Virtual Machine/
System Product

CMS Command Reference

Release 5
SC19-6209-4
Fifth Edition (December 1986)

This edition, SC19-6209-4, is a major revision of SC19-6209-3 and applies to Release 5 of the Virtual Machine/System Product (VM/SP), program number 5664-167, and to all subsequent releases and modifications until otherwise indicated in new editions or Technical Newsletters. Changes are made periodically to the information contained herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/370, 30xx, and 4300 Processors Bibliography, GC20-0001, for the editions that are applicable and current.

Summary of Changes

For a list of changes, see page 873.

Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM’s licensed program may be used. Any functionally equivalent program may be used instead.

Ordering Publications

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers’ comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Dept. G60, PO Box 6, Endicott, NY, U.S.A. 13760. IBM may use or distribute whatever information you supply in anyway it believes appropriate without incurring any obligation to you.

The form for reader’s comments provided at the back of this publication may also be used to comment on the VM/SP online HELP facility.

Use this publication as a reference manual. It contains all of the command formats, syntax rules, and operand and option descriptions for CMS commands and subcommands for general users.

If you are using Virtual Machine/System Product (VM/SP) for the first time, refer to the *VM/SP CMS Primer*, SC24-5236, for introductory tutorial information about VM/SP. If you are using a line-oriented display terminal, then refer to the *VM/SP CMS Primer for Line-Oriented Terminals*, SC24-5242. The *VM/SP CMS User's Guide*, SC19-6210, contains tutorial information and functional descriptions of CMS commands, as well as information on using the editor, EXEC, and debugging facilities of CMS. You should be familiar with the contents of the *VM/SP CMS User's Guide* before you attempt to use this reference manual. For most of the CMS commands described in this publication, you may find additional useful notes in the *VM/SP CMS User's Guide*.

This publication has four chapters:

“Chapter 1. Introduction and General Concepts” describes the components of the VM/SP system and tells you how to enter CMS commands. It lists the notational conventions used in this manual, so that you can interpret the command format descriptions in the following chapters. Chapter 1 also contains information about the CMS command search order and a summary of all the CMS commands available under VM/SP, including those not for general users.

“Chapter 2. CMS Commands” contains complete format descriptions, and operand and option lists, for the CMS commands available to general users. Each command description contains usage notes and lists responses and error messages (with associated return codes) produced by the command.

“Chapter 3. HELP and HELP_CONV Format Words” describes the formats, operands, and defaults of the HELP facility format words. HELP format words are used in HELP description files when the user wants HELP to format output when the HELP file is processed.

“Chapter 4. DEBUG Subcommands” describes the subcommands available in the debug environment of CMS. Each subcommand description contains usage notes and, where applicable, lists the responses to the subcommand.

The chapters that formerly contained the “CMS Functions” and the “CMS Macro Instructions” have been moved to a new publication, the *VM/SP CMS Macros and Functions Reference*, SC24-5284. Refer to that manual for information on functions that are available and for CMS assembler macro instructions you can use when you write programs to execute in CMS.
This publication also has five appendixes:

“Appendix A: VSE/VSAM Functions Not Supported in CMS” lists the restrictions on the use of access method services and VSAM in the CMS/DOS environment of CMS.

“Appendix B: OS/VS Access Method Services and VSAM Functions Not Supported in CMS” lists the restrictions for OS programmers using access method services and VSAM in CMS.

“Appendix C. EDIT Subcommands and Macros” describes the subcommands and macros available in the environment of the CMS editor, which you can invoke using the EDIT command with the OLD option. Each subcommand description contains usage notes and summarizes the types of responses you might receive. Where applicable, additional information is provided for users of display terminals.

“Appendix D: Reserved Filetype Defaults” lists the filetypes that are recognized by the CMS editor and indicates the default settings that the editor supplies for logical tabs, truncation, verification, logical record length, and so on.

“Appendix E. CMS EXEC Control Statements” describes the control statements, special variables, and built-in functions you can use when you create EXEC procedures to execute in CMS. The control statement descriptions contain usage notes, where applicable.
## Contents

### Chapter 1. Introduction and General Concepts
- CMS Environment ........................................ 2
- Entering CMS Commands .................................. 3
- Character Set Usage ...................................... 4
- Naming CMS Files ......................................... 4
- Notational Conventions ................................... 5
- CMS Command Search Order .............................. 7
- CMS Command Execution Characteristics ............. 9
- CMS Command Summary .................................. 10
- CMS Command Summary .................................. 13
- CMS Commands for System Programmers .......... 19

### Chapter 2. CMS Commands
- Command Syntax Error Messages ....................... 24
- ACCESS ................................................ 27
- ALARM VSCREEN ........................................ 33
- AMSERV ................................................ 34
- ASSEMBLE .............................................. 38
- ASSGN .................................................. 47
- CATCHECK .............................................. 51
- CLEAR VSCREEN ....................................... 53
- CLEAR WINDOW ......................................... 54
- CMDCALL ............................................... 56
- CMSBATCH .............................................. 57
- CMSSERV ............................................... 59
- COMPARE .............................................. 61
- CONVERT COMMANDS ................................... 63
- CONWAIT ............................................... 67
- COPYFILE ............................................. 68
- CP ...................................................... 82
- CURSOR VSCREEN ...................................... 84
- DEBUG ................................................ 88
- DEFAULTS ............................................. 89
- DEFINE VSCREEN ....................................... 92
- DEFINE WINDOW ....................................... 97
- DELETE VSCREEN ...................................... 101
- DELETE WINDOW ....................................... 102
- DESBUF ............................................... 103
- DISK .................................................. 104
- DLBL .................................................. 110
- DOSLIB ............................................... 126
- DOSLKED ............................................. 129
- DROPBUF ............................................. 134
- DROP WINDOW ......................................... 135
- DSERV ............................................... 137
- EDIT ................................................... 140
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERASE</td>
<td>144</td>
</tr>
<tr>
<td>ESERV</td>
<td>147</td>
</tr>
<tr>
<td>EXEC</td>
<td>150</td>
</tr>
<tr>
<td>EXECDROP</td>
<td>155</td>
</tr>
<tr>
<td>EXECIO</td>
<td>158</td>
</tr>
<tr>
<td>EXELOAD</td>
<td>176</td>
</tr>
<tr>
<td>EXECMAP</td>
<td>179</td>
</tr>
<tr>
<td>EXECOS</td>
<td>182</td>
</tr>
<tr>
<td>EXECSTAT</td>
<td>184</td>
</tr>
<tr>
<td>EXECUPDT</td>
<td>186</td>
</tr>
<tr>
<td>FETCH</td>
<td>190</td>
</tr>
<tr>
<td>FILEDEF</td>
<td>193</td>
</tr>
<tr>
<td>FILELIST</td>
<td>212</td>
</tr>
<tr>
<td>FINIS</td>
<td>224</td>
</tr>
<tr>
<td>FORMAT</td>
<td>225</td>
</tr>
<tr>
<td>GENDIRT</td>
<td>230</td>
</tr>
<tr>
<td>GENMOD</td>
<td>231</td>
</tr>
<tr>
<td>GENMSG</td>
<td>236</td>
</tr>
<tr>
<td>GET VSCREEN</td>
<td>241</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>243</td>
</tr>
<tr>
<td>GLOBALV</td>
<td>246</td>
</tr>
<tr>
<td>HELP</td>
<td>258</td>
</tr>
<tr>
<td>HELPCONV</td>
<td>269</td>
</tr>
<tr>
<td>HIDE WINDOW</td>
<td>271</td>
</tr>
<tr>
<td>IDENTIFY</td>
<td>273</td>
</tr>
<tr>
<td>IMMCMD</td>
<td>276</td>
</tr>
<tr>
<td>INCLUDE</td>
<td>278</td>
</tr>
<tr>
<td>LABELDEF</td>
<td>283</td>
</tr>
<tr>
<td>LISTDS</td>
<td>289</td>
</tr>
<tr>
<td>LISTFILE</td>
<td>295</td>
</tr>
<tr>
<td>LISTIO</td>
<td>303</td>
</tr>
<tr>
<td>LKED</td>
<td>306</td>
</tr>
<tr>
<td>LOAD</td>
<td>311</td>
</tr>
<tr>
<td>LOADLIB</td>
<td>323</td>
</tr>
<tr>
<td>LOADMOD</td>
<td>327</td>
</tr>
<tr>
<td>MACLIB</td>
<td>329</td>
</tr>
<tr>
<td>MACLIST</td>
<td>333</td>
</tr>
<tr>
<td>MAKEBUF</td>
<td>344</td>
</tr>
<tr>
<td>MAXIMIZE WINDOW</td>
<td>345</td>
</tr>
<tr>
<td>MINIMIZE WINDOW</td>
<td>347</td>
</tr>
<tr>
<td>MODMAP</td>
<td>348</td>
</tr>
<tr>
<td>MOREHELP</td>
<td>349</td>
</tr>
<tr>
<td>MOVEFILE</td>
<td>352</td>
</tr>
<tr>
<td>NAMEFIND</td>
<td>356</td>
</tr>
<tr>
<td>NAMES</td>
<td>364</td>
</tr>
<tr>
<td>NOTE</td>
<td>370</td>
</tr>
<tr>
<td>NUCXDROP</td>
<td>380</td>
</tr>
<tr>
<td>NUCXLOAD</td>
<td>382</td>
</tr>
<tr>
<td>NUCXMAP</td>
<td>387</td>
</tr>
<tr>
<td>OPTION</td>
<td>389</td>
</tr>
<tr>
<td>OSRUN</td>
<td>392</td>
</tr>
<tr>
<td>PARSECMD</td>
<td>393</td>
</tr>
<tr>
<td>PEEK</td>
<td>396</td>
</tr>
<tr>
<td>Command</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>SET RELPAGE</td>
<td>592</td>
</tr>
<tr>
<td>SET REMOTE</td>
<td>594</td>
</tr>
<tr>
<td>SET RESERVED</td>
<td>596</td>
</tr>
<tr>
<td>SET SYSNAME</td>
<td>598</td>
</tr>
<tr>
<td>SET TEXT</td>
<td>599</td>
</tr>
<tr>
<td>SET TRANSLATE</td>
<td>601</td>
</tr>
<tr>
<td>SET UPSI</td>
<td>604</td>
</tr>
<tr>
<td>SET VSCREEN</td>
<td>605</td>
</tr>
<tr>
<td>SET WINDOW</td>
<td>607</td>
</tr>
<tr>
<td>SET WMPF</td>
<td>609</td>
</tr>
<tr>
<td>SETPRT</td>
<td>612</td>
</tr>
<tr>
<td>SHOW WINDOW</td>
<td>615</td>
</tr>
<tr>
<td>SIZE WINDOW</td>
<td>617</td>
</tr>
<tr>
<td>SORT</td>
<td>619</td>
</tr>
<tr>
<td>SSERV</td>
<td>622</td>
</tr>
<tr>
<td>START</td>
<td>624</td>
</tr>
<tr>
<td>STATE/STATEW (ESTATE/ESTATEW)</td>
<td>627</td>
</tr>
<tr>
<td>SVCTRACE</td>
<td>630</td>
</tr>
<tr>
<td>SYNONYM</td>
<td>635</td>
</tr>
<tr>
<td>TAPE</td>
<td>641</td>
</tr>
<tr>
<td>TAPEMAC</td>
<td>650</td>
</tr>
<tr>
<td>TAPPDS</td>
<td>654</td>
</tr>
<tr>
<td>TELL</td>
<td>659</td>
</tr>
<tr>
<td>TXTLIB</td>
<td>661</td>
</tr>
<tr>
<td>TYPE</td>
<td>665</td>
</tr>
<tr>
<td>UPDATE</td>
<td>668</td>
</tr>
<tr>
<td>VALIDATE</td>
<td>682</td>
</tr>
<tr>
<td>WAITREAD VSCREEN</td>
<td>684</td>
</tr>
<tr>
<td>WAITT VSCREEN</td>
<td>688</td>
</tr>
<tr>
<td>WRITE VSCREEN</td>
<td>690</td>
</tr>
<tr>
<td>XEDIT</td>
<td>697</td>
</tr>
<tr>
<td>XMITMSG</td>
<td>704</td>
</tr>
<tr>
<td>Border Commands</td>
<td>711</td>
</tr>
<tr>
<td>B</td>
<td>713</td>
</tr>
<tr>
<td>C</td>
<td>714</td>
</tr>
<tr>
<td>D</td>
<td>715</td>
</tr>
<tr>
<td>F</td>
<td>716</td>
</tr>
<tr>
<td>H</td>
<td>717</td>
</tr>
<tr>
<td>L</td>
<td>718</td>
</tr>
<tr>
<td>M</td>
<td>719</td>
</tr>
<tr>
<td>N</td>
<td>720</td>
</tr>
<tr>
<td>O</td>
<td>721</td>
</tr>
<tr>
<td>P</td>
<td>722</td>
</tr>
<tr>
<td>R</td>
<td>723</td>
</tr>
<tr>
<td>S</td>
<td>724</td>
</tr>
<tr>
<td>X</td>
<td>725</td>
</tr>
<tr>
<td>Immediate Commands</td>
<td>726</td>
</tr>
<tr>
<td>HB</td>
<td>727</td>
</tr>
<tr>
<td>HI</td>
<td>728</td>
</tr>
<tr>
<td>HO</td>
<td>729</td>
</tr>
<tr>
<td>HT</td>
<td>730</td>
</tr>
<tr>
<td>HX</td>
<td>731</td>
</tr>
<tr>
<td>RO</td>
<td>732</td>
</tr>
</tbody>
</table>
Chapter 3. HELP and HELPCONV Format Words ........... 737
.BX (BOX) ........................................ 739
.CM (COMMENT) .................................. 741
.CS (CONDITIONAL SECTION) .......................... 742
.FO (FORMAT MODE) ................................ 744
.IL (INDENT LINE) .................................. 746
.IN (INDENT) ....................................... 747
.MT (MENU TYPE) ................................... 748
.OF (OFFSET) ....................................... 749
.SP (SPACE LINES) .................................. 751
.TR (TRANSLATE CHARACTER) ......................... 752

Chapter 4. DEBUG Subcommands ......................... 755
BREAK ............................................. 756
CAW ................................................. 758
CSW ................................................ 760
DEFINE .......................................... 762
DUMP ............................................... 764
GO ................................................... 766
GPR .................................................. 768
HX .................................................... 769
ORIGIN ............................................ 770
PSW .................................................. 772
RETURN ........................................... 773
SET .................................................. 774
STORE .............................................. 776
X ..................................................... 778

Appendix A. VSE/VSAM Functions Not Supported in CMS .... 779

Appendix B. OS/VS Access Method Services and VSAM Functions Not Supported ................ 781

Appendix C. Edit Subcommands and Macros .................. 785

Appendix D. Edit Reserved Filetype Defaults ................ 839

Appendix E. CMS EXEC Control Statements ................. 841

Summary of Changes .................................. 871

Glossary of Terms and Abbreviations ....................... 881

Bibliography ........................................ 885

Index .............................................. 893
### Figures

1. Character Sets and Their Contents ........................................... 4
2. CMS Command Execution Characteristics .................................. 9
3. CMS Command Summary ...................................................... 13
4. CMS Commands for System Programmers .................................. 19
5. COPYFILE Option Incompatibilities ....................................... 72
6. Determining Which VSAM Catalog to Use .................................. 120
7. Valid File Characteristics for Each Device Type of the FILEDEF Command ......................................................... 197
8. Sample FILELIST Screen .................................................... 223
9. Loader Search Order ......................................................... 316
10. ENTRY Statement Format ................................................... 317
11. LIBRARY Statement Format ............................................... 317
12. LDT Statement Format ..................................................... 318
13. ICS Statement Format ..................................................... 318
14. SLC Statement Format ..................................................... 319
15. REP Statement Format ..................................................... 320
16. SPB Statement Format ..................................................... 321
17. Sample MACLIST Screen ................................................... 342
18. Default Device Attributes for MOVEFILE Command .................... 354
19. Sample ‘userid NAMES’ File ................................................ 363
20. Sample NAMES Screen ..................................................... 369
21. Sample Entry for a List of names. ........................................ 369
22. Sample Note with Short Headings ......................................... 378
23. Example of Long Headings .................................................. 379
24. Sample PEEK Screen ........................................................ 401
25. Header Card Format ........................................................ 415
26. Sample RDRLIST Screen ................................................... 472
27. Default Settings for Message Routing ..................................... 504
28. Sample SENDFILE Menu .................................................... 524
29. Sample FILELIST Screen Invoked from SENDFILE ..................... 525
30. Default Windows ............................................................ 557
31. Default Virtual Screens .................................................... 559
32. Default Windows and Virtual Screens .................................... 560
33. Default Settings for Message Routing ..................................... 560
34. Summary of SVCTRACE Output Lines ..................................... 634
35. System and User-Defined Truncations .................................... 639
36. HELP and HELPCONV Format Word Summary ......................... 738
37. OS Access Method Service Operands NOT Supported in CMS ........ 782
38. Options of OS/VSAM Macros Not Supported in CMS ................... 783
39. Default EDIT Subcommand Settings for CMS Reserved Filetypes 839
Virtual Machine/System Product (VM/SP) is a Licensed Program that, when used in conjunction with VM/370 Release 6, controls "virtual machines." A virtual machine is the functional equivalent of a real machine. However, where the real machine has lights to show status, and buttons and switches on the real system console to control it, the virtual machine does not. It has a virtual system console to display status and a command language to start operations and control them. The virtual system console is your terminal.

VM/SP has command languages, which correspond to the components of the VM/SP system:

- The Control Program (CP) controls the resources of the real machine; that is, it controls the physical machine in your computer room. The CP commands are described in VM/SP CP Command Reference.

- The Conversational Monitor System (CMS) is a conversational operating system designed to run under CP. This publication describes general use CMS commands and subcommands that you can use in the CMS environment.

- The Interactive Problem Control System (IPCS) is an interactive, online facility for reporting and diagnosing software failures and for managing problem information and status. VM/SP IPCS can perform these functions for: CP ABEND dumps, CMS, PVM, RSCS, and GCS dumps, and any dump created by the VMDUMP command. IPCS runs in the CMS command environment. For more information, refer to the VM Diagnosis Guide.

When used in conjunction with VM/370 Release 6, the VM/370 component RSCS is also available to the VM/SP user.

- The Remote Spooling Communications Subsystem (RSCS) is a subsystem designed to supervise transmission of files across a teleprocessing network controlled by CP. For information about RSCS, see the VM/370 Remote Spooling Communications Subsystem (RSCS) User's Guide.

Each of the above components has a unique "command environment" that must be active in order for a command to be accepted. For CMS users, the two basic command environments are the CP command environment and the CMS command environment. By default, CP commands are acceptable input in the CMS command environment; if you enter a CP command, CP executes it, but control returns to the CMS environment. IPCS
DUMPSCAN subcommands must be entered from the DUMPSCAN environment.

CMS Environment

The CMS command language allows you to create, modify, debug, and, in general, manipulate a system of files.

The OS/VS Assembler and many OS/VS and VSE (DOS) language processors can be executed under CMS. For example, the OS/VS BASIC, FORTRAN IV (G1), COBOL and PL/I compilers, as well as the DOS PL/I and DOS/VS COBOL compilers, can execute under CMS. You can find a complete list of language processors that can be executed under CMS in the VM/SP Introduction. CMS invokes the assembler and the compilers when you issue the appropriate CMS commands. The ASSEMBLE command is described in this manual; the supported compiler commands are described in the appropriate Licensed Program publications.

CMS commands allow you to read cards from a virtual card reader, punch cards to a virtual card punch, and print records on a virtual printer. Many commands are provided to help you manipulate your virtual disks and files. The CMS commands are described in “Chapter 2. CMS Commands.”

In addition, CMS commands allow you to use CMS in full-screen mode. You can display CMS in a window, enter commands from anywhere on the physical screen, scroll through information in windows, and define your own windows. The windowing commands are described in “Chapter 2. CMS Commands.”

A special set of CMS commands becomes available to you when you issue the command:

set dos on

These commands, called CMS/DOS commands, simulate various functions of the VSE Operating System (DOS) in your CMS virtual machine. When the CMS/DOS environment is active, the CMS/DOS commands are an integral part of the CMS command language; they are listed alphabetically among the other CMS commands in “Chapter 2. CMS Commands.”

The DEBUG command places your virtual machine in the DEBUG subcommand environment. In this environment you can issue commands to display registers and storage, specify breakpoints (address instruction stops), display the contents of control words, and so on. The DEBUG subcommands are described in “Chapter 4. DEBUG Subcommands.”

The EXEC command executes CMS command procedures, called EXEC files. You can create EXEC files consisting of CMS and CP commands and EXEC control statements. The EXEC facility also has a symbolic capability; by manipulating variable symbols within an EXEC file, you can control the execution of the procedure. These procedures are usually created in the
edit environment. The EXEC control statements, variable symbols, and built-in functions are described in “Appendix E. CMS EXEC Control Statements.”

The HELP format words are used to create HELP ‘text’ information for user-defined commands, EXECs, and messages. The function, formats, and operands of the HELP facility format words are described in “Chapter 3. HELP and HELPConv Format Words.”

**Entering CMS Commands**

A CMS command consists of a command name, usually followed by one or more positional operands and, in many cases, by an option list. CMS commands and other subcommands described in this publication are shown in the format:

```
COMMAND NAME [operands...] [(options...[])]
```

You must use one or more blanks to separate each entry in the command line unless otherwise indicated. For an explanation of the special symbols used to describe the command syntax, see “Notational Conventions.”

**Command Name**

The command name is an alphameric symbol of one to eight characters. In general, the names are based on verbs or verb-noun combinations that describe the function you want the system to perform. For example, you may want to find out information concerning your CMS files. In this case, you would use the either the FILELIST or the LISTFILE command.

**Command Operands**

The command operands are keywords and/or positional operands of one to eight, and in a few cases, one to seven alphameric characters each. In certain cases there are more than eight characters, and CMS may ignore any characters after the first eight. The operands specify the information on which the system operates when it performs the command function.

You must write the operands in the order in which they appear in the command formats in “Chapter 2. CMS Commands,” unless otherwise specified. When you are using CMS, blanks may optionally be used to separate the last operand from the option list. CMS recognizes a left parenthesis “(” as the beginning of an option list; it does not have to be preceded by a blank.
Introduction

Command Options

The command options are keywords used to control the execution of the command. The command formats in "Chapter 2. CMS Commands" show all the options for each CMS command.

The option list must be preceded by a left parenthesis; the closing parenthesis is not necessary.

For most commands, if conflicting or duplicate options are entered, the last option entered is the option in effect for the command. Exceptions to this rule are noted where applicable.

Character Set Usage

CMS commands may be entered using a combination of characters from six different character sets. The contents of each of the character sets is shown in Figure 1.

<table>
<thead>
<tr>
<th>Character Set</th>
<th>Names</th>
<th>Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Separator</td>
<td>Blank</td>
<td></td>
</tr>
<tr>
<td>National</td>
<td>Dollar Sign</td>
<td>$</td>
</tr>
<tr>
<td></td>
<td>Pound Sign</td>
<td>#</td>
</tr>
<tr>
<td></td>
<td>At Sign</td>
<td>@</td>
</tr>
<tr>
<td>Alphabetic</td>
<td>Uppercase</td>
<td>A-Z</td>
</tr>
<tr>
<td></td>
<td>Lowercase</td>
<td>a-z</td>
</tr>
<tr>
<td>Numeric</td>
<td>Numeric</td>
<td>0-9</td>
</tr>
<tr>
<td>Alphameric</td>
<td>National</td>
<td>$, @</td>
</tr>
<tr>
<td></td>
<td>Alphabetic</td>
<td>A-Z</td>
</tr>
<tr>
<td></td>
<td>Numeric</td>
<td>a-z</td>
</tr>
<tr>
<td>Special</td>
<td></td>
<td>All other characters</td>
</tr>
</tbody>
</table>

Figure 1. Character Sets and Their Contents

Naming CMS Files

When you create a CMS file, you can give it any filename and filetype you wish. The rules for forming filenames and filetypes are:

- The filename and filetype can each be from one to eight characters.
- The valid characters are A-Z, a-z, 0-9, $, @, +, - (hyphen), : (colon), and _ (underscore).
Note: Lowercase letters within a fileid are valid for use within the CMS file system. However, some CMS commands do not support fileids that contain lowercase letters.

Notational Conventions

The notation used to define the command syntax in this publication is:

- Truncations and Abbreviations of Commands

Where truncation of a command name is permitted, the shortest acceptable version of the command is represented by uppercase letters. (Remember, however, that CMS commands can be entered with any combination of uppercase and lowercase letters.) The following example shows the format specification for the FILEDEF command.

FILEDEF

This format means that FI, FIL, FILE, FILED, FILEDE, and FILEDEF are all valid specifications for this command name.

Operands and options are specified in the same manner. Where truncation is permitted, the shortest acceptable version of the operand or option is represented by uppercase letters in the command format box. If no minimum truncation is noted, the entire word (represented by all uppercase letters) must be entered.

Abbreviations are shorter forms of command operands and options. Abbreviations for operands and options are shown in the description of the individual operands and options that follow the format box. For example, the abbreviation for DISK in the ASSEMBLE command is DI. Only these two forms are valid and no truncations are allowed. The format box contains

DISK

and the description that follows the format box is

DISK
DI

- Keywords, such as command names, operands, and options, appear in the format box and parameter descriptions in bold letters.

