for more information on spooling.

```
Start the Caretaker background job [Y/N] ? (Y)
```

See Section 12.2 for more information on the system caretaker and logging of error messages.

```
Install LOADABLE DRIVERS on start-up [Y or N] ? (N)
```

If you have chosen loadable driver support (TU58, RX02, or BISYNC) during system generation, you are asked this question. If you answer yes, you are then asked if you want to load a particular driver. See Section 10.4.7 for more information on loadable drivers, and the *DSM-11 BISYNC Programmer's Guide* for more information on the BISYNC driver.

```
Enter printer number for system error messages <1>
Automatic logging of DSM errors [Y/N] ? <N>
Load Mapped Routine Sets on Startup [Y or N] <N> Y
Load Routine Set DEB [Y or N] ? > Y
Load Routine Set NEW [Y or N] ? > N
```

If you answer Y to the preceding questions you load the indicated routine set into memory. You must have previously chosen routine mapping support during system generation, and must have previously created the routine sets you want to map. See Sections 10.4.7, 10.7.6, and 15.1 for more information on routine mapping.

```
Mount additional disk volumes [Y or N] ? <Y> Y
```

```
Enter the disk mounting information. Type ? for help
```

DISK UNIT ----	DATA BASE VOLUME SET -----	LABEL/VOLUME SET NAME -----
DK0	Y	JHM
DR0	Y	JSS

This questions asks if you want to mount additional disk volume sets. See Section 13.2, 13.3, or 14.7 for additional information on disk volume sets.

```
Make this the new start-up command file for configuration TEST [Y/N] ? <Y>
```

This start-up command file now becomes the default start-up file for this configuration. The start-up command file only affects the configuration it is created for - in this case - the configuration TEST. When using [^]STUBLD, the parameters and options that you chose in the start-up command file only come into effect after you reboot the system and start up the configuration.

11.2 Preserving System Integrity

As System Manager of a DSM-11 installation, you should be concerned with the possibility of data loss caused by power failures, by system crashes, or by other unpredictable events. By devising backup strategies that use backup disks and journaling, you can minimize data loss. DSM-11 allows you to perform two kinds of backup: physical backup and logical backup. The following sections describe each type of backup you can perform.

11.2.1 Concept Of Physical Backup (^BACKUP And ^REST)

Physical backup allows you, the System Manager, to save the contents of a DSM-11 system. The system utilities that you can use to save and restore your data base are Disk Backup, ^BACKUP, and Disk Restore, ^REST. You can use the Disk Backup utility to do a scheduled, unattended backup, as well as other kinds of backup. These utilities can be accessed directly or through the SYSTEM UTILITIES menu (^SYS).

11.2.2 Concept Of Logical Backup

Logical backup allows you to save specified routines and globals that are listed in the directory of a particular UCI. Any DSM-11 user can perform a logical backup with the following library utilities: Routine Save (^%RS), Routine Restore (%RR), Global Save (%GTO), and Global Restore (%GTI). These routines are described in Chapter 7. You should be aware that the Routine Save and the Global Save library utilities do not save the system image or the physical disk data structure.

11.2.3 Concept Of Journaling

The journal procedure allows you to record each update you make to your global data base. The journal procedure automatically records new global update data, in the form of a transaction history, on disk or magnetic tape. This occurs each time you issue a SET or KILL command to a global that you previously selected to be journaled. Chapter 16 contains a description of the journal procedure and its related utilities.

11.3 Applying Patches to DSM-11

Some DSM-11 Software Dispatch articles specify patches to the DSM-11 system. These are the only patches supported by Digital. Those patches marked "M" in the upper right hand corner are mandatory, and must be applied to your system if you are to be supported properly. All other patches are optional, and may be applied at your discretion. The two types of patches that may be required are patches to DSM-11 utilities (UTILITY) and patches to the DSM-11 operating system (SYSTEM). These two types of patches require quite different patching techniques, detailed below.

11.3.1 Utility Patching

Most patches involve changes to system and library utility routines. These UTILITY patches are presented in Software Dispatch articles in a form convenient for your application using the "[^]%" editor supplied with DSM-11 (also referred to as the DSM-11 editor).

The patching procedure assumes that you have loaded the routine, using ZLOAD, into a partition of at least 8K bytes, the minimum size for running all DSM-11 system utilities. Occasionally, you are asked to use a larger partition; this is because the [^]% editor itself requires up to 1200 bytes of memory for editing routine lines, and will overflow the partition when you are editing a large routine. In those cases, a 9K byte partition is required.

When a UTILITY patch changes an existing routine line, the Software Dispatch article shows the change exactly as you would type it using the [^]% editor. The characters you are to type are underlined.

When a patch requires that a new line or lines be inserted into a routine, you have three possible means at your disposal. The simplest method is to use the ZPRINT command in Programmer Mode to print the routine line that the new line(s) are to follow. You can then simply type the new line(s) exactly as they are to appear in the routine.

The second method is to use the ZINSERT command, again in Programmer Mode.

The third method is to use the alternate [^]%EDI editor. In fact, you can use the [^]%EDI editor for all your editing. Keep in mind that the [^]%EDI editor requires up to 2K bytes of memory for working space. That means you might need a 10K-byte partition to edit an 8K-byte routine.

Regardless of which editor you use, do not forget to save the routine when you are done. An argumentless ZSAVE command saves the routine under the same name you used to ZLOAD it, eliminating the possibility of typing the routine name incorrectly and accidentally erasing some other routine with a similar name.

11.3.2 System Patching ([^]AUPAT)

DSM-11 is supplied with a system utility routine for creating and applying operating system patches. This routine can be invoked either through the

SYSTEM ROUTINES (MISC) option of the SYSTEM UTILITIES menu (^SYS) or directly by typing:

```
>D ^AUPAT
```

SYSTEM patching should be done in three stages.

1. Type the patch into the system, using the CREATE option of ^AUPAT. When you create system patches, you should number the patch with the Software Dispatch article number from the upper right-hand corner of the article. Do not include the alphabetic flag, just the numbers N.N.N. Some system patches in Dispatch articles may include comment fields to the right of the patch data; these are for information only and are not part of the patch.
2. Apply the patch to memory only. Then, run your system for a period of time that assures you that the patch produces no undesirable side effects peculiar to your system. The start-up routine has a built-in feature that allows you to reapply patches each time the system is booted, so that you can conveniently run with memory-only patches for as long as you wish.
3. Apply the patch to the disk image of the system. After you have applied a patch to disk, all future bootstrap load operations will load the patched system. In other words, the patch is permanent.

Some operating system patches may involve changes to parts of the code required for actually performing the application of patches. This includes changes to the job scheduler and parts of the interpreter. This small class of patches must be patched to disk only, since a system crash might result from patching to memory in a run time environment. When a patch of this nature appears in the Software Dispatch, you are be instructed to apply the patch to disk only. You must reboot your system in order to run with patches applied to disk only.

The following is an example of ^AUPAT. Note that, at the end of the routine, you should use the up arrow (^) in response to the routine questions. This causes you to exit from the routine.

```
>D ^AUPAT
```

```
Specify one of the following:
```

1. Maintain patches
2. Apply to memory patches not already applied to disk
3. Apply to memory and disk patches not already applied to disk
4. Apply to memory and disk all existing patches

```
Enter the number of your choice > 1
```

```
Maintain patches
```

1. CREATE
2. DELETE

3. EDIT

4. LIST

Enter one of the above options > 1

Patch number ? > 1.5.1

Patch date [DD-MMM-YY] ? > 10-APR-81

Patch title ? > SUPPRESS BREAKPOINT AT COLD START

Now you can start to enter your patch data

Module Name ? > BDBTAB

Address Offset ? > 2230

Old contents ? > 3

New contents ? > 240

Address offset = 2232

Old contents? > ^

Address offset ? > 3036

Old contents ? > 5015

New contents ? > 177777

Address offset = 3036

Old contents ? > ^

Address offset ? > ^

Module name ? > ^

Patch number ? > ^

11.4 System Shutdown (^SSD)

You can use the System Shutdown utility routine (^SSD) to bring the system to the point where the processor can be halted so that you can perform various hardware and software maintenance tasks, or to bring up a different DSM-11 configuration. This routine checks to see that all outstanding data base writes and the system Garbage Collector have finished. The utility can be accessed

through the SYSTEMS ROUTINES (MISC) option of the SYSTEM UTILITIES menu (SYS), or by typing:

```
> DO ^SSD
```

All terminal users should log out before performing a system shutdown. You can request a delayed shutdown so that system users are both notified of the impending shutdown and given time to finish what they are doing.

CAUTION

Halting the system without performing the Shutdown routine results in the loss of all partition resident data, and can leave the data base in an inconsistent condition.

Chapter 12

Using the DSM-11 System Utilities

This chapter describes the DSM-11 system utilities.

12.1 Introduction to the System Utilities

The DSM-11 system utilities are a group of privileged-access routines, available only to the System Manager or other users who have access to the manager's UCI (UCI #1). These routines are used for day-to-day maintenance and modification of the DSM-11 operating system.

The DSM-11 system utilities are divided into the following categories:

- Caretaker
- Disk maintenance
- Journal
- Sequential disk
- Spooling
- System definition
- System reports
- Miscellaneous system routines

These categories are presented as options in the SYSTEM UTILITIES menu. Section 12.1.1 explains how to access this menu.

The utilities listed under the JOURNAL menu are discussed in a separate chapter on journaling, Chapter 16.

Utilities listed under the SYSTEM DEFINITION menu are discussed in Chapter 10 and Chapter 11, except for two of the utilities. These two are Global Growth Area (^DGAM) and Global Placement (^GLBPLACE), which are described in Section 13.6.3.

All system utility routines are stored in the manager's UCI (UCI #1), and are listed in the routine directory of the manager's UCI. Several undocumented system and library utility routines are also listed in this directory. These routines are subroutines used by other system and library utilities, or they are utility routines that help tailor the DSM-11 operating system. Although privileged users can modify all utilities, the utilities should not be modified without a thorough understanding of the DSM-11 internal structures.

12.1.1 Running The System Utilities

System utility start-up procedures are the same as those used by the library utilities (refer to Section 7.1.1). However, the initial response to:

```
>D ^ZUTL
```

when issued from the manager's account allows you to select both the system and library utilities.

You can access the SYSTEM UTILITIES menu directly by typing:

```
>D ^SYS
```

The conventions used for the system utility package are the same as those used for the library utility package. The most important convention to remember is that typing a question mark (?) provides a list of valid responses to any prompt. Typing only a carriage return or a ^ backs up the menu one level (unless the carriage return is recognized by that prompt as a default). This in effect provides an exit from a utility if you do not choose to run the utility after receiving the first prompt. This also provides an exit from some utilities that are structured so that they return to the first prompt after you have run the utility.

12.1.2 Stopping The System Utilities

The best way to stop a system utility is by typing the application interrupt key (`CTRL/C`), if that key is enabled by the utility. If the utility does not allow interruption by `CTRL/C`, there is probably a good reason, such as the utility using the VIEW buffer.

You can also stop a system utility by typing a `CTRL/Y`. However, if you stop a routine with `CTRL/Y` while processing data or while I/O is in progress, you can lose data, corrupt data bases, and/or hang devices if output is pending. In any case, the utility cannot be restarted from the point of termination. You must reaccess the utility through the menu-driver.

Many of the system utilities use the VIEW buffer. If you stop a utility while it is using the VIEW buffer, the VIEW buffer may remain OPEN, which prevents other users from accessing the VIEW buffer. You should issue a CLOSE 63 if you interrupt a utility with a `CTRL/Y`.

12.2 Caretaker Utilities (^CARE)

The caretaker utilities are accessed through the SYSTEM UTILITIES menu (`^SYS`). The CARETAKER menu can be accessed directly by typing:

```
>D ^CARE
```

These utilities manage the system caretaker job and the system error log for both hardware and software errors.

Most of the options in this menu must be accessed through the CARETAKER menu. They cannot be accessed directly through the routine name.

Utility	Routine Name
Start System Caretaker	-
Stop System Caretaker	-
Print Hardware Error Log	<code>^KTR</code>
Erase Hardware Error Log	-
Change Error Printer	-
Print Disk Error Summary	<code>^DSKTRACK</code>
Software Error Log	<code>%ER</code>

12.2.1 Start System Caretaker

Starts the caretaker job. This job has three functions:

- It monitors the status of the output-only printers and the system disk. If a printer or the system disk goes off line, a message is printed on the designated error printer, usually the console terminal. A write-locked system disk is also reported this way.
- The caretaker job records other types of disk errors, as well as magnetic tape drive errors, in the ^SYS global.
- The caretaker can optionally log software errors in routines that have \$ZT set to the ^%ET routine. Section 4.10.3 provides more information on the ^%ET routine in the context of error processing and error trapping. This optional software error-logging procedure requires the JOBCOM system feature (selectable during system generation), and a designated JOBCOM channel, which you specify at start-up when specifying the start-up parameters, (^STU), or when building a default start-up parameter file (^STUBLD). See Section 11.1 for more discussion of these utilities.

12.2.2 Stop System Caretaker

Stops the caretaker job. The system status will no longer be monitored; the system operator will not be notified of any special conditions; and errors will no longer be recorded.

12.2.3 Print Hardware Error Log (^KTR)

Prints a list of hardware errors.

12.2.4 Erase Hardware Error Log

Purges the ^SYS global of obsolete hardware error information. The purging is done by date.

12.2.5 Change Error Printer

Changes the error printer. The printer is changed in memory only; it is not changed on the system disk. If you shut down the system and start it up again, the error printer is the one indicated in the start-up command file, not the one you indicated with the Change Error Printer utility. To change the start-up specification for the error printer, use [^]STU, the System Start-up utility, or Define System Start-up Parameters, [^]STUBLD. See Section 11.1 for more discussion of these utilities.

12.2.6 Print Disk Error Summary

Lists disk errors. This report collates logged disk errors by disk drive and read/write head. In this way, problems with read/write heads can be detected before the heads fail completely.

12.2.7 Software Error Log (%ER)

Use routine %ER to show software errors, or to reload the symbol table and the routine where a particular error occurred.

12.3 Disk Maintenance Utilities

These utilities can be accessed through the SYSTEM UTILITIES menu ([^]SYS). The utilities listed on the menu are as follows:

Utility	Routine Name
Backup	^BACKUP
Bad Blocks	^BBTAB
Dismount	^DISMOUNT
Format/Initialize, or New Disk	^DISKPREP
Label	^LABEL
Mount	^MOUNT
Restore	^REST

These utilities are described in the following sections.

12.3.1 Disk Backup (^BACKUP)

Backs up an existing DSM-structured disk to another device to be restored later with Disk Restore (^REST). Use this utility to back up the system. Section 11.2 on preserving system integrity discusses backing up the system.

12.3.2 Disk Bad Blocks (^BBTAB)

Displays the contents of a disk's Bad Block Table. You can also use the utility to add newly discovered bad blocks to the disk's Bad Block Table.

12.3.3 Disk Dismount (^DISMOUNT)

Dismounts a disk volume or volume set.

12.3.4 Disk Format/Initialize (^DISKPREP)

This utility initializes a new disk as a DSM-11 disk. This utility can also "format" RK05 and RP04/05/06 disks to establish sector header information used by the disk hardware controller. The NEW DISK option of the DISK MAINTENANCE menu also uses this utility.

12.3.5 Disk Label (^LABEL)

Labels a DSM-11 disk with a Master/Backup label.

12.3.6 Disk Mount (^MOUNT)

Mounts a disk volume or volume set.

12.3.7 Disk Restore (^REST)

Restores a backed-up copy of a disk. Use this utility to restore the system disk if it was backed up using Disk Backup (^BACKUP).

12.4 Sequential Disk Processor Utility (^SDP)

The Sequential Disk Processor (SDP) is used to access reserved areas of the disk for sequential-access or random-access storage of information. The SDP utility can be accessed as an option through the SYSTEM UTILITIES menu (^SYS), or directly by typing:

```
>D ^SDP
```

You can use this utility to allocate SDP space on a disk, to deallocate SDP space (return it to "free" space), or to display the currently mounted SDP space.

More information on SDP and how to use it to read or write onto SDP space on a disk is available in Section 6.4.

12.5 Spool Utilities

The Spooling Device is a file-structured mechanism used for temporary storage of information. The Spool utilities aid in the use of the Spooling Device. These utilities can be accessed through the SYSTEM UTILITIES (^SYS) menu or by typing:

```
> DD ^SPL
```

Section 6.7 gives more information on the Spooling Device.

Utility	Routine Name
Allocate Spool Space	^SPLALL
Deallocate Spool Space	^SPLREM
Display Spool File Structure	^SPLSTR
Initialize Spool Space	^SPLINI
Start Despooler Lister	DSON^SPL
Start Spooler	SPON^SPL
Stop Despooler	DSOF^SPL
Stop Spooler	SPOF^SPL

12.5.1 Allocate Spool Space (^SPLALL)

Reserves disk space for the Spooling Device to use.

12.5.2 Deallocate Spool Space (^SPLREM)

Returns disk space reserved for spooling to the system.

12.5.3 Display Spool File Structure (^SPLSTR)

Displays the spool file structure.

12.5.4 Initialize Spool Space (^SPLINI)

Deletes all presently defined spool files from your allocated spool space on the disk. The entire spool space that you have allocated for spooling is now available for new spool files.

12.5.5 Start Despooler Lister (DSON^SPL)

Starts a job that lists the spooled output on the default output spool device.

12.5.6 Start Spooler (SPON^SPL)

Enables spooling. If you use device #2 or the default output spool device, the output is sent to your allocated spool space on the disk.

12.5.7 Stop Despooler (DSOF^SPL)

Stops the despooler job.

12.5.8 Stop Spooler (SPOF^SPL)

Disables the spooling function.

12.6 System Reports

These utilities can be accessed through the SYSTEM UTILITIES menu (^SYS). Four major categories of reports are:

- Disk Reports
- Performance Statistics
- Status
- Tables

These reports are described in the following sections.

12.6.1 Disk Reports

These utilities can be accessed through the SYSTEM UTILITIES menu (^SYS). These reports give information on the physical structure of the disk. The report utilities are:

Utilities	Routine Name
Block Dump	^BLDMP
Disk Block Tally	^DBT
Fast Disk Block Tally	^FASTDBT
Integrity Check	^IC

12.6.1.1 Block Dump (^BLDMP) — Provides a dump of a given disk block. You can specify block numbers relative to a disk or a volume set, or DSM-11 block numbers (sequential block numbers beginning with block 0 of Volume Set 0).

12.6.1.2 Disk Block Tally (^DBT) — Provides a summary of disk usage showing allocated and free blocks. For allocated blocks, it reports the number allocated to SDP, spooling, and each UCI.

12.6.1.3 Fast Disk Block Tally (^FASTDBT) — Produces a short DBT report which includes how much space is left on a volume set.

12.6.1.4 Integrity Check (^IC) — Checks the integrity of the logical and physical structure of a data base, such as a disk. This utility is designed to be used with the Data Base Repair utility, ^FIX.

12.6.2 Performance Statistics (^RTHIST)

With this utility, you can gather statistics on how system resources are being used while users are on the system. This includes information on routine and global utilization. This information can be used to identify performance bottlenecks in the system and to indicate ways to improve overall system performance. See Section 15.4 for more information on this utility.

12.6.3 System Status

These utilities can be accessed through the SYSTEM UTILITIES menu (^SYS). They provide status information on specific elements of the system.

Utility	Routine Name
Job Monitor	^%JOB
Switch Register	^SWREG
System Status	^STA
Terminal DDB	^DDR
Line Map Report	^LMAP
Active Job Report	^ACTJOB

12.6.3.1 Job Monitor (^%JOB) — Monitors the output buffer of another job.

12.6.3.2 Switch Register (^SWREG) — Displays the contents of the software switch registers. See Appendix F for more information on the switch registers.

12.6.3.3 System Status (^STA) — Shows the status of the jobs currently running on the system.

12.6.3.4 Terminal DDB (^DDR) — Displays the device descriptor information for a terminal.

12.6.3.5 Line Map Report (^LMAP) — Produces a complete table of all terminal ports on the configuration and their current characteristics.

12.6.3.6 Active Job Report (^ACTJOB) — Produces a list of active jobs on the system.

12.6.4 Table Utilities

These utilities can be accessed through the SYSTEM UTILITIES menu (^SYS). They provide a way of listing the tables.

Utility	Routine Name
Device Table	^DEVTAB
Job Table	^JOBTAB
Locked Variables	^LOCKTAB
Partition Table	^PARTAB
System Table	^SYSTAB
UCI Table	^UCITAB
Partition Vector	^PARVEC
Volume Set Table	^%STRTAB

12.6.4.1 Device Table (^DEVTAB) — Displays the Device Table. This table specifies which devices are available and which devices are currently being used. See Section 14.5 for more information.

12.6.4.2 Job Table (^JOBTAB) — Lists the Job Table. This table contains status about each job currently running on the system. See Section 14.3 for more information.

12.6.4.3 Locked Variables (^LOCKTAB) — Lists all currently locked variables. These are local and global variables that have been locked by a particular job with either the LOCK or ZALLOCATE commands. See Section 14.4 for more information on the Lock Table. For more information on the LOCK and ZALLOCATE commands, see the *DSM-11 Language Reference Manual*.

12.6.4.4 Partition Table (^PARTAB) — Lists the Partition Table. This table specifies the size and location of each UCI in the system. See Section 14.6 for more information.

12.6.4.5 System Table (^SYSTAB) — Lists the System Table. This table specifies system constants and parameters. See Section 14.2 for more information.

12.6.4.6 UCI Table (^UCITAB) — Shows the entries in the UCI Table. This table lists all UCIs in the current volume set, and associated information for each UCI on the disk location of the global directory, routine directory, and growth areas. See Section 14.8 for more information.

12.6.4.7 Partition Vector (^PARVEC) — Displays the contents of a job's partition vector. The partition vector contains basic information about a currently running job that is occupying a partition. See Section 14.13 for more information.

12.6.4.8 Volume Set Table (^%STRTAB) — Shows the Volume Set Table, which lists currently mounted volume sets, and the volumes within each volume set. See Section 14.7 for more information.

12.7 Miscellaneous Routines

This group of routines can be accessed through the SYSTEM UTILITIES menu (^SYS).

Utility	Routine Name
Autopatch	^AUPAT
Broadcast	^BCS
Data Base Repair	^FIX
Peek	^PEEK
Restore Jobs/Devices	^RJD
Set Date	^DAT
Set Time	^TIM
System Shutdown	^SSD
Distributed Data Base Processor	^DDPUTL
Background Job Attacher	^ATTACH
Background Job Detacher	^DETACH
Help Text Driver	^HELP
Magnetic Tape Copy	^TAPECOPY

These utilities are described in the following sections.

12.7.1 Autopatch (^AUPAT)

Provides for creation, installation, and removal of system patches. See Section 11.3.2 for more information on system patching.

12.7.2 Broadcast (^BCS)

Sends messages to specified terminals on the system. This could be a planned system shutdown announcement or some other message.

12.7.3 Data Base Repair (^FIX)

Edits the contents of blocks to repair data base degradation. The utility ^FIX is designed to be used with Integrity Check, ^IC, listed in Section 12.6.1.4.

12.7.4 Peek (^PEEK)

Monitors the output buffer of a specified terminal.

12.7.5 Restore Jobs/Devices (^RJD)

Returns specified jobs or devices to the system.

12.7.6 Set Date (^DAT)

Sets the current date.

12.7.7 Set Time (^TIM)

Sets the current time.

12.7.8 System Shutdown (^SSD)

Performs an orderly shutdown of DSM-11 with a specifiable time delay. System shutdowns are discussed in more detail in Section 11.4.

12.7.9 Distributed Data Base Processor (^DDPUTL)

Allows the setup and control of DDP communications tables and the DDP configuration.

12.7.10 Background Job Attacher (^ATTACH)

Allows you to attach to a running job.

12.7.11 Background Job Detacher (^DETACH)

Allows you to detach from a terminal a currently running job. The terminal is freed. The job can be reattached using the ^ATTACH utility.

12.7.12 Help Text Driver (^HELP)

Provides a complete menu driven help utility for commands, functions, operators, and device parameters.

12.7.13 Magnetic Tape Copy (^TAPECOPY)

Copies a tape to another tape drive.

12.8 Other Miscellaneous System Routines

The other miscellaneous utilities not listed on the menu that are described later in this section are:

Utility	Routine Name
Load a Driver	^LOAD
Unload a Driver	^UNLOAD

12.8.1 Load A Driver (^LOAD)

Loads a loadable device driver, such as a TU58 tape drive, RX02 disk drive, or Binary Synchronous Communications (BISYNC) driver. See Section 10.4.7 for more information on loadable drivers. You may wish to load a driver only when the device is needed if you are short on system space. When the utility prompts for the device, type ? to get a list of the appropriate device driver codes to use.

12.8.2 Unload A Driver (^UNLOAD)

Unloads a loadable device driver.

Chapter 13

Global Structure and Optimization

This chapter describes the logical and physical structure of globals in detail, the growth and development of globals, the placement of globals on disks, and the optimization of the global and data structure.

13.1 Overview of Global Memory-Resident Tables and Disk Blocks

The global module uses the following to place and locate globals on disk and to store data records:

- Volume Set Table
- UCI Table
- Storage Allocation Table
- Disk Table
- Bad Block Table
- Directory Blocks
- Pointer Blocks
- Data Blocks

The tables are discussed in the following paragraphs and in more detail in Chapter 14. The blocks are discussed later in this chapter.

The Volume Set Table is a memory-resident table that indicates which volume sets have been defined and are present on the system, and the volumes and disks contained in the volume set.

All block numbers used by UCIs, globals, and routines are stored as 3-byte numbers relative to the volume set in the range of 0-16777215. Each disk that makes up a volume set is assigned block numbers (relative to the volume set) in volume number order. Block 0 of volume 1 is block 0 of the volume set. Each volume of the set is made up of a series of 400-block units, called "maps." If volume 1 has 24 maps, the first block of volume 2 is block 9600 (relative to the volume set).

The UCI Table (User Class Identification Table) is a disk-resident table that is read into memory when the volume set is mounted. On the disk, the UCI table occupies a full block. Three bytes at location 910, 911, and 912 of the label block of volume 1 of the volume set point to this table. In memory, offset 2 of the Volume Set Table entry points to the UCI table. There is one UCI Table for each volume set. The UCI Table contains information that governs the allocation of disk space for globals (and routines), on a UCI-by-UCI basis.

The Storage Allocation Table (SAT) is a table constructed in memory when the volume set is mounted. The SAT shows, for each volume set, all the maps in the volume set and the maps that have unused blocks. This table is used during global growth to identify maps where the new growth can occur.

Directory, pointer, and data blocks reside on disk memory. Each block type stores a different kind of information. The system stores all information associated with a global on disk in one of these blocks.

The following sections describe disk volumes and DSM-11 volume sets, the general disk layout of DSM-11, and disk blocks.

13.2 Disk Volumes and Volume Sets

A volume is usually one disk (containing data), referred to as a disk volume. The disk volume can be compared to a printed book (that can be thought of as one volume in a set of books). Thus you can have a set of disk volumes that make up a volume set of disks.

In versions previous to Version 3 of DSM-11 several disks mounted on one system can be thought of as one volume set of several disk volumes. With Version 3 you can have more than one volume set if you so desire. Using several volume sets can be a way of organizing a very large data base or handling a large number of users. Each volume set, for instance, can have its own set of 30 UCIs.

A volume set is given a number from 0 to 3, and is referred to as Volume Set 0 (S0), Volume Set 1 (S1), Volume Set 2 (S2), or Volume Set 3 (S3). A volume set also has a 3-character mnemonic, such as SYS. Volume Set 0 is always the system volume set and has as its first volume the system disk.

Disk volumes can be accessible both as individual disks and as data base volume sets. At the lowest level, DSM-11 allows disks to be accessed (using the VIEW command) as series of 1024-byte blocks relative to the start of the volume. Each disk drive has a "physical identity" that is described mnemonically as two letters and a number, for example, "DK0" for RK05 unit 0.

Information can be read off or written onto specific block locations on a disk by using the VIEW buffer. RK05, unit 0, block 5 can be read into the VIEW buffer by the following command:

```
VIEW 5:"DK0"
```

The VIEW buffer can be written to RK05, unit 0, block 5 by:

```
VIEW -5:"DK0"
```

This technique is used by DSM-11 utilities for such tasks as installing the DSM-11 system, formatting and initializing disks, and copying disks.

Part of the initialization procedure is a check for "bad" blocks on the disks. A list of these bad blocks (up to 63 per volume are allowed) is recorded in block 0 of each DSM-11 volume.

A DSM-11 data base volume set is composed of one or more disk volumes whose block numbers are concatenated into a single stream of up to 16,777,216 blocks. The first block number of the second volume is the highest block number of the first volume plus one. Once a volume has become part of a volume set, it must always be mounted when that volume set is mounted. DSM-11 globals grow within a volume set without regard to disk volume boundaries.

Each volume set has a single UCI table that can hold up to 30 user classes. Each UCI is the "root" for both a set of globals and a set of routines. You can access data in one volume set from a UCI in another volume set, see Section 8.2.2 for the syntax to use.

13.3 In-Memory Support for Disk Volume Sets

Each volume set can have a UCI table for that volume set holding up to 30 UCIs. You can also define single-volume volume sets that have no UCI table. Such a volume set can be mounted and used for SDP, journaling, or spooling.

When a volume or volume set that has UCIs is mounted, it becomes one of the four UCI volume sets numbered 0 - 3. The first volume of Volume Set 0 is

always the system disk, from which the system was originally booted. When a volume set is mounted, block 0 is read from volume 1 of the volume or volume set; and information from the second half of that block (called the "label") is used to begin the procedure of establishing the volume set as a mounted volume set. Notice that the first half of block 0 must be reserved for a possible "boot" block.

The following shows the information from the label half of block 0 of volume 1 of each volume set.

Byte	Contains
0	Count of bad blocks
1-191	Space for 63 bad blocks (this information on all volumes)
192-299	Not assigned
300,301	Number of maps (this information on all volumes)
392,393	Code Word (same for each volume in set)
394,395,396	3-character volume set name (same for each volume in set)
397	Volume number (from 1 to 8, ASCII)

(The following information is on volume 1 of the set only.)

398,399,400	Block number of UCI table (low, middle, high)
401	Number of volumes in set (binary number 1-8)
402-443	4 bytes each for volumes 1-8: 1 byte disk type code 1 binary controller type (0-7) 2 bytes number of maps on the volume

13.4 General Disk Layout and Terminology

DSM-11 supports several types of disk drives. Disk types are designated 0 to 7. (See your Software Product Description for the exact disk types DSM-11 supports.) Each disk type represents a controller that can have multiple drives.

Each drive is called a *unit*. Units are designated 0 to 7. Each unit has a unique 3-character code. This code is a mnemonic of two letters and a number. For example, "DK0" is used for RK05, unit 0.

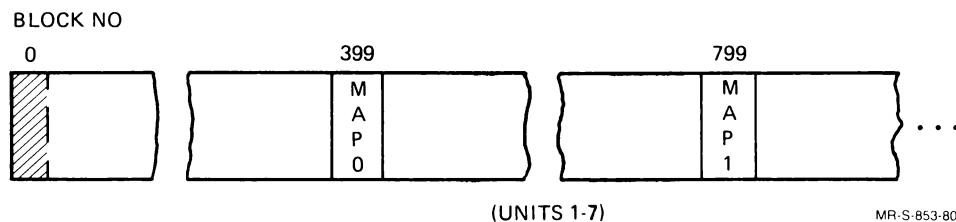
This unit code is encoded within the operating system as a single byte, where bits 5-7 indicate the disk "type." Table 13-1 shows the current disk types supported by DSM-11.