- Lowercase letters, words, and symbols that appear in the command format box in italics represent variables that you will substitute with specific information. For example,

\( fn \) indicates that you should enter a filename with the command.
• Uppercase letters and words, and the following symbols, should be entered as specified in the format box.

asterisk *
colon :
comma ,
equal sign =
hyphen –
parentheses ( )
period .

• The abbreviations fn, ft, and fm refer to filename, filetype, and filemode, respectively. The combination fn ft [fm] is also called the file identifier or fileid.

When a command format box shows the characters fn ft fm or fileid and they are not enclosed by brackets or braces, it indicates that a CMS file identifier must be entered. If an asterisk (*) appears beneath fn, ft, or fm, it indicates that an asterisk may be coded in that position of the fileid. The operand description describes the usage of the *.

• Choices are represented in the command format boxes by stacking.

A
B
C

• An underscore indicates an assumed default option. If an underscored choice is selected, it need not be specified when the command is entered.

For example, the representation:

A
B
C

indicates that either A, B, or C may be selected. However, if B is selected, it need not be specified. Or, if none is entered, B is assumed.

• The use of braces denotes choices, one of which must be selected.

Example: The representation:

\{A
B
C\}

indicates that you must specify either A, or B, or C. If a list of choices is enclosed by neither brackets nor braces, it is to be treated as if enclosed by braces.
• The use of brackets denotes choices, one of which may be selected.

Example: The representation:

\[
\begin{bmatrix}
A \\
B \\
C
\end{bmatrix}
\]

indicates that you may enter A, B, or C, or you may omit the field.

• In instances where there are nested braces or brackets on the text lines, the following rule applies: nested operand selection is dependent upon the selection of the operand of a higher level of nesting.

Example:

Level 1  Level 2  Level 3  
[filename [filetype [filemode]]]

where the highest level of nesting is the operand that is enclosed in only one pair of brackets and the lowest level of nesting is the operand that is enclosed by the maximum number of brackets. Thus, in the previous example, the user has the option of selecting a file by filename only or filename filetype only or by filename filetype filemode. The user cannot select filetype alone because filetype is nested within filename and our rule states: the higher level of nesting must be selected in order to select the next level (lower level) operand. The same is true if the user wants to select filemode; filename and filetype must also be selected.

• An ellipsis indicates that the preceding item or group of items may be repeated more than once in succession.

Example: The representation:

\((options\ldots)\)

indicates that more than one option may be coded within the parentheses.

**CMS Command Search Order**

When you enter a command line in the CMS environment, CMS has to locate the command to execute. If you have EXEC or MODULE files on any of your accessed disks, CMS treats them as commands; they are known as user-written commands.

As soon as the command name is found, the search stops and the command is executed. The search order is:

1. Search for an EXEC with the specified command name:
Introduction

a. Search for an EXEC in storage. If an EXEC with this name is found, CMS determines whether the EXEC has a USER, SYSTEM, or SHARED attribute. If the EXEC has the USER or SYSTEM attribute, it is executed.

If the EXEC has the SHARED attribute, the INSTSEG setting of the SET command is checked. When INSTSEG is ON, all accessed disks are searched and the access mode of the Installation Discontiguous Shared Segment (DCSS) is compared to the mode of an EXEC with the name that resides on disk. If the access mode of the DCSS is equal to or higher than the disk mode, the EXEC in the DCSS is executed. Otherwise, the EXEC on disk is executed.

b. Search for a file with the specified command name and a filetype EXEC on any currently accessed disk. CMS uses the standard search order (A through Z.) The table of active (open) disk files is searched first. An open file may be used ahead of a file that resides on a disk earlier in the search order.

2. Search for a translation or synonym of the specified command name. If found, search for an EXEC with the valid translation or synonym by repeating Step 1.

3. Search for a module with the specified command name:

   a. Search for a nucleus extension module.

   b. Search for a module in the transient area.

   c. Search for a nucleus-resident module.

   d. Search for a file with filetype MODULE on any currently accessed disk. The table of active (open) disk files is searched first. An open file may be used ahead of a file that resides on a disk earlier in the search order.

4. Search for a translation or synonym of the specified command name. If found, search for a module with the valid translation or synonym by repeating Step 3.

If the command is not known to CMS (that is, all of the above fails), it is passed to CP for execution.

When CMS searches for a translation or synonym (as in steps 2 and 4), the translation and synonym tables are searched in the following order:

1. User National Language Translation Table
2. System National Language Translation Table
3. User National Language Translation Synonym Table
4. System National Language Translation Synonym Table
5. CMS User Synonym Table
6. CMS System Synonym Table

See the SET TRANSLATE command for information on tables 1 to 4. See the SYNONYM command for information on tables 5 and 6.

CMS Command Execution Characteristics

Following is an alphabetical list of the CMS commands which require special consideration when called from a user program. For example, a program running in the user area cannot call a CMS command which also runs in the user area.

Any commands which are listed in this book but are not in this table are nucleus resident and will not interfere with the execution of a user program.

The “Code” column indicates the execution characteristics of the command.

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>indicates that this command is an EXEC. It may execute one or more CMS commands which run in the user or transient areas.</td>
</tr>
<tr>
<td>T</td>
<td>indicates that this command executes in the transient area.</td>
</tr>
<tr>
<td>U</td>
<td>indicates that this command executes in the user program area. All OS free storage pointers are reset with the STRINIT macro.</td>
</tr>
</tbody>
</table>

---

**Figure 2. CMS Command Execution Characteristics**

<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
<th>Command</th>
<th>Code</th>
<th>Command</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMSERV</td>
<td>U</td>
<td>EXECMAP</td>
<td>T</td>
<td>LOADLIB</td>
<td>U</td>
</tr>
<tr>
<td>ASSEMBLE</td>
<td>U</td>
<td>EXECUPDT</td>
<td>E</td>
<td>MACLIB</td>
<td>U</td>
</tr>
<tr>
<td>ASSGN</td>
<td>T</td>
<td>FCOBOL</td>
<td>E</td>
<td>MACLIST</td>
<td>E</td>
</tr>
<tr>
<td>CATCHCHECK</td>
<td>U</td>
<td>FILELIST</td>
<td>E</td>
<td>MODMAP</td>
<td>T</td>
</tr>
<tr>
<td>CMSBATCH</td>
<td>U</td>
<td>FORMAT</td>
<td>U</td>
<td>MOREHELP</td>
<td>E</td>
</tr>
<tr>
<td>CMSSERV</td>
<td>E</td>
<td>GENDIRT</td>
<td>T</td>
<td>MOVEFILE</td>
<td>U</td>
</tr>
<tr>
<td>COMPARE</td>
<td>T</td>
<td>GENMSG</td>
<td>U</td>
<td>NAMES</td>
<td>E</td>
</tr>
<tr>
<td>CONVERT COMMANDS</td>
<td>E</td>
<td>GLOBAL</td>
<td>T</td>
<td>NOTE</td>
<td>E</td>
</tr>
<tr>
<td>DDR</td>
<td>U</td>
<td>HELPCONV</td>
<td>T</td>
<td>NUCXDROP</td>
<td>T</td>
</tr>
<tr>
<td>DEFAULTS</td>
<td>E</td>
<td>HNDINT</td>
<td>T</td>
<td>NUCXMAP</td>
<td>T</td>
</tr>
<tr>
<td>DISCARD</td>
<td>E</td>
<td>HNDSSVC</td>
<td>T</td>
<td>OPTION</td>
<td>T</td>
</tr>
<tr>
<td>DISK</td>
<td>T</td>
<td>IOCP</td>
<td>U</td>
<td>OSRUN</td>
<td>U</td>
</tr>
<tr>
<td>DOSLIB</td>
<td>U</td>
<td>LABELDEF</td>
<td>T</td>
<td>PEEK</td>
<td>E</td>
</tr>
<tr>
<td>DOSLKED</td>
<td>U</td>
<td>LANGGEN</td>
<td>E</td>
<td>PSERV</td>
<td>U</td>
</tr>
<tr>
<td>DOSPLI</td>
<td>E</td>
<td>LANGMERG</td>
<td>E</td>
<td>PUNCH</td>
<td>T</td>
</tr>
<tr>
<td>DSERV</td>
<td>U</td>
<td>LISTDS</td>
<td>U</td>
<td>RDR</td>
<td>T</td>
</tr>
<tr>
<td>EDIT</td>
<td>U</td>
<td>LISTIO</td>
<td>T</td>
<td>RDLISTR</td>
<td>E</td>
</tr>
<tr>
<td>ESERV</td>
<td>E</td>
<td>LKED</td>
<td>U</td>
<td>READCARD</td>
<td>T</td>
</tr>
</tbody>
</table>

Chapter 1. Introduction and General Concepts
Introduction

<table>
<thead>
<tr>
<th>Command</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECEIVE</td>
<td>E</td>
</tr>
<tr>
<td>RESERVE</td>
<td>T</td>
</tr>
<tr>
<td>RSERV</td>
<td>U</td>
</tr>
<tr>
<td>RUN</td>
<td>E</td>
</tr>
<tr>
<td>SENDFILE</td>
<td>E</td>
</tr>
<tr>
<td>SETPRT</td>
<td>T</td>
</tr>
</tbody>
</table>

Note: In the list above, HNDINT and HNDSVC are CMS functions, not commands.

CMS Command Summary

Figure 3 on page 13 and Figure 4 on page 19 contain alphabetical lists of the CMS commands and the functions performed by each. Figure 3 lists those commands that are available for general use; Figure 4 lists the commands used by system programmers and system support personnel who are responsible for generating, maintaining, and updating VM/SP. Unless otherwise noted, CMS commands are described in this manual. For those commands not described in this manual, the “Code” column indicates the publication that describes the command:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSE PP</td>
<td>Indicates that this command invokes a VSE Program Product, available from IBM for a license fee.</td>
</tr>
<tr>
<td>EREP</td>
<td>Indicates that this command is described in OS/VS Environmental Recording Editing and Printing (EREP) Program.</td>
</tr>
<tr>
<td>IOCP UG</td>
<td>Indicates that this command is described in the Input/Output Configuration Program User's Guide and Reference.</td>
</tr>
<tr>
<td>DIAG</td>
<td>Indicates that this command is described in VM Diagnosis Guide.</td>
</tr>
<tr>
<td>OS PP</td>
<td>Indicates that this command invokes an OS program product, available from IBM for a license fee.</td>
</tr>
<tr>
<td>PLNGDE</td>
<td>Indicates that this command is described in the VM/SP Planning Guide and Reference.</td>
</tr>
<tr>
<td>INST</td>
<td>Indicates that this command is described in the VM/SP Installation Guide.</td>
</tr>
<tr>
<td>SFPROG</td>
<td>Indicates that this command is described in VM/SP System Facilities for Programming.</td>
</tr>
</tbody>
</table>
CMSPROG Indicates that this command is described in VM/SP CMS for System Programming.

CPPROG Indicates that this command is described in VM/SP CP for System Programming.

Note: If a CMS command is described in this manual, but is also repeated in other VM/SP publications, the chart does not refer to those other publications.

You can enter CMS commands when you are running CMS in your virtual machine, the terminal is idle, and the virtual machine can accept input. However, if CMS is processing a previously entered command and your typewriter terminal keyboard is locked, you must signal your virtual machine via an attention interruption. The system acknowledges the interruption by unlocking the keyboard. Now you can enter commands.

If your terminal is a display device, there is no problem of entering commands while the virtual machine is busy because its keyboard remains unlocked for additional command input. Note that in these circumstances the command you enter is stacked in the terminal input buffer and is not executed until the command that is currently being executed completes. If more commands are entered than CP can handle, a NOT ACCEPTED message is displayed at the display terminal.

In addition to the commands listed in Figure 3 and Figure 4, there are ten Immediate commands and thirteen Border commands. These commands are handled in a different manner from the others.

Immediate commands may be entered while another command is being executed by pressing the Attention key (or its equivalent), and they are executed immediately. The Immediate commands are:

HB  Halt batch execution
HI  Halt Interpretation
HO  Halt tracing
HT  Halt typing
HX  Halt execution
RO  Resume tracing
RT  Resume typing
SO  Suspend tracing
TE  Trace end
TS  Trace start
You can define your own immediate commands by using any of the following:

- The IMMCMD macro in an assembler language program
- The IMMCMD command within an EXEC (CMS EXEC, EXEC 2, System Product interpreter)
- NUXLOAD command with the IMMCMD option specified.

Border commands are single-character windowing commands that you may enter in the corners of window borders. The Border commands are:

- **B** Scrolls the window backward
- **C** Clears the window of scrollable data
- **D** Drops the window
- **F** Scrolls the window forward
- **H** Hides the window
- **L** Scrolls the window to the left
- **M** Changes the location of the window
- **N** Minimizes the window
- **O** Restores the window
- **P** Pops the window
- **R** Scrolls the window to the right
- **S** Changes the size of the window
- **X** Maximizes the window
### CMS Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESS</td>
<td></td>
<td>Identify direct access space to a CMS virtual machine, create extensions and relate the disk space to a logical directory.</td>
</tr>
<tr>
<td>ALARM VSCREEN</td>
<td></td>
<td>Sound the terminal alarm the next time the display is refreshed.</td>
</tr>
<tr>
<td>AMSERV</td>
<td></td>
<td>Invoke access method services utility functions to create, alter, list, copy, delete, import, or export VSAM catalogs and data sets.</td>
</tr>
<tr>
<td>ASSEMBLE</td>
<td></td>
<td>Assemble assembler language source code.</td>
</tr>
<tr>
<td>ASSGN</td>
<td></td>
<td>Assign or unassign a CMS/DOS system or programmer logical unit for a virtual I/O device.</td>
</tr>
<tr>
<td>CATCHECK</td>
<td></td>
<td>Allows a CMS VSAM user (with or without DOS set ON) to invoke the VSE/VSAM Catalog Check Service Aid to verify a complete catalog structure.</td>
</tr>
<tr>
<td>CLEAR VSCREEN</td>
<td></td>
<td>Erase data in the virtual screen by overwriting the data buffer with nulls.</td>
</tr>
<tr>
<td>CLEAR WINDOW</td>
<td></td>
<td>Scroll past all data in the virtual screen to which the window is connected so that no scrollable data is displayed in the window.</td>
</tr>
<tr>
<td>CMDCALL</td>
<td></td>
<td>Convert EXEC 2 extended plist function calls to CMS extended plist command calls.</td>
</tr>
<tr>
<td>CMSBATCH</td>
<td></td>
<td>Invoke the CMS batch facility.</td>
</tr>
<tr>
<td>CMSSERV</td>
<td></td>
<td>Start Enhanced Connectivity Facilities communications between your VM/SP host system and your work station (IBM Personal Computer).</td>
</tr>
<tr>
<td>COMPARE</td>
<td></td>
<td>Compare records in CMS disk files.</td>
</tr>
<tr>
<td>CONVERT COMMANDS</td>
<td></td>
<td>Convert a CMS file containing Definition Language for Command Syntax (DLCS) statements into an internal form for the parsing facility.</td>
</tr>
<tr>
<td>CONWAIT</td>
<td></td>
<td>Causes a program to wait until all pending terminal I/O is complete.</td>
</tr>
<tr>
<td>COPYFILE</td>
<td></td>
<td>Copy CMS disk files according to specifications.</td>
</tr>
<tr>
<td>CP</td>
<td></td>
<td>Enter CP commands from the CMS environment.</td>
</tr>
<tr>
<td>CURSOR VSCREEN</td>
<td></td>
<td>Position the cursor on specified line and column in a virtual screen.</td>
</tr>
<tr>
<td>DDR</td>
<td>CPSP</td>
<td>Perform backup, restore, and copy operations for disks.</td>
</tr>
<tr>
<td>DEBUG</td>
<td></td>
<td>Enter DEBUG subcommand environment.</td>
</tr>
<tr>
<td>DEFAULTS</td>
<td></td>
<td>Set or display default options for the commands: FILELIST, HELP, MACLIST, NOTE, RDRLIST, RECEIVE, PEEK, SENDFILE, and TELL.</td>
</tr>
</tbody>
</table>

Figure 3 (Part 1 of 7). CMS Command Summary
<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFINE VSCREEN</td>
<td></td>
<td>Create a virtual screen.</td>
</tr>
<tr>
<td>DEFINE WINDOW</td>
<td></td>
<td>Create a window.</td>
</tr>
<tr>
<td>DELETE VSCREEN</td>
<td></td>
<td>Remove a virtual screen definition.</td>
</tr>
<tr>
<td>DELETE WINDOW</td>
<td></td>
<td>Remove a window definition.</td>
</tr>
<tr>
<td>DESBUF</td>
<td></td>
<td>Clears the program stack and the terminal input buffers.</td>
</tr>
<tr>
<td>DISK</td>
<td></td>
<td>Perform disk-to-card and card-to-disk operations for CMS files.</td>
</tr>
<tr>
<td>DLBL</td>
<td></td>
<td>Define a VSE filename or VSAM ddname and relate that name to a disk file.</td>
</tr>
<tr>
<td>DOSLIB</td>
<td></td>
<td>Delete, compact, or list information about the phases of a CMS/DOS phase library.</td>
</tr>
<tr>
<td>DOSLKED</td>
<td></td>
<td>Link-edit CMS text decks or object modules from a VSE relocatable library and place them in executable form in a CMS/DOS phase library.</td>
</tr>
<tr>
<td>DOSPLI VSE PP</td>
<td></td>
<td>Compile DOS PL/I source code under CMS/DOS.</td>
</tr>
<tr>
<td>DROP WINDOW</td>
<td></td>
<td>Move a window down in the order of displayed windows.</td>
</tr>
<tr>
<td>DROPBUF</td>
<td></td>
<td>Eliminate a program stack buffer.</td>
</tr>
<tr>
<td>DSERV</td>
<td></td>
<td>Display information contained in the VSE core image, relocatable, source, procedure, and transient directories.</td>
</tr>
<tr>
<td>EDIT</td>
<td></td>
<td>Invoke the VM/SP System Product editor in CMS editor (EDIT) compatibility mode to create or modify a disk file.</td>
</tr>
<tr>
<td>ERASE</td>
<td></td>
<td>Delete CMS disk files.</td>
</tr>
<tr>
<td>ESERV</td>
<td></td>
<td>Display, punch or print an edited (compressed) macro from a VSE source statement library (E sublibrary).</td>
</tr>
<tr>
<td>EXEC</td>
<td></td>
<td>Execute special procedures made up of frequently used sequences of commands.</td>
</tr>
<tr>
<td>EXECDROP</td>
<td></td>
<td>Purge storage-resident EXECs.</td>
</tr>
<tr>
<td>EXECIO</td>
<td></td>
<td>Do I/O operations between a device and the program stack or a variable.</td>
</tr>
<tr>
<td>EXECLOAD</td>
<td></td>
<td>Load EXECs into storage.</td>
</tr>
<tr>
<td>EXECMAP</td>
<td></td>
<td>List storage-resident EXECs.</td>
</tr>
<tr>
<td>EXECOS</td>
<td></td>
<td>Resets the OS and VSAM environments under CMS without returning to the interactive environment.</td>
</tr>
<tr>
<td>EXECSTAT</td>
<td></td>
<td>Obtain status of a specific EXEC.</td>
</tr>
<tr>
<td>EXECUPDT</td>
<td></td>
<td>Produces an updated executable version of a System Product Interpreter source program.</td>
</tr>
</tbody>
</table>

Figure 3 (Part 2 of 7). CMS Command Summary
<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCOBOL</td>
<td>VSE</td>
<td>Compile DOS/VS COBOL source code under CMS/DOS.</td>
</tr>
<tr>
<td>FETCH</td>
<td>PP</td>
<td>Fetch a CMS/DOS or VSE executable phase.</td>
</tr>
<tr>
<td>FILEDEF</td>
<td></td>
<td>Define an OS ddname and relate that ddname to any device supported by CMS.</td>
</tr>
<tr>
<td>FILELIST</td>
<td></td>
<td>List information about CMS disk files, with the ability to edit and issue commands from the list.</td>
</tr>
<tr>
<td>FINIS</td>
<td></td>
<td>Close an open file.</td>
</tr>
<tr>
<td>FORMAT</td>
<td></td>
<td>Prepare disks in CMS fixed block format.</td>
</tr>
<tr>
<td>GENDIRT</td>
<td></td>
<td>Fill in auxiliary module directories.</td>
</tr>
<tr>
<td>GENMOD</td>
<td></td>
<td>Generate nonrelocatable CMS files (MODULE files).</td>
</tr>
<tr>
<td>GENMSG</td>
<td></td>
<td>Convert a message repository file into an internal form.</td>
</tr>
<tr>
<td>GET VSCREEN</td>
<td></td>
<td>Write data from a CMS file to the specified virtual screen.</td>
</tr>
<tr>
<td>GLOBAL</td>
<td></td>
<td>Identify specific CMS libraries to be searched for macros, copy files, subroutines, LOADLIB modules, or DOS executable phases.</td>
</tr>
<tr>
<td>GLOBALV</td>
<td></td>
<td>Set, maintain, and retrieve a collection of named variables.</td>
</tr>
<tr>
<td>HELP</td>
<td></td>
<td>Display information about CP, CMS, or user commands, EDIT, XEDIT, or DEBUG subcommands, EXEC, EXEC 2 and System Product Interpreter control statements, and descriptions of CMS and CP messages.</td>
</tr>
<tr>
<td>HELPCONV</td>
<td></td>
<td>Convert a script file into an acceptable form to be used by the HELP facility.</td>
</tr>
<tr>
<td>HIDE WINDOW</td>
<td></td>
<td>Prevent the specified window from being displayed, and, optionally, connect the window to a virtual screen.</td>
</tr>
<tr>
<td>IDENTIFY</td>
<td></td>
<td>Display or stack userid, nodeid, rcsid, date, time, time zone, and day of the week.</td>
</tr>
<tr>
<td>IMMCMMD</td>
<td></td>
<td>Use the IMMCMMD command to establish or cancel immediate commands from within an EXEC.</td>
</tr>
<tr>
<td>INCLUDE</td>
<td></td>
<td>Bring additional TEXT files into storage and establish linkages.</td>
</tr>
<tr>
<td>IOCP</td>
<td>IOCP</td>
<td>Invoke the Input/Output Configuration Program</td>
</tr>
<tr>
<td>LABELDEF</td>
<td>UG</td>
<td>Specify standard HDR1 and EOF1 tape label description information for CMS, CMS/DOS, and OS simulation.</td>
</tr>
<tr>
<td>LISTDS</td>
<td></td>
<td>List information about data sets and space allocation on OS, DOS, and VSAM disks.</td>
</tr>
</tbody>
</table>

Figure 3 (Part 3 of 7). CMS Command Summary
<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>LISTFILE</td>
<td></td>
<td>List information about CMS disk files.</td>
</tr>
<tr>
<td>LISTIO</td>
<td></td>
<td>Display information concerning CMS/DOS system and programmer logical units.</td>
</tr>
<tr>
<td>LKED</td>
<td></td>
<td>Link edit a CMS TEXT file or OS object module into a CMS LOADLIB.</td>
</tr>
<tr>
<td>LOAD</td>
<td></td>
<td>Bring TEXT files into storage for execution.</td>
</tr>
<tr>
<td>LOADLIB</td>
<td></td>
<td>Maintain CMS LOADLIB libraries.</td>
</tr>
<tr>
<td>LOADMOD</td>
<td></td>
<td>Bring a single MODULE file into storage.</td>
</tr>
<tr>
<td>MACLIB</td>
<td></td>
<td>Create or modify CMS macro libraries.</td>
</tr>
<tr>
<td>MACLIST</td>
<td></td>
<td>List information about all members in a specified maclib, with the ability to edit and issue commands from the list.</td>
</tr>
<tr>
<td>MAKEBUF</td>
<td></td>
<td>Create a new program stack buffer.</td>
</tr>
<tr>
<td>MAXIMIZE WINDOW</td>
<td></td>
<td>Expand a window to the physical screen size.</td>
</tr>
<tr>
<td>MINIMIZE WINDOW</td>
<td></td>
<td>Reduce the size of the window to one line.</td>
</tr>
<tr>
<td>MODMAP</td>
<td></td>
<td>Display the load map of a MODULE file.</td>
</tr>
<tr>
<td>MOREHELP</td>
<td></td>
<td>Obtain either additional or related information about the latest valid HELP command you issued.</td>
</tr>
<tr>
<td>MOVEFILE</td>
<td></td>
<td>Move data from one device to another device of the same or a different type.</td>
</tr>
<tr>
<td>NAMEFIND</td>
<td></td>
<td>Display/stack information from a NAMES file. (default ’userid NAMES’).</td>
</tr>
<tr>
<td>NAMES</td>
<td></td>
<td>Display a menu to create, display or modify entries in a ’userid NAMES’ file. (The menu is available only on display terminals.)</td>
</tr>
<tr>
<td>NOTE</td>
<td></td>
<td>Prepare a ‘note’ for one or more computer users, to be sent via the SENDFILE command.</td>
</tr>
<tr>
<td>NUCXDROP</td>
<td></td>
<td>Delete specified nucleus extensions.</td>
</tr>
<tr>
<td>NUCXLOAD</td>
<td></td>
<td>Load a nucleus extension.</td>
</tr>
<tr>
<td>NUCXMAP</td>
<td></td>
<td>Identify existing nucleus extensions.</td>
</tr>
<tr>
<td>OPTION</td>
<td></td>
<td>Change the DOS/VS COBOL compiler (FCOBOL) options that are in effect for the current terminal session.</td>
</tr>
<tr>
<td>OSRUN</td>
<td></td>
<td>Load, relocate, and execute a load module from a CMS LOADLIB or OS module library.</td>
</tr>
<tr>
<td>PARSECMD</td>
<td></td>
<td>Call the parsing facility from within an exec.</td>
</tr>
<tr>
<td>PEEK</td>
<td></td>
<td>Display a file that is in your virtual reader without reading it onto disk.</td>
</tr>
<tr>
<td>POP WINDOW</td>
<td></td>
<td>Move a window up in the order of displayed windows.</td>
</tr>
<tr>
<td>POSITION WINDOW</td>
<td></td>
<td>Change the location of a window on the physical screen.</td>
</tr>
</tbody>
</table>