Number	Mnemonic	Disk Code
0	DK	RK05
1	DM	RK06,RK07
2	DR	RM02,RM03,RM05
3	DB	RP04,RP05,RP06
4	DL	RL01,RL02
5	DU	RA80,RA60,RA81, RD51,RX50
6	Unassigned	
7	Unassigned	

Bits 2-4 indicate the unit number from 0 to 7. Bits 0 and 1 of this byte is always zero. This byte is the numerical offset from the start of the Disk Table to the beginning of the drive's entry.

Disks are partitioned into storage units called *maps*. Maps, in turn, are composed of subunits called *blocks*. Each block is 1024 bytes long. (A DSM-11 block is composed of two physical disk sectors, each being 512 bytes long.) A map consists of 400 blocks, numbered 0 to 399. The 400th block (block 399) of every map is called the "map block". The map block contains allocation information about the previous 399 blocks. Figure 13-1 shows the general disk layout.

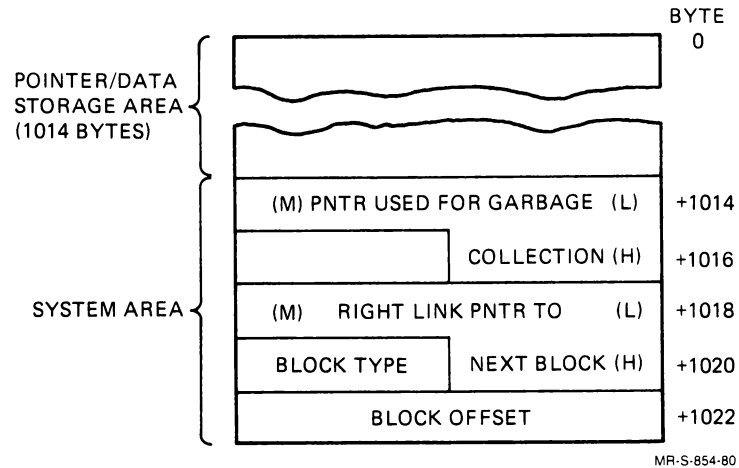
Figure 13-1: General Disk Layout



13.4.1 DSM-11 Block Layout

Figure 13-2 shows the layout of a typical global DSM-11 block.

Figure 13-2: DSM-11 Block



The fields of a DSM-11 block are:

DATA STORAGE AREA

Bytes 0 through 1013 contain information particular to its block type.

POINTER USED FOR GARBAGE COLLECTION

Bytes 1014 to 1016 contain information used by a DSM-11 background job called the *Garbage Collector*. The Garbage Collector returns blocks that have been deleted from a global structure to the system's pool of available blocks. The Garbage Collector gathers deleted blocks in linked chains, called garbage trees. Bytes 1014 to 1016 point to the next right-linked garbage subtree.

RIGHT-LINKED POINTER TO NEXT BLOCK

This 3-byte field points to the next block whose nodes collate immediately following the nodes in the current block.

BLOCK TYPE

Byte 1021 indicates the block type. Block types can be one of six kinds. Block types can also be in a 7-bit or an 8-bit data structure format.

Block Type	7 Bit	8 Bit
Global Directory Block	1	N/A
Pointer Block	2	130
Bottom-level Pointer Block	4	132
Data Block	8	136
Routine Block	16	144
Block Passed to Garbage Collector	32	160

BLOCK OFFSET

Bytes 1022 and 1023 indicate how much of the block is filled, and hence point to the next free byte in the block.

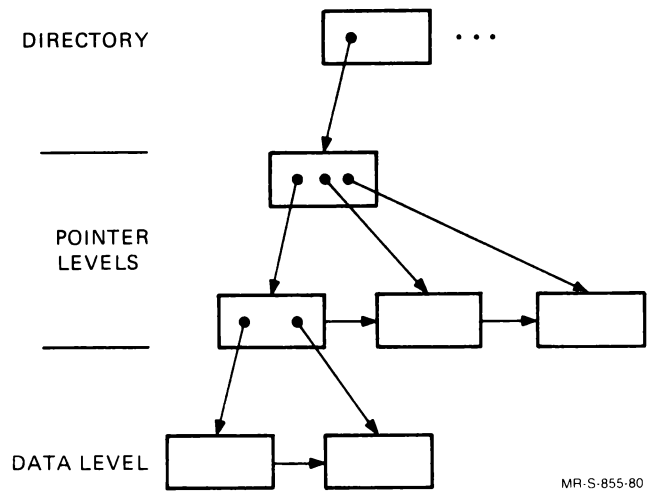
Different block types store different kinds of information. The block types listed previously contain global and routine file information. Directory blocks contain the names of all global or routine files within a UCI.

Pointer blocks hold pointers to subtrees of a global.

Depending on the size of a global, there can be one or more levels of pointer blocks. Bottom-level pointer blocks are the lowest level of pointers in a file. Bottom-level pointers point to either data blocks, if a global, or routine blocks, if a routine. Data blocks store all data records associated with a global. Routine blocks store the lines that belong to a routine.

Appendix B describes the way DSM-11 stores and accesses routine files in more detail. Figure 13-3 shows the relationship between the various block types that form the physical layout of a global.

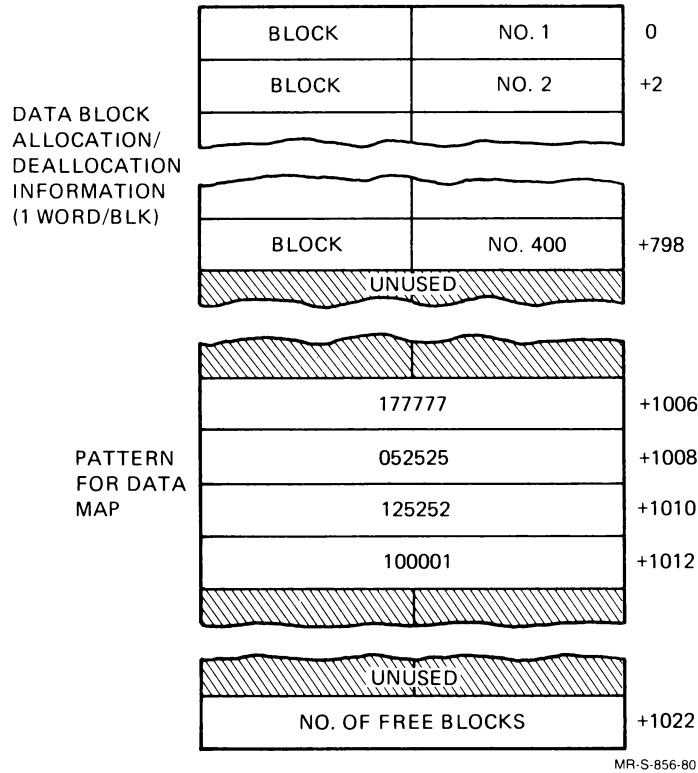
Figure 13-3: Physical Structure of a Global



13.4.2 Map Blocks

DSM-11 uses map blocks to keep track of which blocks are available for use and which blocks are in use. A map block is located after every 399 blocks on the disk. The map block contains allocation information about the previous 399 blocks and itself. Figure 13-4 shows the layout of a map block used by the system to allocate blocks for globals.

Figure 13-4: Map Block



The map block defines the status of each block in the map in the following way. Two bytes are assigned to each block in the map, and a 0, -1, -2, or -3 is placed in the high and low bytes in one of the following combinations:

BYTE		MEANING
HIGH	LOW	
0	0	The block is a free block (available for use).
-1	-1	The block is a system block (a map block or operating system code).
-1	-3	The Disk Block Tally utility changes every two bytes that denote a free block 0,0 to this code if there is discrepancy between the number of blocks allocated and the number of blocks classified as free in bytes 1021 and 1022 of the map block. If this occurs, the system will not allocate any more blocks in this map.

>0 >0 This code is used when a block is in use for a global or routine. The low byte is the UCI number to which the global belongs. The high byte represents the UCI of the job that caused the block to be allocated.

In addition to the allocation/deallocation information described above, the map block contains the following information:

PATTERN FOR DATA MAP

Eight bytes, starting at byte 1006, contain a numerical pattern that defines the block as a global map block.

NUMBER OF FREE BLOCKS

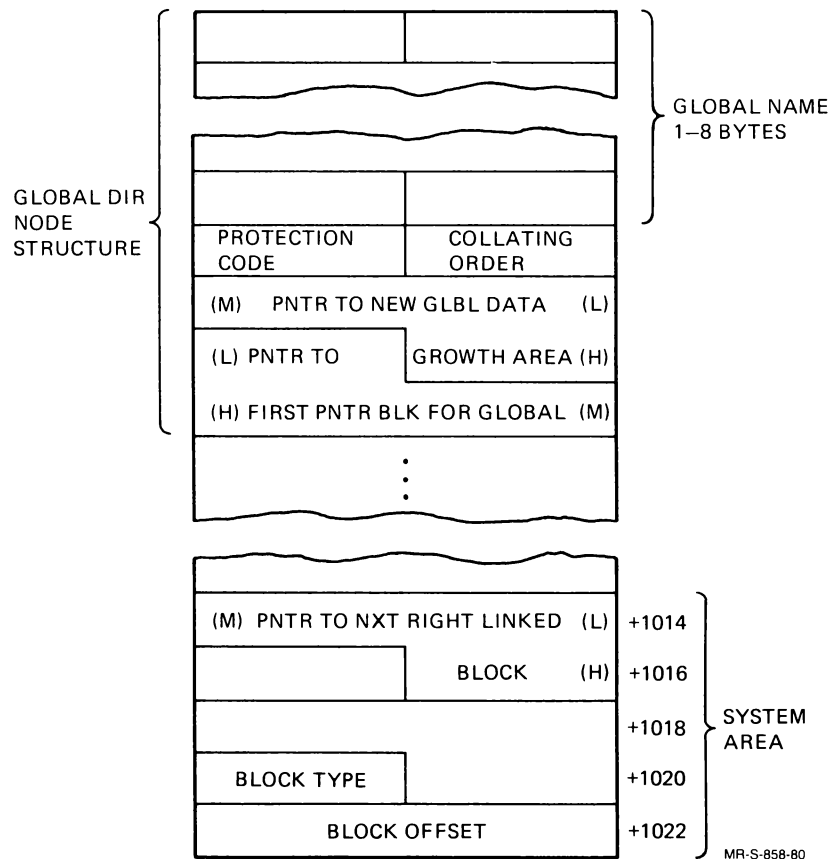
Two bytes, starting at byte 1022, indicate the number of blocks within the map that are free blocks, (that is, the number of occurrences of a 0 entry in both the high and low bytes).

13.4.3 Global Directory Block

Global directory blocks comprise the first level of the global structure that is stored on disk. The global directory level is actually a sequential file that consists of right-linked global directory blocks. Each entry in a block requires 8 bytes plus the global name; therefore, each directory block can point to 63 globals, if each name is the full eight characters long. A directory block can point to as many as 104 names if each name is only one or two characters each. The first block in the global directory file is pointed to by an entry in the UCI table.

Figure 13-5 shows the layout of a global directory block.

Figure 13-5: Global Directory Block



The directory block stores entries in the following format:

GLOBAL NAME

The first field of a directory entry contains the global name. Each name can use up to eight bytes of storage. Global names are stored in compacted form by shifting all bits 1 bit to the left. If the low-order bit is 0, it indicates the end of the global name.

GLOBAL TYPE

The first byte following the global name defines the global type. It contains:

- Bit 0 = 0 if numeric collating
 1 if string collating

- Bit 1 = 0 if 7-bit coded
 1 if 8-bit coded

Bit 2 = 0 not to journal this global
 1 to journal this global

PROTECTION CODE

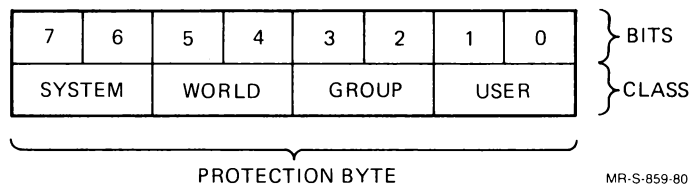
Immediately following the global type is a field that specifies global access restrictions for each of the following user classes:

- SYSTEM
- WORLD
- GROUP
- USER

Section 4.9.3 describes each user class in detail.

This field is 1 byte long. The system assigns two bits to each class within the protection byte. Figure 13-6 shows the bit assignments within the protection code byte.

Figure 13-6: Protection Byte Bit Assignments



The two bit combination within each class can be:

High/Low

- 00 if No Access Privileges
- 01 if READ Only Privilege
- 10 if READ/WRITE Only Privilege
- 11 if READ/WRITE/DELETE Privilege

POINTER TO NEW GLOBAL DATA GROWTH AREA

This 3-byte field stores the block address where the system begins to search for additional data blocks for a defined global. You can change this address through the Growth Area utility (^DGAM).

POINTER TO THE FIRST POINTER BLOCK FOR GLOBAL

This 3-byte field is the block address of the first pointer block for the global. This field specifies where the next level of the global structure begins.

The system area of a global directory block specifies, among other things, the next right-linked block in the directory file. The system area also indicates the block type. The block type for a global directory block is 1.

NOTE

There is no garbage collection on global directory blocks. After a directory block has been created, it cannot be removed.

13.4.4 Global Pointer Block

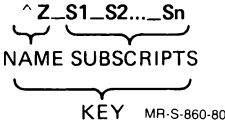
Pointer blocks comprise the intermediate levels of a global tree. Pointer blocks can be one of 2 types:

- Pointer Blocks
- Bottom-level Pointer Blocks

A global can have many pointer levels. Pointer blocks point to bottom-level pointer blocks or to other pointer blocks. Bottom-level pointer blocks are those pointer blocks whose pointers point to data blocks. When a global is created (by the first SET command), one pointer block and one data block are created. The pointer block is a bottom-level pointer block, and contains one pointer (to the data block).

Pointer blocks reference data in related blocks through an identifier called a *global key*. A global key is the compression of a global variable name and its subscripts. (From here on, global keys are referred to simply as *keys*.) The system retrieves data records by referencing the appropriate key. Like records, keys can be variable length. Figure 13-7 illustrates the construction of a variable length key from a global variable name.

Figure 13-7: A Global Key



Pointer blocks (and data blocks) store keys in "compressed" form, by a process called *key compression*. Key compression is accomplished by storing only the characters necessary to uniquely identify a key. It is a feature of DSM-11 that helps conserve storage space and increase performance.

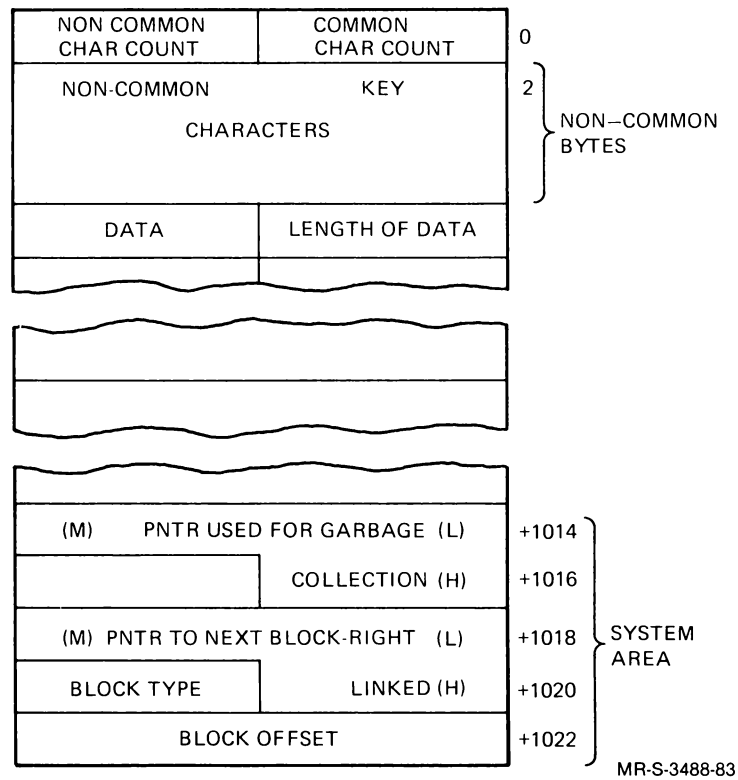
Suppose you defined:

\wedge NAME(DATA)
 \wedge NAME(DATUM)

The system stores \wedge NAME(DATA) as a full key. However, it compiles and stores a compressed key out of \wedge NAME(DATUM). The system compiles the compressed key by omitting the characters it has in common with \wedge NAME(DATA), which leaves UM. UM is the key stored for \wedge NAME(DATUM) (along with the number of characters it has in common with \wedge NAME(DATA), which is called the common character count).

Figure 13-8 shows the layout of a global pointer block.

Figure 13-8: Global Pointer Block



Pointer blocks store pointer entries in the following format:

COMMON KEY CHARACTER COUNT

This 1-byte field contains the number of key characters in common with the key of the previous node.

NONCOMMON KEY CHARACTER COUNT

The following field contains the number of key characters not in common with the key of the previous node. This field is also 1 byte long.

KEY CHARACTERS

The next n bytes contain the noncommon characters of the global key. The key is stored as the concatenation of the global name and its subscript. Key compression occurs at all pointer levels.

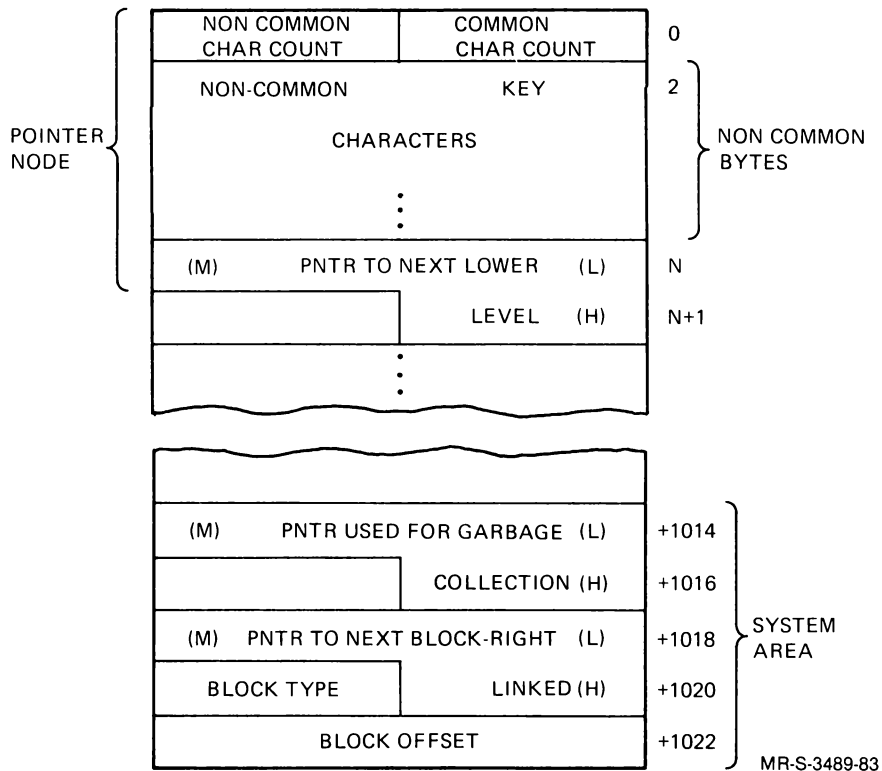
POINTER TO NEXT LOWER LEVEL

The following three bytes specify the address of the next level of the global. If the block is a pointer block, it points to another pointer block or to a data block.

13.4.5 Global Data Block

Data blocks are the deepest level of the DSM-11 data base and the bottom of the global tree. Data blocks store all data records associated with a global. Each data block is pointed to by a bottom-level pointer block. Figure 13-9 illustrates the structure of a global data block.

Figure 13-9: Global Data Block



Records are stored in data blocks in the following format:

COMMON KEY CHARACTER COUNT

This 1-byte field contains the number of key characters in common with the key of the previous node.

NONCOMMON KEY CHARACTER COUNT

The following field contains the number of key characters not in common with the key of the previous node. This field is also 1 byte long.

KEY CHARACTERS

The next *n* bytes contain the noncommon characters of the global key. The key is stored in compressed form, and is the same as the key in the block that pointed to the data block.

LENGTH OF DATA

This 1-byte field indicates the number of characters in the data record that follows. This number can be between 0 and 255.

DATA

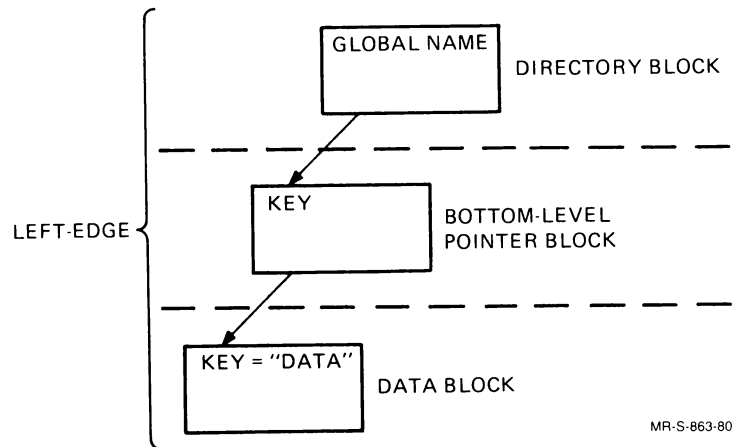
Data records are stored in this area. Storage ranges between 0 and 255 bytes. Individual records are terminated by a -1.

13.5 Global Growth and Development

13.5.1 Global Initialization And Structure

Figure 13-10 shows how DSM-11 initializes a global when you define one node.

Figure 13-10: Global Initialization



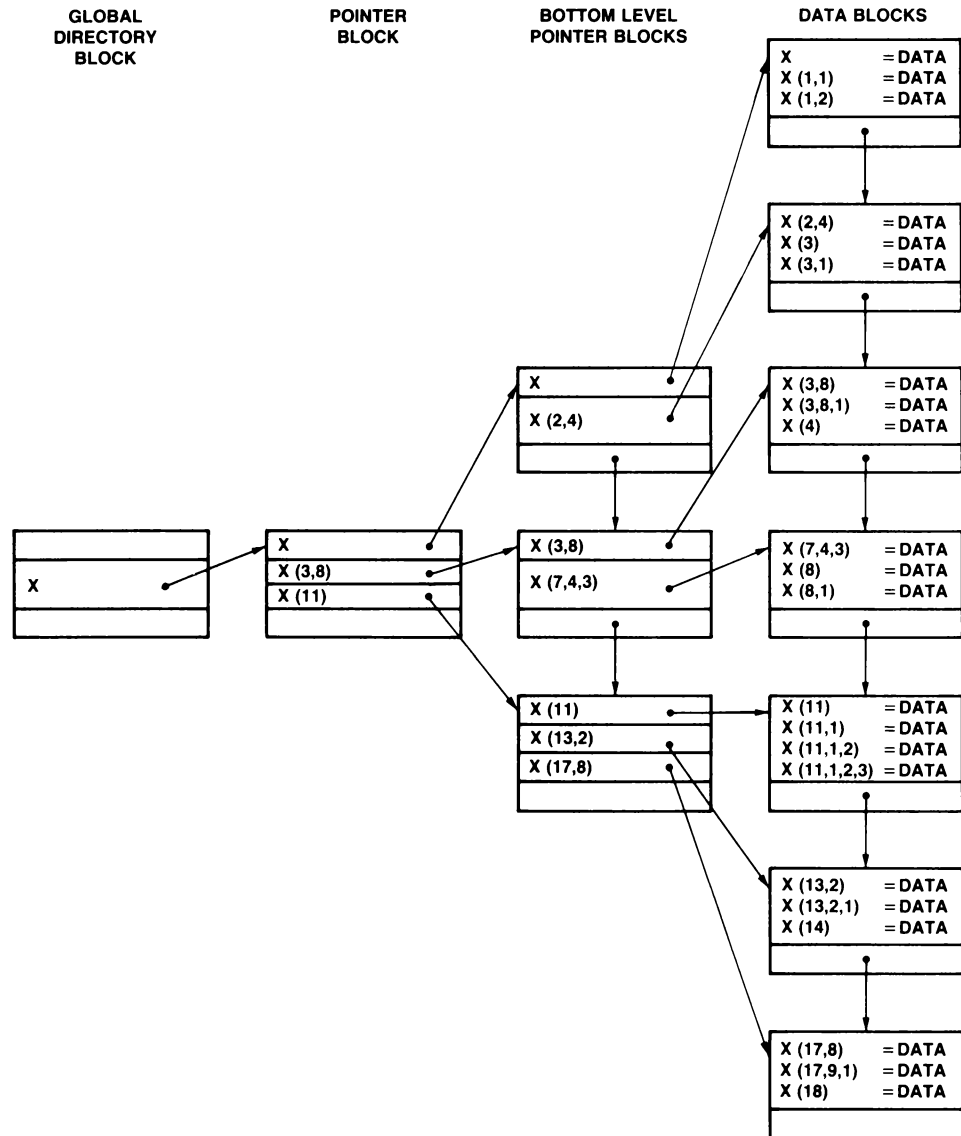
The sequence of blocks shown in Figure 13-10 is referred to as the left edge of the global. Globals grow from their left edge, starting at the data level. When you assign a data value to a node of a global array, the system inserts the record in a data block according to the collating sequence of its key. As more nodes are defined, the data level expands horizontally, and is automatically right linked. As data blocks fill and overflow into the next right-linked data block, global trees expand vertically and to the right. Data level expansion forces pointer blocks to adjust themselves so they continually point to the proper pointer or data blocks. New pointer blocks are automatically right linked in collating order.

The directory level is a sequential file. Thus, the system appends the directory entry for a new global to the end of the directory file. (If you do not define the directory-level node, the global module assigns that node a null value ("")). If four global names exist in a directory block, the global module places the next

entry in the fifth position and accesses it accordingly. The directory entry always points to the first pointer block on the left edge of the global.

Figure 13-11 illustrates the relationship between the various block types, keys, and data records that form a typical global.

Figure 13-11: Typical Global Layout



MR-S 864 80

13.5.2 Global Levels And Disk Access

The number of levels a global generates depends on two factors:

1. The amount of data the global stores
2. The degree to which the keys global stores are unique

The latter factor affects the depth of a global because it forces the system to use more storage space for each key. If keys have mostly unique characters (when compared to each other), the system gains very little through the use of key compression.

Nonetheless, a global consisting of a million records typically generates only three pointer levels: two levels of pointer blocks, and one level of bottom-level pointer blocks. To generate a global with more than three pointer levels, the data records have to be consistently large; and their keys must have many unique values.

After a global generates a level, that level always exists, even if you KILL a large section of the global. Thus, a global never shrinks vertically, only horizontally. When you KILL a portion of a global, the system demarks the region and sends it off to the Garbage Collector so the blocks can be returned to the free block pool. The nodes that remain are linked together with right-linked pointers. Yet the global retains its original depth, and the same number of disk accesses is necessary to retrieve the remaining records at the data level.

To reduce the depth of a multilevel global, you must save it (using the Global Save utility), KILL the directory entry, then restore the global to the disk. This procedure reconstructs the global in its most efficient form.

13.5.3 Reserved Disk Space

You can avoid an unexpected data base overflow due to global growth by reserving disk blocks from each volume set. You are given the opportunity to do this when building a system (using [^]SYSGEN).

When you reserve disk blocks, you set aside blocks from the last map to be used only when *every* map has no blocks remaining. Each time a reserved block is used, the job that caused the reserved block to be used is given a <DKRES> error at the completion of the command that caused the block allocation. The System Manager must now make new space available (presumably by killing unneeded globals).

Finally, when all reserved blocks are consumed, SET or ZSAVE commands receive <DBOVF> errors when they cause block allocation attempts. These errors may indicate that your data base structure is corrupted.

13.5.4 Overflow From One Disk To Another Disk

You can have globals that are larger than one disk volume. Globals naturally grow to higher numbered blocks within the volume set. You can control the extent to which data and routines can grow by using the Global Growth Area utility (^DGAM).

The System Manager's UCI (UCI #1) can never overflow to another disk volume. This is necessary to allow the baseline system to run; the baseline system includes only the system disk. When you are creating additional UCIs, be sure to leave space for ^SYS in UCI 1 to add patches and Caretaker-logged errors. If the system disk becomes full, and further SET commands are done for any reason, a <DBOVF> error results.

13.6 Global Optimization Techniques

This section deals with the aspects of globals that can be controlled and that can enhance the performance and the ability to maintain a data base application if controlled properly.

As a DSM-11 programmer or system manager, you have direct control over the following aspects of your global files:

- Placement of directory, pointer and data growth areas on disk, and placement of the new data growth area for individual globals
- Global key structure
- Data record structure

You can also take advantage of the DSM-11 disk cache by clustering accesses to your data base.

The following sections discuss each of these topics.

13.6.1 Optimizing Disk-Cache Usage

The disk cache is a portion of main memory that stores exact copies of disk blocks. DSM-11 uses the disk cache to minimize physical disk reads and writes.

Segments of the disk cache are referred to as disk buffers. By using disk buffers that are sharable among all users, the system need only keep one copy of any disk block in memory.

DSM-11 allocates disk cache by a least-recently-used algorithm. When the Global Module requires a disk block not already in memory, it overwrites the oldest (least-recently-used) disk buffer. Because of this, blocks that are used frequently (for example, global directory blocks) tend to stay in memory.

Periodically, the system writes all buffers that have been modified to the disk. This ensures that frequently used data always have an up-to-date image on the disk.

The use of a disk cache makes modifying data in certain sequences more efficient. Consider the following example:

Suppose a data base consisted of 20 globals, named A through T. Suppose further that each global had 1000 nodes whose subscripts were 1 to 1000; that is, each global had nodes (1), (2), (3)...(1000).

If it were necessary to modify the first 500 nodes of each global in the data base, the caching algorithm would cause DSM-11 to process the modifications done in the order shown in Sequence 1 faster and more efficiently than the same modifications done in the order shown in Sequence 2:

Sequence 1

1st Pass	2nd Pass	3rd Pass...	20th Pass
$\wedge A(1)$	$\wedge B(1)$	$\wedge C(1)$	$\wedge T(1)$
$\wedge A(2)$	$\wedge B(2)$	$\wedge C(2)$	$\wedge T(2)$
$\wedge A(3)$	$\wedge B(3)$	$\wedge C(3)$	$\wedge T(3)$
$\wedge A(4)$	$\wedge B(4)$	$\wedge C(4)$	$\wedge T(4)$
$\wedge A(500)$	$\wedge B(500)$	$\wedge C(500)$	$\wedge T(500)$

Sequence 2

1st Pass	2nd Pass	3rd Pass...	500th Pass
$\wedge A(1)$	$\wedge A(2)$	$\wedge A(3)$	$\wedge A(500)$
$\wedge B(1)$	$\wedge B(2)$	$\wedge B(3)$	$\wedge B(500)$
$\wedge C(1)$	$\wedge C(2)$	$\wedge C(3)$	$\wedge C(500)$
$\wedge D(1)$	$\wedge D(2)$	$\wedge D(3)$	$\wedge D(500)$

$\wedge T(1)$ $\wedge T(2)$ $\wedge T(3)$ $\wedge T(500)$

The following improvements in system performance occur by modifying the data base in the order shown in Sequence 1:

1. The system transfers most of the pointer blocks for each global into cache and keeps them there. This reduces search time and decreases the number of disk accesses. In Sequence 2, it is likely that a block related to $\wedge A$, for example, would be overwritten by the time you wanted to modify it again.
2. The overall number of physical disk writes is reduced, because Sequence 1 concentrates the modification of data to fewer disk buffers over a fixed period of time.

If you have the most frequently used data in memory, it can improve throughput significantly. Most data base applications require far more data base reads than data base writes; applications of this kind should have as much cache as possible.