Figure 3 (Part 4 of 7). CMS Command Summary
<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINT</td>
<td></td>
<td>Spool a specified CMS file to the virtual printer.</td>
</tr>
<tr>
<td>PSERV</td>
<td></td>
<td>Copy a procedure from the VSE procedure library onto a CMS disk, display the procedure at the terminal, or spool the procedure to the virtual punch or printer.</td>
</tr>
<tr>
<td>PUNCH</td>
<td></td>
<td>Spool a copy of a CMS file to the virtual punch.</td>
</tr>
<tr>
<td>PUT SCREEN</td>
<td></td>
<td>Make a copy of the physical screen and write the image to a CMS file.</td>
</tr>
<tr>
<td>PUT VSCREEN</td>
<td></td>
<td>Write the data from the scrollable data area of a virtual screen to a CMS file.</td>
</tr>
<tr>
<td>QUERY</td>
<td></td>
<td>Request information about a CMS virtual machine.</td>
</tr>
<tr>
<td>RDR</td>
<td></td>
<td>Generate a return code and either display or stack a message that identifies the characteristics of the next file in your virtual reader.</td>
</tr>
<tr>
<td>RDRLIST</td>
<td></td>
<td>Display information about files in your virtual reader with the ability to issue commands from the list.</td>
</tr>
<tr>
<td>READCARD</td>
<td></td>
<td>Read data from spooled card input device.</td>
</tr>
<tr>
<td>RECEIVE</td>
<td></td>
<td>Read onto disk a file or note that is in your virtual reader.</td>
</tr>
<tr>
<td>REFRESH</td>
<td></td>
<td>Update virtual screens and their associated windows.</td>
</tr>
<tr>
<td>RELEASE</td>
<td></td>
<td>Make a disk and its directory inaccessible to a CMS virtual machine.</td>
</tr>
<tr>
<td>RENAME</td>
<td></td>
<td>Change the name of a CMS file or files.</td>
</tr>
<tr>
<td>RESERVE</td>
<td></td>
<td>Use the RESERVE command to allocate all available blocks of a 512-, 1K-, 2K-, or 4K-byte block formatted minidisk to a unique CMS file.</td>
</tr>
<tr>
<td>RESTORE WINDOW</td>
<td></td>
<td>Return a maximized or minimized window to its size and location prior to the maximize or minimize.</td>
</tr>
<tr>
<td>ROUTE</td>
<td></td>
<td>Direct data of a particular message class to a virtual screen.</td>
</tr>
<tr>
<td>RSERV</td>
<td></td>
<td>Copy a VSE relocatable module onto a CMS disk, display it at the terminal, or spool a copy to the virtual punch or printer.</td>
</tr>
<tr>
<td>RUN</td>
<td></td>
<td>Initiate series of functions to be performed on a source, MODULE, TEXT, or EXEC file.</td>
</tr>
<tr>
<td>SCROLL</td>
<td></td>
<td>Move a window to a new location on the virtual screen.</td>
</tr>
<tr>
<td>SENDFILE</td>
<td></td>
<td>Send files or notes to one or more computer users, attached locally or remotely, by issuing the command or by using a menu.(display terminal only)</td>
</tr>
<tr>
<td>SENTRIES</td>
<td></td>
<td>Determine the number of lines currently in the program stack.</td>
</tr>
<tr>
<td>SET</td>
<td></td>
<td>Establish, set, or reset CMS virtual machine characteristics.</td>
</tr>
</tbody>
</table>

Figure 3 (Part 5 of 7). CMS Command Summary
<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>SETPRT</td>
<td></td>
<td>Load a virtual 3800 printer.</td>
</tr>
<tr>
<td>SHOW WINDOW</td>
<td></td>
<td>Place a window on top of all other displayed windows and connect the window to a virtual screen.</td>
</tr>
<tr>
<td>SIZE WINDOW</td>
<td></td>
<td>Change the number of lines and columns for a specified window.</td>
</tr>
<tr>
<td>SORT</td>
<td></td>
<td>Arrange a specified file in ascending order according to sort fields in the data records.</td>
</tr>
<tr>
<td>SSERV</td>
<td></td>
<td>Copy a VSE source statement book onto a CMS disk, display it at the terminal, or spool a copy to the virtual punch or printer.</td>
</tr>
<tr>
<td>START</td>
<td></td>
<td>Begin execution of programs previously loaded (OS and CMS) or fetched (CMS/DOS).</td>
</tr>
<tr>
<td>STATE</td>
<td></td>
<td>Verify the existence of a CMS disk file.</td>
</tr>
<tr>
<td>STATEW</td>
<td></td>
<td>Verify a file on a read/write CMS disk.</td>
</tr>
<tr>
<td>SVCTRACE</td>
<td></td>
<td>Record information about supervisor calls.</td>
</tr>
<tr>
<td>SYNONYM</td>
<td></td>
<td>Invoke a table containing synonyms you have created for CMS and user-written commands.</td>
</tr>
<tr>
<td>TAPE</td>
<td></td>
<td>Perform tape-to-disk and disk-to-tape operations for CMS files, position tapes, and display or write VOL1 labels.</td>
</tr>
<tr>
<td>TAPEMAC</td>
<td></td>
<td>Create CMS MACLIB libraries directly from an IEHMOVE-created partitioned data set on tape.</td>
</tr>
<tr>
<td>TAPPDS</td>
<td></td>
<td>Load OS partitioned data set (PDS) files or card image files from tape to disk.</td>
</tr>
<tr>
<td>TELL</td>
<td></td>
<td>Send a message to one or more computer users who are logged on to your computer or to one attached to yours via RSCS.</td>
</tr>
<tr>
<td>TXTLIB</td>
<td></td>
<td>Generate and modify text libraries.</td>
</tr>
<tr>
<td>TYPE</td>
<td></td>
<td>Display all or part of a CMS file at the terminal.</td>
</tr>
<tr>
<td>UPDATE</td>
<td></td>
<td>Make changes in a program source file as defined by control cards in a control file.</td>
</tr>
<tr>
<td>VALIDATE</td>
<td></td>
<td>Verify the syntax of a file identifier and verify whether or not a disk is accessed.</td>
</tr>
<tr>
<td>VSAPL</td>
<td>OS</td>
<td>Invoke VS APL interface in CMS.</td>
</tr>
<tr>
<td>WAITREAD VSCREEN</td>
<td>PP</td>
<td>Use from an EXEC to update the virtual screen with data, refresh the physical screen, and wait for the next attention interrupt.</td>
</tr>
<tr>
<td>WAITT VSCREEN</td>
<td></td>
<td>Update the virtual screen with data.</td>
</tr>
<tr>
<td>WRITE VSCREEN</td>
<td></td>
<td>Enter information in a virtual screen.</td>
</tr>
<tr>
<td>XEDIT</td>
<td></td>
<td>Invoke the VM/SP System Product Editor to create or modify a disk file.</td>
</tr>
</tbody>
</table>

Figure 3 (Part 6 of 7). CMS Command Summary
### CMS Commands for System Programmers

<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMITMSG</td>
<td></td>
<td>Retrieve a message from a CMS message repository file or your own message repository file.</td>
</tr>
</tbody>
</table>

Figure 3 (Part 7 of 7). CMS Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM3705</td>
<td>INST</td>
<td>Assemble 370x source code.</td>
</tr>
<tr>
<td>ASMGEND</td>
<td>INST</td>
<td>Regenerate the VM/SP assembler command modules.</td>
</tr>
<tr>
<td>CMSGEND</td>
<td>INST</td>
<td>Generate a new CMS disk-resident module from updated TEXT files.</td>
</tr>
<tr>
<td>CPEREP</td>
<td>EREP</td>
<td>Format and edit system error records for output.</td>
</tr>
<tr>
<td>DCSSGEN</td>
<td>PLNGDE</td>
<td>Load, build, and save a DCSS containing the executing copy of frequently used EXECs and Editor Macros.</td>
</tr>
<tr>
<td>DIRECT</td>
<td>PLNGDE</td>
<td>Set up VM/SP directory entries.</td>
</tr>
<tr>
<td>DISKMAP</td>
<td>INST</td>
<td>Summarize the MDISK statements in the CP directory to show gaps and overlaps in minidisk assignments.</td>
</tr>
<tr>
<td>DOSGEN</td>
<td>INST</td>
<td>Load and save CMSDOS and INSTVSAM shared segments.</td>
</tr>
<tr>
<td>DUMPSCAN</td>
<td>DIAG</td>
<td>Provide interactive analysis of CP abend dumps.</td>
</tr>
<tr>
<td>EXPAND</td>
<td>INST</td>
<td>Adds space to a program in object deck form.</td>
</tr>
<tr>
<td>GEN3705</td>
<td>INST</td>
<td>Generate an EXEC file that assembles and link-edits the 370x control program.</td>
</tr>
<tr>
<td>GENTSAF</td>
<td>INST</td>
<td>Builds the TSAF load module from TEXT files.</td>
</tr>
<tr>
<td>ITASK</td>
<td>INST</td>
<td>Performs most of the installation procedure by invoking other execs and commands.</td>
</tr>
<tr>
<td>LANGGEN</td>
<td>SFPROG</td>
<td>Save all text files for a language in a DCSS, and/or save CP message repository.</td>
</tr>
<tr>
<td>LANGMERG</td>
<td>SFPROG</td>
<td>Combine all language-related files for an application into one text file.</td>
</tr>
<tr>
<td>NCPDUMP</td>
<td>DIAG</td>
<td>Process CP spool reader files created by 370x dumping operations.</td>
</tr>
<tr>
<td>PRB</td>
<td>DIAG</td>
<td>Update IPCS problem status.</td>
</tr>
<tr>
<td>PROB</td>
<td>DIAG</td>
<td>Enter a problem report in IPCS.</td>
</tr>
<tr>
<td>PROP</td>
<td>SFPROG</td>
<td>Provide Programmable Operator capability.</td>
</tr>
<tr>
<td>SAMGEN</td>
<td>INST</td>
<td>Load and save the CMSBAM shared segment.</td>
</tr>
<tr>
<td>SAVENCP</td>
<td>INST, CPPROG</td>
<td>Read 370x control program load into virtual storage and save an image on a CP-owned disk.</td>
</tr>
</tbody>
</table>

Figure 4 (Part 1 of 3). CMS Commands for System Programmers
## Introduction

<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>SETKEY</td>
<td>CPPROG</td>
<td>Assign storage protect keys to storage assigned to named systems.</td>
</tr>
<tr>
<td>SNTMAP</td>
<td>INST</td>
<td>Processes DMKSNT macro definitions and produces a saved segment DASD map and a virtual memory map.</td>
</tr>
<tr>
<td>SPGEN</td>
<td>INST</td>
<td>Performs various system generation and maintenance functions.</td>
</tr>
<tr>
<td>SPLOAD</td>
<td>INST</td>
<td>Loads the VM/SP product tapes to the appropriate minidisks during initial installation.</td>
</tr>
<tr>
<td>STAT</td>
<td>DIAG</td>
<td>Display the status of reported system problems.</td>
</tr>
<tr>
<td>TRAPRED</td>
<td>DIAG</td>
<td>Allow the data collected by CPTRAP to be displayed or printed.</td>
</tr>
<tr>
<td>UTILITY</td>
<td>INST</td>
<td>Provides installation functions such as printing system definition files, creating stand-alone service utility tape and service programs on disk, etc.</td>
</tr>
<tr>
<td>VMFASM</td>
<td>INST</td>
<td>Creates an updated source file using IBM updates, PTFs, and user updates, then assembles the source file.</td>
</tr>
<tr>
<td>VMFDOS</td>
<td>INST</td>
<td>Create CMS files for VSE modules from VSE library distribution tape or SYSIN tape.</td>
</tr>
<tr>
<td>VMFLKED</td>
<td>INST</td>
<td>Invokes the CMS LKED command to link-edit modules into a LOADLIB.</td>
</tr>
<tr>
<td>VMFLOAD</td>
<td>INST</td>
<td>Generate a new CP, CMS, or RSCS module.</td>
</tr>
<tr>
<td>VMFMAC</td>
<td>INST</td>
<td>Creates macro libraries using IBM and user updates.</td>
</tr>
<tr>
<td>VMFMERGE</td>
<td>INST</td>
<td>Applies PTFs to object code and maintains a record in the Merge Log.</td>
</tr>
<tr>
<td>VMFNLS</td>
<td>INST</td>
<td>Applies updates to national language files and compiles the updated versions.</td>
</tr>
<tr>
<td>VMFPLC2</td>
<td>INST</td>
<td>Loads source code from product tape, dumps CMS-formatted files from disk to tape, loads previously dumped files from tape to disk, performs various control operations on a specified tape drive, and loads the service installation VMSERV EXEC from the PUT.</td>
</tr>
<tr>
<td>VMFREMOV</td>
<td>INST</td>
<td>Removes PTFs that were applied using VMFMERGE.</td>
</tr>
<tr>
<td>VMFTXT</td>
<td>INST</td>
<td>Creates text libraries using IBM and user updates.</td>
</tr>
<tr>
<td>VMFZAP</td>
<td>INST</td>
<td>Applies ZAPs to object code and maintains a record of them in the ZAP Log.</td>
</tr>
<tr>
<td>VMSERV</td>
<td>INST</td>
<td>Controls the individual service EXECs on the system Program Update Tape.</td>
</tr>
<tr>
<td>VRSIZE</td>
<td>INST</td>
<td>Generates DMKSLC text which is used to generate a V=R area in the system.</td>
</tr>
<tr>
<td>VSAMGEN</td>
<td>INST</td>
<td>Load and save CMSVSAM and CMSAMS shared segments.</td>
</tr>
</tbody>
</table>

Figure 4 (Part 2 of 3). CMS Commands for System Programmers
### Introduction

<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSEVSAM</td>
<td>INST</td>
<td>Build a VSE/VSAM maclib containing the supported VSE/VSAM macros as well as the following VSE macros: CDLOAD, CLOSE, CLOSER, GET, OPEN, OPENR, and PUT.</td>
</tr>
<tr>
<td>ZAP</td>
<td>INST</td>
<td>Modify or dump LOADLIB, TXTLIB, or MODULE files.</td>
</tr>
<tr>
<td>ZAPTEXT</td>
<td>INST</td>
<td>Modifies or dumps individual text files.</td>
</tr>
</tbody>
</table>

Figure 4 (Part 3 of 3). CMS Commands for System Programmers
This part contains reference information for the CMS commands used by general users. Each command description indicates the command format, operands and options; it also lists error messages and return codes the command issues. Usage notes are provided, where applicable.

The formats of the DEBUG, EDIT, XEDIT, and EXEC commands are also listed; for details on the DEBUG or EDIT subcommands or EXEC control statements, see:

- “Chapter 4. DEBUG Subcommands”
- “Appendix C. EDIT Subcommands and Macros”
- “Appendix E. CMS EXEC Control Statements”

For information about the System Product interpreter, see the VM/SP System Product Interpreter Reference, and the VM/SP System Product Interpreter User's Guide.

For details on the XEDIT subcommands and macros, see VM/SP System Product Editor Command and Macro Reference. For usage information on XEDIT subcommands and macros, see VM/SP System Product Editor User's Guide.

For more detailed usage information on CMS commands, see the VM/SP CMS User's Guide.

The following commands and subcommands exist in the CP, CMS, and XEDIT environments:

**CP**  QUERY  SET

The following commands and subcommands exist in the CMS and XEDIT environments:

**HELP**  **LOAD**  **SORT**  **XEDIT**

The following command and subcommand exists in the CP and XEDIT environments:

**RESET**

The command formats are presented in the following order:

- **Command Name**: Identifies the name of the command. The name is also included at the top of the page for easy reference.

- **Function Description**: Describes the general use of the command.
CMS Commands

- **Syntax**: Lists the syntax (format) of the command with all the possible operands that you can use.

- **Operand and Option Description**: Describes the function of each operand and option and any values that you can include.

- **Usage Notes**: Contains notes about special uses of the command, its operands, or combinations of commands or operands.

- **Examples**: Provides one or more examples to show how the command is commonly used.

- **Responses**: Describes the responses sent to the terminal, caused by execution of the command. Some responses are command responses and are not to be construed as VM/SP system messages. When the command responses are not prefixed, they are not contained in *VM/SP System Messages and Codes*.

- **Messages and Return Codes**: Lists the messages issued by the command. In some cases, the text of a message is longer than a line on the display screen. The message text may be divided in the middle of a word and continued on successive lines.

  Refer to *VM/SP System Messages and Codes* for detailed information about the messages. Refer to *VM/SP System Messages Cross-Reference* for a listing of messages.

  For some CMS commands, you may receive messages for syntax errors, that is, for specifying the command format incorrectly. These messages are grouped together and are listed in the following section.

### Command Syntax Error Messages

If you enter a command with an incorrect format, you may receive a syntax error message prefixed by DMSPCL, having a return code of 24. Check the format of the command, make the necessary correction, and enter the command again.

For example, if you issue the WRITE VSCREEN command as WRITE, you receive the message:

**DMSPCL384E** Missing modifier keyword(s)

Check the format of the WRITE VSCREEN command and enter it again, using the correct command name and operands.
<table>
<thead>
<tr>
<th>Number</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>384E</td>
<td>Missing modifier keyword(s)</td>
</tr>
<tr>
<td>385E</td>
<td>Invalid modifier keyword: keyword</td>
</tr>
<tr>
<td>386E</td>
<td>Missing operand(s)</td>
</tr>
<tr>
<td>387E</td>
<td>Missing value for operand operand</td>
</tr>
<tr>
<td>387E</td>
<td>Missing alphanumeric string for operand operand</td>
</tr>
<tr>
<td>387E</td>
<td>Missing application identifier for operand operand</td>
</tr>
<tr>
<td>387E</td>
<td>Missing character for operand operand</td>
</tr>
<tr>
<td>387E</td>
<td>Missing device address for operand operand</td>
</tr>
<tr>
<td>387E</td>
<td>Missing filename for operand operand</td>
</tr>
<tr>
<td>387E</td>
<td>Missing filetype for operand operand</td>
</tr>
<tr>
<td>387E</td>
<td>Missing execname for operand operand</td>
</tr>
<tr>
<td>387E</td>
<td>Missing exectype for operand operand</td>
</tr>
<tr>
<td>387E</td>
<td>Missing filemode for operand operand</td>
</tr>
<tr>
<td>387E</td>
<td>Missing hexadecimal number for operand operand</td>
</tr>
<tr>
<td>387E</td>
<td>Missing integer for operand operand</td>
</tr>
<tr>
<td>387E</td>
<td>Missing negative integer for operand operand</td>
</tr>
<tr>
<td>387E</td>
<td>Missing number for operand operand</td>
</tr>
<tr>
<td>387E</td>
<td>Missing positive integer for operand operand</td>
</tr>
<tr>
<td>387E</td>
<td>Missing mode for operand operand</td>
</tr>
<tr>
<td>387E</td>
<td>Missing character string for operand operand</td>
</tr>
<tr>
<td>388E</td>
<td>Invalid keyword: keyword</td>
</tr>
<tr>
<td>389E</td>
<td>Invalid operand: operand</td>
</tr>
<tr>
<td>389E</td>
<td>Invalid alphanumeric string: string</td>
</tr>
<tr>
<td>389E</td>
<td>Invalid application identifier: string</td>
</tr>
<tr>
<td>389E</td>
<td>Invalid character: character</td>
</tr>
<tr>
<td>389E</td>
<td>Invalid device address: address</td>
</tr>
<tr>
<td>389E</td>
<td>Invalid filename: filename</td>
</tr>
<tr>
<td>389E</td>
<td>Invalid filetype: filetype</td>
</tr>
<tr>
<td>389E</td>
<td>Invalid execname: execname</td>
</tr>
<tr>
<td>389E</td>
<td>Invalid exectype: exectype</td>
</tr>
<tr>
<td>389E</td>
<td>Invalid filemode: filemode</td>
</tr>
<tr>
<td>389E</td>
<td>Invalid hexadecimal number: number</td>
</tr>
<tr>
<td>389E</td>
<td>Invalid integer: integer</td>
</tr>
<tr>
<td>389E</td>
<td>Invalid negative integer: integer</td>
</tr>
<tr>
<td>389E</td>
<td>Invalid number: integer</td>
</tr>
<tr>
<td>389E</td>
<td>Invalid positive integer: integer</td>
</tr>
<tr>
<td>389E</td>
<td>Invalid mode: mode</td>
</tr>
<tr>
<td>389E</td>
<td>Invalid character string: string</td>
</tr>
<tr>
<td>390E</td>
<td>Invalid value value for operand operand</td>
</tr>
<tr>
<td>390E</td>
<td>Invalid alphanumeric string string for operand operand</td>
</tr>
<tr>
<td>390E</td>
<td>Invalid application identifier applid for operand operand</td>
</tr>
<tr>
<td>390E</td>
<td>Invalid character character for operand operand</td>
</tr>
<tr>
<td>390E</td>
<td>Invalid device address address for operand operand</td>
</tr>
<tr>
<td>390E</td>
<td>Invalid filename filename for operand operand</td>
</tr>
<tr>
<td>390E</td>
<td>Invalid filetype filetype for operand operand</td>
</tr>
<tr>
<td>390E</td>
<td>Invalid execname execname for operand operand</td>
</tr>
</tbody>
</table>
### CMS Commands

<table>
<thead>
<tr>
<th>Number</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>390E</td>
<td>Invalid exectype <code>exectype</code> for operand operand</td>
</tr>
<tr>
<td>390E</td>
<td>Invalid filemode <code>filemode</code> for operand operand</td>
</tr>
<tr>
<td>390E</td>
<td>Invalid hexadecimal number <code>number</code> for operand operand</td>
</tr>
<tr>
<td>390E</td>
<td>Invalid integer <code>integer</code> for operand operand</td>
</tr>
<tr>
<td>390E</td>
<td>Invalid negative integer <code>integer</code> for operand operand</td>
</tr>
<tr>
<td>390E</td>
<td>Invalid number <code>number</code> for operand operand</td>
</tr>
<tr>
<td>390E</td>
<td>Invalid positive integer <code>integer</code> for operand operand</td>
</tr>
<tr>
<td>390E</td>
<td>Invalid mode <code>mode</code> for operand operand</td>
</tr>
<tr>
<td>390E</td>
<td>Invalid character string <code>string</code> for operand operand</td>
</tr>
<tr>
<td>391E</td>
<td>Unexpected operand(s): operand</td>
</tr>
<tr>
<td>393E</td>
<td>Missing value for option option</td>
</tr>
<tr>
<td>393E</td>
<td>Missing alphanumeric string for option option</td>
</tr>
<tr>
<td>393E</td>
<td>Missing application identifier for option option</td>
</tr>
<tr>
<td>393E</td>
<td>Missing character for option option</td>
</tr>
<tr>
<td>393E</td>
<td>Missing device address for option option</td>
</tr>
<tr>
<td>393E</td>
<td>Missing filename for option option</td>
</tr>
<tr>
<td>393E</td>
<td>Missing filetype for option option</td>
</tr>
<tr>
<td>393E</td>
<td>Missing exectype for option option</td>
</tr>
<tr>
<td>393E</td>
<td>Missing filemode for option option</td>
</tr>
<tr>
<td>393E</td>
<td>Missing hexadecimal number for option option</td>
</tr>
<tr>
<td>393E</td>
<td>Missing integer for option option</td>
</tr>
<tr>
<td>393E</td>
<td>Missing negative integer for option option</td>
</tr>
<tr>
<td>393E</td>
<td>Missing number integer for option option</td>
</tr>
<tr>
<td>393E</td>
<td>Missing positive integer for option option</td>
</tr>
<tr>
<td>393E</td>
<td>Missing mode for option option</td>
</tr>
<tr>
<td>393E</td>
<td>Missing character string for option option</td>
</tr>
<tr>
<td>394E</td>
<td>Invalid option: option</td>
</tr>
<tr>
<td>395E</td>
<td>Invalid value <code>value</code> for option option</td>
</tr>
<tr>
<td>395E</td>
<td>Invalid alphanumeric string <code>string</code> for option option</td>
</tr>
<tr>
<td>395E</td>
<td>Invalid application identifier <code>applid</code> for option option</td>
</tr>
<tr>
<td>395E</td>
<td>Invalid character <code>character</code> for option option</td>
</tr>
<tr>
<td>395E</td>
<td>Invalid device address <code>address</code> for option option</td>
</tr>
<tr>
<td>395E</td>
<td>Invalid filename <code>filename</code> for option option</td>
</tr>
<tr>
<td>395E</td>
<td>Invalid filetype <code>filetype</code> for option option</td>
</tr>
<tr>
<td>395E</td>
<td>Invalid exectype <code>exectype</code> for option option</td>
</tr>
<tr>
<td>395E</td>
<td>Invalid filemode <code>filemode</code> for option option</td>
</tr>
<tr>
<td>395E</td>
<td>Invalid hexadecimal number <code>number</code> for option option</td>
</tr>
<tr>
<td>395E</td>
<td>Invalid integer <code>integer</code> for option option</td>
</tr>
<tr>
<td>395E</td>
<td>Invalid number <code>number</code> for option option</td>
</tr>
<tr>
<td>395E</td>
<td>Invalid negative integer <code>integer</code> for option option</td>
</tr>
<tr>
<td>395E</td>
<td>Invalid positive integer <code>integer</code> for option option</td>
</tr>
<tr>
<td>395E</td>
<td>Invalid mode <code>mode</code> for option option</td>
</tr>
<tr>
<td>395E</td>
<td>Invalid character string <code>string</code> for option option</td>
</tr>
</tbody>
</table>
ACCESS

Use the ACCESS command to identify a disk to CMS, establish a filemode letter for the files on the disk, and set up a file directory in storage. The specifications you make with the ACCESS command determine the entries in the user file directory.

The format of the ACCESS command is:

```
ACcess [vdev] mode [ / ext [ fn * [ ft * [ fm ] ]]] [ (options... [ ) ] ]
```

**Options:**
- [NOPROF]
- [ERASE]
- [SAVEONLY]
- [NODISK]

**where:**

- **vdev**
  makes available the disk at the specified virtual device address. The default value is 191. Valid addresses are 001 through 5FF for a virtual machine in basic control mode, and 001 through FFF for a virtual machine in extended control mode.

- **mode**
  assigns a one-character filemode letter to all files on the disk being accessed. This field must be specified if vdev is specified. The default value is A.

- **ext**
  indicates the mode of the parent disk. Files on the disk being accessed (vdev) are logically associated with files on the parent disk; the disk at vdev is considered a read-only extension. A parent disk must be accessed in the search before the extension. A blank must not precede or follow the slash (/).

- **fn [ft [fm]]**
  defines a subset of the files on the specified disk. Only the specified files are included in the user file directory and only those files can be read. An asterisk coded in any of these fields indicates all filenames, filetypes, or filemode numbers (except 0) are to be included. (See Usage Notes 4 and 5.) To specify a filemode use a letter and a number, for example:
ACCESS

For OS and DOS disk access restrictions, see Usage Note 12.

Options:

NOPROF
suppresses execution of a PROFILE EXEC file. This option is valid only if the ACCESS command is the first command entered after you IPL CMS. On subsequent ACCESS commands, the NOPROF option is ignored.

ERASE
specifies that you want to erase all of the files on the specified disk. This option is only valid for read/write disks. (See Usage Note 10.)

SAVEONLY
accesses the disk if a saved copy of the file directory information is available in a Discontiguous Shared Segment (DCSS). If it is not available, the disk is not accessed.

NOSAVE
accesses the disk and places the file directory information in your virtual machine. A saved copy of the file directory information in a DCSS is not used.

NODISK
lets you gain access to the CMS operating system with no disks accessed by CMS except the system disk (S-disk) and its extensions. This option is only valid if the ACCESS command is the first command you enter after you IPL CMS.

Usage Notes:

1. If you have defined disk addresses 190, 191, 192, and 19E in the VM/SP directory, or if they are defined before you IPL CMS, these disks are accessed as the S-, A-, D-, and Y-disks respectively. Following an IPL of CMS, you must issue explicit ACCESS commands to access other disks. Ordinarily, you have access only to files with a filemode number of 2 on the system disk, or S-disk. Note that you cannot specify a filemode of S with the ACCESS command.

When ACCESS is the first command issued after an IPL of the CMS system, the A-disk is not automatically defined. Another ACCESS command must be issued to define the A-disk.

2. Associated with each CMS disk is a file directory, which contains an entry for every CMS file on the disk. Specifying ACCESS without the SAVEONLY or NOSAVE options accesses the disk using a saved copy of the file directory that contains entries for only those files that you can reference. If the DCSS containing the saved copy is not available...
or is not current, ACCESS creates a file directory in your virtual machine.

If you use the CP LINK command to link to a new minidisk, issue an ACCESS command each time. Do this so that you obtain the appropriate file directory.

3. If you enter the ACCESS command and any associated operands or options at the VM READ after you IPL CMS, it must be entered in American English; you cannot use any other national language.

4. The filename, filetype, and filemode fields can only be specified for disks that are accessed as read-only extensions. Also, requesting a subset of files creates a file directory in your virtual machine. For example:

   access 195 b/a * assemble

   gives you read-only access to all the files with a filetype of ASSEMBLE on the disk at virtual address 195, and the file directory information is stored in your virtual machine. The command:

   access 190 z/a * * z1

   gives you access to all files on the system disk (190) that have a filemode number of 1.

   When you access any disk in read-only status, files with a filemode number of 0 are not accessed.

5. You can also identify a set of files on a disk by referring to a filename or filetype prefix. For example:

   access 192 c/a abc*

   accesses only those files in the disk at virtual address 192 whose filenames begin with the characters ABC. The command:

   access 192 c/a * a* c2

   gives you access to all files whose filetypes begin with an A and that have a filemode number of 2.

6. Accessing the same disk with different filemodes affects the type of access. Once a disk is accessed using a saved file directory, subsequent ACCESS commands for the same disk place the file directory in your virtual machine. For example, the command sequence

   access 19f g
   access 19f h
   access 19f i

   accesses the disk defined at address 19F with a saved file directory for the G-disk, while the H-disk and I-disk are accessed with the file directory in your virtual machine.
7. You can force a read/write disk into read-only status by accessing it as an extension of another disk or of itself, for example:

```
access 191 a/a
```

forces your A-disk into read-only status.

8. When a disk is made a read-only extension of another disk, commands that typically require or allow you to specify a filemode may search extensions of the specified disk. The exceptions to this are the LISTFILE and DISK DUMP commands. For a detailed description of read-only extensions, see the VM/SP CMS User’s Guide.

9. When you have linked to a formatted disk as read-only and it does not contain any files, you cannot access the disk. A formatted disk linked as read/write can be accessed whether or not it contains files.

10. If you enter the ERASE option by mistake, you can recover from the error as long as you have not yet written any new files onto the disk. (That is, you have not yet caused CMS to rewrite the file directory.) Reissue the ACCESS command without the ERASE option.

11. You should never attempt to access a disk in read/write status if another user already has it in read/write status; the results are unpredictable.

12. When accessing OS and DOS disks:

   a. You cannot specify filename, filetype and filemode when you access OS or DOS disks, nor can you specify any options.

   b. In order to see OS and DOS disks, you must have a read/write CMS A-disk available if you are going to use the LOAD command with the MAP option. (MAP is a default option.)

13. If two or more disks have been accessed in CMS, and CP DEFINE commands are executed that swap virtual addresses, then a subsequent RELEASE command may write the file directory on the wrong disk; for example:

```
(CMS) ACCESS 193 C
(CMS) ACCESS 198 E
(CP) DEFINE 193 293
(CP) DEFINE 198 193
(CMS) RELEASE C
```

This sequence of commands will write the file directory from 193 to 198 since the CP definitions are unknown to CMS.

14. To free an accessed disk, refer to the CMS RELEASE command.
Example:

To access a disk defined at address 498 as your B-disk, then enter the following:

```
access 498 b
```

To access a disk defined at address 498 as your B-disk only if a saved copy of the directory is available, enter the following:

```
access 498 b (saveonly)
```

Responses:

```
DMSACC723I  mode (vdev) {R/O|R/W} [-OS|-DOS]

If the specified disk is a CMS disk, this message is displayed if the disk is read-only. If the disk is in OS or DOS format, the message indicates the format, as well as whether it is a read/write or read-only disk.

DMSACC724I  vdev1 replaces mode (vdev2)

Before execution of the command, the disk represented by vdev2 was the mode disk. The disk vdev1 is now assigned that filemode letter. This message is followed by message DMSACC726I.

DMSACC725I  vdev also = mode [-OS|-DOS] disk

The disk specified by vdev is the mode disk and an ACCESS command was issued to assign it another filemode letter.

DMSACC726I  vdev mode released

The disk being accessed at virtual address vdev as a read/write disk is already accessed at a different mode. It is released from that mode. Or a disk currently accessed at mode is being replaced.
```

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

```
DMSACC003E  Invalid option: option [RC = 24]
DMSACC017E  Invalid device address vdev [RC = 24]
DMSACC048E  Invalid mode mode [RC = 24]
DMSACC059E  vdev already accessed as read/write mode disk [RC = 36]
DMSACC060E  File(s) fn1 fn2 fn3 not found; disk mode(vdev) will not be accessed [RC = 28]
DMSACC066E  option1 and option2 are conflicting options [RC = 24]
DMSACC109S  Virtual storage capacity exceeded [RC = 104]
DMSACC112S  Disk mode(vdev) device error [RC = 100]
DMSACC113S  mode(vdev) not attached [RC = 100]
```
ACCESS

DMSACC230W O/S disk--fileid and/or options specified are ignored [RC = 4]

DMSACP1078E Cannot access saved file directory for this disk [RC = 44]
ALARM VSCREEN

Use the ALARM VSCREEN command to sound the terminal alarm the next time a virtual screen is displayed in a window on the screen.

The format of the ALARM VSCREEN command is:

```
ALARM VScreen  uname
```

where:

```
uname
```

is the name of the virtual screen for which the alarm sounds.

Usage Notes:

None

Responses:

The terminal alarm sounds.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSALA386E Missing operand(s) [RC = 24]
- DMSALA388E Invalid keyword: keyword [RC = 24]
- DMSALA391E Unexpected operand(s): operand [RC = 24]
- DMSALA921E Virtual screen vscreen is not defined [RC = 28]
AMSERV

Use the AMSERV command to invoke access method services to:

- Define VSAM catalogs, data spaces, or clusters
- Alter, list, copy, delete, export or import VSAM catalogs and data sets.

The format of the AMSERV command is:

```
AMserv fn1 [(fn2)] [options...]
```

where:

`fn1` specifies the filename of a CMS file with a filetype of AMSERV that contains the access method services control statements to be executed. CMS searches all of your accessed disks, using the standard search order, to locate the file.

`fn2` specifies the filename of the CMS file that is to contain the access method services listing; the filetype is always LISTING. If `fn2` is not specified, the LISTING file will have the same name as the AMSERV input file (fn1).

The LISTING file is written to the first read/write disk in the standard search order, usually your A-disk. If a LISTING file with the same name already exists, it is replaced.

Options:

**PRINT**

spools the output listing to the virtual printer, instead of writing it to disk. If PRINT is specified, fn2 cannot be specified.
TAPIN \{18n \ TAPn\}
specifies that tape input is on the tape drive at the address indicated by 18n or TAPn. n may be 1, 2, 3, or 4, indicating virtual addresses 181 through 184, respectively.

TAPOUT \{18n \ TAPn\}
specifies that tape output should be written to the tape drive at the address indicated by 18n or TAPn. n may be 1, 2, 3, or 4, indicating virtual addresses 181 through 184, respectively.

Note: If both TAPIN and TAPOUT are specified, their virtual device addresses must be different.

Usage Notes:

1. To create a job stream for access method services, you can use an editor to create a file with the filetype of AMSERV. The editor automatically sets input margins at columns 2 and 72.

2. Restrictions placed on VSAM usage in CMS are listed in this publication in “Appendix A: VSE/VSAM Functions Not Supported in CMS” and “Appendix B: OS/VS Access Method Services and VSAM Functions Not Supported in CMS.” Refer to Using VSE/VSAM Commands and Macros for a description of access method services control statements format and syntax.

3. You must use the DLBL command to identify the master catalog. Disk input and output files may also require a DLBL command. For more information on DLBL requirements for AMSERV see VSE/VSAM Programmer’s Reference.

4. When you use tape input and/or output with the AMSERV command, you are prompted to enter the ddnames; a maximum of 16 ddnames are allowed for either input or output. The ddnames can each have a maximum of seven characters and must be separated by blanks.

   While using AMSERV, only one tape at a time can be attached for either input or output. If you enter more than one tape ddname, specify the tape files in the sequence they are used in the input stream.

5. A CMS format variable file cannot be used directly as input to AMSERV functions as a variable (V) or variable blocked (VB) file because the standard variable CMS record does not contain the BL and RL headers needed by the variable record modules. If these headers are not included in the record, errors will result.

   Most files placed on the CMS disk by AMSERV will show a RECFM of V, even if the true format is fixed (F), fixed blocked (FB), undefined (U), variable or variable blocked. The programmer must know the true
AMSERV

format of the file he is trying to use with the AMSERV command and access it properly, or errors will result.

6. If an AMSERV command abnormally terminates or you issue HX to terminate an AMSERV command, the AMSERV environment may not be reset correctly. If a subsequent AMSERV abends, you must re-IPL CMS.

7. If the AMSERV input file contains the access method services control statement “DELETE” with “IGNORERROR,” the PRINT option on the AMSERV command must be used to send the listing to the virtual printer.

8. The density used for tapes is the default density of the tape drive or the density of the tape mounted on it.

Additional Note for CMS/DOS Users:

AMSERV internally issues an ASSGN command for SYSIPT and locates the source file; therefore, you do not need to assign it. If you use the TAPIN or TAPOUT options, AMSERV also issues ASSGN commands for the tape drives (assigning logical units SYS004 and SYS005).

Any other assignments and DLBL definitions that are in effect when you invoke the AMSERV command are saved and restored when the command completes executing.

Example:

If you enter the AMSERV command with no options, for example:

```
amserv myfile
```

you will get a CMS file with a filename of MYFILE and a FILETYPE of LISTING. LISTING files take up considerable disk space, so you should erase them as you no longer need them. If you do not want to create a disk file from the listing, you can have the output spooled to the virtual printer. The following command:

```
amserv myfile (print
```

spools the output to the virtual printer and no disk file is created.

Responses:

The CMS ready message indicates that access method services has completed processing. If access method services completed with a nonzero return code, the return code is shown in the ready message. Examine the LISTING file created by AMSERV to determine the results of access method services processing.
The publication *VSE/VSAM Messages and Codes* lists and explains the messages access method services generates and the associated reason codes.

DMSAMS367R Enter tape {input|output} DDNAMEs:

This message prompts you to enter the ddnames associated with the tape files.

DMSAMS722I File fn LISTING fm will hold AMSERV output

This message is displayed when you enter a fn2 operand or when the listing is not being written on your A-disk; it tells you the file identifier of the output listing.

Messages and Return Codes:

- DMSAMS001E No filename specified [RC = 24]
- DMSAMS002E File FNAME1 AMSERV not found [RC = 28]
- DMSAMS003E Invalid option: option [RC = 24]
- DMSAMS006E No read/write disk accessed for FNAME2 LISTING [RC = 36]
- DMSAMS007E File FNAME1 AMSERV fm not fixed, 80-character records [RC = 32]
- DMSAMS065E option option specified twice [RC = 24]
- DMSAMS066E option1 and option2 are conflicting options [RC = 24]
- DMSAMS070E Invalid parameter parameter [RC = 24]
- DMSAMS109S Virtual storage capacity exceeded [RC = 104]
- DMSASN113S Tapn(udev) not attached [RC = 100]
- DMSAMS136S Unable to load IDCMAS [RC = 104]
- DMSAMS228E No DDNAME entered [RC = 24]
ASSEMBLE

Use the ASSEMBLE command to invoke the assembler to assemble a file containing source statements. Assembler processing and output is controlled by the options selected.

The format of the ASSEMBLE command is:

<table>
<thead>
<tr>
<th>Assemble</th>
<th>( fn ) [ ( \text{options}...[\text{options}] ) ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Listing Control Options:</td>
<td></td>
</tr>
<tr>
<td>ALOGIC</td>
<td>ESD</td>
</tr>
<tr>
<td>NOALOGIC</td>
<td>NOESD</td>
</tr>
<tr>
<td>LIST</td>
<td>NOLIST</td>
</tr>
<tr>
<td>NOLIBMAC</td>
<td></td>
</tr>
<tr>
<td>XREF (FULL)</td>
<td>XREF (SHORT)</td>
</tr>
<tr>
<td>NOXREF</td>
<td></td>
</tr>
<tr>
<td>Output Control Options:</td>
<td></td>
</tr>
<tr>
<td>DECK</td>
<td>OBJECT</td>
</tr>
<tr>
<td>NODECK</td>
<td>NOOBJECT</td>
</tr>
<tr>
<td>SYSTEM Options:</td>
<td></td>
</tr>
<tr>
<td>NUMBER</td>
<td>STMT</td>
</tr>
<tr>
<td>NONUM</td>
<td>NOSTMT</td>
</tr>
<tr>
<td>Other Assembler Options:</td>
<td></td>
</tr>
<tr>
<td>ALIGN</td>
<td>BUFSIZE ((\text{MIN}))</td>
</tr>
<tr>
<td>NOALIGN</td>
<td>BUFSIZE ((\text{STD}))</td>
</tr>
<tr>
<td></td>
<td>BUFSIZE ((\text{MAX}))</td>
</tr>
<tr>
<td>YFLAG</td>
<td>SYSPARM ((\text{string}))</td>
</tr>
<tr>
<td>NOYFLAG</td>
<td>SYSPARM ((\text{not}))</td>
</tr>
</tbody>
</table>

where:
fn
is the filename of the source file to be assembled and/or the filename of assembler output files. The file must have fixed-length, 80-character records. By default, the assembler expects a CMS file with a filetype of ASSEMBLE.

Listing Control Options:

The list below describes the assembler options you can use to control the assembler listing. The default values are underscored.

**ALOGIC**
lists conditional assembly statements in open code.

**NOALOGIC**
suppresses the ALOGIC option.

**ESD**
lists the external symbol dictionary (ESD).

**NOESD**
suppresses the printing of the ESD listing.

**FLAG** ($nnn$)
**FLAG (0)**
does not include diagnostic messages and MNOTE messages below severity code $nnn$ in the listing. Diagnostic messages can have severity codes of 4, 8, 12, 16, or 20 (20 is the most severe); and MNOTE message severity codes can be between 0 and 255. For example, **FLAG (8)** suppresses diagnostic messages with a severity code of 4 and MNOTE messages with severity codes of 0 through 7.

**LINECOUN** ($nn$)
**LINECOUN (55)**
$nn$ specifies the number of lines to be listed per page.

**LIST**
produces an assembler listing. Any previous listing is erased.

**NOLIST**
does not produce an assembler listing. However, any previous listing is still erased. This option overrides ESD, RLD, and XREF.

**MCALL**
lists the inner macro instructions encountered during macro generation following their respective outer macro instructions. The assembler assigns statement numbers to these instructions. The MCALL option is implied by the MLOGIC option; NOMCALL has no effect if MLOGIC is specified.

**NOMCALL**
suppresses the MCALL option.
MLOGIC
lists all statements of a macro definition processed during macro
generation after the macro instruction. The assembler assigns
statement numbers to them.

NOMLOGIC
suppresses the MLOGIC option.

RLD
produces the relocation dictionary (RLD) as part of the listing.

NORLD
does not print the relocation dictionary.

LIBMAC
lists the macro definitions read from the macro libraries and any
assembler statements following the logical END statement. The
logical END statement is the first END statement processed during
macro generation. It may appear in a macro or in open code; it may
even be created by substitution. The assembler assigns statement
numbers to the statements that follow the logical END statement.

NOLIBMAC
suppresses the LIBMAC option.

XREF (FULL)
includes in the assembler listing a cross-reference table of all symbols
used in the assembly. This includes symbols that are defined but
never referenced. The assembler listing also contains a
cross-reference table of literals used in the assembly.

XREF (SHORT)
includes in the assembler listing a cross-reference table of all symbols
that are referenced in the assembly. Any symbols defined but not
referenced are not included in the table. The assembler listing
contains a cross-reference table of literals used in the assembly.

NOXREF
does not print the cross-reference tables.

PRINT
PR
writes the LISTING file to the printer.

NOPRINT
NOPR
suppresses the printing of the LISTING file.
DISK
DI
places the LISTING file on a virtual disk, searching for the disk in the following order:

1. An accessed READ/WRITE disk from which the assemble source is read.
2. The READ/WRITE “parent” disk, if the disk from which the assemble source is read is a READ/ONLY extension.
3. The primary A disk with READ/WRITE access.

If the primary A disk is not accessed R/W and the first two items do not apply, then an error message is generated.

Output Control Options:

The output control options are used to control the object module output of the assembler.

DECK
writes the object module on the device specified on the FILEDEF statement for PUNCH. If this option is specified with the OBJECT option, the object module is written both on the PUNCH and TEXT files.

NODECK
suppresses the DECK option.

OBJECT
OBJ
writes the object module on the device, which is specified by the FILEDEF statement for TEXT, and erases any previous object modules. If this option is specified with the DECK option, the object module is written on the two devices specified in the FILEDEF statement for TEXT and PUNCH.

NOOBJECT
NOOBJ
does not create the object module. However, any previous object module is still erased.

TEST
includes the special source symbol table (SYM cards) in the object module. This option should not be used for programs to be run under CMS because the SYM cards are not acceptable to the CMS LOAD and INCLUDE commands.
NOTEST
   does not produce SYM cards.

SYSTERM Options:

The SYSTERM options are used to control the SYSTERM file associated with your assembly.

NUMBER
NUM
   writes the line number field (columns 73-80 of the input records) in the SYSTERM listing for statements for which diagnostic information is given. This option is valid only if TERMINAL is specified.

NONUM
   suppresses the NUMBER option.

STMT
   writes the statement number assigned by the assembler in the SYSTERM listing for statements for which diagnostic information is given. This option is valid only if TERMINAL is specified.

NOSTMT
   suppresses the STMT option.

TERMINAL
TERM
   writes the diagnostic information on the SYSTERM data set. The diagnostic information consists of the diagnosed statement followed by the error message issued.

NOTERM
   suppresses the TERMINAL option.

Other Assembler Options:

The following options allow you to specify various functions and values for the assembler.

ALIGN
ALGN
   aligns all data on the proper boundary in the object module; for example, an F-type constant is aligned on a fullword boundary. In addition, the assembler checks storage addresses used in machine instructions for alignment violations.
NOALIGN

NOALGN
does not align data areas other than those specified in CCW
instructions. The assembler does not skip bytes to align constants on
proper boundaries. Alignment violations in machine instructions are
not diagnosed.

BUFSIZE (MIN)

uses the minimum buffer sizes (790 bytes) for each of the utility data
sets (SYSUT1, SYSUT2, and SYSUT3). Storage normally used for
buffers is allocated to work space. Because more work space is
available, more complex programs can be assembled in a given virtual
storage size; but the speed of the assembly is substantially reduced.

BUFSIZE (STD)

chooses the buffer size that gives optimum performance. The buffer
size depends on the amount of virtual storage. Of the assembler
working storage in excess of minimum requirements, 37% is allocated
to the utility data set buffers and the rest to macro generation
dictionaries.

BUFSIZE (MAX)

is useful when many macros and/or large macros are used in an
assembly. The assembler uses up to 15 save areas for input records
and saves the areas according to their frequency of use, optimizing the
macro generation phase. This option has no effect unless a large
enough region is available. The number of allocated save areas is
printed on the statistics page of the assembler listing. Refer to the
OS/VS-VM/370 Assembler Programmer’s Guide, Appendix E for a
description of the effects of BUFSIZE.

RENT

checks your program for a possible violation of program
re-enterability. Code that makes your program nonre-enterable is
identified by an error message.

NORENT

suppresses the RENT option.

YFLAG

does not suppress the warning messages that indicate that relocatable
Y-type address constants have been declared.

NOYFLAG

suppresses the warning messages that indicate relocatable Y-type
constants have been declared.

SYSPARM \{(string)\}

\{0\}

\{?\}

passes a character value to the system variable symbol, SYSPARM.
The variable (string) may be up to 100 characters long, and may not
contain any blanks or parentheses. If you want to enter a string
containing blanks or parentheses, use the SYSPARM (?) format. With the SYSPARM (?) format, CMS prompts you with the message:

ENTER SYSPARM:

You can enter up to 100 characters. SYSPARM () enters a null string of characters.

Note: If ASSEMBLE is called as a command, the SYSPARM information is translated to uppercase.

WORKSIZE \{(2048K)\} \{(nnnnnK)\}
allows the user to delimit the use of region space. The specified value does not include the space for modules and system areas. The allowed range is from 32K to 10240K. The virtual machine size must be large enough to accommodate the WORKSIZE option; otherwise the option has no effect.