13.6.2 Placing UCIs On The Data Base

When you first create a UCI (using the $\wedge UCIADD$ utility), you have the opportunity to place the first global directory block anywhere on the disk volume set. You can also specify the default block addresses for future pointer and data block allocation. Consider the following aspects of the DSM-11 global structure before choosing values for these block numbers:

- Access to a particular global node usually requires several disk reads at the directory and pointer levels, but only one read of a data block

- Data blocks outnumber pointer blocks by a large factor, often 50 to 1 or more

Because there are comparatively few pointer blocks, yet more frequent access to them, you can improve data base efficiency by "bunching" the pointer blocks together. You can do this most effectively if you can predict, at least roughly, the total space required for pointer blocks. You can then specify the pointer growth area separated by enough space from the data growth area so that the two spaces do not overlap. The pointer area should precede the data area.

Figure 13-12 shows the allocation of disk storage for two UCIs. It also shows the UCI Table entries for both, and how they relate to the disk.

Figure 13-12: Disk Allocation for Two UCIs

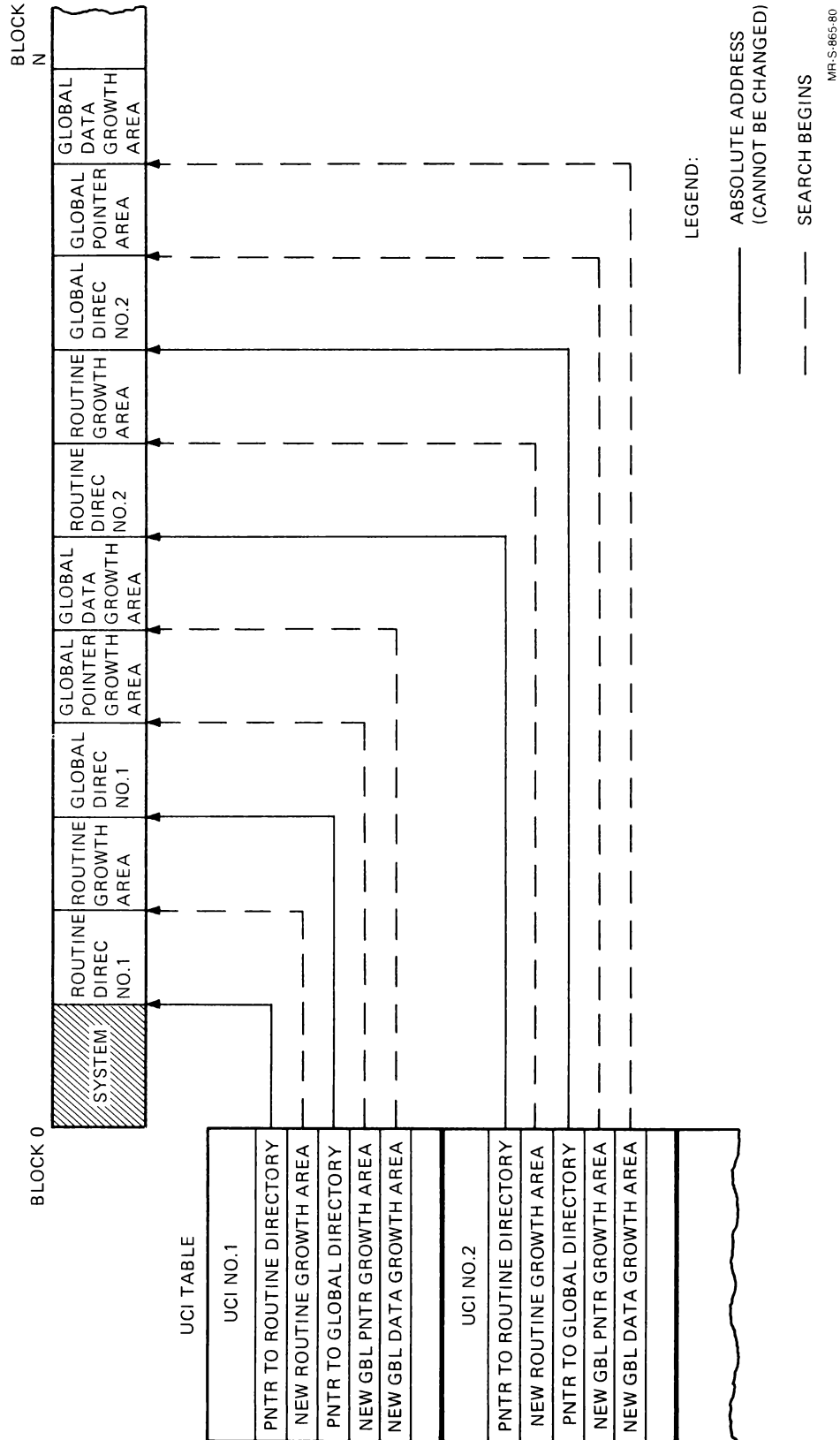
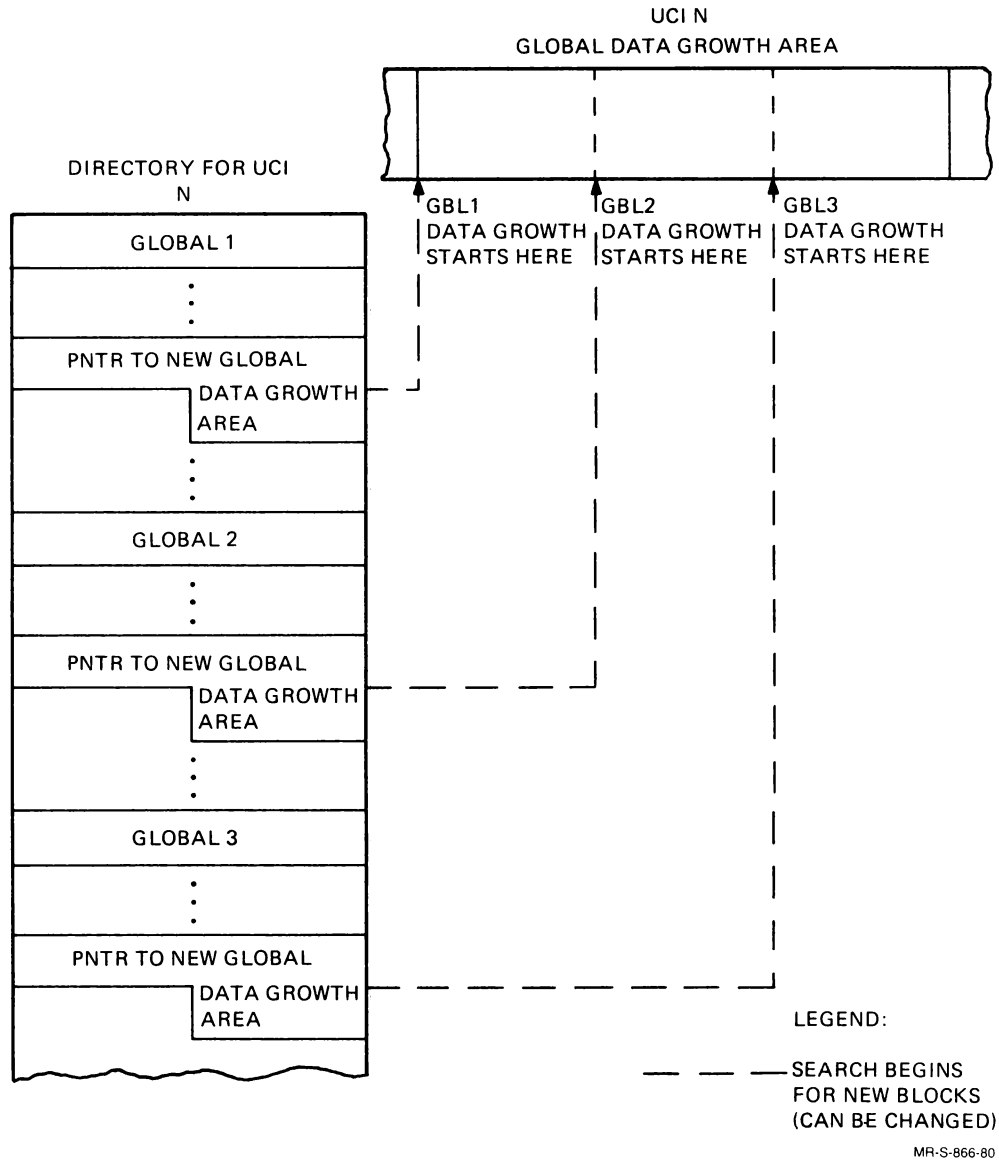


Figure 13-13 shows the allocation of the Global Data Growth Area for individual globals.

Figure 13-13: Global Data Growth Area Allocation for Three Globals



13.6.3 Global Disk Block Allocation (^DGAM And ^%GLOMAN)

You can also allocate disk blocks for globals on a disk-by-disk basis rather than a UCI basis. You can use the following utilities to do this:

- Global Growth Area (^DGAM)
- Global Management (^%GLOMAN)

13.6.3.1 Global Placement — At the time a UCI is created, two default growth areas are assigned. The first growth area is reserved for global directory blocks, global pointer blocks, and routine directory blocks. The second growth area is for both global and routine data blocks. When a specific global is created, however, you can override the UCI default growth area for data blocks using the ^%GLOMAN routine. You can also use this routine to place the first pointer and data blocks of the new global. When you place a global using ^%GLOMAN, the following regime for allocation of new pointer and data blocks occurs:

- Global pointer blocks are assigned starting at the block at which the global is placed, and continuing to the end of map. When the map has been exhausted, blocks are assigned from the default area for the UCI.
- Global data blocks are assigned starting at the block at which the global is placed, and continuing to the end of the map. When the map has been exhausted, blocks are assigned from that global's growth area, if you assigned a growth area using ^%GLOMAN. If you did not assign a special growth area for that global, blocks are assigned from the default data block growth area for the UCI.
- Assignment of disk blocks for all block types proceeds from lower numbered blocks to higher numbered blocks, except where noted. This means that globals should be placed with enough separation to prevent overlapping.

13.6.3.2 Controlling Global Growth — You may wish to use Global Management (^%GLOMAN) and Global Growth Area (^DGAM), together with the system's block allocation algorithm, to control as closely as possible the bunching of disk blocks for a particular global. The following structures are optimum for globals up to approximately 25 megabytes in size:

- The first pointer block of a global should be placed in the first block of a map. The pointer area then grows to higher numbered blocks until that map is filled. One entire map (399 usable blocks) should suffice for a global of at least 25 megabytes.

- The global's first data block should be placed in the first block of the next sequential map, and the growth area should also be assigned that same block.

For globals larger than 25 megabytes, one map is probably not sufficient to hold the entire pointer structure. In that case, you have no choice but to let the pointer blocks fall into the UCI default area. You can still control the placement of the data blocks, however.

Chapter 14

DSM-11 Tables and Memory-Resident Data Structures

This chapter provides information about the tables and memory-resident structures of the DSM-11 operating system.

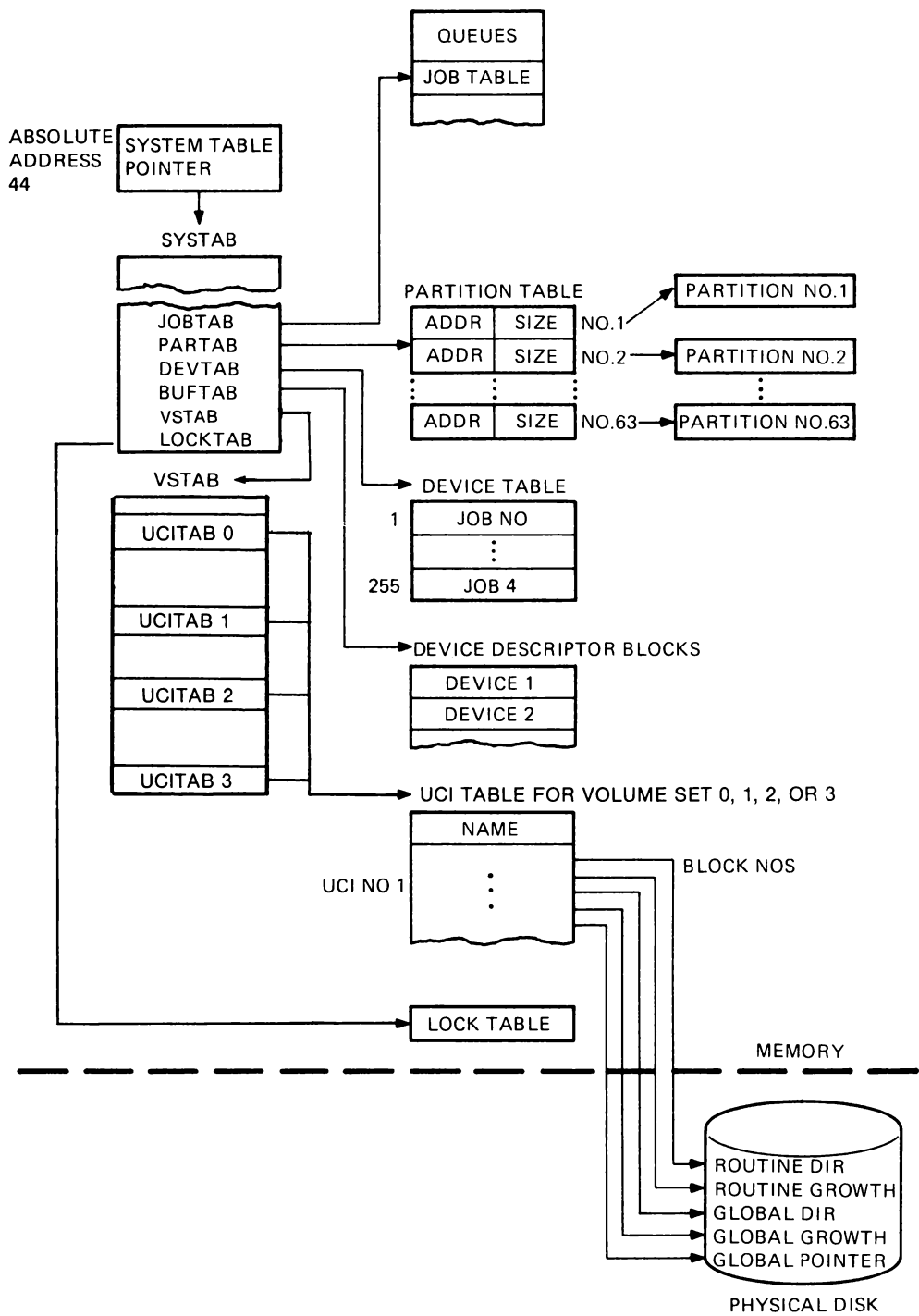
14.1 Introduction

The DSM-11 operating system maintains a number of tables and memory-resident structures that contain system control and status information. Table 14-1 summarizes the function of the tables and data structures described in this chapter. Figure 14-1 shows the linkage between the system tables and data structures described in this chapter.

Table 14-1: DSM-11 Tables and Memory-Resident Data Structures

Table/Data Structure	Function
System Table	Specifies system constants and parameters. Holds pointers to all other tables.
Device Table	Specifies which devices are "OPENed" and by which job, and which devices are not present.
Volume Set Table	Stores the addresses of the associated UCI Table and the Storage Allocation Table, as well as the number and type of volumes within the volume set.
UCI Table	Stores UCI codes for currently mounted disks and the addresses of associated global and routine directories and growth areas.
Storage Allocation Table	Lists all maps within a given volume set and indicates which maps have unused blocks in them.
Disk Table	Lists all disk drives, indicates whether a disk is mounted in the drive, and contains a pointer to the associated Bad Block Table.
Partition Table	Specifies the size and location of each user partition in the system.
Job Table	Contains job status information.
Lock Table	Maintains a list of variables that have been locked by the LOCK command.
Device Descriptor Block	Stores device-specific information. Each device in the system has an associated DDB.
Partition	Stores all job-specific information.

Figure 14-1: System Linkages



MR-S-3490-83

All table information is essential for system operation. The System Manager or programmer can use the data stored in these tables to monitor the state of

the operating system. The VIEW command and \$VIEW function allow privileged users to access and modify table data as required.

NOTE

Modify internal memory (or disk) with extreme caution. Careless alterations can have unpredictable results on system operation.

Privileged users can use the SYSTEM UTILITIES menu (^SYS) to display and examine certain system tables. These utilities are also listed under Section 12.6.4, except for the Terminal Device Descriptor Block utility (^DDR), which is described in Section 12.6.3.4. The tables you can examine with these utilities are:

- System Table
- Job Table
- Lock Table
- Device Table
- Partition Table
- UCI Table
- Partition Vector Table
- Volume Set Table

System utilities are not available to access the Storage Allocation Table and the Disk Table.

14.2 System Table (SYSTAB)

The System Table contains the following information:

- System constants
- Indicators
- Control data
- Special buffer addresses
- Address pointers to other tables in the system
- Errors for magnetic tape or disk

The specific addresses of the other tables in the system and the System Table itself may change in succeeding versions of the operating system. Thus, access to system tables should always be made initially through memory location 44 (54 octal). This address always contains the base address of the System Table.

The relative position of the System Table entries is also fixed so that you can locate all other tables from the base address of SYSTAB. You can obtain the offset into the System Table for all memory-resident tables with the ^SYSTAB dump routine.

Examples:

To obtain the address of the System Table, type:

```
>WRITE $V(44)
```

To obtain the address of the first entry in the Job Table, type:

```
>SET A=$V(44)
```

```
>WRITE $V(A+4)
```

The "4" is the offset into SYSTAB for the Job Table.

To combine the previous two operations, type:

```
>WRITE $V($V(44)+4)
```

14.3 Job Table

The Job Table uses two bytes to specify the status information about each *job* running in the system. A "job" as defined by DSM-11 is any user activity that requires the use of a partition. Thus, logging in to the system initiates a job. A routine started in another partition with the JOB command is also called a job.

Immediately preceding the Job Table are the headers for the various job queues. Each queue contains the number of the next job waiting for a particular service from the CPU or the executive. Jobs in queues are connected in a linked list where each entry points to the next entry in the same queue. The first job number entered in the queue is the first job processed. If a job is not in a queue, it is in a "hung" state. Hung jobs are waiting for an event to complete, such as output to a printer.

The maximum number of user jobs that can be in the Job Table is 63. These jobs are identified by positive integers (1 to 63, inclusive). Note that job numbers 64 to 127 are reserved for system jobs only. The algorithm for computing Job Table system addresses for jobs 1 to 63 is:

$\$JOB = \text{Job Table offset} / 2$

The algorithm for computing the Job Table system addresses for jobs 64 to 127 is:

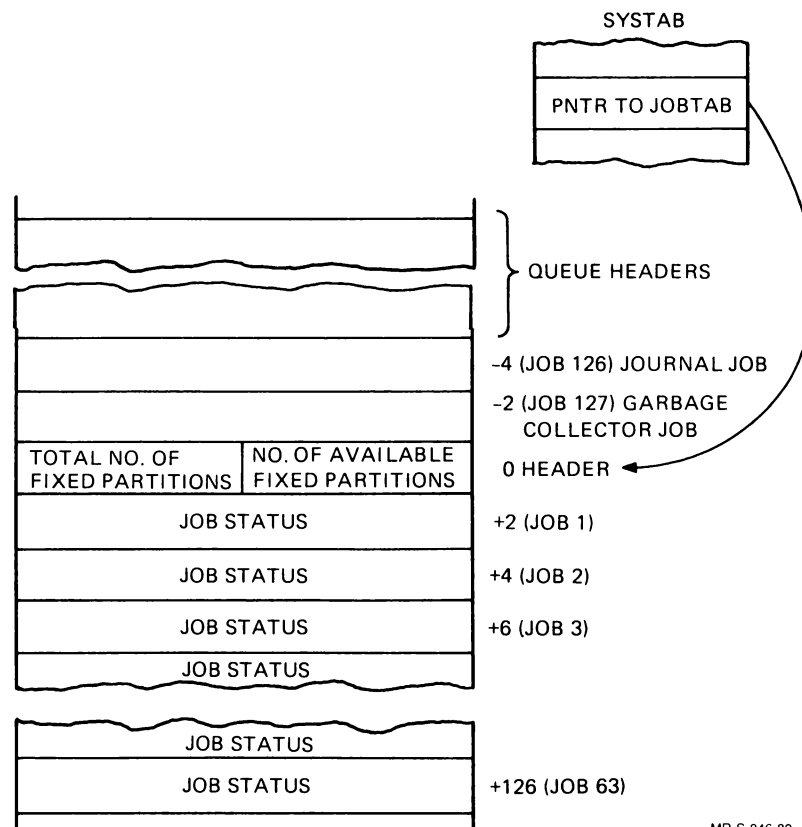
$\$JOB = (\text{Job Table offset} + 256) / 2$

where:

$\$JOB$ is the $\$JOB$ special variable

Figure 14-2 shows the layout of the Job Table.

Figure 14-2: Job Table



As shown, special system jobs are designated by negative offsets from the start of the Job Table.

14.4 Lock Table

The Lock Table contains a list of all local and global variables that have been locked with the LOCK or ZALLOCATE command. A variable entered in the Lock Table by a job cannot be locked by any other job. Like other tables, the

Lock Table has a pointer entry in SYSTAB. The pointer to the base memory management block number of the Lock Table is in SYSTAB at offset +64. (A memory management block number is an absolute physical memory address divided by 64.)

The pointer to the size entry of the Lock Table is in SYSTAB at offset +66, and specifies the number of bytes allocated to the Lock Table. The size of the Lock Table can be specified at SYSGEN and must be in the range of 64 to 8192 bytes.

Use the utility routine, ^LOCKTAB, to show the current contents and the structure of the Lock Table, for example:

```
>DD ^LOCKTAB
```

Contents of Lock Table (1152 bytes allocated)

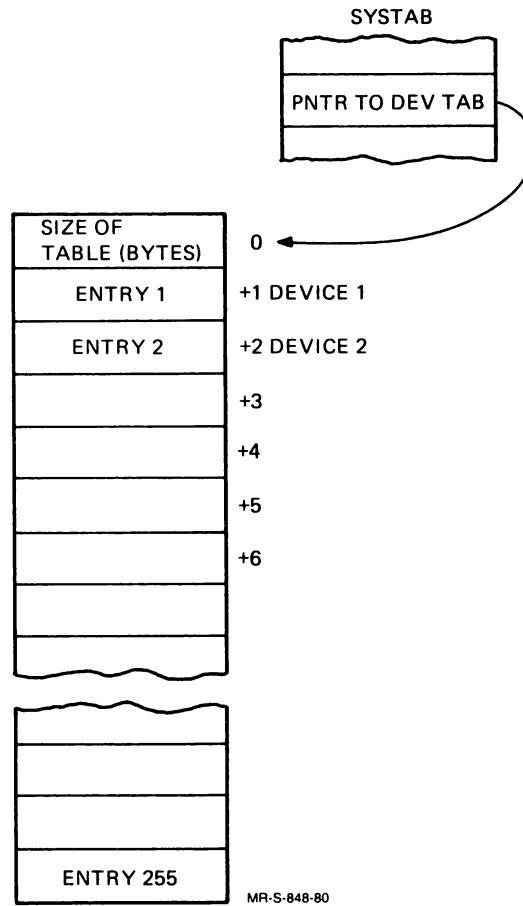
Mode	Job	UCI	Volume Set	Reference
ZA	4	JSS	SYS	^S
ZA	4	JSS	SYS	^D
LOCK	5	JHR	SYS	^U

14.5 Device Table

The Device Table contains entries for each possible I/O device in the system. The first byte in the table (DEV TAB + 0) specifies the number of entries in the table. The number of devices in the table is equal to the value at DEV TAB + 0. Each following byte specifies an I/O device.

The position of the byte in the Device Table is equivalent to the device number. For instance, device 63 is at DEV TAB + 63. The Device Table specifies the status of a device in the following way: if the device does not exist, the contents of the entry is 255; if the device is available, the contents of the entry is 0; otherwise, the entry contains the number, times two, of the job that owns the device. Figure 14-3 shows the layout of the Device Table.

Figure 14-3: Device Table



14.6 Partition Table

The Partition Table specifies the size and base address of each defined partition in the system, including both fixed partitions and currently running dynamic partitions. The entries in the Partition Table are set during SYSGEN. The table consists of two-byte entries. The first entry contains the default partition size in the low byte (expressed in 1024-byte blocks). This size is 8 (8192 bytes), unless modified at SYSGEN time. The high byte is not used. The remaining entries in the table specify partition information.

Bits 0 to 3 of a partition entry specify the size of the partition as the number of 1024-byte increments minus 1. Bits 4 to 15 of the partition entry contain the base address of the partition in 1024-byte multiples. Figure 14-4 shows the layout of the Partition Table.

SYSTAB + 12 points to the Volume Set Table. SYSTAB + 34 contains the size (in bytes) of the Volume Set Table entries. The table is limited to four volume sets.

The Volume Set Table contains:

- The name of the volume set.
- The address of the UCI table for each volume set.
- The address of the Storage Allocation Table for each volume set.
- The type and unit of each volume within the volume set.

At system start-up, UCI tables are set up for the number of volume sets specified by the configuration. To print out the Volume Set Table, run `^%STRTAB`.

Figure 14-5 shows the Volume Set Table.

Figure 14-5: Volume Set Table

VOLUME SET TABLE		0
UCI TABLE ADDRESS		2
SAT TABLE ADDRESS		4
EITHER: NO. OF VOLUMES + 49152 OR: STARTING MAP IF ONLY ONE VOLUME		6
SIZE (IN MAPS) OF VOLUME 1		8
SIZE OF VOL. 2 (LO BYTE)	TYPE/UNIT VOL. 1	10
TYPE/UNIT VOL. 2	SIZE OF VOL. 2 (HI BYTE)	12
•	•	•

MR-S-3492-83

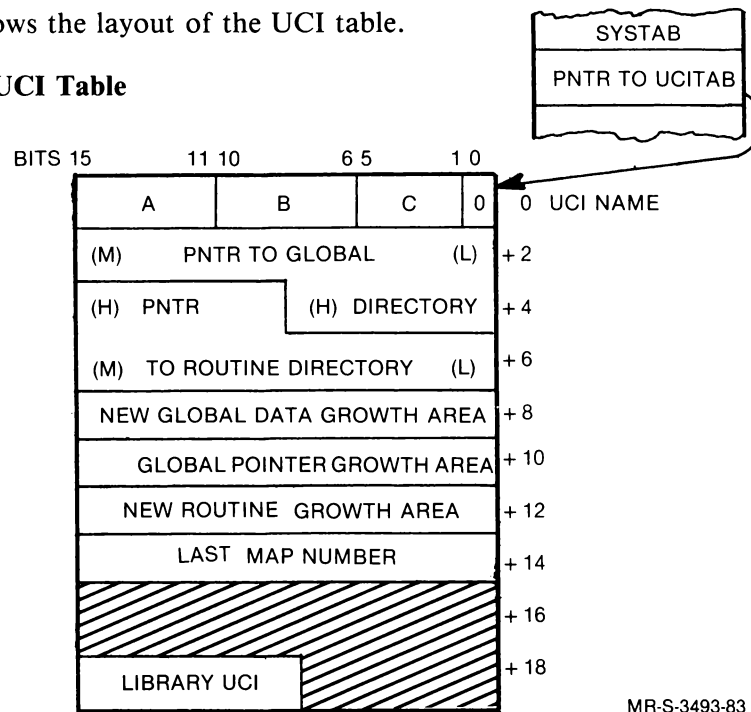
14.8 The UCI Table

Each volume set has an associated UCI (User Class Identification) Table. The UCI Table specifies all UCIs in the volume set, the disk location of the associated global and routine directories, and growth area pointers. These UCIs have contiguous entries in the UCI Table, one for each user class defined during the SYSGEN procedure. There can be as many as 30 UCIs in each volume set. The first entry in the table is defined as the system UCI (UCI #1).

The UCI Table is mapped by the address of the second word of the volume set entry. Each entry in the UCI Table is 20 bytes long and is divided into 8 fields, as shown in Figure 14-6. These fields define the area of the disk that each user class uses for the storage of routines and globals.

Figure 14-6 shows the layout of the UCI table.

Figure 14-6: UCI Table



MR-S-3493-83

Each entry in the UCI table is 20 bytes long and contains the following information:

UCI NAME

The UCI name occupies the first two bytes in each table entry. The UCI name is stored in compacted form (15 bits) as three uppercase ASCII characters.

POINTER TO THE GLOBAL DIRECTORY

The next three bytes contain the block address of the first global directory block.

POINTER TO THE ROUTINE DIRECTORY

The following three bytes contain the block address of the UCI routine directory block.

NEW GLOBAL DATA GROWTH AREA

The following two bytes store the map number where the system begins to search for new global data blocks. You can override this address for a particular global by using the Growth Area utility (^DGAM).

NEW GLOBAL POINTER GROWTH AREA

The following two bytes in each table entry store the map number where the system begins to search for new global pointer blocks.

NEW ROUTINE GROWTH AREA

The following two bytes store the map number of the location where the system begins to search for new blocks to store routine lines.

LAST MAP NUMBER ALLOWED FOR THIS UCI

The following two bytes store the highest map number that this UCI can use. The next three bytes are unused.

LIBRARY UCI

One byte stores the library UCI for this UCI. If this number is 0, then UCI 1 of Volume Set 0 is the library. The library UCI contains all %oroutines and %oglobals that can be accessed by this UCI.

14.9 Storage Allocation Table (SAT)

Each volume set has an in-memory Storage Allocation Table (SAT) to speed up the search for unused blocks when new data space is required. This table is a series of bits, each of which represent the status of a map. Each map is a 400-block disk segment. Bits are set for maps that have blocks available. A bit is set to 0 if the map has no blocks available. Each volume set's SAT starts with a one-word count of the total number of maps in the volume set. The SAT for each mounted volume set is pointed to by the third word of its Volume Set Table entry.

Storage Allocation Table

Byte	Contains
0,1 (word 0)	Number of maps in the volume set
2, bit 0	Set if map 0 has blocks available
2, bit 1	Set if map 1 has blocks available
.	
.	
.	
3, bit 0	Set if map 8 has blocks available

SYSTAB + 394 holds the total number of bytes reserved for SAT space in the system. Volume Set 0 (the system volume set) SAT space always starts at the beginning of the reserved space.

Whenever a volume set is dismounted and a gap is created in the SAT space, the SAT spaces for higher numbered volume sets are moved downward to leave the largest possible continuous open space.

14.10 Disk Table

The Disk Table describes the size, "mount" status, and Bad Block Table (BBT) location for each of as many as 64 total disk drives. There are eight drives for each of the eight possible disk controller types. To print out the Disk Table, run `^DISKMAP`

Each drive has two words in the Disk Table in unit number order within its controller types. The two words are used as follows:

Word	Bit	Description
0	15	Set to 1 if the drive has a disk mounted for "VIEW only".
	14	Set to 1 if a disk is mounted within a volume set
	13-8	Offset, in 64-byte blocks of this disk's 192-byte Bad Block Table. The base address, in 64-byte blocks, of the entire bad block area is at SYSTAB + 86.
	7-0	This byte holds a code that identifies the drive code. If no drive exists, value is 0. If the drive is an RK05, value is 1. If the drive is an RK06, value is 2, and so on.
1		Holds the number of maps on the volume. This word is set whenever the disk is either mounted within a volume set, or mounted for VIEW only but with a Bad Block Table.

14.11 Bad Block Table (BBTAB)

Each DSM-11 disk volume is allowed up to 63 "bad" blocks that cannot be accessed because of some physical defect on the disk. These bad blocks are recorded in a table in block 0 of the volume. This means that block 0 must be accessible on all DSM-11 disks.

When a DSM-11 system is started up, the disk drives that are part of the system are each assigned 192 bytes of memory for holding the Bad Block Table of the disk volume mounted in that drive. An area in memory is reserved for Bad Block Tables; SYSTAB + 86 is the address of this reserved area. The Disk Table contains an entry for each drive pointing to the address of the Bad Block Table (in the reserved Bad Block Table part of memory).

When a disk READ or WRITE request reaches the disk handler, a check is made to determine whether the requested block number is in the Bad Block

Table for that drive. If it is, a replacement block number is computed as follows:

$\text{map count} * 400 + \text{BBTAB position}$

The map count is the number of maps on the volume (also listed in the Disk Table). Since that block can also be bad, it may be necessary to calculate an additional replacement block number. This is done by using the same algorithm, recursively, until a good replacement block is found. The BBTAB position starts with zero.

14.12 Device Descriptor Block (DDB)

Each device on the system has a Device Description Block (DDB) associated with it. Each DDB contains information about the device with which it is associated.

The DDBs for all devices reside in Kernel space in the following order:

1. Single-line terminal-type devices (DL11, DMC11, LP11)
2. Multiplexer terminal-type devices (DH11, DZ11)
3. Magnetic Tape devices
4. In-memory job communication (JOB COM) devices
5. Journal buffers
6. SDP devices
7. DDP lines

The following lists the offset location in the System Table of the pointers to the start of the various classes of DDB:

Device	Pointer Location
Single-line devices	SYSTAB + 10
Multiplexer devices	SYSTAB + 20
Magnetic Tape devices	SYSTAB + 22
JOBCOM devices	SYSTAB + 358
Journal buffers	SYSTAB + 402
SDP devices	SYSTAB + 408
DDP lines	SYSTAB + 422

The DDBs are contiguous. The end of one class of DDBs is also the start of the next class of DDBs. For example, SYSTAB + 10 (the pointer to the start of the single-line devices) actually points to the DDB for device 1, the console terminal. SYSTAB + 20 (the end of the single-line DDBs) is the pointer to the first multiplexer (DH11 and DZ11) DDB for device 64.

Figure 14-7 shows the Device Descriptor Block for the terminal-type devices.

Figure 14-7: Device Descriptor Block for devices 1-19 and 63-191

JSR R0,a(PC)+ (4037)		0
ADDRESS OF INTERRUPT HANDLER		2
DEVICE CSR ADDRESS		4
TERMINAL TYPE CODE	INTERFACE TYPE CODE	6
RING BUFFER ADDRESS		8
TIED PROGRAM NUMBER	MARGIN VALUE	10
\$Y	\$X	12
DEVICE STATUS LOW WORD (\$ZA)		14
CHARACTER IN POINTER	INPUT BUFFER SIZE	16
NO OF CHARACTERS IN BUFFER	CHARACTER OUT POINTER	18
CHARACTER IN POINTER	OUTPUT BUFFER SIZE	20
NO OF CHARACTERS IN BUFFER	CHARACTER OUT POINTER	22
JOB WAITING FOR OUTPUT	JOB WAITING FOR INPUT	24
MODEM CSR ADDRESS		26
LINE PARAMETER REGISTER		28
FIELD LENGTH	DEVICE NUMBER	30
UNUSED	AUTOBAUD SPEED CODE	32
DEVICE STATUS HIGH WORD (\$ZA)		34
\$ZB		36
LINE TERMINATOR		38
BIT MASKS		40
APPLICATION TRAP		42
BIT MASKS		44

MR-S-3494-83

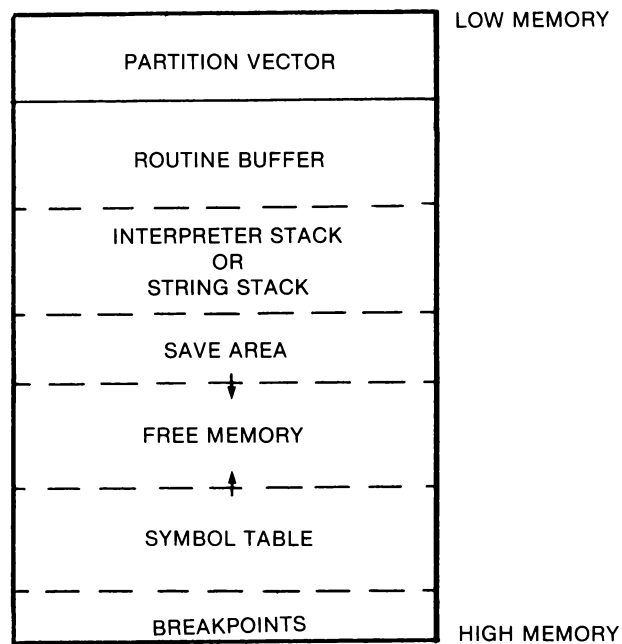
The DDBs Device Status Indicator (offset + 14) describes a number of device-specific parameters described in detail in Table 4-2.

You can examine the DDBs for all single-line and multiplexer devices by running ^DDR, which can be run directly or accessed through the SYSTEM STATUS option of the SYSTEM REPORTS option of the SYSTEM UTILITIES menu (^SYS).

14.13 Partitions

A partition is an area of memory where all elements of a job reside. The components of this area expand and contract dynamically depending on the requirements (such as adding or deleting data) of the job. Figure 14-8 shows the general layout of a partition.

Figure 14-8: Partition Layout



LEGEND:

- FIXED BOUNDARY
- - - - - DYNAMIC BOUNDARY

MR-S-3495-83

As each job enters the system, it is assigned a memory partition where it resides until the job is terminated.

14.13.1 Partition Sections

A partition is divided into five major sections:

Partition Vector	Describes the status of the job in the partition (refer to Section 14.13.2 for further detail).
Routine Buffer	Contains the lines of a DSM-11 routine. The contents of the routine buffer can be modified in Programmer Mode by adding, changing or deleting routine lines. Commands that load routines, such as ZLOAD and JOB, also change the contents of the Routine Buffer.
Interpreter Stack (String Stack)	Stores routine information temporarily while the interpreter processes the routine. The Interpreter Stack is also used to transfer messages during I/O operations. Also the Call Stack for DO and XECUTE commands is stored here.
Save Area	Stores Kernel and User stacks when the system swaps the job out. An area of free memory between the Save Area and the Symbol Table allows for the dynamic expansion of the components of a partition.
Symbol Table	Contains all defined local variables, subscripted or unsubscripted. The size of the Symbol Table varies as variables are defined (with SET) and deleted (with KILL) by the job running in the partition.
Breakpoints	When you set breakpoints for debugging MUMPS routines, up to 10 128-byte spaces are used for the breakpoints.

14.13.2 Partition Vector

As mentioned, the Partition Vector describes the status of the job in a partition. It occupies the first 332 bytes of storage in a partition's memory space.

Within the Partition Vector, there are fields located at offsets relative to \$J. These fields contain such information as:

- Contents of \$TEXT (the name of the routine currently running)
- Contents of \$IO (the current I/O device)
- Contents of \$ZE (the most recent error message)

For information on how to use the special variables \$TEXT, \$IO, and \$ZE, see the *DSM-11 Language Reference Manual*. Another field, the Job Status Word (JSW) at \$J + 2, describes various software and hardware conditions

about the job currently residing in the partition. The individual application can change some of the bits in this field. Table 14-2 shows each JSW bit assignment, and indicates which bits you can set.

Table 14-2: Job Status Word Bit Assignment

Bit	Assignment																
0	Programmer (Direct) Mode, when set*																
1	Unconditional VIEW or ZUSE privileges																
2	BREAK (application interrupt key or DSM-11 BREAK command) is enabled, when set*																
3	Application interrupt key has been received on the terminal**																
4	Timed READ overrun (time is up), used by drivers																
5	DSM-11 Version 2 compatible error processing when set***																
6	Set if this job is a DDP server																
7	Set if this job is a DDP server for a Version 2 system																
8	Specifies the job is using the global module (Data Base Handler)																
9, 10	Determines job's priority, 0-3*																
11	Set to override default spooling (used by spooler and caretaker)																
12 to 14	When bits 12 to 14 are nonzero, the job is flagged for an error on its next swap-in. The value of bits 12 to 14 determine which error is to be issued as shown below.																
	<table> <thead> <tr> <th>Value</th> <th>Error issued on next swap-in</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>DKHER</td> </tr> <tr> <td>2</td> <td>DKSER</td> </tr> <tr> <td>3</td> <td>STORE</td> </tr> <tr> <td>4</td> <td>KILLR</td> </tr> <tr> <td>5</td> <td>INRPT</td> </tr> <tr> <td>6</td> <td>ABORT</td> </tr> <tr> <td>7</td> <td>CRASH</td> </tr> </tbody> </table>	Value	Error issued on next swap-in	1	DKHER	2	DKSER	3	STORE	4	KILLR	5	INRPT	6	ABORT	7	CRASH
Value	Error issued on next swap-in																
1	DKHER																
2	DKSER																
3	STORE																
4	KILLR																
5	INRPT																
6	ABORT																
7	CRASH																
15	Set if job has used reserved disk space.																

* Application can set this bit.

** When a routine has cleared bit 2 (using the B 0 command) to inhibit the application interrupt key, the routine can check this bit and, if desired, perform a controlled interrupt. You can set this bit with the BREAK command (B 1).

*** Set by the BREAK command, (B 2), cleared by B -2.

The partition vector can be displayed by accessing the Partition Vector utility, ^PARVEC.

Chapter 15

Optimizing DSM-11

This chapter contains information on optimizing the DSM-11 system, exclusive of global and data base optimization which is discussed in Chapter 13. Optimizing DSM-11 programs and applications is discussed in Chapter 9. This chapter contains information on mapped routines, and on obtaining performance statistics on the system through the system utility, [^]RTHIST.

15.1 Mapped Routines

Normally, when a routine is used in DSM-11, it is read from disk into a buffer, and then copied byte-by-byte to the job's partition. You can specify that certain routines be placed in the mapped routine area of memory. You thus avoid both using up space in the partition and using CPU time to retrieve the routines from the disk. This is useful when those routines are going to be used frequently.

The mapped routine area is a separate area in memory that contains the specified routines. When a routine is called for, the system checks the directory of the mapped routine area using a binary search before going to the disk to retrieve the routine. Each UCI can have its own routines in the mapped routine area.

You indicate which routines are to be placed in the mapped routine area. The [^]SYSGEN routine asks whether you want the mapped routine option included or not. You indicate which routine sets you want to place in the mapped area when you define the System Start-up parameters. You define these parameters either by using the System Start-up routine, [^]STU, or by using Define Start-up

Parameters, ^STUBLD, You can also use the utility routine ^RMAP to select routines for installation in mapped memory space. The process involves building a routine set using the Routine Set Management utility, ^RMBLD, and then loading the routine set into memory using the Routine Set Loading utility, ^RMLOAD. See Sections 10.4.7, 10.7.5, and 11.1 for more information on these utilities.

The ZLOAD command and the DO command no longer retrieve the same routine when that routine is mapped. If routine ABC is a mapped routine, typing the following means that the routine ABC in the mapped routine area will be accessed:

```
>DO ^ABC
```

Typing the following means that the version of the routine that is resident on the disk will be accessed.

```
>ZLOAD ABC
```

You should be aware of the following limitations on mapped routines:

- The total space taken up by each mapped routine cannot exceed 8K bytes.
- The system accesses the mapped copy of a routine only if the job's partition is 8K bytes or smaller. Notice, however, that the entire 8K bytes is available for the symbol table when using mapped routines.

15.2 Varying Buffer Sizes

You can vary the size of three types of buffers:

- Ring buffers
- Cache buffers
- Journal buffers

You can vary the size or number of these buffers (by using the SYSGEN procedure) in different configurations in an attempt to find the best balance between size of the buffers and overall efficiency. In general, allocating more space to these buffers increases efficiency, but takes up more memory space. The ring and journal buffers are much smaller than the cache buffer; the greatest benefit arises from varying the number of cache buffers.

You should allocate as much space as possible for cache buffers, since disk operations where data is shared for global access are especially optimized by large disk caches. Memory space considerations must be weighed against the number of cache blocks. Room for three partitions plus space for journal overhead (if you select journaling) must be available. Also, system constraints

require that total buffer allocation space does not exceed the value shown during SYSGEN for the maximum space that can be used for buffers.

You may also have additional constraints based on needing more than three partitions, or other memory requirements.

The journal buffers are referred to in Part 7 of SYSGEN and are also discussed in Section 10.4.7. The ring and cache buffers are referred to in Part 8 of SYSGEN and are discussed in Section 10.4.8.

15.3 Limiting Critical Data Areas to Minimize Searching

You can increase efficiency by limiting the size of certain data areas that are frequently searched:

- Global Directory Entries
- UCI Translation Table

15.3.1 Limiting The Size Of The Global Directory

The global directory must be searched each time that a global is referenced. This directory is contained on the disk in one or more blocks. Keeping the size of a directory small decreases the amount of search time.

The directory is arranged according to when each global is first defined. You can establish a global (such as ^A) that is used frequently as the first global in the global directory. Do this by making the very first reference to any global for that data base to ^A. This can be done by a "dummy" set, for example:

```
SET A=""
```

Use short names (one or two letters) for globals to help prevent the global directory from growing beyond one block. If the global directory grows beyond one block, the search time is significantly greater than for one block (if the same total number of globals is involved). See Section 13.4.3 for more information on the global directory block.

15.3.2 Limiting The Size Of The UCI Translation Table

The UCI Translation Table provides a means for providing access to globals across UCIs in a way that is transparent to the user. See Section 10.7.8 for more information on this table.

You should use this table sparingly and keep the number of entries (globals) to a low number; because, for every global reference, this table is first searched to ensure that the reference does not involve a global in the table.

15.4 Performance Statistics

DSM-11 provides a utility (^RTHIST) that can be used to gather various performance statistics for the system as a whole. This utility can also obtain breakdowns for system global and routine calls by individual UCIs. You can then use these statistics for performance evaluation of the system.

15.4.1 Using The Performance Statistics Utility (^RTHIST)

The performance statistics on the DSM-11 system can be obtained by using the Performance Statistics utility, ^RTHIST. This utility can be accessed directly by typing:

```
>DD ^RTHIST
```

It can also be accessed through the SYSTEM UTILITIES menu (^SYS).

The utility allows you to log performance data for a certain period of time (a session), print out a report of data that has been previously logged, or show the on-line help text for the utility. If you select one of the first two options, you are asked additional questions.

15.4.2 Performance Data

Data is gathered on many statistics which are then output as reports or histograms. The types of data are:

- Counts of jobs in queues
- Data base event counts
- Derivative ratios

- Disk usage
- Routine usage
- Global reference counts

15.4.2.1 Queue Data — Jobs occupy various queues in the system when they are not actually running. Abbreviations, such as SHORTQ, are used in the histograms produced by [^]RTHIST.

A job is either waiting for an external event or else ready to run. Jobs that are ready to run are placed in one of several queues, depending upon their priority. Jobs may also be waiting for data base resources to become available. Data provided in each of these categories are statistical averages of job counts in various queues.

SHORTQ	Short Run-Time Queue - is for jobs that are waiting for disk service or access privileges. These jobs have the highest priority for execution, but only for a short time slice.
IORQ	Input/Output Ready Queue - is for jobs that have just finished waiting for I/O, especially terminal or printer I/O.
WAIT1Q	Time Slice Expired Queue - is for jobs when their time slice has expired while executing code. There are 4 queues (WAIT1Q -WAIT4Q) for 4 priorities of jobs.
GLOBQ	Waiting for Global Queue - is for jobs waiting for the global handler because another job has locked the handler for its private use.
GLOLKQ	Global Lock Queue - is for jobs waiting for private use of the global handler. Private use is required whenever a SET adds a block to the data base, or a KILL subtracts a block from the data base.
%GLOCK	Time that Global Handler Is Locked - is the percentage of time that the global handler is locked by some job for its private use.
%GLOCKWAIT	Time in Global Lock Queue - is the percentage of time that the Global Lock Queue has at least one job in it.
DKRBQ	Disk Wait Queue - is for jobs waiting to use the disk handler.
JRNQ	Journal Wait Queue - is for jobs waiting for journal buffers to be written to the journaling tape or disk.

GLOBAL In Global - is not actually a queue, but an average count of the jobs concurrently using the global handler to access the data base.

15.4.2.2 Data Base Event Counts — The data base access handlers automatically accumulate counts of various events. The data represent averages of events per second.

ROUREF Routine References - is the combined count of all DO, GOTO, ZLOAD, and ZSAVE commands that include routine names.

MAPROU Mapped Routine Access - is the count of accesses to routines mapped into memory.

GLOREF Global References - is the combined count of all references to global variables, including SETs and KILLs.

GLOSET Global SETs - is the combined count of both SET and KILL commands to global variables.

GLOKIL Global KILLs - is the count of KILL commands to global variables.

LOGRD Logical Block Reads - is the count of all requests to search the disk cache for a block.

READS Disk Block Reads - is the count of disk block reads to the disk cache only.

TOTRD Total Disk Reads - is the count of all disk reads, including SDP, VIEW, and DMC-11 Block Mode.

LOGWT Logical Block Writes - is the count of requests by the global handler to write disk cache buffers.

WRITES Total Disk Writes - is the count of all transfers from memory to disk.

WTSYNC Synchronous Disk Writes - is the count of disk writes for SDP, VIEW, and DMC-11 Block Mode (but not from the disk cache).

TRYLAST Try Last Block - is the count of attempts to find needed data in the last referenced block for a particular job, thus bypassing searches at the directory and pointer levels.

GOTLAST Got Last Block - is the count of successful attempts to find needed data in the last referenced block.

ALLOC	New Blocks Allocated - is the number of new blocks allocated for global use in 1024-byte blocks per second.
DEALL	Blocks Deallocated - is the number of blocks deallocated from global use in 1024-byte blocks per second. By comparing ALLOC with DEALL you can determine whether your data base is growing or shrinking.
TTYOUT	Character Output - is the count of characters output to terminals and line printers.
TTYIN	Character Input - is the count of characters input from terminal lines.
DDPIN1	DDP Requests Received - is the number of requests to use a DDP line from remote systems.
DDPOUT1	DDP Requests Sent - is the number of requests to use a DDP line sent to other systems. Activity on additional DDP lines are indicated by the counts in DDPIN2, DDPOUT2, DDPIN3, DDPOUT3, and so on.

15.4.2.3 Derivative Ratios — These ratios show the effectiveness of the optimization algorithms of both the disk cache and the global handler. These ratios are derived from the data base event count statistics.

- Block Requests Per Global Reference - shows the effectiveness of the global handler. In general, global and routine references require more than one block request to get at data. The global handler employs optimization algorithms to minimize the number of these block requests. When successive references (within a job) are to the same global, the second and following references may require fewer block requests. If global references within a job are grouped by global name, the ratio declines to approach the value of 1.
- Actual Block Reads Per Read Request - shows the effectiveness of the disk cache in minimizing disk reads.
- Block Write Requests Per SET command - must be at least 1. This ratio can be expected to increase as the rate of growth of the data base increases. When a global block overflows, a split occurs, causing at least four block writes to: the two resulting blocks, the ancestor pointer block, and the "map" block.
- Actual Block Writes Per Write Request - shows the effectiveness of the disk cache and the Write Demon in minimizing disk writing.

15.4.2.4 Disk Usage Histogram — This data shows the percentage of elapsed time used by each disk drive. The data is further broken down into disk reads and disk writes. Currently, DSM-11 allows only a single disk transfer to be performed at any point in time, regardless of the number of physically

connected disk drives. As the total percentage of elapsed time used by all drives increases, a system can be said to be more "disk bound."

15.4.2.5 Routine Name Histogram — This data shows how much CPU time is spent (on the average) by various routines and, conversely, how much of the CPU time is idle. For DSM-11, you begin to perceive the system response time slowing down when CPU idle time declines to 25% or less.

15.4.2.6 Global Name Histogram — This data is an analysis by UCI and global name of the combined count of all references to global variables. This is actually a global-by-global breakdown of the count recorded above as GLOREF.

15.4.3 Optimization Based On Performance Statistics

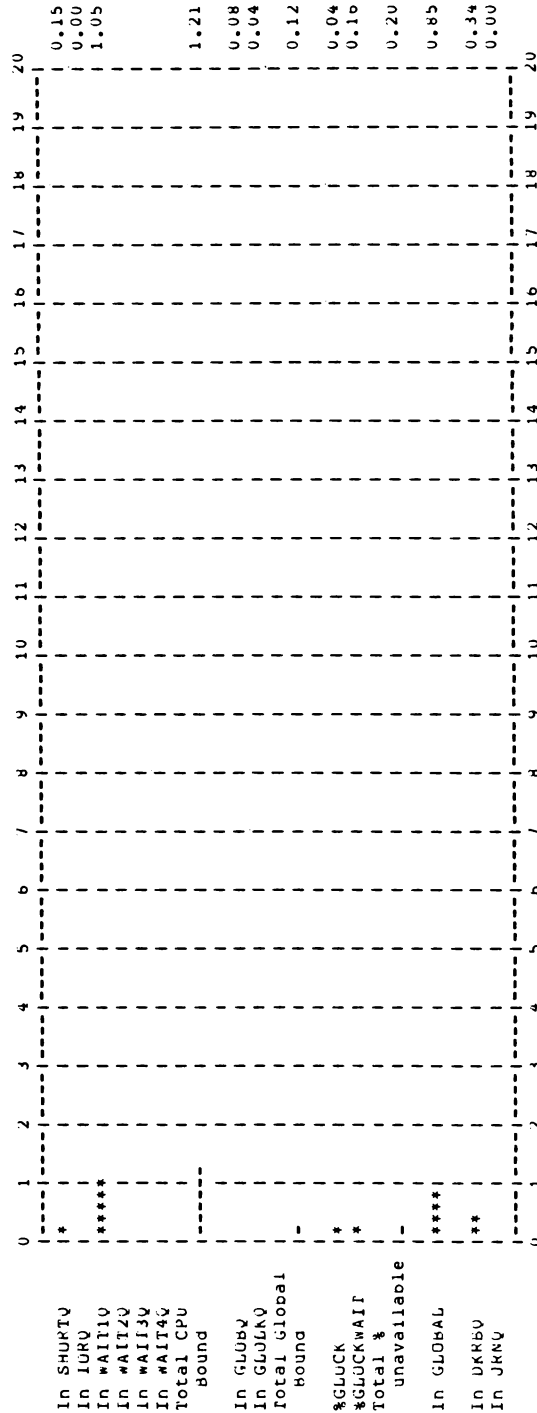
An advantage of the [^]RTHIST utility is that it gathers statistics while an application is running in a normal day-to-day environment, rather than in a test situation. Thus, the statistics can be used to accurately gauge where bottlenecks or performance problems are occurring in the system.

When a system reaches a saturation point, the first question that is usually asked is whether the system is disk bound or CPU bound. The answer can be inferred by examining the [^]RTHIST results.

If the system is CPU bound, you may wish to optimize the DSM-11 coding to improve performance. You can also refer to the [^]RTHIST statistics to see which routines are consuming the most CPU time, and thus determine where your effort will gain the best results.

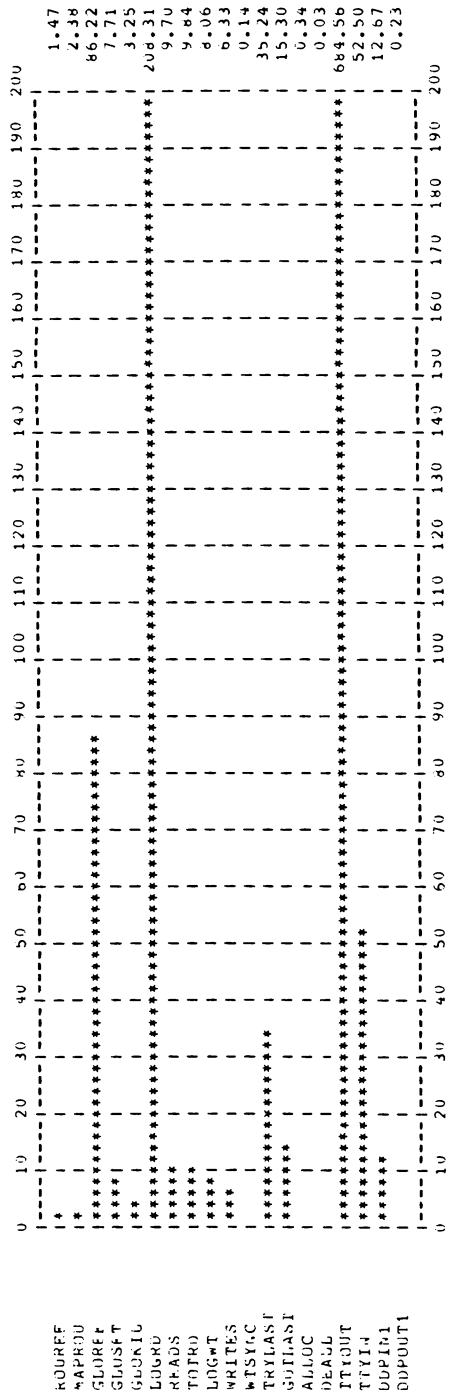
If the system is disk bound, you can use these statistics to show how much load is on each disk drive, how the load is split between reading and writing, and how often each global is being referenced.

Figure 15-1: RTHIST: Average Counts of Jobs



MR-S-3496-83

Figure 15-2: RTHIST: Averages of Data Base Events Per Second



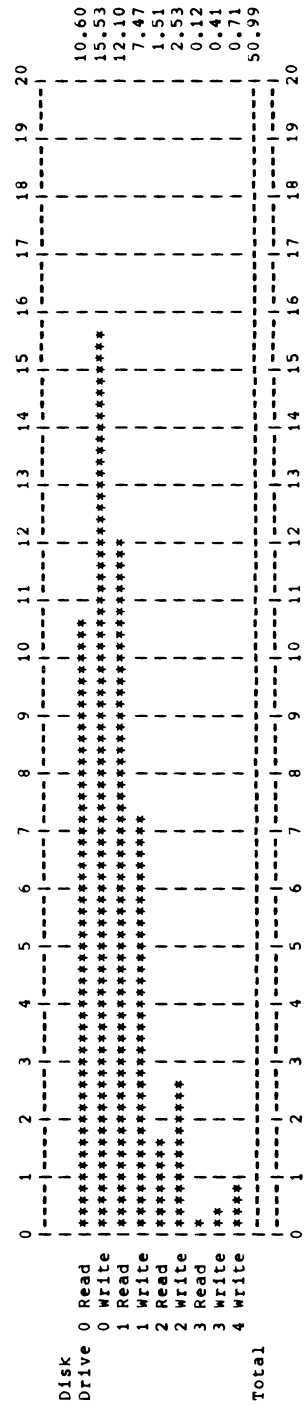
MR-S-3497-83

Figure 15-3: RTHIST: Derivative Ratios

```
Derivative ratios:
| Block Read Requests per Global Reference (LOGRD/(ROUREF+GLOREF)): 2.38
| Actual Block Reads per Block Read Request (READS/LOGRD): 0.05
| Block Write Requests per SET/KILL Command (LOGWT/(GLOSET+GLOKIL)): 1.05
| Actual Block Writes per Block Write Request ((WRITES-WTSYNC)/LOGWT): 0.77
```

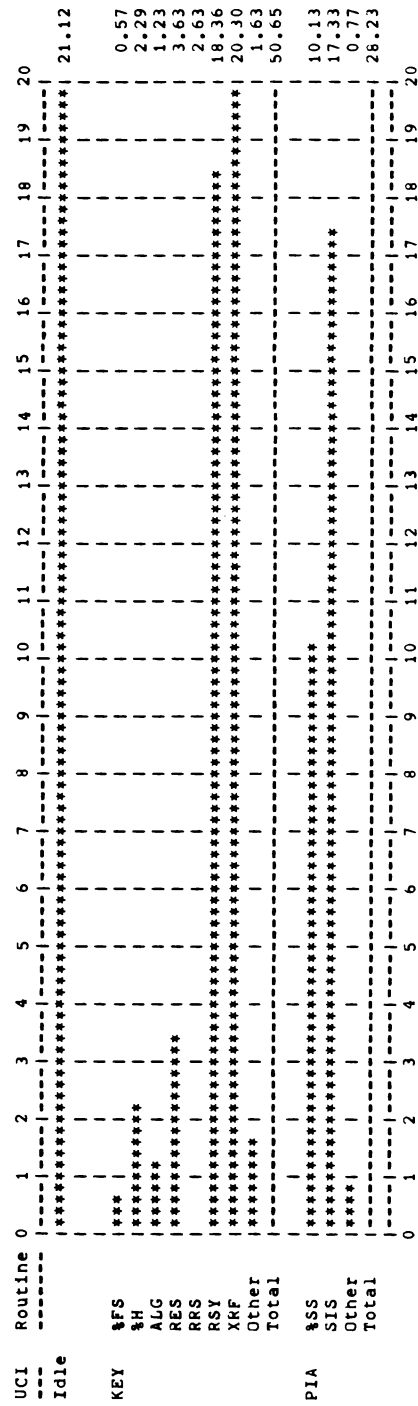
MR-S-3498-83

Figure 15-4: RTHIST: Disk Access as Percentage of Total Time



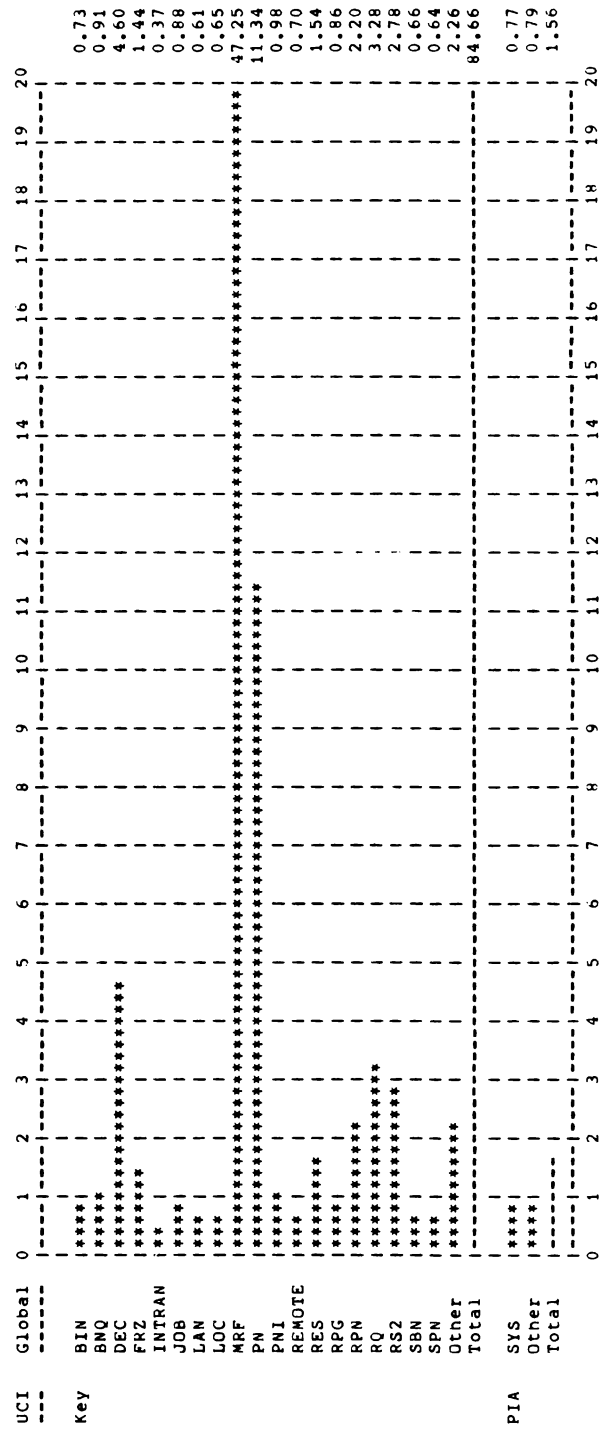
MR-S-3499-83

Figure 15-5: RTHIST: Routine Execution as Percentage of Total Time



MR-S-3612-84

Figure 15-6: RTHIST: Global References Per Second



MR-S-3613-84

Chapter 16

The DSM-11 Journal Procedure

This chapter describes the journal procedure for DSM-11 as well as the individual utilities required to perform the journal procedure.