Usage Notes:

1. When you issue the ASSEMBLE command, default FILEDEF commands are issued for assembler data sets. You may want to override these with explicit FILEDEF commands. The ddnames used by the assembler are:

   ASSEMBLE (SYSIN input to the assembler)
   TEXT (SYSLIN output of the assembler)
   LISTING (SYSPRINT output of the assembler)
   PUNCH (SYSPUNCH output of the assembler)
   CMSLIB (SYSLIB input to the assembler)
   SYSUT1 (workfile of the assembler)
   SYSUT2 (workfile of the assembler)
   SYSUT3 (workfile of the assembler)

   The default FILEDEF commands issued by the assembler for these ddnames are:

   FILEDEF ASSEMBLE DISK fn ASSEMBLE fm (RECFM FB LRECL 80 BLOCK 800
   FILEDEF TEXT DISK fn TEXT fm
   FILEDEF LISTING DISK fn LISTING fm (RECFM FBA BLOCK 1210
   FILEDEF PUNCH PUNCH
   FILEDEF CMSLIB DISK CMSLIB MACLIB * (RECFM FB LRECL 80 BLOCK 800
   FILEDEF SYSUT1 DISK fn SYSUT1 fm4 (BLOCK 7294 AUXPROC asmproc
   FILEDEF SYSUT2 DISK fn SYSUT2 fm4 (BLOCK 7294 AUXPROC asmproc
   FILEDEF SYSUT3 DISK fn SYSUT3 fm4 (BLOCK 7294 AUXPROC asmproc
At the completion of the ASSEMBLE command, all FILEDEFs that do not have the PERM option are erased.

2. If you want to use any CMS macro or copy libraries during an assembly, issue the GLOBAL command to identify the macro libraries before you issue the ASSEMBLE command. For example:

```
global maclib dmssp cmslib osmacro testlib
```

identifies the MACLIB files named CMSLIB, DMSSP, OSMACRO, and TESTLIB.

3. To use OS macro libraries during an assembly, issue the FILEDEF command for the OS data set. Use a ddname of CMSLIB and assign a CMS file identifier; the filetype must be MACLIB, and you must use the filename on the GLOBAL command line. For example:

```
filedef cmslib disk oldtest maclib c dsn oldtest macros
global maclib oldtest
```

assigns the OS data set OLDTEST.MACROS, on the disk accessed as mode C, a CMS fileid of OLDTEST MACLIB and identifies it as the macro library to be used during assembly.

4. You cannot assemble programs using DOS macros from the DOS/VS source statement libraries under CMS/DOS. You should use the SSERV, ESERV, and MACLIB commands to create CMS MACLIBs to contain DOS macros for assembly under CMS/DOS. See VM/SP CMS for System Programming for examples.

5. You need not make any logical assignments for input or output files when you use the assembler under CMS/DOS. File definitions are assigned by default under CMS, as described in Usage Note 1.

6. Usage information about the VM/SP Assembler Language and assembler options can be found in OS/VS and VM/370 Assembler Programmer’s Guide and OS/VS, DOS/VS, and VM/370 Assembler Language.

7. ASSEMBLE uses the extended plist for processing the SYSPARM parameter. If you are calling ASSEMBLE from an assembler language program and using SYSPARM, you should supply an extended plist. VM/SP CMS for System Programming has more information on how an assembler language program can supply an extended plist.
Example:

Specifying the following:
assemble myfile (print

assembles the assembler language source file MYFILE ASSEMBLE into object module format and directs the output listing to printer.

Messages and Return Codes:

For the messages and return codes associated with the ASSEMBLE command, see the OS/VS and VM/370 Assembler Programmer's Guide.
ASSGN

Use the ASSGN command in CMS/DOS to assign or unassign a system or programmer logical unit for a virtual I/O device.

The format of the ASSGN command is:

```
ASSGN SYSxxx [Reader
PUnch
PRInnert
Terminal]
TAP [n]
mode
[options... [ ] ]
```

where:

SYSxxx

specifies the system or programmer logical unit to be assigned to a particular physical device. SYS000 through SYS241 are valid programmer logical units in CMS/DOS; they may be assigned to any valid device. The system logical units you may assign, and the devices to which they may be assigned, are:

**SYSxxx Valid Assignments**

- SYSRDR Reader, disk, tape
- SYSIPT Reader, disk, tape
- SYSIN Reader, disk, tape
- SYSPCH Punch, disk, tape
- SYSLST Printer, disk, tape
- SYSLOG Terminal, printer
- SYSOUT Tape
- SYSSLB Disk
- SYSRLB Disk
- SYSCLB Disk
- SYSCAT Disk

The assignment of a system logical unit to a particular device type must be consistent with the device type definition for the file in your program.
**ASSGN**

**Reader**
- is the spooled card reader (card reader I/O must not be blocked).

**PUnc**
- is the spooled punch.

**PRinter**
- is the spooled printer.

**Terminal**
- is your terminal (terminal I/O must not be blocked).

**TAP[n]**
- is a magnetic tape. \( n \) is the symbolic number of the tape drive. It is either 1, 2, 3, or 4, representing virtual addresses 181, 182, 183, and 184, respectively. If \( n \) is omitted, TAP1 is assumed.

**mode**
- specifies the one-character mode letter of the disk being assigned to the logical unit (SYSxxx). The disk must be accessed when the ASSGN command is issued. SYSRDR, SYSIPT, and SYSIN cannot be assigned to a DOS-formatted FB-512 disk.

**IGN**
- (ignore) specifies that any attempt to read from the specified device results in an end-of-file indication; any attempt to write to the device is ignored. IGN is not valid when associated with SYSRDR, SYSIPT, SYSIN, or SYSCLB.

**UA**
- indicates that the logical unit is to be unassigned. When you release a disk for which an assignment is active, it is automatically unassigned.

**Options:**

**UPCASE**
- translates all terminal input data to uppercase.

**LOWCASE**
- retains all terminal input data as keyed in.

**7TRACK**
- is the tape setting.

**9TRACK**
- is the tape setting.

**TRTCH \( a \)**
- refers to the tape recording technique for 7-track tapes. Use the following chart to determine the value of \( a \).
ASSGN

<table>
<thead>
<tr>
<th>a</th>
<th>Parity</th>
<th>Converter</th>
<th>Translator</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>odd</td>
<td>off</td>
<td>off</td>
</tr>
<tr>
<td>OC</td>
<td>odd</td>
<td>on</td>
<td>off</td>
</tr>
<tr>
<td>OT</td>
<td>odd</td>
<td>off</td>
<td>on</td>
</tr>
<tr>
<td>E</td>
<td>even</td>
<td>off</td>
<td>off</td>
</tr>
<tr>
<td>ET</td>
<td>even</td>
<td>off</td>
<td>on</td>
</tr>
</tbody>
</table>

DEN *den*  
is tape density: *den* can be 200, 556, 800, 1600, or 6250 bits per inch (bpi). If 200 or 556 are specified, 7TRACK is assumed. If 800, 1600, or 6250 are specified, 9TRACK is assumed. (See Usage Note 8.)

Usage Notes:

1. When you enter the CMS/DOS environment with the command SET DOS ON, SYSLOG is assigned by default to TERMINAL. If you specify the mode letter of the VSE system residence on the SET DOS ON command line, SYSRES is assigned to that disk mode.

2. You cannot assign any of the following VSE system logical units with the ASSGN command:

   SYSRES  SYSLNK  SYSDMP  
   SYSCTL  SYSREC

3. If you assign the logical unit SYSIN to a virtual device, SYSRDR and SYSIPT are also assigned to that device. If you make a logical assignment for SYSOUT, both SYSLST and SYSPCH are assigned.

4. To obtain a list of current assignments, use the LISTIO command.

5. To cancel all current assignments (that is, to unassign them), you can enter, in succession, the commands:

   set dos off  
   set dos on [mode]

6. If you want to access VSE private libraries, you must assign the logical units SYSSLB (source statement library), SYSRLB (relocatable library), and SYSCLB (core image library), and you must issue the DLBL command to establish a file definition.

7. An assignment to disk (mode) should be accompanied by a DLBL command that provides the disk file identification.

8. If no tape options are specified, the default for a 7-track tape is 800 bpi, data converter off, translator off, and odd parity. If the tape is 9-track, the density defaults to the highest density of the tape drive. 1600 bpi is the default for 9-track 800/1600 dual-density tapes and 6250 bpi is the default for 9-track 1600/6250 dual-density tapes. If the tape drive is
phase-encoded, density defaults to the density of the tape. If the tape drive is NRZI, the reset condition is 800 bpi.

9. 8809 tape drives require the 9TRACK and DEN 1600 options. These are the default options; it is not necessary to state them explicitly.

10. Assignment of Programmer Logical units to 'T' and 'R' is restricted to terminal and reader respectively.

Example:

To assign logical unit SYS010 to a the B-disk, enter the following:

assign sys010 b

Responses:

None.

Messages and Return Codes:

- **DMSASN003E** Invalid option: option [RC = 24]
- **DMSASN027E** Invalid device devtype [for SYSaaa] [RC = 24]
- **DMSASN028E** No logical unit specified [RC = 24]
- **DMSASN029E** Invalid parameter parameter in the option option field [RC = 24]
- **DMSASN035E** Invalid tape mode [RC = 24]
- **DMSASN050E** Parameter missing after SYSaaa [RC = 24]
- **DMSASN065E** option option specified twice [RC = 24]
- **DMSASN066E** option1 and option2 are conflicting options [RC = 24]
- **DMSASN069E** Disk mode not accessed [RC = 36]
- **DMSASN070E** Invalid parameter parameter [RC = 24]
- **DMSASN087E** Invalid assignment of SYSaaa to device devtype [RC = 24]
- **DMSASN090E** Invalid device class devclass for devtype [RC = 36]
- **DMSASN099E** CMS/DOS environment not active [RC = 40]
- **DMSASN113E** Tapn(vdev) not attached [RC = 100]
CATCHECK

As a CMS VSAM user (with or without DOS set ON), you can use the CATCHECK command to invoke the VSE/VSAM Catalog Check Service Aid to verify a complete catalog structure. CATCHECK produces a print file containing the catalog analysis. If you do not specify a catalog name with CATCHECK, it uses the default catalog specified via the DLBL command.

The format of the CATCHECK command is:

```
CATCHECK [catname
            [catname/password]
```

where:

catname

is the catalog name of the catalog to be checked. The name can be a maximum of 44 characters and must follow the VSE/VSAM catalog naming conventions.

password

is the password for the catalog “catname” as specified when the catalog was defined. The maximum length is 8 characters. If you specify the password, you must put a slash between the catalog name and the password.

Usage Notes:

1. If a catalog name is not specified on the command line, the default catalog is used. The default catalog is the job catalog identified via a ddname of “IJSYSUC” on the DLBL command. If a job catalog was not specified, the default catalog name will be the master catalog identified via the ddname of “IJSYSCT” on the DLBL command.

2. When a catalog name is specified, a DLBL need not be issued for the catalog if it is not the master catalog. A DLBL for the master catalog must always be in effect when running VSAM.

3. The output must always go to the virtual printer.

4. CATCHECK uses the extended plist for processing the catname/password parameter. If you are calling CATCHECK from an assembler language program and using catname or catname/password, you should supply an extended plist. VM/SP CMS for System
Example:

If you have only issued a DLBL command for the master catalog, then specifying the following:

catch check private.cat1

produces a print file containing the catalog analysis for PRIVATE.CAT1 catalog.

Messages and Return Codes:

- DMSCCK803E  Invalid parameter specification [RC = 4]
- DMSCCK804S  Error establishing CMS/DOS environment [RC = 8]
- DMSCCK805S  Error assigning output to printer [RC = 12]
- DMSCCK806S  VSE/VSAM phase IKQVCHK not found [RC = 16]
- DMSCCK807S  Error encountered issuing ASSGN for catalog [RC = 20]

*Programming* has more information on how an assembler language program can supply an extended plist.
CLEAR VSCREEN

Use the CLEAR VSCREEN command to erase data in a virtual screen.

The format of the CLEAR VSCREEN command is:

```
CLEAR VSCREEN vname
```

*where:*

vname

is the name of the virtual screen to be cleared.

Usage Notes:

1. The CLEAR VSCREEN command
   - Clears the scrollable data area by overwriting the data buffer with nulls.
   - Purges data in the queue.
   - Reconnects all windows connected to the virtual screen to the top of the virtual screen.
   - Leaves the reserved areas and cursor position unchanged.

2. The field attribute buffer and the extended attribute buffers for color, extended highlighting (exthi) and Programmed Symbol (PS) sets are reset to the default attributes as specified on the DEFINE VSCREEN and SET VSCREEN commands.

Responses:

None.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSCLR386E  Missing operand(s) [RC = 24]
- DMSCLR388E  Invalid keyword: keyword  [RC = 24]
- DMSCLR391E  Unexpected operand(s): operand  [RC = 24]
- DMSCLR921E  Virtual screen vname is not defined  [RC = 28]
- DMSCLR928E  Command is not valid for virtual screen vname  [RC = 12]
CLEAR WINDOW

Use the CLEAR WINDOW command to scroll past all data in the virtual screen to which the window is connected so that no scrollable data is displayed in the window.

The format of the CLEAR WINDOW command is:

```
CLEAR WINDOW [wname]
```

where:

- `wname` is the name of the window that is cleared. An "=" indicates that the topmost window is cleared. If this operand is not specified, "=" is assumed.

Usage Notes:

1. CLEAR WINDOW is valid only when the window is connected to a virtual screen (see HIDE WINDOW and SHOW WINDOW).

2. If the window you want to clear is variable size, the window is not displayed when the physical screen is refreshed.

3. If you are writing output (see the WRITE VSCREEN command), CLEAR WINDOW provides a means for starting output at the top of the window. It does not erase data in the virtual screen but instead works like scrolling. CLEAR WINDOW positions the window at the line following the last data line in the virtual screen. When output is written, lines of data are appended to the virtual screen and are displayed starting at the first nonreserved line of the window.

4. In a System Product editor session, you cannot clear the window that the System Product editor is using.
**Responses:**

None.

**Messages and Return Codes:**

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSCLR386E Missing operand(s) [RC = 24]
- DMSCLR388E Invalid keyword: keyword [RC = 24]
- DMSCLR391E Unexpected operand(s): operand [RC = 24]
- DMSCLR921E Window wname is not defined [RC = 28]
- DMSCLR929E Window wname is not connected to a virtual screen [RC = 36]
CMDCALL

CMDCALL

Use the CMDCALL command to convert EXEC 2 extended PLIST function calls to CMS extended or standard PLIST command calls.

The format of the CMDCALL command is:

```
CMDCALL [cmd [operand1 operand2 ... operandn]]
```

**where:**

`cmd`

is the command that is to be invoked with the CMS extended PLIST, indicating invocation as a command, rather than as a function.

`operand1 operand2 ...`

are the operands to be passed with the command.

**Usage Notes:**

1. The extended PLIST and the standard CMS PLIST are adjusted for the command or function called, and that command or function is invoked via SVC 202.

2. If an extended PLIST is not available, the command or function called is invoked only with the standard PLIST adjusted for the command or function called.

3. CMDCALL invoked with no calling command or function returns a return code of zero. Otherwise, the return code is that of the command invoked via the CMDCALL function.

**Example:**

For example, if, from within your EXEC 2 EXEC, you want to erase the file TEST EXEC from your A-disk, you would specify the following:

```
CMDCALL erase test exec a
```
CMSBATCH

Use the CMSBATCH command to invoke the CMS batch facility. Instead of compiling or executing a program interactively, virtual machine users can transfer jobs to the virtual card reader of an active CMS batch virtual machine. This frees their terminals for other work.

The format of the CMSBATCH command is:

```
CMSBATCH [sysname]
```

where:

sysname
is the eight-character identification of the saved system that is specifically generated for CMS batch operations via the CP SAVESYS command and the NAMESYS macro. Refer to the VM/SP CMS for System Programming publication for details on SAVESYS and NAMESYS use.

Note: If sysname is not supplied on the command line, then the system that the system operator is currently logged onto becomes the CMS batch virtual machine.

Usage Notes:

1. The CMSBATCH command must be invoked immediately after an IPL of the CMS system. Alternatively, BATCH may be specified following the PARM operand on the IPL command line.

2. Do not issue the CMSBATCH command if you use a virtual disk at address 195; the CMS batch virtual machine erases all files on the disk at address 195.

3. For a description of how to send jobs to the CMS batch virtual machine, see the VM/SP CMS User’s Guide. For an explanation of setting up a batch virtual machine, see the VM/SP Operator’s Guide.

4. The CMS batch virtual machine can be utilized by personnel who do not have access to a terminal or a virtual machine. This is accomplished by submitting jobs via the real card reader. For details on this, see the VM/SP CMS User’s Guide.

5. If the CMSBATCH command encounters recursive abends, the message “CMSBATCH system ABEND” appears on the system operator’s console.
CMSBATCH

Messages and Return Codes:

DMSBTTB100E  No batch processor available [RC = 40]
DMSBTTB101E  Batch not loaded [RC = 88]
DMSBTP105E   No job card provided
DMSBTP106E   /JOB card format invalid
DMSBTP107E   CP/CMS command command [,devtype] not allowed [RC = 100]
DMSBTP108E   /SET card format invalid [RC = 88]
DMSBTP109E   {CPU|Printer|Punch} limit exceeded
Use the CMSSERV command to start Enhanced Connectivity Facilities communications between your VM/SP host system and your workstation (IBM Personal Computer).

The format of the CMSSERV command is:

```
CMSSERV
```

### Usage Notes:

1. When you enter CMSSERV, the CMSSERV panel is displayed. If you enter CMSSERV and have not previously started your workstation communications program, you’ll be prompted to do so. From the CMSSERV panel, you can:

   - Swap (jump) to your workstation (IBM Personal Computer) environment to use the services of Enhanced Connectivity Facilities.

   You can also:

   - Enter any command on the command line that you can enter from CMS.
   - Press PF3 to end the communications program.


2. If you want to issue CMSSERV from an EXEC program, you should precede it with the EXEC command; that is, specify

```
exec cmsserv
```
CMSSERV

Messages:

DMSDFT639E  Error in routine routine; return code was nnnn
DMSDFT716E  SRPI subcommand environment was not found.
DMSDFT718E  Unable to link to work station.
DMSDFT719E  Work station communications not active.
DMSDFT720E  No longer linked to work station; error code was nnn
DMSDFT812E  Input was ignored.

Return Codes:

0  Normal termination.
38  CMSSERV was called to start communications, but CMSSERV communications were already started.
40  General internal CMS error (for example, DMSDFT not a nucleus extension or error trying to NUCXLOAD DMSDFT).
55  Enhanced Connectivity Facilities communications error.
100  Console usage error.
104  Storage error.
nnnn  Propagated return code from routine (accompanies message DMSDFT639E).
COMPARE

Use the COMPARE command to compare two CMS disk files of fixed- or variable-length format on a record-for-record basis and to display dissimilar records at the terminal.

The format of the COMPARE command is:

```
COMPARE fileid1 fileid2 [(option...)]
```

Option: COL [mmm[-]nnn] lrecl

where:

fileid1 fileid2
are the file identifiers of the files to be compared. An equal sign may be coded for one or more of the file identifiers for fileid2 in any combination except '=' = '='. All three file identifiers (filename, filetype, filemode) must be specified for each fileid. An equal sign (=) coded in fileid2 implies that the file identifier in that position is identical to the corresponding file identifier in fileid1.

Option:

COL [mmm[-]nnn] lrecl

defines specific columns to be compared. The comparison begins at position mmm of each record. The comparison proceeds up to and including column nnn. The hyphen (-) may be used in place of a blank if the total number of characters required for mmm-nnn is not more than eight (maximum parameter length). If column nnn is specified, the hyphen may not follow or precede a blank. If column nnn is not specified, the default ending position is the last character of each record (the logical record length).
Usage Notes:

1. To find out whether two files are identical, enter both file identifications, as follows:
   
   ```
   compare testl assemble a testl assemble b
   ```
   
   or
   
   ```
   compare testl assemble a = = b
   ```
   
   Any records that do not match are displayed at the terminal.

2. To stop the display of dissimilar records, use the CMS Immediate command HT.

3. If a file does not exist on a specified disk, the read-only extensions of that disk are also searched. The complete fileids of the files being compared are displayed in message DMSCMP179I.

Responses:

```
DMSCMP179I   Comparing fn ft fm with fn ft fm

This message identifies the files being compared. If the files are the same (in the columns indicated), this message is followed by the CMS ready message. If any records do not match, the records are displayed. When all dissimilar records have been displayed the message DMSCMP209W is issued.
```

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

```
DMSCMP002E   File fn [ft [fm]] not found [RC = 28]
DMSCMP003E   Invalid option: option [RC = 24]
DMSCMP005E   No column specified [RC = 24]
DMSCMP009E   Column col exceeds record length [RC = 24]
DMSCMP010E   Premature EOF on file fn ft [fm] [RC = 40]
DMSCMP011E   Conflicting file formats [RC = 32]
DMSCMP019E   Identical fileids [RC = 24]
DMSCMP029E   Invalid parameter parameter in the option option field [RC = 24]
DMSCMP054E   Incomplete fileid specified [RC = 24]
DMSCMP062E   Invalid * in fileid [RC = 20]
DMSCMP069E   Disk mode not accessed [RC = 36]
DMSCMP104S   Error nn reading file fn ft fm from disk [RC = 100]
DMSCMP109S   Virtual storage capacity exceeded [RC = 104]
DMSCMP209W   Files do not compare [RC = 4]
DMSCMP211E   Column fields out of sequence [RC = 24]
```
CONVERT COMMANDS

Use the CONVERT COMMANDS command to convert a CMS file containing Definition Language for Command Syntax (DLCS) statements into an internal form for the parsing facility.

The format of the CONVERT COMMANDS command is:

```
CONVert COMmands [ fn [ ft [ DLCS [ fm * ] ] ] [ (options... [ ] ) ] ]
```

**Options:**

**SYStem**
- Specifies that only valid system functions and subsets are in the file.
- This is the default.

**USER**
- Specifies that valid user functions and/or system functions and their subsets are in the file. You must load user functions as a nucleus extension for CONVERT COMMANDS to process them correctly.

**where:**

- **fn**
  - is the filename of the file to be converted. When editing a file, omit the filename (and filetype and filemode) to convert the active file.

- **ft**
  - is the filetype of the file to be converted. If you do not specify ft, the default of DLCS is assumed.

- **fm**
  - is the filemode of the file to be converted. The default for fm is an asterisk (*), which means the first file in the disk search order that satisfies the filename and filetype qualifications is converted.
CONVERT COMMANDS

**ALL**

specifies that the file contains user functions or subset values for user functions. They are not checked for validity.

**Check**

checks only the contents of the DLCS file for validity and does not produce an output text deck.

**OUTmode**

specifies the filemode of the disk to which the converted output files are written. The `fm` may be any read/write disk that you have accessed. If you omit this parameter or specify an asterisk (*), the default is the first read/write disk in the disk search order. OUTMODE * is the default if CHECK or OUTMODE `fm` is not specified.

**STACK**

causes the fileids of the output files to be placed in the program stack. The stacked line has the format:

```
fn1 ft1 fm1 fn2 ft2 fm2
```

The `fn1 ft1 fm1` is the fileid of the file to be converted (DLCS file). The `fn2 ft2 fm2` is the fileid of the output file containing translations.

The information is stacked either FIFO (first in first out) or LIFO (last in first out). The default order is FIFO.

**FIFO**

(first-in first-out) is the default option for STACK. FIFO causes the fileids of the output files to be placed in the program stack. The information is stacked FIFO. The options STACK, STACK FIFO, and FIFO are all equivalent.

**LIFO**

(last-in first-out) causes the fileids of the output files to be placed in the program stack. The information is stacked LIFO. This option is equivalent to STACK LIFO.

Using information from the file containing DLCS statements, CONVERT COMMANDS creates the following output files:

```
applidxPA TXTlangid
applidxSY TXTlangid
```

where:
applid

is the application id.

\( x \)

indicates whether the output file is a System file or User file. The \( x \) may be an S (System) or U (User).

PA

indicates that the file contains command syntax information.

SY

indicates that the file contains command name translations and synonyms.

langid

is the language id.

Note: Four utility files, COMMANDS CMSUT1, COMMANDS CMSUT2, COMMANDS ASSEMBLE, and COMMANDS TEXT are temporarily created to hold the converted data before it is placed in the output files.

Usage Notes:

1. For detailed usage information, refer to the VM/SP CMS for System Programming publication.

2. When editing a file with DLCS statements, you can issue CONVERT COMMANDS for that file from the XEDIT command line. If an error is detected, the line containing the error becomes the current line of the file.

3. System function subsets are always verified and cause an error when incorrect.

4. If both OUTmode and CHECK are specified, the one specified last is recognized.

5. If STACK, LIFO, or FIFO is specified with the CHECK option, a blank line is stacked.

6. Specifying LIFO or FIFO overrides STACK.

7. If conflicting options are specified, the last one specified is used.

8. If you want to issue CONVERT COMMANDS from an EXEC program, you should precede it with the EXEC command; that is, specify

exec convert commands
CONVERT COMMANDS

Example:

Suppose you have a CMS file called TEST DLCS with the following statements:

```plaintext
:DLCS DMS USER AMENG ;
:CMD MMYCMD1 MYCMD1 ;
:SYN MY1 3 ;
:OPR FCN(FN) ;
:OPR FCN(FT) ;
:OPT KWL(<DISK 4> <PRINT 5>) ;
:OPT KWL(<NUMRECS 3>) FCN(PINTEGER) ;
:CMD YYOURCMD YOURCMD YOURCMD 4 ;
:OPR FCN(STRING) ;
:OPT KWL(<TYPE 4>) ;
```

To convert your file into a format that can be used by SET LANGUAGE and the parsing facility, enter:

```plaintext
convert commands test dlcs (system
```

The output files are DMSUPA TXTAMENG and DMSUSY TXTAMENG.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

- **DMSCON006E** No Read/Write {disk|diskmode disk} accessed [RC = 36]
- **DMSCON639E** Error in routine routine; return code was retcode [RC = 256]
- **DMSCON950I** Conversion of fn ft fm [from XEDIT] complete
- **DMSCON950I** No errors found in fn ft fm [from XEDIT]
- **DMSPCA947E** Line line: syntax error [RC = 8]
- **DMSPCC946E** XEDIT is not active. Specify a file name. [RC = 40]
- **DMSPCR002E** File fn ft fm not found [RC = 28]
- **DMSPCR069E** Disk diskmode not accessed [RC = 36]
- **DMSPCR948E** Line line: syntax error [RC = 8]
- **DMSPCR949E** Line line: syntax error [RC = 8]
- **DMSPCT396E** Maximum number of command table entries exceeded [RC = 32]
- **DMSPCT639E** Error in routine routine; return code was retcode [RC = 256]
- **DMSPCW396E** Maximum number of command table entries exceeded [RC = 32]
- **DMSPCW639E** Error in routine routine; return code was retcode [RC = 256]
CONWAIT

Use the CONWAIT command to cause a program to wait until all pending terminal I/O is complete.

The format of the CONWAIT command is:

CONWAIT

Usage Note:

The CONWAIT command synchronizes input and output to the terminal; it ensures that the output console stack is cleared before the program continues execution. Also, you can ensure that a read or write operation is finished before you modify an I/O buffer.

Responses:

None

Messages and Return Codes:

None
COPYFILE

Use the COPYFILE command to copy and/or modify CMS disk files. The manner in which the file identifiers are entered determines whether or not one or more output files are created.

The format of the COPYFILE command is:

\[
\text{COPYfile} \quad \text{[fileidi1...]} \quad \text{[fileidi2]} \quad \text{[fileido]} \quad \text{[(options...[)]]} \quad \text{[options...[]]}
\]

\text{Options:}

- \text{Type [NOType]}
- \text{NEWDate [OLDDate]}
- \text{NEWFile [REPlace] [PRompt] [NOPrompt]}
- \text{FROM recno [FLabel xxxxxxxx]}
- \text{FOR numrec [TLabel xxxxxxxx] [SSpecs] [NOSpecs]}
- \text{OVly [APPend] [RECfm [F] [V] [Lrecl nnnn] [TRUnc] [NOTRunc]}
- \text{PAck UNPack [FILL c [EBcdic] [UPcase [TRAns] [FILL 40] [LOwcase]}
- \text{[SIngle]}

\text{where:}

\text{fileidi1}

is the first (or only) input file. Each file identifier (filename, filetype, and filemode) must be specified either by indicating the specific identifier or by coding an asterisk.

\text{fileidi2}

is one or more additional input files. Each file identifier (filename, filetype, and filemode) must be specified. In single output mode, any of the three input file identifiers may be specified either by indicating the specific identifier or by coding an asterisk. However, all three file identifiers of fileidi2 cannot be specified by asterisks. In multiple output mode, an asterisk (*) is an invalid file identifier. An equal sign (=) may be coded for any of the file identifiers, indicating that it is the same as the corresponding identifier in fileidi1.
the output file(s) to be created. Each file identifier (filename, filetype, and filemode) must be specified. To create multiple output files, an equal sign (=) must be coded in one or more of the identifier fields. If there is only one input file, fileido may be omitted, in which case it defaults to = = = (the input file represented by fileid1 is replaced).

The COPYFILE command options are listed below, briefly. For usage notes and examples, see “Using the COPYFILE Command” on page 73.

Options:

**Type**
- displays, at the terminal, the names of the files being copied.

**NOType**
- suppresses the display of the names of the files being copied.

**NEWDate**
- uses the current date as the creation date of the new file(s).

**OLDDate**
- uses the date on the (first) input file as the creation date of the new file(s).

**NEWFile**
- checks that files with the same fileid as the output file do not already exist. If one or more output files do exist, an error message is displayed and the COPYFILE command terminates. This option is the default so that existing files are not inadvertently destroyed.

**REPlace**
- causes the output file to replace an existing file with the same file identifier. REPLACE is the default option when only one fileid is entered or when the output fileid is specified as “= = =.”

**PRompt**
- displays the messages that request specification or translation lists.

**NOPRompt**
- suppresses the display of prompting messages for specification and translation lists.
COPYFILE

Copy Extent Options:

**FRom recno**

is the starting record number for each input file in the copy operation.

**FRLabel xxxxxxxx**

xxxxxxxx is a character string that appears at the beginning of the first record to be copied from each input file. Up to eight nonblank characters may be specified.

**FOR numrec**

is the number of records to be copied from each input file.

**TOLabel xxxxxxxx**

xxxxxxxx is a character string which, if at the beginning of a record, stops the copy operation for that input file. The record containing the given character is not copied. Up to eight nonblank characters may be specified.

**SPecs**

indicates that you are going to enter a specification list to define how records should be copied. See “Entering a COPYFILE Specification List” on page 77 for information on how you can define output records in a specification list.

**NOSPecs**

indicates that no specification list is to be entered.

**OVly**

overlays the data in an existing output file with data from the input file. You can use OVLY with the SPECS option to overlay data in particular columns.

**APPend**

at the end of the output file.

Data Modification Options:

The following options can be used to change the record format of a file. See “Modifying Record Formats” on page 76 for more details.

**RECfm [F V]**

is the record format of the output files. If not specified, the output record format is the same as that of the input file.

**LReel nnnnn**

is the logical record length of the output file(s) if it is to be different from that of the input file(s). The maximum value of nnnnn is 65535.
COPYFILE

TRUnc
removes trailing blanks (or fill characters) when converting fixed-length files to variable-length format.

NOTRunc
suppresses the removal of trailing blanks (or fill characters) when converting fixed-length files to variable-length format.

PAck
compresses records in a file so that they can be stored in packed format.

Note: A file in packed format should not be modified in any way. If such a file is modified, the UNPACK routines are unable to reconstruct the original file.

UNPack
reverses the PACK operation. If a file is inadvertently packed twice, you can restore the file to its original unpacked form by issuing the COPYFILE command twice.

FIl c
FIl h h
FIl 40
is the padding and truncation character for the TRUNC option or the principal packing character for the PACK option. The fill character may be specified as a single character, c, or by entering a two-digit hexadecimal representation of a character. The default is 40 (the hexadecimal representation for a blank in EBCDIC).

Character Translation Options:

EBcdic
converts a file that was created with 026 keypunch characters (BCD), to 029 keypunch characters (EBCDIC). The following conversions are made:

< to )
& to +
% to (  
# to =
@ to ’
’ to :

UPcase
converts all lowercase characters in each record to uppercase before writing the record to the output file.
COPYFILE

L0wcase
converts all uppercase characters in each record to lowercase before writing the record to the output file.

TRAns
indicates that you are going to enter a list of character translations to be made as the file is copied. See “Entering Translation Specifications” on page 79 for details on entering a list of characters to be translated.

SIngle
suppresses multiple output mode regardless of how the file identifiers are specified.

Incompatible Options:

Figure 5 below shows combinations of options that should not be specified together in the same COPYFILE command. If the option in the first column is specified, do not code any of the options in the second column.

<table>
<thead>
<tr>
<th>Options</th>
<th>Incompatible Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPEND</td>
<td>LRECL, NEWDATE, NEWFILE, OLDDATE, OVLY, PACK, RECFM, REPLACE, UNPACK</td>
</tr>
<tr>
<td>EBCDIC</td>
<td>PACK, UNPACK</td>
</tr>
<tr>
<td>FOR</td>
<td>PACK, TOLABEL, UNPACK</td>
</tr>
<tr>
<td>FRLABEL</td>
<td>FROM, PACK, UNPACK</td>
</tr>
<tr>
<td>FROM</td>
<td>FRLABEL, PACK, UNPACK</td>
</tr>
<tr>
<td>LOWCASE</td>
<td>PACK, UNPACK</td>
</tr>
<tr>
<td>LRECL</td>
<td>APPEND, PACK, UNPACK</td>
</tr>
<tr>
<td>NEWDATE</td>
<td>APPEND, OLDDATE</td>
</tr>
<tr>
<td>NEWFILE</td>
<td>APPEND, OVLY, REPLACE</td>
</tr>
<tr>
<td>NOPROMPT</td>
<td>PROMPT</td>
</tr>
<tr>
<td>NOSPECS</td>
<td>SPECS</td>
</tr>
<tr>
<td>NOTRUNC</td>
<td>TRUNC</td>
</tr>
<tr>
<td>NOTYPE</td>
<td>TYPE</td>
</tr>
<tr>
<td>OLDDATE</td>
<td>APPEND, NEWDATE</td>
</tr>
<tr>
<td>OVLY</td>
<td>APPEND, NEWFILE, PACK, REPLACE, UNPACK</td>
</tr>
<tr>
<td>PACK</td>
<td>APPEND, EBCDIC, FOR, FRLABEL, FROM, LOWCASE, LRECL, OVLY, RECFM, SPECS, TOLABEL, TRANS, TRUNC, UNPACK, UPCASE</td>
</tr>
<tr>
<td>PROMPT</td>
<td>NOPROMPT</td>
</tr>
</tbody>
</table>

Figure 5 (Part 1 of 2). COPYFILE Option Incompatibilities
COPYFILE Option Incompatibilities

<table>
<thead>
<tr>
<th>Options</th>
<th>Incompatible Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECFM</td>
<td>APPEND, PACK, UNPACK</td>
</tr>
<tr>
<td>REPLACE</td>
<td>APPEND, NEWFILE, OVLY</td>
</tr>
<tr>
<td>SPECS</td>
<td>NOSPECS, PACK, UNPACK</td>
</tr>
<tr>
<td>TOLABEL</td>
<td>FOR, PACK, UNPACK</td>
</tr>
<tr>
<td>TRANS</td>
<td>PACK, UNPACK</td>
</tr>
<tr>
<td>TRUNC</td>
<td>NOTRUNC, PACK, UNPACK</td>
</tr>
<tr>
<td>TYPE</td>
<td>NOTYPE</td>
</tr>
<tr>
<td>UNPACK</td>
<td>APPEND, EBCDIC, FOR, FRLABEL, FROM, LOWCASE, LRECL, OVLY, PACK, RECFM, SPECS, TOLABEL, TRANS, TRUNC, UPCASE</td>
</tr>
<tr>
<td>UPCASE</td>
<td>PACK, UNPACK</td>
</tr>
</tbody>
</table>

Using the COPYFILE Command

Two simple uses of the COPYFILE command are: (1) to copy a single CMS file from one disk to another, or (2) to make a duplicate copy of the file on the same disk. For example:

```plaintext
copyfile test1 assemble a test2 assemble a
```

makes a copy of the file TEST1 ASSEMBLE A and names it TEST2 ASSEMBLE A.

For those portions of the file identifier that you want to stay the same, you may code an equal sign in the output fileid. Thus, the command line above can be entered:

```plaintext
copyfile test1 assemble a test2 = =
```

The equal sign may be used as a prefix or suffix of a file identifier. For example, the command:

```plaintext
copyfile a b c file= type= =
```

creates an output file called FILEA TYPEB C.

When you copy a file from one virtual disk to another, you specify the old and new filemodes, and any filename or filetype change you want to make; for example:

```plaintext
copyfile test3 assemble c good = a
```

This command makes a copy of the file TEST3 ASSEMBLE C, and names it GOOD ASSEMBLE A.

If you want to copy only particular records in a file, you can use the FROM/FOR FRLABEL/TOLABEL options. For example:
COPYFILE

copyfile old test a new test a (frlabel start for 41

copies 41 records from the file OLD TEST A1, beginning with the record starting with the character string START into the file NEW TEST A1. Since the user's command line, as passed to COPYFILE in the PLIST, has been translated into uppercase letters, any FRLABEL or TOLABEL character string consisting of either all lowercase or mixed case letters is not found in the input file. Error message DMSCPY157E is issued if the FRLABEL character string is not found. If the TOLABEL character string is not found, the copy operation continues as if TOLABEL was not specified.

Note: If the input filemode is an '*', then you should specify an explicit filemode, not an ' = ', for the output filemode. If you do not specify an explicit output filemode, it is possible to create an output file that would be recognized as an input file which generates the error message DMSCPY024E stating that the file already exists. For example, if you have a file named 'C B A' and you issue the command 'COPY C * * = D =', COPY will first create an output file named 'C D A'. This file will then match the input fileid of the file 'C * *' and copy will attempt to write an output file with the name 'C D A', which already exists.

Multiple Input and Output Files

You can combine two or more files into a single file with the COPYFILE command. For example:

copyfile test data1 a test data2 = test data3 b

copies the files TEST DATA1 and TEST DATA2 from your A-disk and combines them into a file, TEST DATA3, on your B-disk.

Note that if any input file has a filemode number of 3, it is possible that the file will be copied in a sequence different from its order on the disk.

If you want to combine two more files without creating a new file; use the APPEND option. For example:

copyfile new list a old list a (append

appends the file NEW LIST A to the bottom of the existing file labeled OLD LIST A.

Note: If the file NEW LIST A has a different LRECL from the file OLD LIST A, the appended data is padded, or truncated, to the LRECL of the file OLD LIST A.

Whenever you code an asterisk (*) in an input fileid, you may cause one or more files to be copied, depending upon the number of files that satisfy the remaining conditions. For example:

copyfile * test a combined test a
COPYFILE

copies all files with a filetype of TEST on your A-disk into a single file named COMBINED TEST. If only one file with a filetype of TEST exists, only that file is copied.

If you want to copy all the files on a particular disk to another disk, you could enter:

copyfile * * b = = a

All the files on the B-disk are copied to the A-disk. The filenames and filetypes remain unchanged.

You can also copy a group of files and change all the filenames or all the filetypes. For example:

copyfile * assemble b = test a

copies all ASSEMBLE files on the B-disk into files with a filetype of TEST on the A-disk. The filenames are not changed.

You can use the SINGLE option to override multiple output mode. For example:

copyfile * test a = = B (single

copies all files on the A-disk with a filetype of TEST to the B-disk as one combined file, with the filename and filetype equal to the first input file found.

Whenever an asterisk appears, it indicates that all files are to be copied; whenever an equal sign (=) appears, it indicates that the same files are to be copied. For example:


copyfile x * a1 = file =

combines all files with a filename of X on the A-disk into a single file named X FILE A1.

Whenever an equal sign appears in the output fileid in a position corresponding to an asterisk in an input fileid, multiple input files produce multiple output files. When you perform copy operations of this nature you might wish to use the TYPE option, which displays the names of files being copied. For example:


copyfile * test a = output a = summary = (type

might result in the display:

COPY 'ALPHA TEST A1' TO 'ALPHA SUMMARY A1' (NEW FILE)
COPY 'ALPHA OUTPUT A'
COPY 'BETA TEST A1' TO 'BETA SUMMARY A1' (NEW FILE)
COPY 'BETA OUTPUT A.'

which indicates that files ALPHA TEST A and ALPHA OUTPUT A were copied into a file named ALPHA SUMMARY A and that files BETA TEST
COPYFILE

A and BETA OUTPUT A were copied into a file named BETA SUMMARY A.

Modifying Record Formats

You can use the RECFM and LRECL options to change the record format of a file as you copy it. For example:

copyfile data file a (recfm f lrecl 130

converts the file DATA FILE A1 to fixed-length 130-character records.

If you specify an output fileid, for example:

copyfile data file a fixdata file a (recfm f lrecl 130

the original file remains unchanged. The file FIXDATA FILE A contains the converted records.

If the records in a file being copied are variable-length, each output record is padded with blanks to the specified record length. If any records are longer than the record length, they are truncated.

When you convert files from fixed-length records to variable-length records, you can specify the TRUNC option to ensure that all trailing blanks are truncated:

copyfile data file a (recfm v trunc

If you specify the LRECL option and RECFM V, the LRECL option is ignored and the output record length is taken from the longest record in the input file.

When you convert a file from variable-length to fixed-length records, you may also specify a fill character to be used for padding instead of a blank. If you specify:

copyfile short recs a (recfm f fill *

then each record in the file SHORT RECS is padded with asterisks to the record length. Assuming that SHORT RECS was originally a variable-length file, the record length is taken from the longest existing record. Note that if SHORT RECS is already fixed-length, it is not altered.

Similarly, when you are converting back to variable-length a file that was padded with a character other than a blank, you must specify the FILL option to indicate the pad character, so that character is truncated.
The FILL option can also be used to specify the packing character used with the PACK option. When you use the PACK option, a file is compressed as follows: all occurrences of two or more blanks are encoded as one character, and four or more occurrences of any other character are written as three characters. If you use the FILL option to specify a fill character, then that character is treated as a blank when records are compressed. You must, of course, specify the FILL option to unpack any files packed in this way. Since most fixed-length files are blank-padded to the record length, you do not need to specify the FILL option unless you know that some other character appears more frequently.

A file which is packed on an 800 byte blocksize disk will be fixed format with a logical record length of 800. On a 512, 1K, 2K, or 4K blocksize disk, the file will be fixed format with a logical record length of 1024. A packed file of either logical record length can be unpacked back to its original specifications regardless of the disk blocksize it resides on. A packed file with logical record length 800 on a disk with blocksize 512, 1K, 2K, or 4K, and packed files with logical record length 1024 on 800 byte disks should be unpacked and re-packed if minimal disk block usage is needed.

When you convert record formats on packed files with the COPYFILE command you can specify single or multiple output files, in accordance with the procedures outlined under "Modifying Record Formats" on page 76. For example:

```
copyfile * assemble a (pack
```

compresses all ASSEMBLE files in the A-disk without changing any file identifiers. The command:

```
copyfile * assemble a = script = (recfm v trunc
```

creates copies of all ASSEMBLE files residing on your A-disk. The copies will have variable-length record formats and filetypes of SCRIPT.

**Entering a COPYFILE Specification List**

When you use the COPYFILE command, you can specify particular columns of data to be manipulated or particular characters to be translated. Again, how you specify the file identifier determines how many files are copied or modified.

When you use the SPECS option on the COPYFILE command, you receive the message:

```
DMSCPY601R Enter specification list:
```

Chapter 2. CMS Commands 77
COPYFILE

The system waits for you to enter a specification list. If you do not wish to receive this message, use the NOPROMPT option. The specification list you enter may consist of one or more pairs of operands in the following format:

\[
\begin{align*}
\{ & \text{nnn-mmm} \\
& \text{/string/} \\
& \text{hxx...} \} \quad \text{col}
\end{align*}
\]

where:

- **nnn-mmm** specifies the start and end columns of the input file that are to be copyed to the output file. If mmm exceeds the length of the input record, the end of the record is the assumed ending position.

- **string** is any string of uppercase and lowercase characters or numbers delimited by any non-alphameric character.

- **hxx...** is an even number of hexadecimal digits prefixed with an h.

- **col** is the column in the output file at which the copy operation is to begin.

You can enter as many as 20 pairs of specifications resulting in as many as 130 characters per line. If you want to enter more than one line of specifications, enter two plus signs (++) before column 130 at the end of one input line as continuation indicators.

A specification list may contain any combination of specification pairs; for example:

```
copyfile sorted list a (specs

DMSCPY601R Enter specification list:

/1/ 1 1-8 3 /1/ 12 /***/ 14 ++
9-80 18
```

After this command is executed, each record in the file SORTED LIST will look like the following:

```
| oooooooo | *** oooo....
```

where the o's in columns 3 through 10 indicate information originally in columns 1 through 8; the o's following the asterisks indicate the remainder of each record, columns 9 through 80.

When you enter a specification list, you are actually constructing a file column by column. If you specify multiple input or output files, the same copy operation is performed for each record in each output file.
Those columns for which you do not specify any data are filled with blanks or, if you use the FILL option, the fill character of your choice. For example:

```
COPYFILE sorted list a (specs noprompt lrecl 20 fill $ 1-15 6
```

copies columns 1 through 15 beginning in column 6 and writes dollar signs($) in columns 1 through 5.

If you do want to modify data in particular columns of a file but want to leave all of the rest of each record unchanged, you can use the OVLY (overlay) option. For example, the sequence:

```
COPYFILE * bracket a (specs ovly noprompt had 1 hbd 80
```

overlays the characters [ (X'AD') and ] (X'BD') in columns 1 and 80 of all the files with a filetype of BRACKET on your A-disk.

When you copy fixed-length files, records are padded or truncated to the record length; variable-length files are always written as specified.

**Entering Translation Specifications**

You can perform conversion on particular characters in CMS files or groups of files with the TRANS option of the COPYFILE command.

When you enter the TRANS option, you receive the message:

```
DMSCPY602R Enter translation list:
```

and a read is presented to your virtual machine. You may enter the translation list. If you do not wish to receive this message, use the NOPROMPT option.

A translation list consists of one or more pairs of characters or hex digits, each pair representing the character you want to translate and the character you want to translate it to, respectively. For example:

```
copy test file a (trans
```

```
DMSCPY602R Enter translation list:
```

```
* - A f0 00 ff
```

specifies that all occurrences of the character * are to be translated to -, all character A's are to be translated to X'F0' and all X'00's are to be translated to X'FF's.

If any translation specifications you enter conflict with the LOWCASE, EBCDIC, or UPCASE options specified on the same command line, the translation list takes precedence. In the preceding example, if LOWCASE had also been specified, all A's would be translated to X'F0's, not to a's.
COPYFILE

You can enter as many as 130 characters per line. You can enter translation pairs on more than one line if you enter two plus signs (++) before column 130 at the end of one input line as continuation indicators.

Responses:

DMSCPY601R Enter specification list:

This message prompts you to enter a specification list when you use the SPECS option.

DMSCPY602R Enter translation list:

This message prompts you to enter a translation list when you use the TRANS option.

DMSCPY7211 Copy fn ft fm {tolappendoverlay} fn ft fm

This message appears for each file copied with the TYPE option. It indicates the names of the input file and output file. When you have multiple input files, the output fileid is displayed only once.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSCPY002E File[(s)] [fn [ft [fm]]] not found [RC = 28]
DMSCPY002E Input file[(s)] [fn [ft [fm]]] not found [RC = 28]
DMSCPY002E Overlay file[(s)] [fn [ft [fm]]] not found [RC = 28]
DMSCPY003E Invalid option: option [RC = 24]
DMSCPY024E File fn ft fm already exists; specify REPLACE option [RC = 28]
DMSCPY029E Invalid parameter parameter in the option option field [RC = 24]
DMSCPY030E File fn ft fm already active [RC = 28]
DMSCPY037E Disk mode[vedev] is accessed as read/only [RC = 36]
DMSCPY042E No fileid(s) specified [RC = 24]
DMSCPY048E Invalid mode mode [RC = 24]
DMSCPY054E Incomplete fileid specified [RC = 24]
DMSCPY062E Invalid character char in fileid fn ft fm/ [RC = 20]
DMSCPY063E No [sort|translation|specification] list {entered|given} [RC = 40]
DMSCPY064E Invalid [translate] specification at or near list [RC = 24]
DMSCPY065E option option specified twice [RC = 24]
DMSCPY066E option1 and option2 are conflicting options [RC = 24]
DMSCPY067E Combined input files illegal with PACK or UNPACK options [RC = 24]
DMSCPY068E Input file fn ft fm not in packed format [RC = 32]
DMSCPY069E Disk mode not accessed [RC = 36]
DMSCPY101S SPECS temp string storage exhausted at storarea [RC = 88]
<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSCPY102S</td>
<td>Too many fileids [RC = 88]</td>
</tr>
<tr>
<td>DMSCPY103S</td>
<td>Number of SPECS exceeds maximum nn [RC = 88]</td>
</tr>
<tr>
<td>DMSCPY156E</td>
<td>FROM nnn not found--the file fn ft fm has only nnn records [RC = 32]</td>
</tr>
<tr>
<td>DMSCPY157E</td>
<td>Label label not found in file fn ft fm [RC = 32]</td>
</tr>
<tr>
<td>DMSCPY172E</td>
<td>TOLABEL label {equals</td>
</tr>
<tr>
<td></td>
<td>FRLABEL label [RC = 24]</td>
</tr>
<tr>
<td>DMSCPY173E</td>
<td>No records were copied to output file fn ft fm [RC = 40]</td>
</tr>
<tr>
<td>DMSCPY901T</td>
<td>Unexpected error at vstor1: plist function fn ft fm at vstor2, base vstor3, rc nn [RC = 256]</td>
</tr>
<tr>
<td>DMSCPY903T</td>
<td>Impossible PHASE code xx [RC = 256]</td>
</tr>
<tr>
<td>DMSCPY904T</td>
<td>Unexpected UNPACK error at vstor1, base vstor2 [RC = 256]</td>
</tr>
</tbody>
</table>
Usage Notes:

1. You must use the CP command to invoke a CP command:
   
   - From within a CMS EXEC or an EXEC 2 EXEC.
   
   - If the implied CP (IMPCP) function is set to OFF for your virtual machine.
   
   - In a job that you send to the CMS batch facility.

2. To enter a CP command from the CMS environment without CMS processing the command line, use #CP.

3. When you enter an invalid CP command following the CP command, you receive a return code of -1. In an EXEC, this return code is +1.