16.1 Introduction

The DSM-11 journal procedure is a means of keeping a record on a secondary storage device (disk or magnetic tape) of the SET and KILL operations that users perform on the global data base. If the system malfunctions, the journal procedure provides additional backup so you do not have to repeat all SET and KILL operations after you have restored the data base.

As System Manager, you must specify which globals are to be journaled. After you mark globals for journaling, the following information is recorded with each SET and KILL of those globals:

- Date and time of the transaction as recorded in \$HOROLOG (5 bytes binary)
- The UCI and volume set number of the global that is affected by the SET or KILL operation (1 byte binary)
- The job number of the user who issued the SET or KILL (1 byte binary)
- A letter to identify the transaction: S for SET, K for KILL (1 ASCII byte)

- The full global reference name and its subscripts, if any, separated by commas (a variable-length ASCII string)
- The data value of the global if a SET (a variable-length ASCII string)

During the journal procedure, DSM-11 records operations in the order in which they are performed. Therefore, when you run the dejournal utility, DSM-11 restores each operation in its original sequence.

You can enable or disable the journal capability on a systemwide basis. In addition, each application can select specific globals that should be journaled. Section 16.2 describes the DSM-11 utilities you must run to identify the globals to be journaled.

The journal utilities perform such functions as starting and stopping the journal procedure, allocating and deallocating journal space, initializing the journal space, and restoring journaled information from disk or magnetic tape.

16.2 Selection of Globals

To select globals for the journal procedure, you must run the Global Management utility (^%GLOMAN). Among several options, these utilities allow you to modify the journal capability of any global in the system. Selectively journal only data that is necessary to recover the system. It is inefficient to journal all computer transactions. Backing up scratch globals or other noncrucial data may serve no real purpose. Moreover, it uses system resources wastefully. You may find that the total amount of journal data affects your hardware configuration and your system maintenance strategy.

16.3 The Journal Utilities

You can run each journal utility individually. Except for ^JRNRECOV, you can also access each journal utility through the SYSTEM UTILITIES (^SYS) menu.

The journal utilities are:

Utility	Routine Name
Start Journal	^JRNSTART
Stop Journal	^JRNSTOP
Show Journal Spaces	^JRNLSHOW
Allocate New Journal Space	^JRNALL

Initialize Journal Space	^JRNINIT
Deallocate Journal Space	^JRNDEALL
Journal Recover	^JRNRECOV
Dejournal	^DEJRNL

A useful nonjournal utility is the fast option of Disk Block Tally, ^DBT, which shows free blocks on a disk. This information can be used in allocating journal space. The Allocate New Journal Space utility, ^JRNALL, also provides the same information about block allocations.

The following is a brief description of each journal utility.

16.3.1 Start Journal (^JRNSTART)

Invokes the journal procedure if it is not already running as a result of a system start-up. Requests the disk space or magnetic-tape unit number you want to use for the journal procedure. Displays a message informing you that the journal procedure has begun.

16.3.2 Stop Journal (^JRNSTOP)

Allows the current journal operation to finish, then terminates the journal procedure.

16.3.3 Show Journal Spaces (^JRNLSHOW)

Lists all currently allocated journal spaces. Displays the starting, ending, and next available block numbers for all journal space. You must know this information when you run the start Journal utility.

16.3.4 Allocate New Journal Space (^JRNALL)

Allocates new journal space on disk. Verifies that all maps within that area are not in use for any other purpose. Calls the Initialize Journal Space utility to initialize the blocks in that area to all zeros. Updates the ^SYS global to indicate that the new journal space exists. Note that the space you designate as new journal space cannot overlap existing journal space.

16.3.5 Initialize Journal Space (^JRNINIT)

Initializes the specified journal space by setting the blocks to zeros. This updates the ^SYS global to indicate that the space does not contain any journaled data.

16.3.6 Deallocate Journal Space(^JRNDEALL)

Clears the space that you specify. This space must be space that you previously specified for the journal procedure. Deletes from the SYS global all entries for this space and changes the related map blocks to indicate that they are not journal maps.

16.3.7 Journal Recover (^JRNRECOV)

Closes journaling on one space and continues journaling on another space. No data is lost, but the job can hang until the alternate space is available. Provides several options when an error or an out-of-space condition occurs on disk or magnetic tape.

- Switch to another magnetic-tape reel
- Switch to alternate journal space on the same disk
- Allocate new journal space
- Mount a different disk and allocate space on it if space is not already there
- Stop journal

If there is no more space, the caretaker utility is responsible for detecting the condition. The caretaker utility then notifies the operator to invoke ^JRNRECOV. See Section 12.2 for more information on the caretaker utility.

16.3.8 Dejournal (^DEJRNL)

Restores to disk all SET and KILL information that you stored on disk or magnetic tape. Provides the option to restore the information on all or selected journaled globals.

16.4 Journaling Globals

To enable the journal procedure, you must include it during SYSGEN and specify it as part of your startup parameter specifications. Next, you must enable the journal capability for each existing global that you want to journal. To do this, run the DSM-11 Global Management utility (^%GLOMAN). These utilities asks you to name each global, one at a time, that you want to journal. When you have listed all selected globals, you can exit from the utility by typing a carriage return.

At any time after you have enabled journaling for specific globals, you can run the journal utilities in one of two ways:

1. Run the SYSTEM UTILITIES menu (^SYS) to select a journal utility from the menu.
2. Run any of the journaling utilities individually.

While journaling is running, you cannot do the following:

- Dismount the disk or magnetic tape on which journaling is running
- Initialize or remove the currently used journal space

16.4.1 Journaling To Disk

When you request journaling space on disk, DSM-11 allocates the space in increments of 400-block map areas. You can allocate journal spaces on any of the disks mounted in the running configuration. DSM-11 keeps track of each area designated for journaling on any currently mounted disk: whether the area is empty, full, or partially full. Normally you allocate contiguous map areas on the same disk as the primary journaling space. You can also designate

other map areas as alternate journal space in case the initial space becomes full.

16.4.2 Journaling To Magnetic Tape

When you specify journaling to magnetic tape, the magnetic tape drive that you specify is reserved for the journal procedure exclusively. Journal information is stored in sequential blocks on a single unlabeled magnetic tape file. Each block is 1024 bytes long. The end of each section of journal information is indicated by two magnetic tape marks.

NOTE

You cannot perform journaling to magnetic tape and disk simultaneously.

16.5 The Dejournal Procedure

The dejournal procedure is a means of restoring to disk the SET and KILL information from either disk or magnetic tape. The example dejournal utility, `^DEJRNL`, which is included with your DSM-11 system, is designed to recover as much of the data base as possible following an event that causes corruption of the data base. You should devote a single-user system configuration to the dejournal procedure. Also, you can create your own utility that more specifically suits your needs.

The example utility allows you to specify:

- The media from which you want to dejournal.
- The individual variables (or ALL) you want to dejournal. You can specify the variables by UCI, variable name, or a wildcard.
- The UCI of the job that journaled the globals.

Chapter 17

Volume Sets and Clustered Systems

This chapter discusses volume sets within a single DSM-11 system and within a cluster of DSM-11 systems. A cluster is a set of DSM-11 systems connected by communications. Communication across systems is referred to as Distributed Data Processing (DDP). DDP allows a job running on a system within the cluster to access and lock globals residing on a volume set mounted on another system. The two systems must have a direct connection.

This access can be through two different approaches:

- Using an explicit syntax, the extended global reference.
- Using the UCI Translation Table to provide transparent communication across systems.

Volume sets provide a way of organizing data on disks. Having separate volume sets is a way of having independent data bases on separate disks. By referring to the volume set, you can access data in the volume set, regardless of where the volume set resides.

This chapter discusses how to set up a clustered system in general, and how to access volume sets across systems. The utilities and mechanisms involved include:

- SYSGEN - establishes DDP (Distributed Data Base Processing) arrangements during system generation procedure.
- ^DDPUTL - modifies DDP arrangements by using the ^DDPUTL system utility.

- Explicit syntax - allows access to data across volume sets and across systems.
- [^]UCITRAN - sets up user-transparent access to data in another volume set by using the [^]UCITRAN system utility to modify the UCI Translation Table.
- Journaling and backup - allows journaling and backup of a data base in a clustered system.
- [^]%GLOMAN - controls access to globals across systems by means of protection codes assigned to globals by [^]%GLOMAN.
- Dual porting - allows the same disk drive to be switched between two systems easily.
- Library UCI - changes the library UCI by using the [^]UCIEDIT utility.
- \$ZR - checks on whether a global access across systems actually connects to the other system by using the \$ZREFERENCE special variable.

17.1 Volume Sets

Volume sets are a basic way of organizing distinct data bases on one PDP-11 system, and among clustered PDP-11 systems. This section summarizes information on volume sets. Additional information is provided in:

- Section 10.4.7 - providing support for volume sets during SYSGEN.
- Section 11.1 - mounting volume sets during system start-up.
- Section 12.3.6 - mounting volume sets while the system is running using [^]MOUNT.
- Section 12.6.4.8 - printing out the Volume Set Table using [^]%STRTAB.
- Section 13.2 and 13.3 - volumes and volume sets.
- Section 14.7 - structure of the internal Volume Set Table.

17.1.1 Analogy To A Set Of Books

A volume set can be compared to a set of books. The set of books forms a whole representing one author's works: for example, the books of Mark Twain. However, you can easily have several sets of books sitting in your library: such as the books of Thomas Jefferson, the books of Charles Dickens, and the books of Edgar Allen Poe. Each set of books is logically separate from the other books, but can easily be included in one library. The books within one set of books are logically related since they were written by the same author.

By analogy, data on a computer can be contained on disks; each disk can be thought of as one volume (compared to a book in a set). A set of disks with related data make up a volume set (compared to a set of books). In actuality a volume set can be included in totality on one disk (in the way that the works of a poet can be included all in one book), or on several disks.

17.1.2 DSM-11 Conventions For Volume Sets

DSM-11 has several conventions for how data in volume sets are treated. The two general classes of volume sets are:

- System - contains a bootable system image with a manager's UCI and all system utilities. If it is booted as the first volume set, it is the system disk for that CPU and cannot be dismounted.
- Nonsystem - contains no system image, and can only be mounted by a running system.

Both classes can contain data.

New data growth proceeds sequentially within a volume set, that is, a global can grow from one disk to another disk, if both disks are in the same volume set.

There can be a maximum of four volume sets, and there must always be at least one volume set. Each volume set represents a separate class of 30 UCIs. All UCIs in a volume set can access globals within that volume set. Thus, any UCI in the volume set A can access the global, ^XXX. But a global, ^YYY, that exists in another volume set, volume set B, cannot normally be accessed from a UCI in volume set A (unless an explicit syntax or the UCI Translation Table is used).

In versions of DSM-11 previous to Version 3, the single DSM-11 system can be thought of as a system that has only one volume set. Additional disk drives can be added to the system and it can be upgraded to Version 3. Then the data base could be organized into different data base groupings (volume sets) established on different disks.

17.1.3 Practical Uses Of Volume Sets

Within a single DSM-11 system, volume sets can provide a practical way to logically divide up a data base and restrict user access to different parts of the data base.

In a multiple CPU environment (not a clustered system) each CPU can have a designated backup CPU which can take over its data base in the event of a CPU failure. In the event of a failure, a volume set can be moved from the failed CPU to the backup CPU.

In a clustered system, you can also move volume sets around if one processor is down. Thus the cluster can function without one CPU, and the same data base can be accessed.

You can move volume sets from one system in a cluster to another without rebooting the running system (if planning is done properly for this situation):

- The backup system must be running a configuration that provides support for multiple volume sets.
- There must be disk drives available on the system.
- There must be space in the Volume Set Table to mount another volume. The maximum number of mounted volume sets is four per system.
- You may have to change references in the UCI Translation Tables (using `^UCITRAN`).
- You may have to change DDP line labels (using `^DDPUTL`).
- You may have to change library UCI references (using `^UCIEDIT`).

You can use the `^MOUNT` system utility to mount a nonsystem volume set while the system is running.

Volume sets can be useful in a software development situation. Each application development team can have its own volume set that can be moved from one system to another. Thus each separate application can be developed and debugged without conflicting with other applications.

17.1.4 Logging In A Volume Set

When you have more than one volume set on a system, you can log into a specific volume set on that system. By default you log in the system volume set (Volume Set 0), if you do not specify a volume set.

To indicate a specific volume set log in as shown in this example:

```
DSM-11 Version 3 Device #64 UCI: JHM,WWW:XXX
```

In this case, JHM indicates a UCI in volume set WWW. The PAC password is XXX.

17.2 Clustered Systems

A DSM-11 clustered system is created when individual DSM-11 systems are connected by DDP. Typically, a master data base would reside on a volume set on disks connected to a central PDP-11 CPU. Users logged in on other CPUs would access the master data base without logging into the central CPU. All journaling and backup should be handled from the central CPU.

Any actual planning for a clustered system must take into account any special requirements on your particular situation, and should involve a Digital representative or support person.

This section discusses how to set up clustered systems with movable volume sets.

17.2.1 System Generation For Clustered Systems (^SYSGEN)

This example shows the information needed by SYSGEN for a configuration that supports several volume sets and clustered systems. This SYSGEN is for a clustered system with DDP provided by by DMC11 or DMR11 communication devices. Cables must connect the DMC11s or DMR11s, which are in turn attached to the PDP-11 systems.

You can run either a manual SYSGEN or an autoconfiguration. If you run an autoconfiguration, most questions will be answered automatically with the default answers. You are asked the questions in Part 4 on configuring DMC11s or DMR11s, even if you run an autoconfiguration.

If you run a manual SYSGEN, you need to know the CSR and Vector addresses for devices connected to your system. You can obtain these addresses by running the ^CONFIG utility, (see Section 10.6 for more information).

The example manual SYSGEN below skips over irrelevant parts of SYSGEN, and focuses on the questions and parts of SYSGEN relevant to volume sets and clustered systems. Chapter 10 provides a full discussion of the SYSGEN procedure.

Answers to each SYSGEN question must be followed with a carriage return. In general the carriage returns are not indicated in this example.

On any question asked during SYSGEN, you can enter:

^ to return to the previous question
? for additional help

If a value appears between angle brackets, < >, it is a default value. Typing a **RET** causes the default value to be used as the answer to the question.

Example:

```
> DO ^SYSGEN
```

```
System generation for DIGITAL Standard MUMPS
```

```
Type ? for HELP at any time
```

```
PART 1: SYSGEN
```

```
PART 3: SYSTEM DEVICES
```

```
3.5 How many DMC11s are there (max = 4) ? <0> 2
```

Type the number of DMC11 or DMR11 synchronous controllers in this configuration. Be sure to enter all DMC11s or DMR11s regardless of their use in the system. Note that when SYSGEN refers to DMC11s it is referring to both DMC11s and DMR11s.

```
Enter the VECTOR address, in OCTAL, for DMC11 controller 1 > 340
```

```
Enter the CSR address, in OCTAL, for DMC11 controller 1 > 160110
```

```
Enter the VECTOR address, in OCTAL, for DMC11 controller 2 > 360
```

```
Enter the CSR address, in OCTAL, for DMC11 controller 2 > 160120
```

Do not use the Vector and CSR addresses shown here when running SYSGEN. You must use the addresses that are appropriate for your system.

PART 4: CONFIGURE DMC11s

4.1 Is DMC controller 1 HALF DUPLEX [Y OR N] ? <N> N

4.1 Is DMC controller 2 HALF DUPLEX [Y OR N] ? <N> N

Indicate whether the DMC11 or DMR11 is to be half or full duplex. Typically, for DMC11s or DMR11s used in a DDP configuration (clustered system), the DMC11 or DMR11 is used in full duplex mode.

4.3 How many of your DMC11s do you wish to use for DDP [Y OR N] ? >2

At this point you indicate the number of DMC11s or DMR11s you want to use for DDP. If you have more than one DMC11 or DMR11, questions 4.4 through 4.6 are repeated for each DMC11 or DMR11.

4.4 Which DMC11 controller do you wish to use for DDP line 1 > 1

4.5 Enter the 3-letter codes for DDP line 1 > AAA, BBB, CCC, DDD

The code must be unique for each line since the code is used by DSM-11 routines to specify that line for extended global references. The codes can also be used in the UCI Translation Table. The code must contain three uppercase alphabetic characters. These codes should refer to volume sets that reside on another PDP-11 system connected by DDP lines. There can be as many as four codes, referring to the four volume sets that can be mounted on a remote system.

Labeling the DDP line with the volume set codes provides for automatic and smooth access to the volume sets in a remote system. By referring to this code from the other system (connected by DDP) you automatically connect to a DDP line, and then connect into the designated volume set.

4.6 Will DDP line 1 be connected to a Version 2 system [Y or N]? <N>

If the line is to be connected to a Version 2 system, you should answer this question Y to insure compatibility between the Version 3 and Version 2 systems.

4.4 Which DMC11 controller do you wish to use for DDP line 2 > 1

4.5 Enter the 3-letter codes for DDP line 2 > EEE, FFF, GGG, HHH

4.6 Will DDP line 2 be connected to a Version 2 system [Y or N] ? <N>

PART 7: SOFTWARE OPTIONS

7.8 Include support for UCI TRANSLATION TABLES [Y OR N] ? <Y> Y

At this point you must indicate that you want support for UCI Translation Tables if you wish to use these tables to provide transparent access to globals residing in a volume set on another system (through) DDP.

7.9 Include support for MOUNTABLE DATA BASE VOLUME SETS [Y OR N]? <Y> Y

Here you must indicate support for more than one volume set on this system that you are now configuring. In Part 9 of SYSGEN you are asked for the exact number of volume sets.

PART 9: SYSTEM DATA STRUCTURES

9.2 Enter the number of ADDITIONAL mountable DATA BASE VOLUME SETS <3> 3

Your system is allowed up to four concurrently mounted data base volume sets. Volume Set 0, of which the system disk is the first volume, remains mounted at all times. You can specify that space be reserved for one, two, or three additional UCI tables (one for each volume set).

Enter comment (max. 200 chars.)) EXAMPLE SYSGEN

The system global ^SYS has been built by SYSGEN.
^SYS is a reserved global and should not be altered.

17.2.2 The DDP Utility (^DDPUTL)

You can use this utility to modify the DDP arrangements on a running configuration.

EXAMPLE:

```
>DO ^DDPUTL
```

```
Distributed Data Processing Control and Status Utility
```

```
[S]tatus [E]nable [D]isable [V]erify [C]hange
```

```
Option ( type ? for help ) > S
```

DDP Status

System Codes	Line State	Job	Activity	Errors
AAA,WWW	Up	3	0	0
SYB,BBB	Up	4	0	0

The status option shows the current status of the DDP system. System codes in this case refer to the codes given to the DDP lines, which should also represent volume set codes. In this case AAA and WWW represent volume sets on one system, and SYB and BBB represent volume sets on another system.

```
[S]tatus [E]nable [D]isable [V]erify [C]hange
```

```
Option ( type ? for help ) > D
```

```
System Name > AAA
```

The disable option is used to disable access across one of the DDP lines. Access to volume set AAA is disabled, but this also affects access to WWW since both AAA and WWW are both on the same DDP line. Running the status option would indicate that the DDP line indicated by code AAA and WWW is down.

```
[S]tatus [E]nable [D]isable [V]erify [C]hange
```

```
Option ( type ? for help ) > E
```


System Name > AAA

DDP line AAA is now running through job number 3.

System AAA is up and running configuration SYS-2A.

Access to volume set AAA is reestablished. The utility reports what configuration is running on the CPU that is attached to volume set AAA.

[S]tatus [E]nable [D]isable [V]erify [C]hange

Option (type ? for help) > V

System Name > SYB

System SYB is up and running configuration SYS-2B

You can use this option to verify that a remote system is up and running and is connected by DDP.

System Name > BBB

System BBB is up and running configuration VOLUME SET BBB.

In this case the utility verifies the connection to volume set BBB (connected to system SYS-2B). If you try to verify a connection to a volume set that has no manager's account, you can get an error.

[S]tatus [E]nable [D]isable [V]erify [C]hange

Option (type ? for help) > C

System Name > AAA

You can use this option to change the names of the volume set being accessed across DDP lines. The volume set code must match the name of a volume actually mounted on the remote system. This is the kind of change you may desire to make in case of a failure in a clustered system. The specific change must conform to the requirements of your cluster of systems.

Enter the 3 letter codes for DDP line 1 (AAA,WWW) ? CCC,XXX

DDP line 1 SYSTEM CODES modified

17.2.3 Explicit Syntax

Once you have established a running configuration using `^SYSGEN` or `^DDPUTL`, and mounted the volume sets; you can use an explicit syntax to access a volume set on a remote system connected to your system by DDP lines.

This syntax is the extended global reference and is also described in Section 8.2.3. The syntax is summarized here as follows:

`^["UCI","SYS"]GLOBAL REFERENCE`

UCI refers to a UCI in the remote volume set. *SYS* refers to the volume set on the remote system. If the remote system has only one volume set then *SYS* is, in effect, the name of the remote system. The *UCI* must exist in the remote volume set, the volume set must be properly mounted on the remote system, and a DDP line must be labeled with the same volume set name as in *SYS*. You will get a `<NOUCI>` or `<NOSYS>` error if the UCI, system, or volume set does not exist.

GLOBAL REFERENCE refers to a standard MUMPS global reference, for example:

```
^S Ψ=^["SCT","BBB"]X(1,2)
```

DSM-11 uses a search algorithm to locate the volume set to which you are referring. First, DSM-11 searches through the list of volume sets that are currently mounted on the local system. When the name, such as BBB, is not found, DSM-11 automatically searches through the labels attached to any DDP lines. When BBB is found as one of the names of a DDP line, DSM-11 seeks to access the global in the designated UCI in the BBB volume set on the remote system.

17.2.4 Using The UCI Translation Table (^UCITRAN)

The UCI Translation Table provides a way of accessing a global in a volume set on a remote system. This access is transparent to the user or the application involved.

Once the global (such as `^X`) is entered in the UCI Translation Table, when you access `^X` in your UCI, you automatically access the global, `^X`, in the UCI and volume set indicated in the UCI Translation Table. If the volume set is on another system, and DDP lines have been properly set up, then the access is to the other system.

You can use the UCI Translation Table if your application routines have already been written without using extended global references (see Section 8.2). Rather than changing your application routines, you can create an entry in the UCI Translation Table. This can be particularly useful if you must move a volume set to another system. By changing the references in the UCI Translation Table you can access the same volume set even though it has been moved.

The UCI Translation Table is maintained as part of your system configuration global (^SYS) and is loaded into memory when you start up your system. You must also set aside memory space for this table during system generation.

Using the UCI Translation Table can affect performance on the system where the table is set up. The UCI Translation Table must be searched each time any global reference (including LOCK and ZA) is made. When used in the context of DDP, this disadvantage must be weighed against the advantages of the clustered system. See Section 15.3.

EXAMPLE:

```
> DO ^UCITRAN
```

```
UCI Translation:
```

1. DISABLE UCI TRANSLATION (DISAB^TRANTAB)
2. EDIT TRANSLATION TABLES (EDIT^TRANTAB)
3. ENABLE UCI TRANSLATION (ENAB^TRANTAB)
4. SHOW UCI TRANSLATION TABLE (SHOW^TRANTAB)

```
Type of operation> 1. DISABLE UCI TRANSLATION
```

If you choose this option, all references in the UCI Translation Table are disabled. You can choose this, if you wish to access globals in a local volume set that have the same name as globals in a remote volume set.

```
Type of operation > 3. ENABLE UCI TRANSLATION
```

This option reenables the UCI Translation Table.

```
Type of operation > 2. EDIT TRANSLATION TABLES
```

```
Enter the name of the configuration you wish to alter <TEST10>
```

```
Configuration "TEST10" is running now. If you continue,  
both memory and disk will be modified.
```

```
Are you sure you want to proceed? <NO> Y
```

```
Terminate by responding with carriage return to table entry # question:
```

```
Table entry # ? > 1
```

UCI name ? > JSS

System/Volume Set name ? > FFF

Global name ? > X

New UCI name ? > YYY

New System/Volume Set name ? > HHH

These entries allow you to access the global, ^X, in the UCI, YYY, in volume set HHH, from the UCI, JSS, in volume set FFF. Both UCIs (and any volume sets you refer to) must be previously defined. If the volume set HHH indicates a DDP line (labeled HHH), that connects to a system with a volume set HHH mounted on it, then the access is to volume set HHH on the remote system.

Table entry # ? > (RET)

Reloading the UCI TRANSLATION TABLE

Please enter your initials > JSS

Enter comment (max. 200 chars.) > TEST FOR DOCUMENTATION

Type of operation > 4. SHOW UCI TRANSLATION TABLE

UCI Translation Table for configuration "TEST10":

Table entry #	UCI name	Volume set	Global name	New UCI name	New volume set
1	JSS	FFF	X	YYY	HHH

17.2.5 Journaling And Backup

General information on journaling is provided in Chapter 16, and information on backup in Chapter 11.

In a clustered system your master data base resides on one central system and the data is accessed by DDP lines from other systems. The basic principle for journaling is to establish journaling at the central system. This covers all accesses to the master data base from other systems. There is no need to provide journaling at the peripheral systems, unless critical data base updates are also occurring on the peripheral system's local data base.

The same principle applies to backup. You must backup the master data base on the central system. You should not try to backup the master data base from

one of the peripheral systems. If a peripheral system's data base contains its own local critical data, then the peripheral system's data base should be backed up on its own. The backup of the peripheral system's local data base should be done at the peripheral system.

17.2.6 Setting Access Codes And Categories For Globals (^%GLOMAN)

In order to access a global from a remote system, the global must have the appropriate codes. For globals to be accessed in clustered systems, the global must be placed in the WORLD category. This code allows access across volume sets. Note that the GROUP category is not adequate since it allows access only from users who are in the local volume set.

Use the Global Management utility (^%GLOMAN) to change the codes for a given global. You can set READ, WRITE, or DELETE access codes depending on what you want users to be able to do to the global. See Section 4.9.3 for more information on access codes.

17.2.7 Dual Porting

Dual porting is a feature that can be used with certain disk drives. These drives have two ports and can be connected to two CPUs at the same time. Dual porting can be used in two ways:

- Static dual porting - only one CPU at a time has access to the disk drive. The drive can be manually switched from one CPU to another by pressing the appropriate button on the disk drive. This approach is most useful in clustered systems, or in other situations to guard against a CPU failure.
- Dynamic dual porting - both CPUs can access the disk drive at the same time. This approach can cause data base problems, and can greatly increase disk access times. It should be used cautiously.

In a clustered system a disk drive can be connected to two CPUs with static dual porting. The primary CPU is actively connected to the dual-ported disk, and second backup CPU does not have access to the disk drive (though it is connected physically to the dual port). In the clustered system, the backup CPU has access to the data on the disk through the primary CPU using normal DDP arrangements.

If the primary CPU fails, then the System Manager can activate the connection to the backup CPU from the disk drive. He does this by pressing buttons

on the disk drive (see the appropriate disk drive user's guide for more information). He does not need to physically transfer the disk, but he must run the mount utility (^MOUNT) for that disk from the backup CPU.

The System Manager may then need to perform other tasks to insure that other systems can access the volume set in the dual ported disk drive, such as changing entries in the UCI Translation Table on other systems.

17.2.8 Library UCI

The library UCI is the one that contains all %routines and all %globals. By default this UCI is UCI1 of Volume Set 0. Each UCI has a reference to the library UCI it is supposed to use for %routines and %globals. You can use ^UCIEDIT to change the library UCI for each UCI.

In clustered systems, the library UCI should be the first UCI of the volume set for all UCIs in that volume set. In this way you can avoid problems if you must move the volume set from one system to another.

17.2.9 Using \$ZR To Verify DDP

You can use the \$ZREFERENCE special variable to verify whether DDP is actually working, instead of using the verify option of ^DDPUTL. This variable always contains the last completely resolved global reference. If the reference was to another volume set, this information is contained in \$ZR.

Assume that the UCI Translation Table is set up so that a reference to global ^XXX is made to ^XXX in the UCI, JHM, in volume set WWW on a remote system.

First, access global ^XXX; then WRITE the value of \$ZR.

```
>S B=^XXX(2,2)
>W $ZR
^["JHM","WWW"]XXX(2,2)
```

Thus, you can verify that the DDP access was actually made.

Appendix A

Detecting and Recovering from Errors

This Appendix describes DSM-11 error detection and recovery.

A.1 Types of Errors

The four major types of errors that can occur in a DSM-11 system are:

- MUMPS Programming Errors
- Mass storage hardware errors
- System crash errors
- Other system failures

A.2 Mumps Programming Errors

DSM-11 reports these errors either as typed messages to the relevant job's terminal, or as traps through the \$ZTRAP special variable for the job. General programming errors normally occur during routine development, and result from faulty routines. For example, insufficient partition space for a routine or its variables, or incorrect syntax. Other errors, such as accidental disconnection of a remote terminal or data base overflow, are out of the control of the programmer, but still must be anticipated when designing and managing an application system. See Section 4.10 for a discussion of MUMPS error handling, and Section A.5 for a list of possible errors.

A.3 Mass Storage Hardware Errors

DSM-11 maintains a log of the hardware errors that are described in this appendix. Two examples of mass storage hardware errors that DSM-11 records are disk read errors and magnetic-tape read errors. DSM-11 stores the device controller status information for these errors in the system table for subsequent logging on disk.

The occurrence of mass storage errors is usually transparent to the user because DSM-11 always makes a number of attempts to successfully perform the operation that caused the error. DSM-11 logs each unsuccessful attempt.

DSM-11 has a fixed number of retries for any particular device. After this fixed number is exceeded, DSM-11 displays the appropriate error message.

Hardware error logging is performed by a system utility routine, the caretaker utility, that runs as a background job in its own partition. This system utility routine monitors the system for error conditions, and logs them chronologically in a special system global. DSM-11 also has a utility routine (^KTR) that displays the information that is recorded in the error log. See Section 12.2 for more information on the caretaker utility.

A.4 Errors That Cause System Failure

The two types of system failures are: soft crashes and hard crashes. A soft crash results in an orderly shutdown of the DSM-11 operating configuration; however, the processor and the Executive Debugging Technique, XDT, (if included in the configuration) remain operable. See the *DSM-11 XDT Reference Manual* for more information on XDT.

You can restart the system if a soft crash resulted from certain hardware or operating system errors that are detected by the PDP-11 processor. These soft crash errors can include:

- Bus timeout error
- Illegal instruction error
- Memory parity error
- Segmentation error
- System power failure

When a soft crash occurs, DSM-11 logs the error information in the System Table and on the console terminal. See Section A.4.1 for more information on soft crashes. See Section A.4.2 for information on restarting the system.

A hard crash is caused by unpredictable events. Occurrence of either type of crash indicates an abnormal hardware or software condition that may require the assistance of Digital service personnel.

A.4.1 DSM-11 System Soft Crash

The DSM-11 operating system can detect certain system failures by trapping through hardware error vectors. All of these vectors transfer control to a single memory-resident trap handler. The trap handler can tell which vector caused the trap, because each error vector in DSM-11 has a unique processor status word (PSW) value. The trap handler stores a number of important CPU registers into a reserved crash block in the System Table (SYSTAB).

The trap handler prints (on the console terminal) the crash block, followed by the Kernel Stack, the User Stack, and the System Map. This entire printout is called a "crash report." If DSM-11 XDT is included in the crashed system, the trap handler gives control to XDT just as if the trap were an XDT breakpoint.

If the trap is a power failure, however, the crash report is not printed until the trap occurs a second time, indicating that power has been restored. Also, a power fail trap does not give control to XDT, but allows the system to continue.

A Digital service representative can use a crash report to determine the probable cause of the system failure. You should save any crash reports that occur, and write down as much information about the state of the crashed system (which terminals were active, names of running routines, and so on) as you can.