Example:

If the implied CP function is set to OFF for your virtual machine (QUERY IMPCP to find out setting), and you want to specify the CP SCREEN command, you can precede it by the CP command. For example:

```
cp screen inarea red
```
Responses:

All responses are from the CP command that was issued; the CMS ready message follows the response.
CURSOR VSCREEN

CURSOR VSCREEN

Use the CURSOR VSCREEN command to position the cursor on a specified line and column in a virtual screen.

The format of the CURSOR VSCREEN command is:

```
CURsor VScreeen  vname  line  col  [(options [])]
```

**Options:**

**Reserved**

places the cursor in the reserved area of the virtual screen.

The `line` number must be less than or equal to the number of lines in the reserved area. A negative line number positions the cursor in the bottom reserved area, and a positive line number positions the cursor in the top reserved area.

The `col` must be less than or equal to the number of columns in the virtual screen.

You cannot specify a `line` or `col` of zero when placing the cursor in the RESERVED area.

**Data**

places the cursor in the scrollable data area at the specified line and column. DATA is the default.
The line number must be less than or equal to the number of lines in the data area, and must be zero or greater. A value of zero positions the cursor at the line following the current bottom of the virtual screen.

The col must be less than or equal to the number of columns in the virtual screen, and must be zero or greater. A value of zero positions the cursor in column two. This allows for a start field in column one.

Usage Notes:

1. In a virtual screen, the lines in the top reserved area are numbered starting from the top. The top line is 1, the second line is 2, etc. In the bottom reserved area, lines are numbered starting at the bottom and have negative values. The bottom line is line -1, the second line up is -2, the third line up is -3, etc.

2. When the physical screen is read, the following cursor information is saved:
   a. Cursor position on the physical screen
   b. Cursor position in the window and the window name
   c. Cursor position in the virtual screen and the virtual screen name.

   If the cursor is not in a window or if it is on a border, the window and virtual screen cursor information is not updated.

   In addition, the cursor position can be changed by:
      • The CURSOR VSCREEN command
      • Lines written to a virtual screen

   The last window and last virtual screen to be updated with cursor location information are said to ‘own’ the cursor.

   When the physical screen is refreshed, the following rules determine the position of the cursor on the physical screen:
      a. If a border command (with the exception of X and H) was issued, position the cursor on the corner of the window in which the last processed border command was typed. If the window position is no longer displayed, processing continues with the next step.
      b. Place the cursor in the virtual screen that last owned the cursor:
         1) Determine if the last window to own the cursor is connected to this virtual screen. If it is, check if the window contains the virtual screen cursor position and if it is visible on the physical screen.
CURSOR VSCREEN

2) If the position is not visible and this is the CMS virtual screen, place the cursor on the command line.

3) Otherwise, check other windows connected to this virtual screen and determine if any of these windows contain the virtual screen cursor position and are visible on the physical screen. Place cursor in the first window that meets these criteria.

4) For each window connected to the virtual screen that owns the cursor, starting with the top-most window, determine if the virtual screen displayed by that window has a visible cursor position. Place the cursor in the first position visible on the physical screen.

5) Calculate the cursor positions of each window visible on the physical screen (starting with the top-most window) and place the cursor in the first position that is visible on the physical screen.

c. Place the cursor in the window that last owned the cursor:

1) Place the cursor in the window that last owned it if that position is visible on the physical screen.

2) If the position not visible, determine if any position in this window is visible on the screen and place it in the first position of this window that is visible.

3) Calculate the cursor position for the top window. Place the cursor in this position if it is visible. Otherwise, place it in the first visible position of the top window.

d. If the cursor still has not been positioned, place it in row 1, column 1 of the physical screen.

Responses:

The next time the screen is displayed, the cursor will be positioned at the specified location in the virtual screen, assuming that the location is displayed somewhere in an associated window on the physical screen.
Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSCUR386E  Missing operand(s) [RC = 24]
DMSCUR388E  Invalid keyword: keyword [RC = 24]
DMSCUR389E  Invalid operand: operand [RC = 24]
DMSCUR391E  Unexpected operand(s): operand [RC = 24]
DMSCUR394E  Invalid option: option [RC = 24]
DMSCUR921E  Virtual screen vname is not defined [RC = 28]
DMSCUR923E  Specified location is outside the virtual screen [RC = 32]
DMSCUR928E  Command is not valid for virtual screen vname [RC = 12]
DEBUG

Use the DEBUG command to enter the debug environment from the CMS environment. In the debug environment you can use a variety of DEBUG subcommands that allow you to test and debug your programs. The DEBUG subcommands are described in “Chapter 4. DEBUG Subcommands.” For tutorial information, including examples, see the VM/SP CMS User’s Guide.

The format of the DEBUG command is:

```
DEBUG
```

Usage Notes:

1. The debug environment is also entered as a result of an external interruption or the result of a breakpoint (address stop) encountered during program execution.

2. Once you are in the debug environment, you can enter only DEBUG subcommands and CP commands via the #CP function.

3. To return to the CMS environment, enter the DEBUG subcommand RETURN.

Responses:

DMSDBG728I Debug entered

This message indicates that you are in the debug environment.
Use the DEFAULTS command to set up default options for the commands that are listed below. Each time you enter one of these commands, the options specified in the DEFAULTS command are in effect. However, the options specified with each invocation of the various commands override the ones set up in the DEFAULTS command. Thus, you can customize the options by using DEFAULTS, yet override them when you desire. DEFAULTS can also be used to display the current default options for one or more of the commands.

The format of the DEFAULTS command is:

```
DEFAULTS [Set command options...]
[      List [command]   ]
```

where:

**Set**

specifies that default options are to be set up for the command indicated.

**List**

specifies that the current default options for the command indicated are to be displayed. If no command is specified, all the commands listed below and the current default options are displayed.

**command**

is one of the commands listed below.

**options**

is one or more options associated with a particular command, as shown below.

The commands and options that can be specified as defaults are listed below. Valid abbreviations for both the command names and the keyword options are indicated by uppercase letters. Mutually exclusive options are listed one under the other. The system defaults are underscored.
## DEFAULTS

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filelist</td>
<td>Profile fn</td>
</tr>
<tr>
<td></td>
<td>Filelist</td>
</tr>
<tr>
<td></td>
<td>Profile PROFLST</td>
</tr>
<tr>
<td></td>
<td>NoFilelist</td>
</tr>
<tr>
<td>Help</td>
<td>Brief</td>
</tr>
<tr>
<td></td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>DETail</td>
</tr>
<tr>
<td></td>
<td>DESCript</td>
</tr>
<tr>
<td></td>
<td>Format</td>
</tr>
<tr>
<td></td>
<td>Parms</td>
</tr>
<tr>
<td></td>
<td>Options</td>
</tr>
<tr>
<td></td>
<td>NOTes</td>
</tr>
<tr>
<td></td>
<td>Errors</td>
</tr>
<tr>
<td></td>
<td>Screen</td>
</tr>
<tr>
<td>Maclist</td>
<td>MList</td>
</tr>
<tr>
<td></td>
<td>Profile fn</td>
</tr>
<tr>
<td></td>
<td>Compact</td>
</tr>
<tr>
<td></td>
<td>Profile PROFMLST</td>
</tr>
<tr>
<td></td>
<td>NoCompact</td>
</tr>
<tr>
<td>Note</td>
<td>Profile fn</td>
</tr>
<tr>
<td></td>
<td>Short</td>
</tr>
<tr>
<td></td>
<td>LOG</td>
</tr>
<tr>
<td></td>
<td>NOAck</td>
</tr>
<tr>
<td></td>
<td>NOTEbook fn</td>
</tr>
<tr>
<td></td>
<td>NOTEbook ALL</td>
</tr>
<tr>
<td></td>
<td>NOTEbook *</td>
</tr>
<tr>
<td></td>
<td>NONotebook</td>
</tr>
<tr>
<td>Peek</td>
<td>Profile fn</td>
</tr>
<tr>
<td></td>
<td>FROM reco</td>
</tr>
<tr>
<td></td>
<td>FOR numrec</td>
</tr>
<tr>
<td></td>
<td>Profile PROFPEEK</td>
</tr>
<tr>
<td></td>
<td>FROM 1</td>
</tr>
<tr>
<td></td>
<td>FOR 200</td>
</tr>
<tr>
<td></td>
<td>FOR *</td>
</tr>
<tr>
<td>Sendfile</td>
<td>Sfile</td>
</tr>
<tr>
<td></td>
<td>New</td>
</tr>
<tr>
<td></td>
<td>NOType</td>
</tr>
<tr>
<td></td>
<td>NOFilelist</td>
</tr>
<tr>
<td></td>
<td>Log</td>
</tr>
<tr>
<td></td>
<td>NOAck</td>
</tr>
<tr>
<td></td>
<td>Filelist</td>
</tr>
<tr>
<td></td>
<td>NOlog</td>
</tr>
<tr>
<td>Receive</td>
<td>Log</td>
</tr>
<tr>
<td></td>
<td>Olddate</td>
</tr>
<tr>
<td></td>
<td>NOTEbook fn</td>
</tr>
<tr>
<td></td>
<td>NOTEbook ALL</td>
</tr>
<tr>
<td></td>
<td>Minprompt</td>
</tr>
<tr>
<td></td>
<td>NOTEbook *</td>
</tr>
<tr>
<td></td>
<td>NOPrompt</td>
</tr>
<tr>
<td>Tell</td>
<td>Message</td>
</tr>
<tr>
<td></td>
<td>MSG</td>
</tr>
<tr>
<td></td>
<td>MSGNOH</td>
</tr>
<tr>
<td></td>
<td>MSGSMG</td>
</tr>
<tr>
<td></td>
<td>Warning</td>
</tr>
</tbody>
</table>

### Usage Notes:

1. The DEFAULTS command uses the GLOBALV command, which maintains a LASTING GLOBALV file on your A-disk. This file contains the options specified in a DEFAULTS command. However, do not edit the LASTING GLOBALV file to change the options. Use the DEFAULTS command, instead. For more information on GLOBALV files, see the description of the GLOBALV command.

2. You must be an authorized class B user to send messages with the CP MSGNOH command.

3. If you want to issue DEFAULTS from an EXEC program, you should precede it with the EXEC command; that is, specify

   `exec defaults`
Example:

To change the SENDFILE command default from NEW to OLD, you would enter:

defaults set sendfile old

Responses:

The following is a list of your default options for the cmdname command:
  option...
    ...
To change these default options enter 'DEFAULTS Set Cmdname Opt1 <Opt2..>'.

The following default options have been set:
  commandname option...
To change any default options enter 'DEFAULTS Set Cmdname Opt1 <Opt2..>'.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSWDF003E   Invalid option: option [RC=24]
DMSWDF641E   No options specified [RC=24]
DMSWDF653E   Error executing GLOBALV, RC=nn [RC=40]
DEFINE VSCREEN

Use the DEFINE VSCREEN command to create a virtual screen. A virtual screen is a functional simulation of a physical display screen. It is a "presentation space" where data is written.

The format of the DEFINE VSCREEN command is:

```
DEFINE VSCREEN vname lines cols rtop rbot [ (optionA optionB optionC optionD [ ] ) ]
```

where:

- **vname**
  is the name assigned to the virtual screen. You may specify a name up to eight characters in length.

- **lines**
  is the number of scrollable data lines that the virtual screen contains. The number of lines must be one or greater.

- **cols**
  is the number of columns that the virtual screen contains.

- **rtop**
  is the number of reserved lines maintained in the top reserved area of the virtual screen.

- **rbot**
  is the number of reserved lines maintained in the bottom reserved area of the virtual screen.

**OptionA:**
- **TYPE**
- **NOType**

**OptionB:**
- **PRoect**
- **NOProect**
- **High**
- **NOHigh**

**OptionC:**
- **color**
- **exthi**
- **psset**

**OptionD:**
- **USer**
- **SYstem**
DEFINE VSCREEN

<table>
<thead>
<tr>
<th>Option A:</th>
<th>Option A controls the actions when information is written to the specified virtual screen.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The following may be specified:</td>
</tr>
<tr>
<td></td>
<td><strong>TYPE</strong> specifies that data is moved to the virtual screen when the virtual screen queue is processed. TYPE is the default.</td>
</tr>
<tr>
<td></td>
<td><strong>NOType</strong> specifies that the virtual screen is not updated.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option B:</th>
<th>Option B indicates the default attributes of the data in the virtual screen.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The following may be specified:</td>
</tr>
<tr>
<td></td>
<td><strong>PROtect</strong> the data is protected.</td>
</tr>
<tr>
<td></td>
<td><strong>NOProtect</strong> the data is not protected. NOProtect is the default.</td>
</tr>
<tr>
<td></td>
<td><strong>High</strong> data is displayed in high intensity.</td>
</tr>
<tr>
<td></td>
<td><strong>NOHigh</strong> data is displayed in a normal intensity. NOHigh is the default.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option C:</th>
<th>Option C indicates the default extended attributes for the virtual screen.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The following may be specified:</td>
</tr>
<tr>
<td></td>
<td><strong>color</strong> the color may be Default, Blue, Red, Pink, Green, Turquoise, Yellow, or White.</td>
</tr>
<tr>
<td></td>
<td><strong>exthi</strong> the extended highlighting may be None (default), REVvideo, BLInk, or Underline.</td>
</tr>
<tr>
<td></td>
<td><strong>psset</strong> the Programmed Symbol Set (PSset) may be specified as PS0 (the default), PS1, PSA, PSB, PSC, PSD, PSE, or PSF.</td>
</tr>
</tbody>
</table>
**DEFINE VSCREEN**

**Option D:**

Option D indicates whether or not the virtual screen is retained when a task abnormally ends (abend) or when the HX (halt execution) command is issued. The following may be specified:

**USER**

indicates that the virtual screen is deleted when a task abnormally ends (abend) or when the HX (halt execution) command is issued.

**SYSTEM**

indicates that the virtual screen is retained when a task abnormally ends (abend) or when the HX (halt execution) command is issued.

**Usage Notes:**

1. For information on attributes, extended attributes, fields, and Programmed Symbol Sets, refer to the *IBM 3270 Information Display System Data Stream Programmer’s Reference, GA23-0059.*

2. Use the `SET VSCREEN` command to change the options for a defined virtual screen.

3. When you specify NOTYPE, the virtual screen is not updated when the queue is processed. However, the data in the queue is logged to a CMS file if logging is set on (see the `SET LOGFILE` command).

4. The virtual screen options are accepted whether or not the device has the ability to use those options. However, the action taken depends upon the device. For example, color is ignored on a 3278 display, and color and extended highlighting are ignored on a 3277 display. In addition, if Programmed Symbol Sets are supported by the device, then the PSset specified must be loaded in the display. If not, the default PSset for the device is used.

   `QUERY VSCREEN` displays all the options, even those that may be ignored. The `QUERY DISPLAY` command displays the options that are supported by the device.

5. When you define a virtual screen, the following buffers are allocated for the data and option information:

   - Data
   - Attribute
   - Color
   - Extended highlight
   - PSset

   The number of buffers allocated depends on the options supported by the device. For example, if color is not available, a color buffer is not allocated. The data buffer and the attribute buffer are always defined.
The structure allows you to assign different characteristics to each character in a virtual screen. For example, adjacent characters can be displayed with different colors if the device supports character options. The SET CHARMODE command allows you to specify whether character attributes should be used when displaying virtual screen data.

In addition to the buffers, a queue is also defined. Data is queued to the virtual screen when writes are done.

6. The option information sets the default display characteristics of the data in the virtual screen. When data is written to the virtual screen, the virtual screen defaults are used unless the write specifies different characteristics, in which case the write options override the characteristics of the virtual screen. See the WRITE VSCREEN command for more information.

7. The reserved lines are maintained outside the area defined as scrollable data in the virtual screen. For example:

```
define vscreen message 20 80 1 1
```

defines a virtual screen named MESSAGE that contains 20 lines of scrollable data, one line in the top reserved area, and one line in the bottom reserved area. Each line is 80 columns wide. The WRITE VSCREEN command enables you to write to the scrollable data area or, using the RESERVED option, write to the reserved areas.

8. A virtual screen may not be defined with the same name as a virtual screen that already exists. The rules for naming a virtual screen are the same as those for naming files:

- The name can be from one to eight characters.
- The valid characters are A-Z, a-z, 0-9, $, #, @, +, - (hyphen), :(colon), and _ (underscore).

9. To view the contents of a virtual screen in a window, a window must be connected to the virtual screen using the SHOW WINDOW or HIDE WINDOW command.

**Responses:**

None.
DEFINE VSCREEN

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSDEF014E Invalid function function [RC = 24]
DMSDEF386E Missing operand(s) [RC = 24]
DMSDEF389E Invalid operand: operand [RC = 24]
DMSDEF391E Unexpected operand(s): operand [RC = 24]
DMSDEF394E Invalid option: option [RC = 24]
DMSDEF622E Insufficient free storage [RC = 104]
DMSDEF913E Invalid virtual screen name: uname [RC = 20]
DMSDEF920E Virtual screen uname already exists [RC = 3]
DMSDEF926E Command is only valid on a display terminal [RC = 88]
DEFINE WINDOW

Use the DEFINE WINDOW command to create a window with the specified name, size, and position on the physical screen.

The format of the DEFINE WINDOW command is:

```
DEFine WINdow  wname  lines  cols  psline  pscol  [(options [])]
```

Options:
- **VARiable**
- **FIXed**
- **BORder**
- **POP**
- **NOBorder**
- **NOPop**
- **TOP**
- **NOTop**
- **USer**
- **SYstem**

Where:

- **wname**
  - is the name assigned to the window. You may specify a name up to eight characters in length.

- **lines**
  - is the number of lines the window displays.

- **cols**
  - is the number of columns the window displays.

- **psline**
  - is the line on the physical screen where the upper or lower edge of the window is positioned. A positive number positions the upper edge of the window on the specified line relative to the top of the screen. A negative number positions the lower edge of the window on the specified line relative to the bottom of the screen. For more information, see Usage Note 7.

- **pscol**
  - is the column on the physical screen where the left edge of the window is positioned.
DEFINE WINDOW

Options:

VARiable
indicates that the number of lines in the window may vary depending on the amount of scrollable data displayed.

FIXed
indicates that the number of lines in the window is always constant. FIXED is the default.

BORder
indicates that the borders are displayed when possible. BORDER is the default.

NOBorder
indicates that borders are not displayed.

POP
specifies that the window is displayed on top of all other windows when the virtual screen that the window is showing is updated.

NOPop
specifies that there is no effect on the window’s position in the ordered list of windows when the virtual screen that the window is showing is updated. NOPOP is the default.

TOP
specifies that the window may qualify as the topmost window. Most windowing commands process the topmost window by default or when = is specified as the window name.

NOTop
specifies that the window cannot qualify as the topmost window. Windows defined as NOTOP are not processed by default or when a command is specified with = for the window name.

USER
indicates that the window is deleted when a task abnormally ends (abend) or when the HX (halt execution) command is issued.

SYSTEM
indicates that the window is retained when a task abnormally ends (abend) or when the HX (halt execution) command is issued.
Usage Notes:

1. A maximum of 255 windows may be defined at any time.

2. A window may not be defined with the same name as a window that already exists. A window name of "*" and "=" is invalid.

3. Defining a window does not automatically display it. To display a window, it must be connected to a virtual screen (see the SHOW WINDOW and HIDE WINDOW commands for details).

4. Use the SET WINDOW command to change the options for a window.

5. A window displays as many top and bottom reserved lines as possible, regardless of the position on the virtual screen where the window is connected. The reserved lines are defined and maintained in the virtual screen. See the SET RESERVED command to change the way in which reserved lines are displayed in a window.

6. A window's size and location must be specified in such a way that, excluding borders, the entire window fits on the physical screen.

7. Psline may be specified as either a positive or a negative number. When positive, the window's upper left corner is placed on the physical screen at the line number specified. When negative, the window's lower left corner is placed relative to the bottom of the physical screen. Thus, a psline value of +1 positions a window by its upper left corner to the top line of the physical screen. A psline value of -1 positions it by its lower left corner to the bottom line of the physical screen.

8. Window borders are built outside the area defined for the window. Therefore, it is possible that some or all of the borders may not fit on the physical screen due to the size and position of the window. Borders are highlighted and displayed using the following characters:
   - top border character is a dash, '-'
   - bottom border character is a dash, '
   - left border character is a vertical bar, 'I'
   - right border character is a vertical bar, 'I'

   Border corners are identified with a plus (+) sign. Use the SET BORDER command to alter border attributes and characters.


10. If the window, virtual screen, and physical screen do not have the same number of columns, it is recommended that you define the window with one column greater than the number of columns in the virtual screen.
DEFINE WINDOW

that it is displaying. This provides for the additional field definition character (Start Field) that is necessary for the proper display of the window on the screen, and ensures that a maximum number of columns of virtual screen data are displayed.

11. When you specify a window as VARIABLE, the current number of lines in the window may vary from 0, in which case the window is not displayed, to the number of lines specified for the window. The window size depends upon the amount of scrollable data being displayed.

12. If multiple windows defined with the POP option are showing the same virtual screen, all the windows are displayed on top of the other windows when the virtual screen is updated. Their position in relation to each other is maintained.

Responses:

None.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSDEF014E Invalid function function [RC = 24]
DMSDEF386E Missing operand(s) [RC = 24]
DMSDEF389E Invalid operand: operand [RC = 24]
DMSDEF391E Unexpected operand(s); operand [RC = 24]
DMSDEF394E Invalid option: option [RC = 24]
DMSDEF622E Insufficient free storage [RC = 104]
DMSDEF676E Invalid character {I|} for window name [RC = 20]
DMSDEF915E Maximum number of windows already defined [RC = 12]
DMSDEF920E Window wname already exists [RC = 3]
DMSDEF922E Window does not fit entirely on the screen [RC = 32]
DMSDEF926E Command is only valid on a display terminal [RC = 88]
DELETE VSCREEN

Use the DELETE VSCREEN command to remove a virtual screen definition.

The format of the DELETE VSCREEN command is:

```
DELete VSCreen  vname
```

where:

`vname`

is the name of the virtual screen to be deleted.

Usage Notes:

1. The CMS virtual screen cannot be deleted when SET FULLSCREEN is ON or SUSPEND (see SET FULLSCREEN).

2. When deleting a virtual screen and data is in the queue, handling of data depends on whether or not you are using full-screen CMS. When SET FULLSCREEN is ON, the data is redirected to the CMS virtual screen and window. When SET FULLSCREEN is OFF or SUSPEND, the data is typed out on your screen.

3. When you delete a virtual screen, all windows connected to it are disconnected. Any ROUTE command message classes that have been directed to the virtual screen are rerouted to the CMS virtual screen.

Responses:

None.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSDEL386E  Missing operand(s) [RC = 24]
- DMSDEL388E  Invalid keyword: `keyword` [RC = 24]
- DMSDEL391E  Unexpected operand(s): `operand` [RC = 24]
- DMSDEL919E  The CMS virtual screen cannot be deleted [RC = 24]
- DMSDEL921E  Virtual screen `vname` is not defined [RC = 28]
DELETE WINDOW

**DELETE WINDOW**

Use the DELETE WINDOW command to remove a window definition.

The format of the DELETE WINDOW command is:

```
DELETE WINDOW wname
```

*where:*

* wname is the name of the window to be deleted.

**Usage Notes:**

1. If the window is connected to a virtual screen, the virtual screen (vscreen) is not affected.

2. The CMS window cannot be deleted when SET FULLSCREEN is ON or SUSPEND (see SET FULLSCREEN).

**Responses:**

None.

**Messages and Return Codes:**

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSDEL386E  Missing operand(s)  [RC = 24]
- DMSDEL388E  Invalid keyword: keyword  [RC = 24]
- DMSDEL391E  Unexpected operand(s): operand  [RC = 24]
- DMSDEL919E  The CMS window cannot be deleted  [RC = 24]
- DMSDEL921E  Window wname is not defined  [RC = 28]
DESBUF

Use the DESBUF command to clear the console and program stack input and output buffers.

The format of the DESBUF command is:

```
DESBUF
```

Usage Notes:

1. Note that DESBUF clears the output buffers as well as the input buffers. Use the CONWAIT command before DESBUF to halt program execution until all output lines are displayed at the terminal.

**Warning:** Be careful when using the DESBUF command because the input and output console and program stack buffers are used to communicate information between programs.

Response:

None

Messages and Return Codes:

None
Use the DISK command to:

- Punch CMS disk files to the virtual spooled card punch in a special format which allows the punched deck to be restored to disk in the form of the original disk file.

- Restore punched decks created by the DISK DUMP command to a disk file.

The format of the DISK command is:

| DISK | [DUMP fn ft ftm] | [LOAD (options... [])] | Options: [Fullprompt Minprompt NOPrompt Replace NOReplace] [OLDDate] |

where:

**DUMP fn ft ftm**

punches the specified file (fn ft ftm). The file may have either fixed- or variable-length records. After all data is punched, an end-of-file card is created with an N in column 5. This card contains directory information and must remain in the deck. The original disk file is retained.

**LOAD**

loads a file or files from the spooled card reader and writes them as CMS files on your A-disk. A file may also be loaded to a disk other than your A-disk through the use of the prompting facility. The filename and filetype are obtained from the card stream. If a file exists with the same filename and filetype as one of those in the card stream, it is replaced if the REPLACE option is in effect.