After the crash report has printed (if it is not a power failure), you may wish to attempt restart of the system to minimize data loss. See Section A.4.2 for a discussion of restarting the system.

The error vectors that cause crash reports are:

- | | | |
|----|----------------------|--|
| 4 | CPU Error | This is the most common symptom of a system failure, either hardware or software. The CPU error register at crash block location 2104 shows which of several possible errors has occurred. |
| 10 | Reserved Instruction | This trap is caused by either an accidental modification of memory, or by an erroneous attempt to execute a data value as an instruction. |
| 24 | Power Fail | This trap is neither a software nor computer error, and the system can be restarted without data loss. |

- 114 Memory System Error This is most likely a hardware error. See the Memory System Error Register (MEMSYS) at crash block location 1706. If bit 1 is set in MEMSYS, then a parity error has been detected in a memory location. The address of the faulty memory location is indicated by LOWADD and HIADD at crash block locations 1710 and 1712 respectively.
- 250 Memory Because of the structure of DSM-11, this trap occurs only if a Page Description Register is accidentally modified, which is a highly unlikely event.

For more information about error vectors and error registers, see your *PDP-11 Processor Handbook*. Here's a detailed description of the Crash Block:

SYSTAB Address	Hardware Register	Description
1644		PSW of error vector (flag for trap handler)
1646	-	General Register 0
1650	-	1
1652	-	2
1654	-	3
1656	-	4
1660	-	5
1662	-	6 (Kernel Stack Pointer)
1664	-	User-mode Stack Pointer
1666	177776	Processor Status Word (PSW)
1670	-	General Register 7 (Program Counter (PC))
1672	172352	Kernel Page Address Register 5 (PAR5K)
1674	172354	Kernel Page Address Register 6 (PAR6K)
1676	177652	User Page Address Register 5 (PAR5U)
1700	177654	User Page Address Register 6 (PAR6U)
1702	177656	User Page Address Register 7 (PAR7U)
1704	177766	CPU Error Register Bit 7 = illegal halt 6 = odd address error 5 = nonexistent memory (cache) 4 = Unibus timeout 3 = yellow-zone stack limit 2 = red-zone stack limit
1706	177744	Memory System Error Bit 0 = Main memory timeout 1 = Main memory address parity error
1710	177740	Low 16 bits of address of memory error (LOWADD)

A.4.2 Soft Crash Recovery

The safest way to restart is at memory location 40 (octal). This restart location is provided for just this purpose. When a 40 restart is done, the CPU registers and all terminals are reinitialized. To restart from DSMXDT, simply type 40G. To find out how to restart from your CPU console, consult your *PDP-11 Processor Handbook*, or PDP-11 processors user's, system installation, or system operating guide.

Restarting the system allows all jobs to resume operation without data loss. The job that is running at the time of the failure is given a <CRASH> error. After restarting the system, you should run the System Status utility routine (^STA) to obtain the current system status information. The Restore Jobs/Devices utility routine (^RJD) can be used to restore resources belonging to lost jobs and locked-out devices.

As a general rule, you should permit those jobs that were running at the time of a system failure to terminate before performing a system shutdown. However, soon you should shut down the system using the System Shutdown utility, ^SSD, and reboot. Continued running with a damaged system could cause problems.

A.4.3 DSM-11 Hard Crash

A hard crash is the unconditional termination of all processor operations. This error condition occurs when the system accidentally causes the processor to execute a HALT instruction (machine code 000000). This condition is caused by a hardware or software malfunction that results in the modification of the contents of one or more memory locations that contain the operating system. This error condition is identified by the abrupt termination of all system operations and the extinguishing of the processor's RUN light. Possible causes of this error condition include:

- Improper use of VIEW command
- Hardware failure
- Operating System failure
- HALT console switch being pressed

When a hard crash occurs, you should obtain the following information for diagnosing the cause of the error:

- The address displayed in the processor's ADDRESS REGISTER
- Contents of registers R0 through R6
- Contents of User Mode stack pointer
- Contents of the first five memory word locations (ascending order) beginning with the address contained in R6 Stack Pointer

After recording this information, you must reboot the system to continue operation.

A.5 DSM-11 Error Messages

The following is a list of the DSM-11 error messages.

MESSAGE	MEANING
<ABORT>	Indicates that you pressed the programmer abort key while in Programmer Mode during the execution of a routine. This is an unconditional interrupt of the routine.
<BREAK>	Indicates a BREAK command was encountered while executing. The BREAK command is used for diagnostic purposes. This message can be generated by a BREAK command in a routine, by setting a breakpoint (through \$ZBREAK), or by pressing CTRL/B during Programmer Mode.
<BLPRT>	Indicates an attempt to access a global while using the VIEW device in protected mode, and at the same time owning the VIEW device.
<CLOBR>	This error results from trying to modify the current routine by using the ZLOAD commands within the routine.
<CMMND>	Indicates illegal use of a command or a command that is undefined in the DSM-11 language.
<CRASH>	Error to the job in the RUN queue at the time of a restart following a system crash.
<DBDGD>	Indicates the contents of a block in the global does not conform to the defined structures given to directory, pointer, or data nodes.
<DBOVF>	Indicates that a global SET or ZSAVE has been aborted because of insufficient room on disk to allocate new space. The integrity of the data base may be degraded as a result of this condition.

- <DIVER> Indicates an attempt to perform division by zero.
- <DKHER> Indicates disk hardware error.
- <DKRES> Indicates that a global SET or a ZSAVE has resulted in the allocation of disk blocks from reserved disk space. Although the routine aborts because of this condition, the SET or ZSAVE is allowed to complete.
- <DKSER> Indicates a reference to a disk block number that is not in the range of blocks accessible by the DSM system. This error can also be generated by a degraded global or routine pointer, SDP, SPOOLING, or VIEW.
- Indicates (if not a system software error) an attempt to use the VIEW command to access a block number larger than the size of the referenced disk, or on a nonexistent disk.
- <DMCER> Indicates that either an illegal DMC11 I/O operation has been attempted, or a data transfer error has occurred in the DMC11 link.
- <DSCON> Indicates a telephone line has disconnected.
- <DSTDB> Indicates a DMC11 line has failed during a DDP data transfer.
- <FORMT> Indicates incorrect data format encountered while reading data from magtape, the TU58, or the RX02.
- <FUNCT> Indicates an undefined function or faulty use of a defined function.
- <INDER> Indicates that an indirection argument has incorrect syntax.
- <INRPT> Indicates that you pressed the `(CTRL/C)` key (or another designated application interrupt key) and that this key was enabled. See Section 4.7 for more discussion of interrupting routines and the application interrupt key.
- <ISYNT> Indicates a line of DSM code cannot be inserted into the routine buffer. The line label may not be legal, the line may contain illegal control characters, or the line may be too long.
- <KILLR> Indicates an internal error was generated by the routines `^RJD` and `^SSD` to terminate a job.
- <LINER> Indicates a reference has been made to a line that does not exist.

- <LPERR> Indicates a line-printer error. This error message is displayed only if \$ZTRAP is enabled, and the device is not using the Caretaker utility to monitor and report errors.
- <LVLER> Indicates an illegal attempt to transfer control to another level from a higher level execution block when using block-structured programming. See the *DSM-11 Language Reference Manual* for more discussion of block-structured programming.
- <MINIM> Indicates the value of a number is below the valid lower limits for numerics.
- <MODER> Indicates an attempt to execute a command that is not allowed in the designated mode while using a TU58 or RX02 device.
- <MTERR> Indicates magnetic-tape hardware or operator error as determined by the current contents of the \$ZA special variable. The system generates this error only if you SET the \$ZT special variable.
- <MXNUM> Indicates that the value of a number is outside the range expected at that point.
- <MXSTR> Indicates that the string exceeds the maximum length allowed (255 characters).
- <NAKED> Indicates an attempt to reference a global variable using naked syntax:
- Before any full syntax reference
 - After another user killed the global variable using the KILL command
- <NOBUF> Indicates that a ring buffer is not available for the current terminal I/O operation. Repeated occurrence of this error indicates insufficient ring buffer space has been allocated for the system.
- <NODEV> Indicates an attempt to open a device that is not included in the system, or an attempt to open a device with an illegal device number.
- <NOEBC> Indicates an attempt to open a device for use with the EBCDIC character set, but EBCDIC support is not configured into the running system.
- <NOPEN> Indicates an attempt to USE a device that has not been opened.

<NOPGM>	Indicates that a reference is made to a routine name that does not exist in the routine directory for this UCI, and is not in the directory of library (%) routines.
<NOSYS>	Indicates a reference to a system (through Distributed Data Processing) or to a volume set (through an extended reference) that is undefined.
<NOTSY>	Indicates an attempt to branch to location 0. This error may indicate corruption of the system image in memory.
<NOUCI>	Indicates a reference to a UCI (in an extended reference) that is undefined.
<PARn>	Indicates that there was an error during an OPEN or USE command on the parameter indicated by the number n. For example, <PAR1> refers to an error in parameter 1 of the OPEN or USE command.
<PGMOV>	Indicates that there is insufficient space available in the routine buffer, when loading a routine via ZLOAD, DO, GOTO, or QUIT.
<PLDER>	Indicates that the system cannot retrieve the routine being loaded, called, or started. The routine is corrupted on disk.
<PROT>	Indicates that an attempt was made to use the view command from a nonlibrary (%) routine, or when the System Manager has restricted the use of VIEW. This error also indicates an attempt to access a protected global. This error is also given to the owner of a VIEW job when it attempts to access global or routine data that it has protected by using the P switch.
<SBSCR>	Indicates illegal subscript usage: <ul style="list-style-type: none"> • A null string for a subscript • Too many characters in a subscript • Illegal characters in a subscript
<STKOV>	Indicates an overflow in machine stack space. This condition could arise from deeply nested indirection, endless program loops, or some other unusual condition.
<STORE>	Indicates there is insufficient space in the partition.
<SYNTAX>	Indicates that the current command being executed has an error in syntax. Syntax errors include illegal punctuation, illegal use of operators, and illegal use of parentheses.

- <UNDEF> Indicates a reference to an undefined local or global variable.
- <\$SERR> Indicates the \$SELECT command did not find a truth valued expression.
- <VWERR> Indicates an attempt to access a device in shared VIEW buffer mode without ownership of the VIEW device (device #63). This error may also result from an attempt to close the VIEW device before closing any devices still open in shared VIEW buffer mode.

Appendix B

Routine File Structure

Routine files consist of routine lines that are stored on disk. Routine names are stored in routine directory blocks. Routine lines are stored in routine data blocks.

Disk allocation for routine files is handled by the global module. The global module treats the routines in each UCI as if they constituted a single global. Routine names are inserted in routine directory blocks as if they were the first subscript of a global whose name was the space character (40 octal). The routine names are a single logical level of subscripts in that global. You can access this global using the \$DATA and \$ORDER functions.

Routine directories and data growth areas are set up at SYSGEN during the Modify UCI Data session. Therefore, the first routine directory block is pointed to by an entry in the UCI Table. The global module initializes each routine directory by placing one dummy entry in it. The identification field for the dummy routine is the space character without a first subscript (routine name).

Routine files only use the pointer structure and data level of the global hierarchy. In all respects, routine directory blocks are identical to global pointer blocks. They also behave the same. As the number of routine files increase, bottom-level pointer blocks split. The split creates two pointer levels. The original bottom-level pointer block becomes a pointer block and the new pointer block becomes the bottom-level pointer.

Each node of a pointer block points to *one* routine data block. That data block is fully dedicated to the routine, and may point to continuation blocks (using the "garbage" pointer) if the routine is larger than one block. Routine data blocks are not right-linked.

Figure B-1 shows the physical structure of a routine file that has continuation blocks.

Figure B-1: Routine File Structure

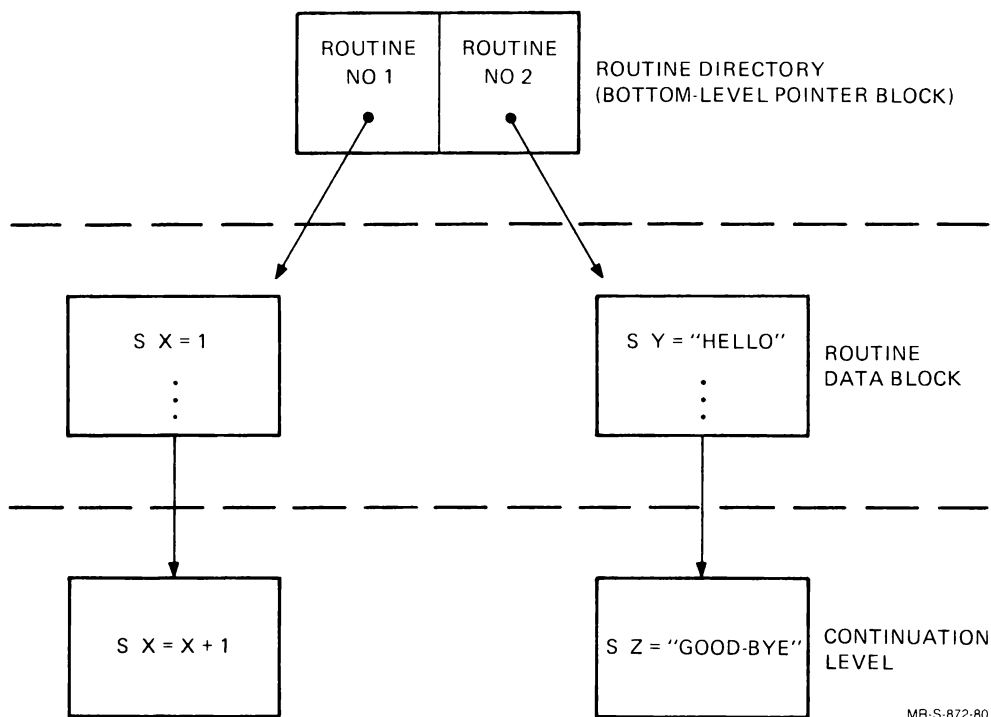
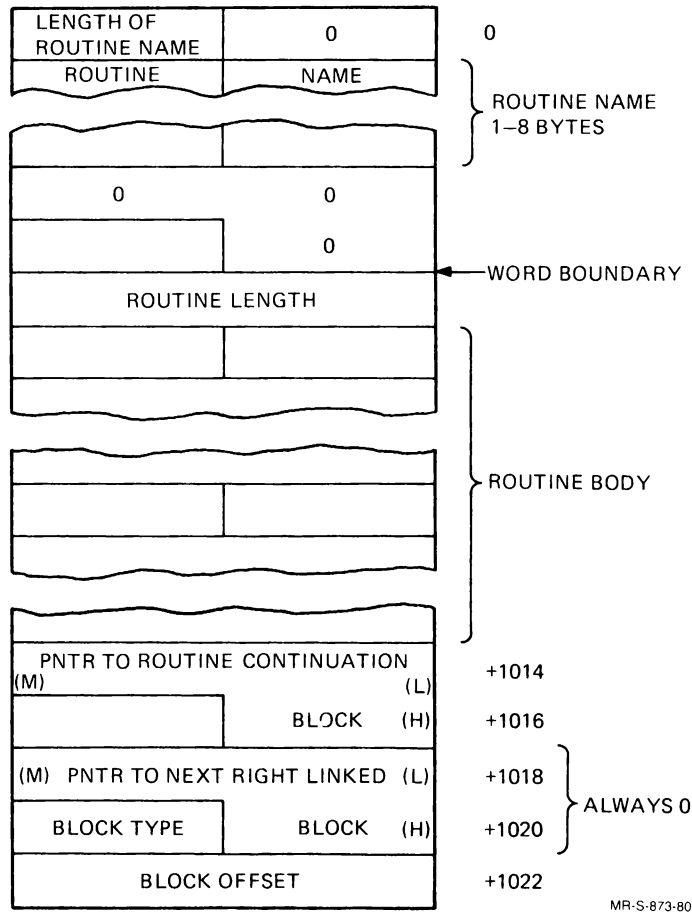


Figure B-2 shows the structure of the routine data block.

Figure B-2: Routine Data Block



Routine data blocks store routine lines in the following format:

LENGTH OF ROUTINE NAME

This 1-byte field stores the number of characters (bytes) in the routine name.

ROUTINE NAME

The following field stores the name of the routine. Routine names can use as many as 8 bytes.

ROUTINE LENGTH

This 1-word field stores the number of characters (bytes) in the entire routine.

ROUTINE BODY

This field stores the routine lines of a DSM routine. The maximum size of this field depends on the length of the routine name (since a routine name can use between one and eight bytes).

POINTER TO ROUTINE CONTINUATION BLOCK

Bytes 1014 to 1016 store the block address of the next continuation block in the routine file. This entry is zero if the file has no continuation blocks or if it is the last continuation block in the file. The address stored in this field is also used by the Garbage Collector. When you KILL a routine, the Garbage Collector is not used. The global handler follows the continuation pointers and collects the blocks immediately.

POINTER TO THE NEXT RIGHT-LINKED POINTER BLOCK

This 3-byte field always contains zeros.

BLOCK TYPE

This field stores the block type of the routine block:

- Type 10 (octal) if a routine data block
- Type 20 (octal) if a routine directory block

This field is 1 byte long.

BLOCK OFFSET

One word starting byte 1022 indicates how much of the block is filled, by pointing to the next free byte in the block.

Appendix C

DSM-11 Editors

This appendix discusses two DSM-11 editors used for editing routines. The first is the basic DSM-11 editor. The second editor is referred to as the EDI editor. Additional information on DSM-11 editors is given in Section 4.6.

C.1 The DSM-11 Editor

The DSM-11 editor consists of seven prompts that are displayed sequentially. After the editor displays a prompt, you simply enter the requested data, and the editor performs the specified operation. Section 4.6.2 contains information on how to access the editor; and the *Introduction to DSM* contains additional examples and a tutorial on how to use the editor. The following chart describes each editor prompt in detail. It lists the valid responses to each prompt and the results of their entry.

PROMPT 1: LINE

<i>Response</i>	<i>Result</i>
Carriage RETURN	The editor interprets a carriage return to mean that you wish to edit the last line edited. If this is the first line request, or the line does not exist for the label reference, the editor writes "LINE NOT FOUND" and returns to the LINE prompt. Otherwise, it proceeds to the prompt 2.

	Ends the edit session.
;	Proceeds to the fourth edit prompt, CHANGE EVERY.
; followed by any DSM code	Temporarily leaves edit mode and executes the DSM code, then reenters the editor at the LINE prompt.
Label or label & offset	Attempts to obtain the \$TEXT of the label reference and, if unsuccessful, writes "LINE NOT FOUND" and returns to the LINE prompt. Otherwise, it proceeds to the prompt 2.
+ or +N, where N is a positive integer	Interprets this entry as a positive offset from the last line label entered. Treatment is as described in response 1: Carriage RETURN. Note that a +0 works exactly like a carriage return.

PROMPT 2: REPLACE

<i>Response</i>	<i>Result</i>
Carriage RETURN	Writes the line being edited and returns to prompt 1.
END	This entry has two meanings. If the line you are editing contains the string "END", the editor proceeds to prompt 3 to replace the occurrence of "END" with the response to prompt 3. Otherwise, the editor proceeds to prompt 3 to add something to the end of the line.
A string containing either code, a comment, or string literal	If the line contains the specified string, the editor proceeds to prompt 3 to replace it with the response to prompt 3.
SUBSTRING...	This entry has two meanings. If the line contains the string literal "SUBSTRING...", the editor proceeds to prompt 3 to replace that string with the response to prompt 3. If it does not contain this string literal, the editor proceeds to prompt 3 to replace all text from the occurrence of SUBSTRING to the end of the line with the response to prompt 3.

SUBSTRING 1
..SUBSTRING 2

Two meanings exist for this response. If the line contains the string literal "SUBSTRING 1...SUBSTRING 2," the editor proceeds to prompt 3 to replace the occurrence of the string with the response to prompt 3. If it does not contain this string literal, the editor proceeds to prompt 3 to replace all text between the occurrences of SUBSTRING 1 and SUBSTRING 2 inclusive, with the response to prompt 3.

PROMPT 3: WITH

<i>Response</i>	<i>Result</i>
Carriage RETURN	Replaces the string specified in prompt 2 with nothing; that is, it removes it from the line and returns to prompt 2.
A string containing either code, a comment, or string literal	Replaces the previously specified string from prompt 2 with this response, and returns to prompt 2.

PROMPT 4: CHANGE EVERY

<i>Response</i>	<i>Result</i>
Carriage RETURN	Returns to prompt 1.
A string you wish to search for	Accepts whatever is typed, (including a ?) as the string to be searched for, and proceeds to prompt 5.

PROMPT 5: FROM LINE

<i>Response</i>	<i>Result</i>
Carriage RETURN	Assumes that you want to start searching from the first line in the routine; the editor writes "FIRST," and then proceeds to prompt 6.
A line label and optional offset	If the line does not exist, the editor writes "LINE NOT FOUND," and repeats the prompt. Otherwise, the editor proceeds to prompt 6.

PROMPT 6: TO LINE

<i>Response</i>	<i>Result</i>
Carriage RETURN	Assumes that you want to search to the last line in the routine buffer, writes "LAST," and proceeds to prompt 7.

A line label and optional offset If the line does not exist, the editor writes "LINE NOT FOUND," and repeats the prompt. Otherwise, the editor proceeds to prompt 7.

PROMPT 7: CHANGE TO

<i>Response</i>	<i>Result</i>
Carriage RETURN	Every occurrence of the "CHANGE EVERY" string is removed from the routine buffer. Answering this question begins the search. Each time the editor finds a line with the "CHANGE EVERY" string in it, it removes the string and prints out the line in its edited form. When all lines specified have been searched, the editor returns to prompt 1.
" (single quote)	Leaves all occurrences of the "CHANGE EVERY" string as they are; that is, the editor just performs a search. Each line that contains the "CHANGE EVERY" string is written out unchanged. When the editor is finished, it returns to prompt 1.
The string you want to insert	Proceeds with the search. Each time the editor finds an occurrence of the "CHANGE EVERY" string, it is changed to the "CHANGE TO" string and the edited line is written out. Note that the line is written out only once. Multiple occurrences of the "CHANGE EVERY" string are all edited before the line is written out. Editor returns to prompt 1.

C.2 The ^%EDI Editor

The ^%EDI editor is a line editor that operates on the routine that is currently in your routine buffer. The editor maintains a pointer that points at the current line of the routine buffer. The ^%EDI editor uses commands that you must enter after the EDI prompt, which is an asterisk (*). Most of these commands operate on the current line.

C.2.1 Using The ^%EDI Editor

The routine you wish to edit must be in the routine buffer. If the routine is on the disk, you must load it into the buffer by using the ZLOAD command before entering the EDI editor. To create a new routine, type:

```
> ZR  
> X ^ZEDI
```

To edit an existing routine, type:

```
> ZL Routine Name  
> X ^ZEDI
```

The editor responds with:

```
*** Editing <Routine Name> ***
```

```
*
```

Help information can be obtained by typing HELP or H. To get general information, type HELP. To get information on a particular EDI command, type H and the command name. The following example gives you information on the EDI ADD command.

```
* H ADD
```

To get a listing of all the EDI help information type:

```
* HELP ALL
```

You are then prompted for a device to list the information on. The default device is your terminal.

To exit from the EDI editor type:

```
* EXIT
```

C.2.2 EDI Commands

This section summarizes the EDI commands. Commands are indicated with uppercase and lowercase letters, such as "BOttom". The uppercase parts represent the abbreviated version of the command, in this case, BO.

Note that DSM-11 accepts the EDI commands in either uppercase or lowercase letters. For instance, you could type bo for the BOTTOM command.

In many cases the argument, such as string, is optional. If you do not use the argument, the editor usually prompts for a string. Many commands operate over more than one line of text. These commands have the letter n in their format. Substitute for n the number of lines of text (starting with the current line) that you want the command to cover.

Some commands use delimiters (such as /) to separate strings:

C/string1/string2/

Other delimiters can be used in addition to the slash (/), as long as that delimiter does not appear in the strings themselves. For example, a period can also be used as a delimiter:

C.string1.string2.

The final delimiter (after string2) is optional.

The EDI commands are as follows:

Add {string}

The indicated string is appended to the current line.

AP {string}

Same as the ADD command, except that the new line is printed.

BOttom or End

Moves the pointer to the beginning of the last line of the program.

Begin or Top

Moves the pointer to the dummy line preceding the top line of the routine.

{n}Change/string1/string2{/}}

Changes string1 to string2 in the current line. String1 and string2 cannot contain a / character.

CCcharacter

Refers to the concatenation character, which allows you to give multiple commands on one line.

Delete {n}

Causes a line of text to be deleted.

DP {n}

Same as the DELETE command, except that the new current line is printed.

EXit

Terminates the current editing session.

{n}Find {string}

Searches the routine for the specified string, beginning at the line following the current line.

Insert {string}

Inserts the specified string immediately following the current line.

KILL

Deletes all lines in the routine.

{n}Locate {string}

Causes a search for the indicated string beginning on the line following the current line.

{n}LC{/string1/string2/{/}}

Changes all occurrences of string1 in the current line to string2. String1 and string2 cannot contain a / character.

LI

Causes all remaining lines in the routine to be typed to the terminal.

MACRO number definition

This command is used to define a macro, which (in this situation) is a series of EDI commands.

MCall {global node}

This command allows you to retrieve up to three macro definitions previously stored in a global.

MSave {global node}

Saves the current macro in a global for future use.

{n}Mx {a}

This command executes the macro specified (by number) n times. The optional argument a is used by the macro as its argument.

Next {n}

This command moves the current pointer backwards or forwards in the file.

Overlay {n}

This command deletes n lines and replaces them with any number of lines that you type.

Print {n}

This command prints out the current line and the next n-1 lines on the terminal.

PAste{/string1/string2{/}}

This command is identical to the LC command except that all remaining lines in the program are searched and all occurrences of string1 are replaced with string2. String1 and string2 cannot contain a / character.

Retype {string}

This command replaces the current line with a string.

SC{/string1/string2{/}}

This search and change command searches for string1 in the routine and replaces it with string2. The located line becomes the current line. String1 and string2 cannot contain a / character.

SAve {n} {global node}

This command causes the current line plus n-1 lines to be saved.

TYpe {n}

This command is similar to the PRINT command except that TYPE does not move the line pointer after displaying the requested text.

UNsave {global node}

This command retrieves all the lines in a specified global and copies them after the current line.

Verify {ON}
Verify OFF

This command controls the display of lines specified by the LOCATE and CHANGE commands.

X string

This command allows the user to execute a sequence of DSM-11 commands before returning to EDI command level again.

`(ESC)`

This command prints the previous line in the program.

`(RET)`

This command prints the next line of the program.

`n <definition >`

This command defines and executes a macro in one step. The macro is executed n times.

‘

Appendix D

DSM-11 Disk Capacities and Calculations

This appendix describes disk capacities and how to calculate a DSM-11 block address.

D.1 DSM-11 Disk Capacities

Table D-1 lists DSM-11 disk capacities.

Table D-1: DSM-11 Disk Capacities

Disk Mnemonic	Type	Type Designation	Maps per Disk	Maximum Usable Block Numbers on Disk (Approximate)
DK	RK05	0	6	2399
DM	RK06	1	33	13199
DM	RK07	1	66	26399
DR	RM02	2	164	65599
DR	RM03	2	164	65599
DR	RM05	2	625	249999
DB	RP04	3	212	84799
DB	RP05	3	212	84799
DB	RP06	3	424	169599
DL	RL01	4	12	4799
DL	RL02	4	25	9999
DU	RA80	5	296	118399

DU	RA60	5	500	199999
DU	RA81	5	1100	439999
DU	RD51	5	24	9599
DU	RX50	5	1	399

D.2 Computing Block Numbers

DSM-11 and the system backup utilities store information on disks in disk blocks. A physical disk block holds 512 (decimal) bytes of information; while a logical DSM-11 block holds 1024 bytes. There are several ways to specify a particular disk block; the way that you specify a block number must be appropriate for the particular DSM-11 command, function, or utility that you are using. You can specify a block:

1. By its cylinder, track and sector numbers
2. By its absolute block number: the block at cylinder 0, track 0, sector 0 is absolute block 0; the block at cylinder 0, track 1, sector 0 is absolute block m, if m is the number of blocks per track on the disk; the block at cylinder 1, track 0, sector 0 is absolute block #m*n, if there are n tracks per cylinder on the disk.

If m is the number of disk blocks per track for a given disk type, and n is the number of tracks per cylinder, then the block at cylinder, c, track t, sector s, has the block number A as in:

$$A = ((c * n) + t) * m + s$$

If you already know the DSM-11 block number, M, you can calculate the absolute block number, A, using the following formula:

$$A = 2 * M (M - K - (u * 262144))$$

where the constant K depends on the disk type, and u is the unit number (0 to 7) by which you refer to the disk in your system.

3. By its DSM-11 block number (not a preferred syntax). If you know the absolute block number, you can use the following formula to calculate the DSM-11 block number:

$$\text{TYPE} * 2097152 + (\text{UNIT} * 262144) + \text{ABSOLUTE BLOCK NUMBER}$$

If you know the block number within the map, add

$$+ (\text{MAP} * 400)$$

to the formula shown above.

The DSM-11 block number is always equal to half the absolute block number, plus a fixed constant, which depends on the DSM-11 disk type and unit number.

Note that this syntax becomes ambiguous at block 262144. The preferred syntax is the one described in the following paragraphs.

4. By its block number relative to a disk; for example, the following is block number 1278 of DM1:

1278:"DM1"

5. By its block number relative to a volume set; for example, the following is block number 1278 of Volume Set 1:

1278:"S1"

‘

Appendix E

Using the \$ZCALL Function

You can use \$ZCALL to add your own functions to those already written in the DSM-11 language. The system has an area in memory reserved for user-written MACRO-11 subfunctions used by the \$ZCALL function.

NOTE

You can use the Autopatch utility to install your subfunctions in the system. To use \$ZCALL, you must have a working knowledge of the PDP-11 instruction set, and a thorough understanding of the internal conventions of the DSM-11 operating system, particularly as they apply to the interpreter and routine buffer structure. It is the user's responsibility to decode and manage any optional arguments that follow the subfunction name referenced by \$ZCALL.

The system table, SYSTAB, contains a pointer to the area reserved subfunctions, and defines the size of this area.

Several conditions must be met before you can use the \$ZCALL function:

- You must create a table that defines each of your subfunctions. Conversely, you must ensure that there is a subfunction for every entry in the table.
- You must maintain certain DSM-11 general purpose register conventions.

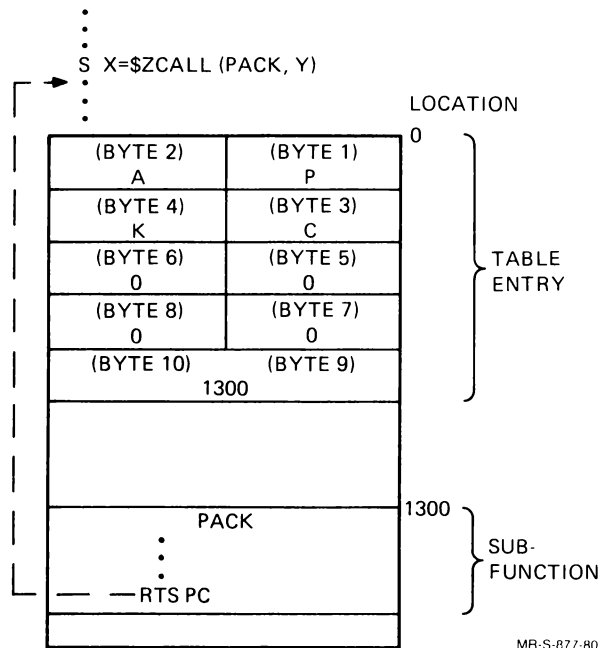
The following sections describe each of these conditions in further detail.

E.1 Creating the \$ZCALL Name Table

The beginning of the area reserved for subfunctions must be a table. This table contains the names and locations of each subfunction you create to be called by \$ZCALL.

Each entry in the table must be 10 bytes long. The first 8 bytes must contain the name of the subfunction as it is referenced by \$ZCALL. If the name is less than 8 characters, the remaining bytes must be filled with zeroes. The last 2 bytes of the table entry must contain the location of the subfunction relative to the beginning of the user-written subfunction area. Figure E-1 shows the table entry of a \$ZCALL subfunction called PACK. Note that ZCALL names that are preceded by a "%" are reserved for use by Digital.

Figure E-1: Memory Area for User-Written Subfunctions



All bytes, of both the name and locations fields, of the last entry in the table must contain zeroes.

E.2 Using DSM-11 General Purpose Registers

General purpose registers R0, R1, R4, and R5 must be properly managed as \$ZCALL enters and exits the subfunction. On entry into the subfunction:

- R0 contains the byte address of the character that terminates the subfunction name. For example:

```
S X=$ZCALL(PACK,Y)
      ↑
      R0
```

This character can be a comma. If, however, there are no arguments following the subfunction name, the character is the close parenthesis,).

- R1 and R5 both point to the next available byte in the Interpreter Stack (String Stack) in your partition.

NOTE

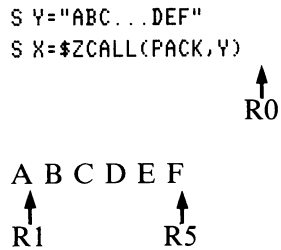
The Interpreter Stack temporarily stores information while the interpreter processes the subfunction. (For more information, refer to Section 14.13.)

If necessary, you can use this area to pass back any values that subfunction produces.

- R4 points to the beginning of the current job's partition.

Upon completion of the subfunction:

- R0 must point to the close parenthesis character that terminates the argument string.
- R1 points to the first character your subfunction has stored in the Interpreter Stack (that is, it has the same position as on entry into the subfunction).
- R2 and R3 can be changed by your subfunction.
- R4 must point to the the beginning of the current job's partition.
- R5 points to the next available byte in the Interpreter Stack.



Interpreter Stack

As shown in Figure E-1, your subfunction must end with the MACRO-11 instruction, RTS PC. This transfers program operation from the subfunction back to the location recorded by the Program Counter (PC) in the processor stack.

E.3 Error Conditions

There are several conditions that cause an error message when you use the \$ZCALL function:

- If you reference a subfunction that has not been defined in the table, DSM-11 issues a <FUNCT> error.
- When the subfunction finishes, if R0 does not point to the terminator of the \$ZCALL function string, DSM-11 issues a <FUNCT> error.
- If your subfunction tries to store more characters in the Interpreter Stack than the stack has space for, DSM-11 issues a <STORE> error.

Appendix F

Using the Operations Control Register

This appendix contains reference information for using the operations control register.

F.1 Operations Control Register Use

The operations control register is a concept, rather than a single hardware or software feature. The operations control register determines both the function of the console terminal and the general operational status of the system.

You can set or clear switches by using a system utility, `^SWREG`, that allows you to examine and alter the software switch register. Setting the hardware control switches does not affect the software switches, which ignore the hardware settings.

F.2 Operations Control Switches

The following paragraphs describe the function and use of each operations control register switch. Table F-1 contains the octal and decimal values for the switch functions.

Switch 4

When this switch is ON, it disables error logging to disk.

Switch 5

When this switch is ON, all system errors that are normally logged through the system caretaker utility are printed on the system console terminal. You should use this switch if you want to diagnose errors as they occur.

Switch 6

When this switch is ON, it prevents all users, except the console terminal user, from logging in. When you set this switch, all users attempting to log in will receive the following message:

```
SORRY, LOGIN DISABLED
```

Switch 10

When this switch is ON, it indicates that a warm system restart is desired. This function cannot be controlled through the hardware switch register.

Switch 13

When this switch is ON, it prevents the writing of any data to the disk except by VIEW. This is a lockout on data base modification operations. This function cannot be controlled through the hardware switch register.

Switch 14

When this switch is ON, it prevents the reading or writing of any disk data except by VIEW. This switch performs a total lockout on all data base operations. This function cannot be controlled through the hardware switch register.

Table F-1 lists the octal and decimal values for each DSM operation control register switch.

Table F-1: Values for Operations Control Register Switches Function Assignments

Function	Switch	Octal Value	Decimal Value
Disables Error Logging	4	20	16
Prints System Errors	5	40	32
Disables Log In	6	100	64
Enables Warm Restart	10	2000	1024
Disables Data Base Writes	13	20000	8192
Disables Data Base Reads And Writes	14	40000	16384

(

Appendix G

Using Bit-Masking Techniques

To examine the condition of one or more bits within two bytes, you must use a technique called bit masking. Bit masking involves shifting the bit(s) you want to examine to the right into a "viewing" position.

NOTE

Remember that two bytes consist of 16 bits numbered 0 to 15 from right to left.

To mask a bit in DSM-11, use the following procedure:

If the bit to be examined is in a two-byte segment that is interpreted as a DSM-11 number, integer divide the segment by the appropriate power of 2. For example, to test if bit 12 is set in XYZ, do the following:

```
I XYZ\4096#2...
```

This shifts bits 0 to 11 to the right in XYZ, leaving bit 12 in bit 0s position, then tests whether the right-most bit is equal to 0.

The following chart gives a general guide to the use of bit masking, where X is the value to be tested.

To Examine Bit n Use

0	X#2
1	X\2#2
2	X\4#2
3	X\8#2

To acquire certain information about the operating system, you may need to shift one or more bits in a memory location in order to examine other bits. In the following examples, LOC represents a memory address in the system table.

```
LOC\256 ;shift right 8 bits (high byte into low byte)
```

```
LOC*256 ;shift left 8 bits (low byte into high byte)
```

Combining operations:

```
LOC\256#2 ;shift high byte to low byte and check  
;for bit 8 set.
```

To mask for more than 1 bit at a time, use the following procedure. First, integer divide by the power of 2 corresponding to the starting bit (counting from right to left) of the group that you want to shift; that is, if you want to shift out bits 0-8, integer divide the two-byte segment by 512. Next, check the bits by means of a modulo 2 of the appropriate power. The following examples illustrate specific cases:

```
LOC\512#8 ;check if any of bits 9 to 11 are set
```

```
LOC#256 ;check for any bit set in low byte
```

To shift left (zero fill), multiply by corresponding powers of 2.

The following routine checks the status of all bits in the \$ZA (magnetic tape status) special variable by shifting the contents of the word to the right (dividing by 2) for successive bit positions. If this routine finds an error, it executes the routine ERR and then returns to check the next bit.

```
A S A=$ZA F I=0:1:15 D ERR:A#2 S A=A2
```

Appendix H

Internal Subscript Format for a DSM-11 Global

DSM-11 uses a method of global encoding referred to as 8-bit encoding. This approach allows subscripts to be composed of all possible 8-bit byte values except 0 (ASCII null). Negative subscripts collate in the proper numeric sequence. A flag is present in global directory entries to indicate whether a particular global is encoded as 8-bit or 7-bit. (Versions of DSM-11 previous to Version 3 use 7-bit encoding.)

This description applies to 8-bit globals. Each subscript in the node reference for a DSM-11 global is constructed of a variable-length string of bytes, as follows:

coll. desc.	1st byte	2nd byte	...	last byte	null byte
-------------	----------	----------	-----	-----------	-----------

Each key is a concatenation of subscripts, each with a trailing zero byte, as follows:

C	B	B	B...B	0	C	B	B	B...B	0	...C	B	B...B	0
---	---	---	-------	---	---	---	---	-------	---	------	---	-------	---

The collation descriptor byte, the first byte in the subscript, has a value that depends upon the type of subscript. These values are:

0	reserved
1	null subscript
2-127	negative canonic number having 125-0 integer digits
128	canonic zero
129-253	positive canonic number having 0-124 integer digits
254	ASCII string
255	reserved

When the subscript is an ASCII string, the bytes of the subscript following the collation descriptor byte are the bytes of the string.

When the subscript is a positive canonic number, the bytes after the collation descriptor byte are the ASCII values of the digits (including decimal point, if any) of the number.

When the number is a negative canonic number, the bytes following the collation descriptor byte are the nine's complement of the digits. In this case, the minus character is not included, but a byte of value 254 is appended to the end of the subscript.

All types of subscripts have a trailing null byte, which acts as a separator between subscripts.

The following examples show how particular global nodes are represented:

```

^( "ABCD" ) 254 "A" "B" "C" "D" 0
^(12.5)      131 "1" "2" "." "5" 0
^(-2.5)     126 '2' '.' '5' 254 0

```

A number written without quotes is the actual decimal value of the byte. Double quotes indicate that the value of the byte is the ASCII value of the quoted character. Single quotes indicate that the value of the byte is the nine's complement of the ASCII value of the quoted character.

Glossary

Address

A label, name, or number that designates a location where data is stored.

Argument

An expression that controls the action, location, direction, or range of the command or function with which it is used.

Array

An ordered set of local or global elements or nodes referenced by subscripts and a common variable name.

ASCII

The American Standard Code for Information Interchange. A series of 128 characters defined in ANSI (the American National Standards Institute) Standard X3.4-1968.

Auxiliary Storage

This term refers to any device that stores data, except core memory.

Background Job

A job that:

1. Is started by the JOB command

2. Does not own any devices
3. Has a principal device (the device that started its execution)

Also see Job.

Balanced Tree

A graph that represents the structure of DSM-11 arrays. Balanced trees are drawn like a family tree with the root (name) at the top and the nodes arranged below by their depth of subscripting. All nodes with one subscript are on the first level, all nodes with two subscripts are on the second level, and so forth.

This tree structure is only a logical picture of an array; it does not reflect how DSM-11 physically stores the array. For example, the tree structure shows all nodes that point to defined data nodes. DSM-11 does not store such pointer nodes, but does recognize their logical existence.

Baud

A term used to describe data transmission rates, equivalent to one bit per second. For example a 1200 baud rate equals 1200 bits per second.

Bootstrap

A technique or device designed to bring itself into a desired state by means of its own action. For example, a "bootstrap" routine is one whose first few instructions are sufficient to bring the rest of itself into the computer from an input device.

Breakpoint

A location at which routine operation is suspended in order to examine partial results.

Collating Sequence

An ordering assigned to a set of items. For storage of array nodes or elements DSM-11 collates in a numeric collating sequence. It first stores all nodes with subscripts that are canonic numbers (subscripts that contain a string of valid, numeric characters) in the ascending numeric order of their subscripts. It then stores all nodes with subscripts that are strings or noncanonic numbers (such as 01) in the ascending order of their ASCII-character codes.

Command

A name or mnemonic, that, by virtue of its syntax and position on an input line, causes a computer to perform a predefined action.

Command Line

One or more statements input to DSM-11 for immediate execution. DSM-11 distinguishes a command line from a routine line (entered for later execution) by the absence of a line label and TAB character.

Command Mode

One of two modes of system operation. This mode allows you to:

1. Enter commands for immediate execution
2. Create or modify routines
3. Execute routines

Comment

A brief, nonexecuted explanation of the purpose of a routine or of a routine line. You must precede all comments with a semicolon (;). You can use any valid DSM-11 graphic character in a comment.

Concatenation

The process of evaluating two or more string expressions to produce a single string that is a joining of the values of the expressions. Concatenation is designated by the underline character.

Configuration

A particular selection of hardware and software resources that are tailored to provide a system environment.

Constant

A constant is a value or string that does not change its value from one execution of a routine to the next.

Control Characters

The result of pressing the CTRL key and a letter key simultaneously.

Data

The data (facts, numbers, and symbols) entered into the system for processing and/or storage. DSM-11 has only one data type: the variable length string. (DSM-11 regards numbers as a special interpretation of a string.)

Data Base

The body of disk-stored data that resides in global arrays.

Data Record

The individual pieces of data associated with the nodes of a global array. Data records can be variable length, up to 255 characters.

Default

The value of an argument, operand, or field assumed by a routine if a specific assignment is not supplied by the user.

Descendant

For any array node (element), any other array node on a lower (deeper) subscripting level that can be reached from that node and that shares the first n subscripts in common with that node. For example, the nodes A(1,2,2) and A(1,2) are descendants of A(1).

Device

Any part of the computer system other than the central processing unit, memory, or their associated structures. A device can be a physical piece of equipment, such as a terminal or a line printer, or a logical piece of equipment.

Device Controller

A hardware unit that electronically supervises one or more of the same type of devices. The device controller acts as a link between the CPU and the I/O devices.

Duplex

A communications line that can send data in both directions simultaneously is called a full-duplex line. A line that can send data in both directions but not simultaneously is called a half-duplex line. Duplex can also refer to a computer system that contains two interconnected CPUs.

Expression

A string of characters that yields a value when executed. An expression is composed of expression atoms (variables, functions, and literals), binary operators, unary operators, formatting characters, parentheses, and indirection operators.

Fatal Error

An error from which a process cannot recover. For example, errors that cause the CPU to stop, or disk write errors that are not caused by the disk drive being powered down or write-locked. Errors that a process can recover from are not fatal. For example, errors that are reported in \$ZA or trapped by error processing routines.

Function

A name preceded by a dollar sign (\$) and followed by an argument or argument list enclosed in parentheses. The function and its argument list defines and performs a procedure.

Global

A simple global variable or global array stored on disk.

Global Variable

A named reference to data in a global on disk can be either simple (a variable name that references a single datum). Global variables are not unique to a user. They can be examined and/or changed by any authorized user.

Indirection

A means of using the value of an expression instead of the expression itself. Indirection is denoted by the at (@) sign followed by an element that evaluates to an expression atom.

Whenever DSM-11 finds an occurrence of indirection, it substitutes the value previously given the expression atom for the occurrence of indirection. This allows DSM-11 to execute the same segment of code repeatedly for different values of the expression atom.

Interpreter

A part of the DSM-11 operating system that translates DSM-11 commands and statements into machine instructions and executes the specified operations. The interpreter translates a DSM-11 routine each time it is run.

Interrupt

A signal which, when activated, causes a transfer of control to a specified location in memory, thereby breaking the normal flow of control of the routine being executed.

Job

Any system activity that requires the use of a partition. For example, logging into the system, or JOBing a routine.

Kernel Mode

The hardware mode in which the operating system normally executes. In this mode, all machine instructions can be executed.

Line

A string of characters terminated with a carriage-return/line-feed combination.

Line Label

An optional name at the beginning of a routine line that identifies the line within that routine. A line label should have no more than eight alphanumeric characters. DSM-11 evaluates only the first eight characters. If the first character of the line label is an uppercase alphabetic character or a percent (%) character, the remaining characters in the label can be any combination of uppercase alphabetic characters or digits. If the first character of the line label is a digit, the remaining characters in the label must be digit characters and must be unique within a particular routine.

Literal

A string of characters delimited by double quotation marks that occurs within the context of a routine and that never changes value from one execution of the routine to another. DSM-11 recognizes two types of literals:

- Numeric literals
- String literals

Local Variable

A variable that exists only in memory. Local variables are unique to a user and can usually be inspected and/or changed only by that user.

Map

A disk storage unit. A map consists of 400 DSM-11 blocks (which are 1024-bytes long). The 400th block of every map is a map block which contains allocation data about previous 399 blocks in the map, and itself.

Modem

Acronym for MOdulator DEModulator. A modem is a device that converts data generated by data processing equipment to a form that can be transmitted over telephone lines, and reconverts data transmitted over telephone lines to a form that is compatible with data processing equipment.

Naked Reference

A shorthand method of referring to a node in a global array. Naked references permit you to refer to the sibling or descendant of a previously referenced node by using only the circumflex (^) character and the unique portion of the sibling's or descendant's subscript.

Nesting

A relationship between two DSM-11 language statements. When two statements are nested, the second (or inner) statement is wholly contained within the range of the first (or outer) statement. Generally, the inner

statement completes its execution cycle, then returns to the outer statement so it can complete its execution cycle.

Node

A global or local array element addressed by the name (common to all members of the array) and a unique subscript.

Operator

A symbolic character that indicates the action to be performed on operands.

Partition

The area of memory that contains all elements of a job. Partitions contain routine and local variable storage areas, and status data about routines that is used by the executive's time-sharing algorithm.

Principal Device

The terminal at which you log in. Although each terminal has a unique device number associated with it, DSM-11 always interprets Device 0 to mean the principal device.

Prompt

A word or message printed by the system that requests some action on your part.

Queue

A queue is an ordered list in which the first item entered is the first item removed.

Right-Linked Pointer

A pointer stored in a blocks used by globals and routine files that specifies the address of a related block. If the block is a global pointer or bottom-level pointer block, or a routine directory block, the right-linked pointer points to the block whose nodes collate immediately following the nodes in the current block. If the block is a global directory block or global data block, the right-linked pointer specifies the address of the next block in the file (since these blocks form sequential files). If the block is a routine data block, right-linked pointers are not specified.

Routine

A collection of DSM-11 lines, saved, loaded, called (using DO), or overlaid (using GOTO) as a single unit. A set of computer instructions or symbolic statements combined to perform a task.

Routine Buffer

One of five major sections of a partition. The routine buffer stores the lines of a DSM-11 routine. As the routine buffer fills, it grows toward the high end of the partition's memory. As the routine buffer empties, it shrinks toward the low end of the partition's memory.

Routine Line

One or more statements input to DSM-11 for later execution as part of a routine. You must precede the statements in a routine line with a TAB character. You can precede the TAB with an optional line label.

Run Queue

The run queue is a system queue that contains the number of the job currently executing in its time slice. This queue is effectively a one-entry queue.

Sparse Array

An array that need not be predefined to its maximum size. A sparse entry contains only those nodes that have been explicitly defined.

The sparse array is a dynamic structure. As more nodes are defined, DSM-11 allocates space for them.

Special Variable

A variable that is permanently defined within DSM-11. These variables provide system and control data. The first character of a special variable is always a dollar sign (\$).

Stand-alone

This term refers to a system or piece of equipment that is capable of doing its job without being connected to anything else.

Storage Reference

A name symbolic representation of a storage location that has the value of the contents of that location. DSM-11 recognizes three types of storage reference:

- Local Variables
- Global Variables
- Special Variables

String Data

Any set of between 0 and 255 characters taken as a string data entry and referenced by a local or global variable.

String Literal

A string of characters enclosed in double quotation marks within the context of a line. The value of a string literal is a function of its spelling.

Statement

A unit consists of a command (and, optionally, its modifying arguments) that specify an operation to be performed.

String

A set of up to 255 ASCII characters. DSM-11 stores all data as variable-length character strings.

Subroutine

A group of statements arranged so that control can pass to the subroutine and back to the main routine again. Subroutines usually perform tasks required more than once in a routine.

Subscript

A numeric or string-interpreted value appended to a local or global variable name to identify specific elements or nodes in an array. Subscripts are enclosed in parentheses. Multiple subscripts must be separated by commas.

Subscripted Variable

A local or global variable to which a subscript is affixed. An element or node in a local or global array.

Substring

Any contiguous part of a specified string.

System Queue

This term refers to the set of queues used by the DSM-11 operating system to control the allocation of system resources.

Tied Terminal

A terminal that can only run preselected routines and access preselected globals. When you log in at a tied terminal, DSM-11 forces the automatic and protected startup of the application to which the terminal is tied.

Timeout

An integer-valued expression preceded by a colon that can be appended to the argument of an OPEN, LOCK, READ, or JOB command. The integer specifies the number of seconds DSM-11 is to try to complete the operation the command specifies.

Time Slice

This term refers to the period of time allocated by the operating system to process a particular partition's routine.

User Mode

In this hardware mode, certain instructions cannot be executed. This mode is used to prevent one user in a multiuser system from altering data in other partitions, or in the operating system itself.

Utility Routine

This term refers to any general-purpose routine that is included in an operating system to perform common functions.

Variable

A symbolic name for a location where a datum is stored. DSM-11 recognizes three types of variables:

- Local Variables
- Global Variables
- Special Variables

Local variables are stored in memory. Global variables are stored in arrays on disk. Special variables are maintained by DSM-11 for all users.

INDEX

- Abort key
 - see control Y
- Active Job Report utility
 - see ACTJOB
- ACTJOB, 12-11
- Add UCIs utility
 - see UCIADD
- Application interrupt key, 10-15, 10-26, 12-3
 - see also control C
 - defining the key, 4-17
 - OPEN parameter, 6-11
 - recognition of, 4-16
- Application Mode, 2-3
 - definition of, 2-2
 - login, 3-8, 3-9
 - logout, 3-11
 - routine execution, 4-13
 - stopping routine execution, 4-14
- Applications
 - structure, 9-9
- Arrays
 - see globals
- ATTACH, 12-15
- AUPAT, 11-1, 11-5 to 11-7, 12-14
- Autobaud, 10-29
- Autoconfiguration, 1-4, 1-5, 1-25, 10-1
 - completion, 1-31
 - definition of, 1-4
 - example, 10-27 to 10-29
 - installation, 1-4, 1-8, 1-26
 - requirements, 10-2
 - utility
 - see CONFIG
- Autopatch utility
 - see AUPAT
- Background Job Attacher utility
 - see ATTACH
- Background Job Detacher utility
 - see DETACH
- BACKUP, 11-1, 11-4, 12-6
 - clustered systems, 17-13
- Bad Block Table, 1-23, 10-24, 14-13, 14-13 to 14-14
- Bad blocks, 13-3
 - see also Bad Block Table
 - in installation, 1-7
 - installation, 1-23
 - test, 1-23
 - test patterns, 1-23
- Baseline System, 1-1, 1-2 to 1-3, 1-4, 1-5, 1-21, 1-25, 1-30
 - PAC, 1-3, 1-30
 - UCI, 1-3, 1-30
- BBTAB, 12-6
- BCS, 12-14
- BISYNC, 11-3
 - SYSGEN, 10-7, 10-9, 10-22
- Bit-masking, G-1 to G-2
- BLDMP, 12-10
- Block numbers
 - computing, D-2
 - SDP, 6-35
- Block structuring
 - optimizing routines, 9-1, 9-2
- BREAK, 3-6, 4-15, 4-17, 4-20, 4-29, 6-11
 - arguments, 4-16
- Break Mode, 3-8
 - description of, 4-17
 - entering, 4-17
- Breakpoints, 4-19
 - actions, 4-20

- clearing, 4-21
- continuing execution, 4-21
- examining, 4-21
- setting with \$ZBREAK, 4-19
- setting with BREAK, 4-20
- Broadcast utility
 - see BCS
- Buffers
 - sizes, 15-2 to 15-3
- CARE, 12-3 to 12-5
 - Change Error Printer, 12-5
 - Erase Hardware Error Log, 12-4
 - Print Disk Error Summary, 12-5
 - Start System Caretaker, 12-4
 - Stop System Caretaker, 12-4
- Caretaker job
 - installation, 1-31
 - line printer errors, 6-17
 - start-up, 11-2
 - utilities
 - see CARE
- Cassettes
 - see TU58
- Characters
 - Binary Pattern Match operator, 9-7
 - BREAK key, 4-17
 - carriage return with routines, 7-3
 - circumflex, 9-14
 - circumflex and globals, 8-2
 - control, 3-5 to 3-6
 - formatting, 5-6, 6-17
 - function keys, 3-4
 - indicating octal number, 7-3
 - indirection operator, 9-8
 - percent, 9-11
 - question mark, 10-16
 - terminal escape keys, 6-12 to 6-15
 - up arrow, 9-14, 10-3, 10-16
 - up arrow and globals, 8-2
 - up arrow with routines, 7-3
 - use of uppercase and lowercase, 4-2
- CLOSE, 5-4, 6-22, 6-35, 6-49
- Command lines
 - definition of, 4-4
- Commands
 - abbreviations, 4-2, 9-6
 - debugger, 4-18
 - entering commands, 4-2 to 4-4
- Comments, 4-3, 9-10
- Communications
 - between computers
 - see DMC11
 - between jobs
 - see JOBCOM
 - between routines
 - see routine interlocks
- CONFIG, 10-3, 10-27
- Configuration
 - autoconfiguration, 1-4, 10-1, 10-2
 - definition of, 1-4, 10-4
 - installation, 1-25
 - manual, 10-1
 - name, 10-17
 - size, 10-17
- Control B, 4-15, 4-17
- Control C, 4-15, 4-16, 10-15, 10-26, 12-3
 - see also application interrupt key
 - Programmer Mode interrupt, 4-15
- Control characters, 3-5
- Control Y, 4-15, 10-15, 10-26, 12-3
 - Programmer Mode interrupt, 4-15
- %CRF, 7-8
- %CURSOR, 7-14
- Cursor Control utility
 - see %CURSOR
- Customization
 - installation, 1-30
 - using system generation, 1-1
- %D, 7-10
- DAT, 12-14
- Data Base Repair utility
 - see FIX
- Data Base Supervisor
 - see operating system
- Data records, 8-5
- Date
 - Date and Time utility
 - see %H
 - Show Current Date utility
 - see %D

Date and Time utility
 see %H
 DBT, 12-10, 13-10, 16-3
 DDCMP, 6-40
 DDP
 global access syntax, 8-8
 SYSGEN, 8-8, 10-5, 10-6, 10-21,
 17-7
 volume sets, 17-1 to 17-15
 DDPUTL, 12-15, 17-9 to 17-10
 DDR, 12-11, 14-16
 Debugger
 MUMPS, 3-6, 4-17 to 4-22
 breakpoint actions, 4-20
 breakpoints, 4-19
 clearing breakpoints, 4-21
 continuing execution, 4-21
 enabling and disabling, 4-18
 examining breakpoints, 4-21
 setting breakpoints, 4-19, 4-20
 system, 3-6, 4-17
 see also XDT
 Decimal
 as default base for numbers, 7-3
 Decimal/Octal Conversion utility
 see %DOC utility
 to Octal Conversion utility
 see %DO utility
 Define Start-up parameters
 see STUBLD
 DEJRNL, 16-5, 16-6
 DESPOOL, 6-47, 6-51
 DETACH, 12-15
 Device Descriptor Block, 14-14 to
 14-16
 Device Status Word, 6-4
 Device Table, 5-2, 14-7 to 14-8
 utility
 see DEVTAB
 DEVTAB, 12-12
 DGAM, 13-12, 13-26, 14-11
 DH11, 6-1, 6-8
 SYSGEN, 10-5, 10-20
 %DIAL, 7-12
 Direct Mode, 3-8
 see also Programmer Mode
 definition of, 4-4
 description of, 4-5
 prompt, 4-5
 Directories, 4-22 to 4-24
 global, 4-23
 routine, 4-23, 14-11
 Disk Table, 14-13, 14-13
 utility
 see DISKMAP
 Diskettes
 see RX02
 DISKMAP, 14-13
 DISKPREP, 12-7
 Disks
 see also Disk Table
 backup, 11-4
 Backup utility
 see BACKUP
 Bad Blocks utility
 see BBTAB
 Block Dump report utility
 see BLDMP
 block layout, 13-5
 blocks, 13-5
 capacities, D-1
 computing block numbers, D-2
 Data Base Repair utility
 see FIX
 Disk Block Tally report utility
 see DBT
 Disk Table utility
 see DISKMAP
 disk-cache usage, 13-21
 Dismount utility
 see DISMOUNT
 dual porting, 17-14
 Fast Disk Block Tally report
 utility
 see FASTDBT
 Format/Initialize utility
 see DISKPREP
 general layout, 13-5
 global directory block, 13-10
 Integrity Check report utility
 see IC
 journaling, 16-5
 Label utility
 see LABEL
 layout, 13-4 to 13-17
 maintenance utilities, 12-5 to
 12-7
 map blocks, 13-8
 maps, 13-2, 13-5, 14-12
 Mount utility
 see MOUNT
 mounting at start-up, 11-3

- overflow, 13-20
- print error summary utility
 - see DSKTRACK
- report utilities, 12-10
- reserved space, 13-19
- restore, 11-4
- Restore utility
 - see RESTORE
- SYSGEN
 - disk information, 10-4, 10-18
 - disk reserve, 10-13
 - disk-tape buffers, 10-10, 10-22, 10-24
 - write check, 10-13
- types, D-1
- units, 13-4
- DISMOUNT, 12-6
- Distributed Data Base Processor
 - see DDPUTL
- Distributed Data Processing
 - see DDP
- Distribution kit
 - backup, 1-21
 - distribution media, 1-2, 1-5
 - DSM-11, 1-1, 1-2
 - media backup, 1-6
 - mounting media, 1-14
- Distribution media, 1-2, 1-5
 - backup, 1-21
 - mounting, 1-14
- DL11
 - SYSGEN, 10-19, 10-22
- DMC11, 6-40 to 6-46
 - Block Mode, 6-41
 - Buffer Mode, 6-43
 - commands, 6-40
 - DDCMP, 6-40
 - DDP, 8-8
 - device numbers, 6-40
 - error conditions, 6-45
 - Message Mode, 6-41
 - modes, 6-40
 - OPEN in Buffer Mode, 6-43
 - switched telephone network, 6-44
 - SYSGEN, 10-5, 10-19, 10-20, 10-22, 17-6, 17-7
 - USE in Buffer Mode, 6-43
 - VIEW buffer, 6-43
 - WRITE * with Block Mode, 6-42
 - WRITE * with Buffer Mode, 6-44
 - \$ZA, 6-43, 6-45, 6-46
 - \$ZA Status Bit Assignments, 6-45
 - DO, 4-14, 9-1, 9-2, 9-5, 15-2
 - %DO utility, 7-13
 - %DOC utility, 7-9
 - DOS-11, 6-27, 10-35
 - DSM-11
 - data access, 2-2
 - I/O, 5-1
 - language description, 2-1
 - system access, 2-3
 - system capacity, 2-2
 - system hardware, 2-6
 - system overview, 2-1
 - system software, 2-3 to 2-6
 - DSM-11 editor
 - see editors
 - DSOF SPL, 12-9
 - DSONSPL, 12-9
 - Dual porting, 17-14
 - DUP11
 - SYSGEN, 10-7, 10-9, 10-20, 10-22
 - DUV11
 - SYSGEN, 10-7, 10-9, 10-22
 - DZ11, 6-1, 6-8, 6-9
 - SYSGEN, 10-5, 10-20
 - DZV11
 - SYSGEN, 10-20
 - EBCDIC
 - translation, 10-6, 10-8, 10-35
 - %EDI, 4-12, C-4
 - see also editors
 - Editors, 4-11 to 4-13
 - DSM-11 Editor, 4-12, C-1 to C-4
 - %EDI, 4-12 to 4-13, C-4 to C-9
 - editing using ZPRINT and ZREMOVE, 4-6
 - Global Editor (%GEDIT), 4-13
 - Load DSM-11 Editor utility
 - see %LOAD%ED
 - Programmer Mode, 4-11
 - types of editors, 4-11
 - %EGD, 7-6
 - %ER, 4-27, 12-5
 - %ERD, 7-8
 - Error processing, 4-25 to 4-29
 - BREAK 2 control of, 4-29

- crash
 - hard, A-5
 - recovery, A-5
 - report, A-3
 - soft, A-3
- default error processing, 4-25
- DMC11, 6-45
- error messages, A-6 to A-10
- error-processing routines, 4-25
 - to 4-26
- error-trapping routines, 4-27
- exiting from an error handler,
 - 4-28 to 4-29
- hardware errors, A-2
- I/O, 5-8
- line printers, 6-17
- magnetic tape, 6-31
- MUMPS programming errors, A-1
- nested contexts, 4-27 to 4-28
- RX02, 6-65
- SDP, 6-36
- spooling, 6-52
- system failure, A-2
- terminals, 6-15
- TU58, 6-70
- types of errors, A-1
- %ET, 4-27, 12-4
- Executive
 - see operating system
- Extended Global Directory utility
 - see %EGD
- Extended global references, 8-6
 - to 8-9
- Extended Routine Directory
 - utility
 - see %ERD

- FASTDBT, 12-10
- %FIND, 7-5
- First Line utility
 - see %FL
- FIX, 12-10, 12-14
- %FL, 7-7
- Floppy disks
 - see RX02
- FOR, 9-2
- Formatting
 - disk during installation, 1-22, 1-24
- Formatting Characters, 5-6, 6-17

- Function keys, 3-4

- %G, 7-5
- Garbage Collector, 11-7, 13-6, 13-13, 13-19
- GC, 7-4
- %GD, 4-24, 7-4
- %GE, 7-4
- %GEDIT, 4-13
- Get UCI utility
 - see %GUCI
- Globals, 8-1 to 8-9
 - for system global see SYS
 - access to other UCIs, 8-6, 8-7, 8-9
 - access to other volume sets, 8-8
 - arrays, 8-1
 - ASCII sequence, 8-3
 - backup, 11-4
 - balanced tree, 8-4
 - 7-bit, H-1
 - 8-bit, H-1
 - characteristics, 13-11
 - characteristics in SYSGEN, 10-13, 10-25
 - circumflex, 8-2
 - collating sequences, 8-2
 - concepts, 8-1 to 8-6
 - Copy utility
 - see GC
 - data base supervisor, 8-1
 - data blocks, 13-15
 - data records, 8-5
 - data structures, 8-4 to 8-5
 - DDP, 8-8
 - directories, 4-23, 14-11, 15-3
 - directory block, 13-10
 - Directory utility
 - see %GD
 - extended directory
 - see %EGD
 - editor
 - see %GEDIT
 - Efficiency utility
 - see %GE
 - extended global references, 8-6
 - to 8-9, 17-11
 - global module, 8-1, 13-21
 - grouping accesses, 9-2

growth and development, 13-17
 to 13-20
 initialization, 13-17
 journaling, 16-5
 keys, 13-13, 13-14
 layout of directory block,
 13-10
 levels, 13-19
 library utilities, 2-6, 7-3 to
 7-6
 List utility
 see %G
 Management utility
 see %GLOMAN
 numeric sequence, 8-2
 optimization, 13-20
 pointer blocks, 13-13
 programming strategy, 8-5 to
 8-6
 protection categories, 4-24,
 13-12, 17-14
 Restore utility
 see %GTI
 Save utility
 see %GTO
 Selector utility
 see %GSEL
 sparse arrays, 8-3
 structure and optimization,
 13-1 to 13-27
 Subscript Filter utility
 see %FIND
 subscripts, 8-2
 internal format, H-1
 tree structure, 8-4
 up arrow, 8-2
 using UCI Translation Table,
 8-9
 variables, 2-2, 8-2 to 8-3
 Version 1 Restore utility
 see %GR
 Version 1 Save utility
 see %GS
 %GLOMAN, 4-24, 7-4, 8-3, 10-13,
 13-26, 16-2, 16-5, 17-14
 GOTO, 4-14, 4-28, 9-2, 9-5
 %GR, 7-11
 %GS, 7-11
 %GSEL, 7-5
 %GTI, 7-5, 11-4
 %GTO, 7-5, 11-4
 %GUCI, 7-14
 %H, 7-12
 HALT, 3-10, 4-14, 4-15, 5-4, A-5
 %HDR, 7-11
 Header Formatter utility
 see %HDR
 HELP, 12-15
 Help Text Driver utility
 see HELP
 \$HOROLOG, 16-1

 I/O
 assigning devices, 5-3
 commands, 5-4
 device characteristics, 6-1 to
 6-71
 Device Selector utility
 see %IOS
 device specifiers, 5-2
 Device Table, 5-2
 error processing, 5-8
 introduction, 5-1
 Modes, 3-8
 I/O Monitor
 see operating system
 IC, 12-10, 12-14
 In-memory Job Communications
 see JOBCOM
 Indirect Mode, 3-8
 Indirection
 optimizing, 9-8
 Input
 see I/O
 Input/Output
 see I/O
 Installation
 autoconfiguration, 1-4, 1-5,
 1-8, 1-25, 1-26, 1-31
 bad blocks, 1-7, 1-23
 Baseline System, 1-21, 1-30
 booting the system, 1-5, 1-14
 to 1-18, 1-21
 caretaker job, 1-31
 checklist, 1-5 to 1-8
 customization, 1-30
 default answers, 1-21
 define start-up command file,
 1-30

- dialogue, 1-21 to 1-31
- formatting disks, 1-22, 1-24
- help, 1-21, 1-25
- MICRO/PDP-11, 1-19 to 1-20
- mounting media, 1-14
- of DSM-11, 1-1 to 1-31
- overview, 1-4 to 1-5
- PAC, 1-8, 1-29
- PDP-11 hardware, 1-1
- power line frequency, 1-8, 1-29
- power on console, 1-9
- power on PDP-11, 1-10 to 1-13
- standard software options, 1-27
- start-up command file, 1-30
- starting the system, 1-5, 1-14
 - to 1-18, 1-21
- STU, 1-30
- STUBLD, 1-30
- SYSGEN, 1-25
- system generation, 1-25
- system global, 1-29
- upgrade of DSM-11, 1-1

Interactive routines, 9-11

Interpreter

- see operating system

\$IO, 5-3, 6-12

%IOS, 7-11

JOB, 6-16, 14-5

Job Communications

- see JOBCOM

Job Monitor utility

- see %JOB utility

\$JOB special variable, 14-6

Job Table, 14-5 to 14-6

- utility
 - see JOBTAB

%JOB utility, 12-11

JOBCOM, 6-38 to 6-39, 12-4

- commands, 6-39
- device numbers, 6-38
- READ, 6-39
- receivers, 6-38
- SYSGEN, 10-6, 10-8, 10-10, 10-23, 10-24
- transmitters, 6-38
- WRITE, 6-39
- ZLOAD, 6-39
- ZPRINT, 6-39

Jobs

- see also Job Table
- definition of, 14-5

JOBTAB, 12-12

Journaling, 11-4, 13-3, 15-13 to 16-6

- clustered systems, 17-13
- DDP, 17-13
- dejournal, 16-6
- disks, 16-5
- globals, 16-5
- magnetic tape, 16-6
- SDP access to journal space, 6-33, 6-35, 6-37
- start-up, 11-2
- SYSGEN, 10-6, 10-7, 10-22
- utilities, 16-2 to 16-5

JRNALL, 6-33, 16-4

JRNDEALL, 16-4

JRNINIT, 16-4

JRNLSHOW, 16-3

JRNRECOV, 16-2, 16-4

JRNSTART, 16-3

JRNSTOP, 16-3

KILL, 16-1

KTR, 12-4

LA120

- power on, 1-9

LABEL, 12-7

Language Interpreter

- see operating system

Library utilities, 7-1 to 7-15

- conventions, 7-3
- globals, 7-3 to 7-6
- menu, 7-2
- miscellaneous utilities, 7-9 to 7-12
- other miscellaneous utilities, 7-12 to 7-15
- routines, 7-6 to 7-8
- running, 7-2
- types of, 7-1

Library utility routines, 2-5

Line Map Report utility

- see LMAP

Line printers, 6-16 to 6-17

- queing, see spooling
- commands, 6-16
- error conditions, 6-17
- format control, 6-16

- formatting characters, 6-17
- \$ZA, 6-17
- Lines
 - command lines, 4-4
 - entering routine lines, 4-5
 - line structure, 4-3 to 4-4
 - routine lines, 4-4
- List First Line utility
 - see %FL
- List MENU Global utility
 - see %MENLIS
- LMAP, 12-11
- LOAD, 6-61, 6-66, 10-9, 12-16
- Load a Driver utility
 - LOAD
- Load DSM-11 Editor utility
 - see %LOAD%ED
- %LOAD%ED, 7-10
- Loadable drivers
 - start-up, 11-2
 - SYSGEN, 10-7, 10-8, 10-22, 10-23
- Local variables, 2-2
- LOCK, 10-11, 12-12, 14-6
- Lock Table, 14-6 to 14-7
 - SYSGEN, 10-11, 10-24, 14-7
 - utility
 - see LOCKTAB
- LOCKTAB, 12-12, 14-7
- Login, 3-7 to 3-10
 - see also MUX
 - access parameters, 3-7
 - Application Mode, 3-9
 - autobauded, 3-10
 - echo, 10-14, 10-26
 - modem with autobauding, 3-10
 - modes, 3-8
 - PAC, 3-8
 - partition size, 3-8
 - Programmer Mode, 3-9
 - tied terminals, 3-10
 - UCI, 3-7
 - volume sets, 17-5
- Logout, 3-10 to 3-11
 - Application Mode, 3-11
 - Programmer Mode, 3-10
- Lowercase characters, 4-2, 10-31
- LP11, 6-16
 - SYSGEN, 10-19, 10-22
- Magnetic tape, 6-17 to 6-33
 - CLOSE, 6-22
 - commands, 6-18
 - Continuous Mode, 6-30
 - Copy utility
 - see TAPECOPY
 - data formats, 6-28
 - device numbers, 6-18
 - DOS-11 label, 6-27
 - error conditions, 6-31
 - fixed-length data format, 6-29
 - journaling, 16-6
 - labeling, 6-27
 - Magtape Default Mode utility
 - see MMD
 - OPEN parameters, 6-18 to 6-22
 - operations, 6-24
 - READ, 6-24
 - standard EBCDIC label, 6-27
 - Status Check utility
 - see %MTCHK
 - stream data format, 6-28
 - streaming, 6-30
 - SYSGEN, 10-18
 - disk-tape buffers, 10-10, 10-24
 - unlabeled, 6-28
 - USE, 6-22
 - USE and B switch, 6-26
 - variable-length record format, 6-29
 - VIEW buffer, 6-26
 - VIEW buffer and streaming, 6-30
 - WRITE, 6-22
 - WRITE *, 6-22
 - \$ZA, 6-24, 6-31, 6-32
 - \$ZA status bit assignments, 6-24
 - \$ZB, 6-25
 - ZLOAD, 6-24
 - ZPRINT, 6-22
 - \$ZTRAP, 6-32
- Magtape Default Mode utility
 - see MMD
- Manuals
 - MICRO/PDP-11, 1-3
 - PDP-11, 1-3
- MBP, 10-2, 10-12 to 10-16, 10-25
 - application interrupt key, 10-15, 10-26
 - assigning PAC and UCI, 2-3

line frequency, 10-16, 10-26
 log-in sequence echo, 10-14, 10-26
 PAC, 10-16, 10-26
 programmer abort key, 10-15, 10-26
 significant digits in division computations, 10-15, 10-26
 system restart after power failure, 10-15, 10-26
 telephone disconnect delay, 10-15, 10-26
 VIEW protection, 10-14, 10-26
 ZUSE, 10-14, 10-26
 %MENLIS, 7-10
 %MENU, 7-12
 Menu Manager utility
 see %MENU
 MICRO/PDP-11
 installation of DSM-11, 1-19 to 1-20
 manuals, 1-3
 Miscellaneous utilities, 7-9 to 7-12
 Miscellaneous utilities (other), 7-12 to 7-15
 MMD, 6-17, 10-2, 10-34 to 10-35
 Modem Autodialer utility
 see %DIAL
 Modems
 see also MUX
 Modes, 3-8 to 3-9
 Modify Basic Parameters utility
 see MBP
 MOUNT, 8-8, 12-7
 %MTCHK, 6-32, 7-14
 Multiplexers
 DH11, 6-1
 DZ11, 6-1
 line parameter register, 6-8
 MUMPS
 see also DSM-11
 MUX, 10-2, 10-29 to 10-32
 autobaud, 10-29
 CRT, 10-29
 device number, 10-29
 edit comment, 10-32
 example, 10-32
 login, 10-30
 lowercase, 10-31
 modem control, 10-30
 output margin, 10-31
 output only, 10-30
 parity, 10-29
 receiver speed, 10-30
 routine numbers, 10-31
 stall count, 10-30
 tab control, 10-31
 transmitter speed, 10-30
 ZUSE, 10-30

 Octal
 Decimal/Octal Conversion utility
 see %DOC utility
 to Decimal Conversion utility
 see %OD utility
 using # with octal numbers, 7-3
 %OD, 7-13
 OPEN, 5-3, 5-4, 6-3, 6-15, 6-18, 6-34, 6-35, 6-37, 6-43, 6-48, 6-58, 6-61, 6-64, 6-67
 Operating system, 2-4
 data base supervisor, 2-4, 8-1
 executive, 2-4
 I/O monitor, 2-4
 language interpreter, 2-4
 system supervisor, 2-4
 Operations Control Register, F-1 to F-3
 Output
 see I/O

 PAC, 3-7, 3-8, 10-16, 10-26
 Baseline System, 1-3, 1-30
 description of, 2-3
 installation, 1-8, 1-29
 PARTAB, 12-12
 Partition Table, 14-8 to 14-9
 utility
 see PARTAB
 Partition Vector, 14-17 to 14-20
 utility
 see PARVEC
 Partitions
 see also Partition Vector
 see also Partition Table
 description of, 2-2
 partition size at login, 3-8
 SYSGEN, 10-12, 10-24

PARVEC, 12-13, 14-20
 Patches, 11-4 to 11-7
 applied to memory, 11-2
 system, 11-5
 utility, 11-5
 see AUPAT
 Pattern matching
 optimizing, 9-7
 PDP-11, 2-6
 manuals, 1-3
 power on, 1-10 to 1-13
 PEEK, 12-14
 Peek utility
 see PEEK
 Percent character, 9-11
 Percent editor
 see DSM-11 editor under editors
 Performance statistics, 15-4 to 15-13
 utility
 see RTHIST
 Power failure
 system restart, 10-15, 10-26
 Print Disk Error Summary
 see DSKTRACK
 Printers
 queuing, see spooling
 see line printers
 Programmer abort key, 10-15, 10-26
 Programmer Access Code
 see PAC
 Programmer Mode, 2-3, 4-2
 definition of, 2-2
 description of, 3-8
 Direct Mode, 4-4, 4-5
 login, 3-8, 3-9
 logout, 3-10
 prompt, 4-5
 routine execution, 4-13
 stopping routine execution, 4-15
 Programming
 bit-masking, G-1 to G-2
 conventions, 9-9 to 9-14
 globals, 8-5 to 8-6
 interactive routines, 9-11
 Protection categories
 see Globals
 protection categories
 QUIT, 4-14, 4-15, 4-28

 %RCE, 7-8
 %RCMP, 7-6
 %RCOPY, 7-7
 %RD, 4-7, 4-23, 7-7
 READ, 5-4, 6-12, 6-24, 6-35, 6-39, 9-7
 REST, 11-1, 11-4, 12-6, 12-7
 Restore Jobs/Devices utility
 see RJD
 Ring buffers
 input, 6-3
 output, 6-3
 space unavailable for terminals, 6-15
 RJD, 12-14
 RMAP, 10-3, 10-9, 10-35 to 10-39, 15-2
 RMBLD, 10-36, 10-38, 15-2
 RMDIS, 10-36, 10-37
 RMLOAD, 10-36, 10-38, 15-2
 RMSHO, 10-36, 10-37
 Routine communications
 see routine interlocks
 Routine directories, 4-7, 4-23
 Routine interlocks, 6-57
 commands, 6-57
 device numbers, 6-57
 Routine lines
 definition of, 4-4
 Routine Mapping utility
 see RMAP
 Routines
 see also system utilities
 backup, 11-4
 carriage return, 7-3, 12-2
 Change Every utility
 see %RCE
 communications
 see routine interlocks
 Compare utility
 see %RCMP
 Copy utility
 see %RCOPY
 Cross Reference utility
 see %CRF
 default answers, 7-3

- definition of, 4-4
- deleting and renaming, 4-8 to 4-9
- directories, 4-23
- Directory utility
 - see %RD
 - extended directory
 - see %ERD
- entering routine lines, 4-5
- entering routines, 4-2 to 4-4
- exiting, 7-3
- file structure, B-1 to B-4
- help, 7-3
- Indirect Mode, 4-5
- interactive
 - conventions, 9-11
- library utilities, 2-5, 7-6 to 7-8
- List First Line utility
 - see %FL
- loading, 4-6 to 4-8
- loading from a sequential file, 4-10
- mapped, 10-6, 10-9, 10-12, 10-23, 10-24, 15-1 to 15-2
 - start-up, 11-3
- number for tied terminals, 10-31
- optimizing, 9-1 to 9-9
 - abbreviating commands, 9-6
 - calling by line labels, 9-5
 - direct, top-down execution path, 9-2
 - DO, 9-2
 - GOTO, 9-2
 - indirection, 9-8
 - pattern matching, 9-7
 - short line labels, 9-6
 - short variable names, 9-6
 - XECUTE, 9-8
- question mark, 12-2
- Restore utility
 - see %RR
- return to previous question, 7-3
- routine directories, 4-7
- routine line structure, 4-4
- Routine Mapping utility
 - see RMAP
- Save utility
 - see %RS
- saving, 4-6 to 4-8
- Search utility
 - see %RSE
- Selector utility
 - see %RSEL
- starting and stopping, 4-13 to 4-17
- storing in sequential files, 4-9
- Summaries utility
 - see %SUM
- system, 2-5
- up arrow, 12-2
- writing into a sequential file, 4-10

- %RR, 4-9, 7-7, 11-4
- %RS, 4-9, 7-7, 11-4
- %RSE, 7-8
- %RSEL, 7-8
- RTHIST, 12-11, 15-1, 15-4 to 15-13
- RX02, 6-61 to 6-66, 11-3
 - commands, 6-61
 - Control Mode, 6-64
 - error codes, 6-65
 - Mixed Mode, 6-63
 - OPEN and USE examples, 6-64
 - OPEN and USE parameters, 6-61 to 6-63
 - Read Mode, 6-63
 - SYSGEN, 10-4, 10-5, 10-7, 10-9, 10-20, 10-22
 - View Buffer Mode, 6-64
 - Write Mode, 6-63
 - \$ZB, 6-66

- SDP, 4-9, 4-10, 6-33 to 6-37, 13-3
 - access to journal space, 6-33, 6-37
 - OPEN, 6-35
 - CLOSE, 6-35
 - commands, 6-34
 - device numbers, 6-34
 - error conditions, 6-36
 - OPEN, 6-37
 - journal space, 6-35
 - OPEN and USE parameters, 6-34 to 6-35
 - READ, 6-35

- special characteristics, 6-36
- SYSGEN, 10-6, 10-7, 10-22
- USE, 6-37
- WRITE, 6-35
- \$ZA, 6-36
- \$ZB, 6-36
- ZLOAD, 6-35
- ZPRINT, 6-35
- ZWRITE, 6-35
- SDP utility, 6-33, 12-7
- Sequential Disk Processor
 - see SDP
- SET, 8-2, 16-1
- Set Date utility
 - see DAT
- Set Time utility
 - see TIM
- Show Current Date utility
 - see %D
- Show Current Time utility
 - see %T
- Shutdown utility
 - see SSD
- Software Error Log utility
 - see %ER
- Sparse arrays
 - see globals
- SPL, 6-47, 6-51, 12-8
- SPLALL, 12-8
- SPLINI, 12-9
- SPLREM, 12-8
- SPLSTR, 12-9
- SPOFSPL, 12-9
- SPONSPL, 12-9
- Spooling, 6-46 to 6-56, 10-23, 13-3
 - Allocate Spool Space utility
 - see SPLALL
 - CLOSE, 6-49
 - Deallocate Spool Space utility
 - see SPLREM
 - DESPOOL, 6-47
 - despooling, 6-46, 6-48
 - device commands, 6-48
 - device numbers, 6-47
 - Display Spool File Structure utility
 - see SPLSTR
 - error conditions, 6-52
 - explicit, 6-47, 6-48, 6-51
 - Initialize Spool Space utility
 - see SPLINI
 - OPEN, 6-48
 - spool file structure, 6-52 to 6-56
 - Start Despooler Lister utility
 - see DSONSPL
 - Start Spooler utility
 - see SPONSPL
 - start-up, 11-2
 - Stop Despooler utility
 - see DSOFSP
 - Stop Spooler utility
 - see SPOFSPL
 - transparent, 6-47, 6-48, 6-50
 - USE, 6-49
 - utilities, 12-8 to 12-9
 - see SPL
 - \$ZA, 6-49, 6-52
- SSD, 11-1, 11-7, 12-15
- STA, 10-32, 12-11
- Start-up command file, 1-30, 11-2
 - see also STU and STUBLD
 - define in installation, 1-30
- Start-up utilities
 - see STU and STUBLD
- Storage Allocation Table, 14-10, 14-12 to 14-13
 - description of, 13-2
- Streaming
 - magnetic tape, 6-30
- String data, 4-3
- String literals, 4-3
- %STRTAB, 12-13, 14-10
- STU, 1-30, 6-51, 8-8, 11-1, 11-1 to 11-3, 12-4, 12-5, 15-1
- STUBLD, 1-30, 11-1, 11-1 to 11-3, 12-4, 12-5, 15-2
- Subscript Filter utility
 - see %FIND
- %SUM, 7-8
- Switch Register utility
 - see SWREG
- SWREG, 12-11
- SYS global, 10-27, 12-4
 - description of, 2-6
 - installation, 1-29
- SYS utility, 12-2
- SYSDEF, 10-29 to 10-41
 - customization, 1-30
 - in installation, 1-30, 1-31
 - list of utilities, 10-2

SYSGEN, 10-1 to 10-27
 application interrupt key,
 10-15, 10-26
 assign device numbers, 10-7,
 10-21
 autoconfiguration, 10-1, 10-2
 example, 10-27 to 10-29
 basic system parameters
 see MBP
 carriage return, 10-16
 clustered systems, 17-5 to 17-8
 concept, 10-1
 configuration, 10-4, 10-17
 CSR addresses, 10-1
 DDP, 17-5 to 17-8
 default answers, 10-16
 definition of, 1-25, 10-1
 device characteristics
 see MUX
 disk information, 10-4, 10-18
 disk-tape buffers, 10-10, 10-22,
 10-24
 example, 10-16 to 10-27
 global characteristics, 10-25
 help, 10-16
 installation, 1-25
 journaling, 10-22
 magnetic tape, 10-18
 manual configuration, 10-1
 mapped routines
 see routines
 memory buffer allocation, 10-10
 modify system parameters
 see SYSDEF
 partition size and login, 3-8
 partitions, 10-12, 10-24
 parts, 10-3
 procedure, 10-3
 programmer abort key, 10-15,
 10-26
 question mark, 10-16
 ring buffers, 10-10, 10-23
 routines
 mapped
 see routines
 session, 10-3 to 10-16
 software
 configuration, 10-6, 10-21
 options, 10-7, 10-22
 standard options, 10-21
 standard system, 10-6

SYSDEF utilities, 10-2
 see also SYSDEF
 system data structures, 10-11
 to 10-12, 10-24, 17-8
 system devices, 10-5, 10-18,
 17-6
 terminal device characteristics
 see MUX
 up arrow, 10-3, 10-16
 Vector addresses, 10-1
 volume sets, 10-10, 10-11,
 10-23, 10-24, 17-8

SYSTAB, 12-12, 14-5

System Definition
 see SYSDEF

System generation
 see SYSGEN

System management
 See also journaling
 see also SYSDEF
 see also SYSGEN
 see also system utilities
 buffer sizes, 15-2 to 15-3
 data access, 2-2
 global structure and
 optimization, 13-1
 limiting data areas, 15-3 to
 15-4
 operating and maintaining
 DSM-11, 11-1
 operations control register,
 F-1 to F-3
 optimizing DSM-11, 15-1 to
 15-13
 performance statistics, 15-4 to
 15-13
 preserving system integrity,
 11-3 to 11-4
 system access, 2-3
 system capacity, 2-2
 system hardware, 2-6
 system linkages, 14-3
 system operation, 1-3
 system overview, 2-1
 system reports, 12-10
 system software, 2-3 to 2-6
 system status, 12-11
 table utilities, 12-12
 tables, 14-1 to 14-20

System patches
 see patches

- System Shutdown utility
 - see SSD
- System Start-up utility
 - see STU
- System Status utility
 - see STA
- System Table, 14-4 to 14-5, 14-7, 14-10, 14-14
 - utility
 - see SYSTAB
- System tables, 14-1 to 14-20
- System utilities, 2-5, 12-1 to 12-16
 - running, 12-2
 - stopping, 12-3
- System-Configuration global
 - see SYS

- %T, 7-10
- Tables
 - system
 - see system tables
- Tape
 - see magnetic tape
- TAPECOPY, 12-16
- Terminal DDB utility
 - see DDR
- Terminal Device Characteristics utility
 - see MUX
- Terminals, 6-1 to 6-16
 - autobauded login, 3-10
 - commands, 6-2
 - description of, 3-1
 - device characteristics, 10-29 to 10-32
 - see also MUX
 - device numbers, 6-2
 - Device Status Word, 6-4
 - error conditions, 6-15
 - escape processing, 6-12 to 6-15
 - JOB, 6-16
 - nonautobauded login, 3-7
 - OPEN, 6-3, 6-15
 - OPEN parameters, 6-3 to 6-12
 - READ, 6-12
 - tied
 - see tied terminals
 - see TTTG
 - tied terminal login, 3-10
 - types of, 3-1
 - USE, 6-8, 6-12, 6-15
 - \$ZA, 6-12
 - \$ZB, 6-11, 6-12, 6-13, 6-14
 - \$ZTRAP, 6-16
 - ZUSE, 6-8, 6-15, 6-16
- \$TEST, 6-11, 6-21
- Tied terminals
 - disabling of control C, 4-16
 - login, 3-10
 - routine number in MUX, 10-31
 - show table utility
 - see TTTSHO
 - utility
 - see TTTG
- Tied terminmls
 - description of, 2-3
- TIM, 12-15
- Time
 - Date and Time utility
 - see %H
 - Show Current Time utility
 - see %T
- Translation Table utility
 - see UCITRAN
- Tree structure
 - see globals
- TTTG, 10-3, 10-32, 10-39 to 10-41
- TTTSHO, 10-40
- TU58, 6-66 to 6-71, 11-3
 - commands, 6-66
 - Control Mode, 6-69
 - error codes, 6-70
 - Mixed Mode, 6-68
 - OPEN and USE examples, 6-69
 - OPEN and USE parameters, 6-67 to 6-68
 - Read Mode, 6-68
 - SYSGEN, 10-7, 10-9, 10-20, 10-22
 - View Buffer Mode, 6-69
 - Write Mode, 6-69
 - \$ZA, 6-70
 - \$ZB, 6-71

- UCI, 3-7, 3-7
 - see also UCI Table
 - Add UCIs utility
 - see UCIADD
 - description of, 2-3

- Get UCI utility
 - see %GUCI
- library, 10-34, 14-12, 17-15
- List utility
 - see UCILI
- Name Change utility
 - see UCIEDIT
- placing on data base, 13-22
- Translation Table, 10-6, 10-9, 10-23, 15-4, 17-8, 17-11 to 17-13
 - see also UCITRAN
- Translation Table utility
 - see UCITRAN
- UCI List utility
 - see UCILI
- UCI Name Change utility
 - see UCIEDIT
- UCI Table, 13-3, 14-10, 14-10 to 14-12
 - description of, 13-2
 - utility
 - see UCITAB
- UCI Translation Table utility
 - see UCITRAN
- UCIADD, 10-2, 10-33, 13-22
- UCIEDIT, 10-2, 10-34, 17-15
- UCILI, 10-2, 10-34
- UCITAB, 12-13
- UCITRAN, 10-3, 10-10, 10-41, 17-11 to 17-13
- UNLOAD, 6-61, 6-66, 10-9, 12-16
- Unload a Driver utility
 - UNLOAD
- Upgrade
 - DSM-11, 1-1
- Uppercase characters, 4-2
- USE, 5-3, 5-4, 6-8, 6-12, 6-15, 6-22, 6-26, 6-34, 6-37, 6-43, 6-49, 6-60, 6-61, 6-64, 6-67
- Utilities
 - global
 - see globals
 - library
 - see library utilities
 - miscellaneous library
 - see miscellaneous utilities
 - other miscellaneous library
 - see miscellaneous utilities (other)
 - routine

- see routines
- system
 - see system utilities
- Utility patches
 - see patches
- UTL, 12-2
- Variables
 - see also global variables
 - see also local variables
 - naming conventions, 9-11
- Version 1 Global Restore utility
 - see %GR
- Version 1 Global Save utility
 - see %GS
- VIEW, 6-26, 6-30, 6-43, 6-60, 10-14, 10-26, 12-3, 13-3, 14-4, A-5
- VIEW device, 6-57 to 6-61
 - OPEN parameters, 6-58 to 6-60
 - USE parameters, 6-60
 - \$VIEW, 6-60
 - VIEW, 6-60
- \$VIEW function, 6-60, 14-4
- Volume Set Table, 14-9 to 14-10, 14-12
 - description of, 13-2
 - utility
 - see %STRTAB
- Volume sets, 13-2 to 13-4
 - see also Volume Set Table
 - access to globals in different, 8-8
 - clustered systems, 17-1 to 17-15
 - explanation of, 13-2
 - SYSGEN, 10-10, 10-11, 10-23, 10-24, 17-8
- VT100, 3-2, 6-1
 - escape keys, 6-14
 - keyboard, 3-3
 - power on, 1-9
- VT52, 3-2, 6-1
 - escape keys, 6-13, 6-14
 - keyboard, 3-3
- WRITE, 5-4, 5-5, 6-16, 6-22, 6-35, 6-39
- WRITE *, 6-22, 6-42, 6-44

\$X, 5-7, 5-8, 6-3, 6-8, 6-50
 XDT, 3-6, A-2, A-3
 SYSGEN, 10-6, 10-9, 10-23
 XECUTE, 9-2, 9-8

 \$Y, 5-7, 5-8, 6-8, 6-50

 \$ZA, 5-8, 6-12, 6-17, 6-24, 6-31,
 6-32, 6-36, 6-43, 6-45, 6-46,
 6-49, 6-52, 6-70
 ZALLOCATE, 10-11, 12-12, 14-6
 \$ZB, 6-11, 6-12, 6-13, 6-14, 6-25,
 6-36, 6-66, 6-71
 \$ZBREAK, 4-17, 4-18, 4-19, 4-20,
 4-21

 ZBREAK, 3-6, 4-21
 \$ZCALL, E-1 to E-4
 \$ZERROR, 4-25, 4-26
 ZGO, 4-18, 4-21
 ZINSERT, 4-11
 ZLOAD, 4-8, 5-4, 5-5, 6-24, 6-35,
 6-39, 15-2
 ZPRINT, 4-6, 5-4, 5-5, 6-22, 6-35,
 6-39
 ZQUIT, 4-14, 4-15, 4-28
 \$ZR, 17-15
 ZREMOVE, 4-6, 4-8, 4-11
 ZSAVE, 4-7
 \$ZTRAP, 4-25, 4-27, 4-29, 5-8,
 6-16, 6-32, 9-4, 12-4, A-1
 ZUSE, 6-8, 6-15, 6-16, 10-14,
 10-26, 10-30
 ZWRITE, 5-4, 5-5, 6-35

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

- Assembly language programmer
- Higher-level language programmer
- Occasional programmer (experienced)
- User with little programming experience
- Student programmer
- Other (please specify) _____

Name _____ Date _____

Organization _____ Telephone _____

Street _____

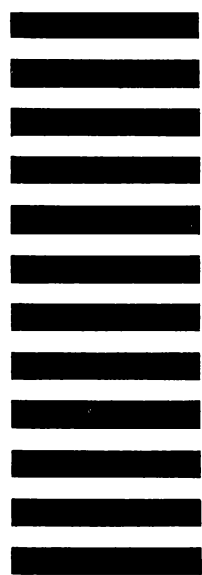
City _____ State _____ Zip Code _____
or Country

----- 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

SOFTWARE PUBLICATIONS
200 FOREST STREET MRO1-2/L12
MARLBOROUGH, MA 01752

----- Do Not Tear -- Fold Here and Tape -----