The card-image sequence numbers on all files being loaded are checked. A message notifies the user of any record numbers missing or out of order. The file is loaded whether or not a problem is found in the sequence number check.
The DISK LOAD function checks for invalid characters in the fileid field of the reader file to be loaded. If an invalid character is found, message DMDSK496S is printed at the console informing the user that an invalid fileid has been found in the input record. The file is left in the reader. A file is not loaded when the last card of the reader file does not match the filename, filetype, and filemode of the first card in the reader file.

Options for DISK LOAD:

**Fullprompt**

specifies that a prompt is issued for each file in the spool file.

**Minprompt**

specifies that a prompt is issued when the name of the first (or only) file differs from the name of the spool file; the prompt for the first file is suppressed when it has the same name as the spool file. A prompt is always issued for the second and subsequent files.

**NOPrompt**

specifies that a prompt is not issued to you as a file is received.

NOPROMPT is the default.

**Replace**

specifies that if a file of the same filename and filetype exists on the disk onto which the incoming file is to be loaded, it is to be replaced with this one. REPLACE is the default.

**NOReplace**

specifies that a file is not received that would overlay an existing file on the receiving disk.

**OLDDate**

when specified, OLDDATE retains the date and time of the most recent update of the file prior to it being sent to your virtual reader. This date becomes the creation date for the file being loaded. Otherwise, the date and time of execution of the DISK LOAD command will be used as the creation date for the output file produced by the DISK LOAD.

Usage Notes:

1. To read files with the DISK LOAD command, they must have been created by the DISK DUMP command. To identify the proper method to use in loading spooled reader files, use the 'RDR' command. Also see the 'RECEIVE' command.

2. To load reader files created by DISK DUMP, you must issue the DISK LOAD command for each spool file. For example, if you enter:

   ```
   disk dump source1 assemble
   disk dump source2 assemble
   ```
the virtual machine that receives the files must issue the DISK LOAD command twice to read the files onto disk. If you use the CP SPOOL command to spool continuous, for example:

```plaintext
cp spool punch cont
disk dump source1 assemble
disk dump source2 assemble
cp spool punch nocont close
```

then you only need to issue the DISK LOAD command once to read both files.

You may send multiple files by continuous spooling (using CP SPOOL PUNCH CONT) or by a series of DISK DUMP commands but those methods are discouraged. As a sender you are encouraged to do the following:

- Always use SENDFILE, which resets any continuous spooling options in effect.
- Do not spool the punch continuous.

3. DISK LOAD loads a file from the reader into a temporary work file called “DISK CMSUT1.” The existing file with the same name as the one being loaded from the reader is then erased. The name of the temporary work file just created is changed to the name of the work file just read in. If the file you are loading has the name “DISK CMSUT1,” it is changed to “DISK CMSUT2.” “DISK CMSUT1” is a reserved work filename for the DISK command.

4. DISK LOAD or DISK DUMP may cause a file to occupy one extra block on the disk. If the file is close to filling or exactly fills the last block on a 512, 1024, 2048, or 4096 formatted disk, the last record produced by the DISK DUMP or DISK LOAD may be filled with X’00’s causing the file to occupy one extra block consisting of X’00’s on the disk.

5. If you specify the FULLPROMPT or MINPROMPT option the valid responses include:

- One of the digits specified in the prompt
- One of the parenthetical words that follows a digit or any initial truncation of the word.

The meanings of these responses are:
<table>
<thead>
<tr>
<th>Response</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>0 or No</strong></td>
<td>If this file is one of a set of files that constitutes a single spool file, the file is not received and prompting continues for the next file, if there is one. If this is the last file of a set of files or if this is the only file in the spool file, the command is ended.</td>
</tr>
<tr>
<td><strong>1 or Yes</strong></td>
<td>Receives the file under the name <code>fn1 ft1 fm1</code> (or <code>fn3 ft3 fm3</code>).</td>
</tr>
<tr>
<td><strong>2 or Quit</strong></td>
<td>Ends the command.</td>
</tr>
<tr>
<td><strong>3 or Rename</strong></td>
<td>Requests prompt message DMSDSK1080R so that the incoming file can be received using a different name.</td>
</tr>
</tbody>
</table>

6. If you receive prompt message DMSDSK1081R the valid responses include:
   - One of the digits specified in the prompt
   - One of the parenthetical words that follows a digit or any initial truncation of the word.

The meanings of these responses are:

<table>
<thead>
<tr>
<th>Response</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>0 or No</strong></td>
<td>Does not receive the file under the name <code>fn ft fm</code> and repeats the original prompt message DMSDSK1080R. This allows you to specify a different name for the incoming file.</td>
</tr>
<tr>
<td><strong>1 or Yes</strong></td>
<td>Receives the file under the name <code>fn ft fm</code>.</td>
</tr>
<tr>
<td><strong>2 or Quit</strong></td>
<td>Ends the command.</td>
</tr>
</tbody>
</table>

**Responses:**

1. When DISK LOAD has completed loading each incoming file, you receive one of the following responses, depending on the situation.
   - If the incoming file (`fn1 ft1 fm1`) does not already exist and it is received without being renamed, you receive
     
     `fn1 ft1 fm1 created`
   - If the incoming file (`fn1 ft1 fm1`) is renamed to a filename (`fn2 ft2 fm2`) that does not already exist, you receive
     
     `fn2 ft2 fm2 created from fn1 ft1 fm1`
• If the incoming file \((fn1 \ ft1 \ fm1)\) is copied to an existing data set that has the same name as the incoming file, you receive

\(fn1 \ ft1 \ fm1\) replaced

• If the incoming file \((fn1 \ ft1 \ fm1)\) is copied to an existing file \((fn2 \ ft2 \ fm2)\) with a name different from that of the incoming file, you receive

\(fn2 \ ft2 \ fm2\) replaced by \(fn1 \ ft1 \ fm1\)

• If the incoming file \((fn1 \ ft1 \ fm1)\) replaces an existing file \((fn2 \ ft2 \ fm2)\), but is given a mode \((fm1)\) that differs from the mode of the existing file \((fm2)\), you receive

\(fn1 \ ft1 \ fm1\) replaced \(fn2 \ ft2 \ fm2\)

• If the incoming file \((fn1 \ ft1 \ fm1)\) replaces an existing file \((fn2 \ ft2 \ fm2)\), but is given a mode \((fm3)\) that differs from the mode of the existing file \((fm2)\), you receive

\(fn3 \ ft3 \ fm3\) replaced \(fn2 \ ft2 \ fm2\) sent as \(fn1 \ ft1 \ fm1\)

2. If you specify the FULLPROMPT or MINPROMPT option, one of these prompts is displayed:

DMSDSK1079R Receive \(fn1 \ ft1 \ fm1\)?
Reply 0 (NO), 1 (YES), 2 (QUIT), or 3 (RENAME)

Receive \(fn1 \ ft1 \ fm1\) and replace the existing file of the same name?
Reply 0 (NO), 1 (YES), 2 (QUIT), or 3 (RENAME)

Receive \(fn1 \ ft1 \ fm1\) and replace \(fn2 \ ft2 \ fm2\)?
Reply 0 (NO), 1 (YES), 2 (QUIT), or 3 (RENAME)

Receive \(fn1 \ ft1 \ fm1\) as \(fn3 \ ft3 \ fm3\)?
Reply 0 (NO), 1 (YES), 2 (QUIT), or 3 (RENAME)

Receive \(fn1 \ ft1 \ fm1\) as \(fn3 \ ft3 \ fm3\) and replace the existing file of the same name?
Reply 0 (NO), 1 (YES), 2 (QUIT), or 3 (RENAME)

Receive \(fn1 \ ft1 \ fm1\) as \(fn3 \ ft3 \ fm3\) and replace \(fn2 \ ft2 \ fm2\)?
Reply 0 (NO), 1 (YES), 2 (QUIT), or 3 (RENAME)

• The fileid \(fn1 \ ft1 \ fm1\) is the name from the card stream in the spool file.

• The phrase “and replace the existing file of the same name?” appears when the operation replaces an existing file and the filemode of that file is the same as \(fm1\).

• The phrase “and replace \(fn2 \ ft2 \ fm2\).” appears when the operation replaces an existing file and the filemode of that file is not \(fm1\).
3. If you respond with a 3 (or RENAME) to prompt message DMSDSDK1079R, the following message appears and you must enter a fileid in the form fn [ft [fm]].

DMSDSDK1080R Enter the new name for fn ft fm

4. If you respond to prompt message DMSDSDK1080R with a fileid that names an existing file, you receive this prompt:

DMSDSDK1081R Replace fn ft fm?
   Reply 0 (NO), 1 (YES), or 2 (QUIT)

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSDSDK002E File fn [ft [fm]] not found
DMSDSDK014E Invalid function function
DMSDSDK024E File fn [ft fm] already exists[; specify REPLACE option] [RC = 28]
DMSDSDK037E Disk mode[vdev]] is accessed as read/only
DMSDSDK047E No function specified
DMSDSDK054E Incomplete fileid specified
DMSDSDK062E Invalid * in fileid
DMSDSDK069E Disk mode[vdev]] not accessed
DMSDSDK069E Output disk mode[vdev]] not accessed
DMSDSDK070E Invalid parameter parameter
DMSDSDK077E End card missing from input deck
DMSDSDK078E Invalid card in input deck
DMSDSDK078W Sequence error detected loading fn ft--expected seqno1 found seqno2
DMSDSDK104S Error nn reading file fn ft fm from disk
DMSDSDK105S Error nn writing file fn ft fm on disk
DMSDSDK106S Virtual storage capacity exceeded
DMSDSDK118S Error punching file
DMSDSDK124S Error reading card file
DMSDSDK205W Reader empty or not ready
DMSDSDK257T Internal system error at address address (offset offset)
DMSDSDK445W Invalid data in sequence field, bypassing sequence check
DMSDSDK496S Invalid fileid fn ft fm found in input record [RC = 100]
DMSDSDK550W Date/Time data not present for file fn ft
DMSDSDK638E Error in routine routine; return code was nnnn
DMSDSDK671E Error loading file fn ft fm; rc = nn from RENAME
DMSDSDK1123E Unknown response text ignored
DMSDSDK1124W Spool file spoolid has been left in your reader because one or more files were not received [RC = 1]
Use the DLBL command:

- In CMS/DOS, to define VSE and CMS sequential disk files for program input/output; to identify VSE files and libraries; to define and identify VSAM catalogs, clusters, and data spaces; and to identify VSAM, VSE, or CMS files used for VSAM program input/output and access method services functions. In many situations, VSE/VSAM does not require the DLBL command. Information on when a DLBL statement is required can be found in the VSE/VSAM Programmer’s Reference.

- In CMS, to define and identify VSAM catalogs, clusters, and data spaces; to identify VSAM files used for program input/output; and to identify input/output files for AMSERV.

The format of the DLBL command is:

```
DLBL [ddname] [mode] [CMS fn ft] [(option A option B [ ])]
      [CMS FILE ddname]
      [DSN qual1 qual1] [(qual2 qualn) ]
      [DSN ?]

{ddname} CLEAR

Option A: [SYS xxx]
Option B: [PERM] [CHANGE NOCHANGE]
Option C: [VSAM] [EXTENT] [CAT catdd] [BUFSP nnnnn]
```

Note: The operands and options of the DLBL command are described below. Usage notes are provided for general usage, followed by additional notes for CMS/DOS users, and then additional notes for OS VSAM users.
where:

ddbname
specifies a one- to seven-character program ddname (OS) or filename (VSE), or dname (as specified in the FILE parameter of an access method services control statement). An asterisk (*) entered with the CLEAR operand indicates that all DLBL definitions, except those that are entered with the PERM option, are to be cleared.

mode
specifies a valid CMS disk mode letter and optionally, filemode number. A letter must be specified; if a number is not specified, it defaults to 1. The disk must be accessed when the DLBL command is issued. In the CMS/DOS and CMS/VSAM environments, filemodes “R” and “T” cannot be used on the DLBL command. This is because “R” and “T” are used as abbreviations for reader and terminal in the CMS/DOS and CMS/VSAM environments.

DUMMY
specifies that no real I/O is to be performed. A read operation results in an end-of-file condition and a write operation results in a successful return code. DUMMY should not be used for OS VSAM data sets (see Usage Note 3 on page 113).

CLEAR
removes any existing definitions for the specified ddname. Clearing a ddname before defining it ensures that a file definition does not exist and that any options previously defined with that ddname no longer have any effect.

CMS fn ft
indicates that this is a CMS file, and the file identifier (fn ft) that follows is a CMS filename and filetype.

FILE ddbname is the default CMS file identifier associated with all non-CMS data sets. (See Usage Note 3 on page 113 for CMS/DOS users.)

DSN
indicates that this is a non-CMS file.

?
indicates that you are going to enter the data set name interactively. When prompted, you enter the data set name or fileid in its exact form, including embedded blanks, hyphens, or periods.

qual1 [qual2...qualn]
-- or --
qual1 [qual2...qualn]
is an OS data set name or VSE file-id. Only data sets named according to standard OS conventions may be entered this way; you may omit the periods between qualifiers, or specify the full dataset name, including periods between qualifiers. (See Usage Note 2 on page 113.)
Options:

SYSSxx
(CMS/DOS only) indicates the system or programmer logical unit that is associated with the disk on which the disk file resides. The logical unit must have been previously assigned with the ASSGN command. In many situations VSE/VSAM does not require a SYSSxxx operand. Thus no previous ASSGN is required. See VSE/VSAM Programmer's Reference for information on when the SYSSxxx operand is required.

PERM
indicates that this DLBL definition can be cleared only with an explicit CLEAR request. It will not be cleared when the DLBL * CLEAR command line is entered.

All DLBL definitions, including those entered with the PERM option, are cleared as a result of a program abend or HX (halt execution) Immediate command.

CHANGE
indicates that any existing DLBL for this ddname is not to be canceled, but that conflicting options are to be overridden and new options merged into the old definition. Both the ddname and the file identifier must be the same in order for the definitions to be merged.

NOCHANGE
does not alter any existing DLBL definition for the specified ddname, but creates a definition if none existed.

VSAM
indicates that the file is a VSAM data set. This option must be specified for VSAM functions unless the EXTENT, MULT, CAT, or BUFSP options are entered or the ddnames IJSYSCT or IJSYSUC are used.

EXTENT
indicates that you are going to use access method services to define a VSAM catalog, data space, or unique cluster and you want to enter extent information.

MULT
indicates that you are going to reference an existing multivolume data set and you want to enter the volume specifications.

Note: In many situations VSE/VSAM does not require EXTENT or MULT information. See VSE/VSAM Programmer's Reference for information on when EXTENT or MULT information is required.

CAT catdd
identifies the VSAM catalog (defined by a previous DLBL definition) which contains the entry for this data set. You must use the CAT option when the VSAM data set you are creating or identifying is not
Usage Notes:

1. To display all of the disk file definitions in effect, enter:

   dlbl

   The response will be:

   ddname DISK fn ft
   . . . .

   If no DLBL definitions are in effect, the following message is displayed:

   DMSDLB324I No user defined DLBLs in effect

2. You may enter an OS or VSE file identification on the DLBL command line. The maximum length of the file identification is 44 characters, including periods. For example, the file TEST.INPUT.SOURCE.D could be identified as follows:

   dlbl ddl c dsn test input source d (options...
   -- or --
   dlbl ddl c dsn test.input.source.d (options...

   Or, it may be entered interactively, as follows:

   dlbl ddl c dsn ? (options
   DMSDLB220R Enter data set name:

   test.input.source.d

   If the dataset name is entered interactively, the dataset name must be entered in its exact form. If it is entered as a command, or from EXEC 2, the dataset name may be entered in its exact form. If the command is entered with blanks separating the qualifiers, DLBL replaces them with periods. If it is entered via CMS EXEC, the periods between qualifiers must be omitted, and the qualifiers must be 1 to 8 characters long.

3. In VSE, a VSAM data set that has been defined as DUMMY is opened with an error code of X'11'. CMS supports the DUMMY operand of the DLBL command in the same manner. OS users should not use the
DLBL

DUMMY operand in CMS, since a dummy data set does not return, on open, an end-of-file indication.

4. Do not use the same ddname for a CMS disk if a DLBL already exists with the same ddname for a DOS disk. The use of DSN and CMS is not interchangeable.

5. DLBL uses the extended plist for processing the DSN quail放过qual2...qualn] parameter. If you are calling DLBL from an assembler language program and using DSN quail放过qual2...qualn], you should supply an extended plist. VM/SP CMS for System Programming has more information on how an assembler language program can supply an extended plist.

Additional Notes for CMS/DOS Users:

1. Each DLBL definition must be associated with a system or programmer logical unit assignment, previously made with an ASSGN command. Specify the SYSxxx option on the first, or only, DLBL definition for a particular ddname. Many DLBL definitions may be associated with the same logical unit. For example:

   ```
   assgn sys100 b
   dlb1 dd1 b cms test file1 (sys100
   dlb1 dd2 b cms test file2 (sys100
   dlb1 dd2 cms test file3
   ```

   is a valid command sequence.

   In many situations VSE/VSAM does not require the DLBL command. See VSE/VSAM Programmer's Reference for information on when the DLBL command is required.

2. The following special ddnames must be used to define VSE private libraries, and must be associated with the indicated logical units:

<table>
<thead>
<tr>
<th>ddname</th>
<th>Logical Unit</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>IJSYSSL</td>
<td>SYSSLB</td>
<td>Source statement</td>
</tr>
<tr>
<td>IJSYSLR</td>
<td>SYSRLB</td>
<td>Relocatable</td>
</tr>
<tr>
<td>IJSYSLC</td>
<td>SYSLCD</td>
<td>Core Image</td>
</tr>
</tbody>
</table>

   These libraries must be identified in order to perform librarian functions (with the SSERV, ESERV, DSERV, or RSERV commands) for private libraries; or to link-edit or fetch modules or phases from private relocatable or core image libraries (with the DOSLKED and FETCH commands).

3. Each VSE file has a CMS file identifier associated with it by default; the filename is always FILE and the filetype is always the same as the ddname. For example, if you enter a DLBL command for a VSE file MOD.TEST.STREAM as follows:

   ```
   dlb1 test c dsn mod.test.stream
   ```
then you can refer to this data set as FILE TEST when you use the STATE command:

```
state file test
```

When you enter a DLBL command specifying only a ddname and mode, as follows:

```
dlbl junk a
```

CMS assigns a file identifier of FILE JUNK A1 to the ddname JUNK.

4. The FILEDEF command performs a function similar to that of the DLBL command; you need to use the FILEDEF command in CMS/DOS only:

- When you want to override a default ddname for an assembler input or output file.
- When you want to use the MOVEFILE command to process a file.

5. If you use the DUMMY operand, you must have issued an ASSGN command specifying a device type of IGN, or ignore, for the SYSxxx unit specified in the DLBL command, for example,

```
assign sys003 ign
dlbl test dummy (sys003
```

**Specifying VSAM Extent Information:** You may specify extent information when you use the access method services control statements DEFINE SPACE, DEFINE MASTERCATALOG, DEFINE USERCATALOG, DEFINE CLUSTER (UNIQUE); or when you use the IMPORT or IMPORTRA functions for a unique file.

In many situations, VSE/VSAM does not require EXTENT information. See VSE/VSAM Programmer's Reference for information on when EXTENT information is required.

When you enter the EXTENT option of the DLBL command, you are prompted to enter the disk extents for the specified file. You must enter extent information in accordance with the following rules:

- For count-key-data devices, you must specify the starting track number and number of tracks for each extent, as follows:

```
19 38
```

This extent allocates 38 tracks, beginning with the 19th track, on a 3330 device.

- For fixed-block devices, you must specify the starting block number and the number of blocks for each extent. The following example allocates 200 blocks, starting at block number 352, on a fixed-block device.

```
352 200
```
Because VSAM rounds the starting block to the next highest cylinder boundary, it is advisable to specify the starting block on a cylinder boundary.

- All count-key-data extents must begin and end on cylinder boundaries, regardless of whether the AMSERV file contains extent information in terms of cylinders, tracks, or records.

- Multiple extent entries may be entered on a single line separated by commas or on different lines. Commas at the end of a line are ignored.

- Multiple extents for the same volume must be entered in numerically ascending order; for example:

```
20 400, 600 80
```

These extents are valid for a 2314 device.

- When you enter multivolume extents, you must specify the mode letter and logical unit associated with each disk that contains extents; extents for each disk must be entered consecutively. For example:

```
assgn sys001 b
assgn sys002 c
assgn sys003 d
dbl file1 b (extent sys001
DMSDLB331R Enter extent specifications:
100 60, 400 80, 60 40 d sys003
200 100 c sys002
400 100 c sys002
(null line)
```

specifies extents on disks accessed at modes B, C, and D. These disks are assigned to the logical units SYS001, SYS002, and SYS003. Since B is the mode specified on the DLBL command line, it does not need to be respecified along with the extent information.

- A DASD volume must be mounted, accessed, and assigned for each disk mode referenced in an extent.

When you are finished entering extent information, you must enter a null line to terminate the DLBL command sequence. If you do not, an error may result and you will have to reenter the DLBL command. If you make any error entering the extents, you must reenter all the extent information.

The DLBL command does not check the extents to see whether they are on cylinder boundaries or whether they are entered in the proper sequence. If you do not enter them correctly, the access method services DEFINE function will terminate with an error.

CMS assigns sequence numbers to the extents according to the order in which they were entered. These sequence numbers are listed when you use the LISTDS command with the EXTENT option.
In order to display the actual extents that were entered for a VSAM data set at DLBL definition time, the following commands may be entered:

DLBL (EXTENT) or QUERY DLBL EXTENT

Either of these commands will provide the following information to the user:

**DDNAME**  The VSE filename or OS ddname.

**MODE**  The CMS disk mode identifying the disk on which the extent resides.

**LOGUNIT**  The VSE logical unit specification (SYSxxx). This operand will be blank for a data set defined while in CMS/OS environment; that is, the SET DOS ON command had not been issued at DLBL definition time.

**EXTENT**  Specifies the relative starting track number and number of tracks for each extent entered for the given dataset ddname.

If no DLBL definitions with extent information are active, the following message is issued:

DMSDLB3241  No user defined EXTENTS in effect

**Identifying Multivolume VSAM Extents:**  When you want to execute a program or use access method services to reference an existing multivolume VSAM data set, you may use the MULT option on the DLBL command that identifies the file.

In many situations, VSE/VSAM does not require this information. See VSE/VSAM Programmer's Reference for information on when this type of EXTENT information is required.

When you use the MULT option, you are prompted to enter additional disk mode letters, as follows:

```plaintext
assgn sys001 c
assgn sys002 d
assgn sys003 e
assgn sys004 f
assgn sys005 g
dlbl infile c (mult sys001
DMSDLB330R  Enter volume specifications:

d sys002, e sys003 , f sys004
  g sys005 (null line)
```

The above identifies a file that has extents on disks accessed at modes C, D, E, F, and G. These disks have been assigned to the logical units SYS001, SYS002, SYS003, SYS004, and SYS005. The rules for entering multiple extents are:
• All disks must be mounted, accessed, and assigned when you issue the DLBL command.

• You must not repeat the mode letter and logical unit of the disk that is entered on the DLBL command line (C in the above example).

• If you enter more than one mode letter and logical unit on a line, they must be separated by commas; trailing commas on a line are ignored.

• A maximum of nine disks may be specified; you do not need to specify them in alphabetical order.

You must enter a null line to terminate the command when you are finished entering extents; if not, an error may result and you must reenter the entire command sequence.

In order to display the volumes on which all multivolume data sets reside, the following commands are issued:

```
DLBL (MULT) or QUERY DLBL MULT
```

The following information concerning multiple volume datasets is provided:

**DDNAME**  The VSE filename or OS ddname.

**MODE**  The CMS disk mode identifying one of the disks on which the dataset resides.

**LOGUNIT**  The VSE logical unit specification (SYSxxx). This operand will be blank for a data set defined while in CMS/OS environment; that is, the SET DOS ON command had not been issued at DLBL definition time.

If no DLBL definitions with multiple volume specifications are active, the following message is issued:

```
DMSDLB324I No user defined MULTs in effect
```

**Using VSAM Catalogs:**  There are two special ddnames you must use to identify a VSAM master catalog and job catalog:

**IJSYSCT**  identifies the master catalog when you initially define it (using AMSERV), and when you begin a terminal session. You should use the PERM option when you define it.

You must assign the logical unit SYSCAT to the disk on which the master catalog resides. If you are redefining a master catalog that has already been identified, you may omit the SYSCAT option on the DLBL command line.
**IJSYSUC** identifies a job catalog to be used for subsequent AMSERV jobs or VSAM programs.

Any programmer logical unit may be used to assign a job catalog.

Only one VSAM catalog is ever searched when a VSAM function is performed. If a job catalog is defined, you may override it by using the CAT option on the DLBL command for a data set. The following DLBL command sequence illustrates the use of catalogs:

```plaintext
assign syscat c
dlbl ijsysct c dsn mastcat (perm syscat

identifies the master catalog, MASTCAT, for the terminal session.

assign sys010 d
dlbl ijsysuc d dsn mycat (perm sys010

identifies the job (user) catalog, MYCAT, for the terminal session.

assign sys100 e
dlbl inttest1 e dsn test.case (vsam sys100

identifies a VSAM file to be used in a program. It is cataloged in the job catalog, MYCAT.

assign sys101 f
dlbl cat3 f dsn testcat (cat ijsysct sys101

identifies an additional user catalog, which has an entry in the master catalog. Since a job catalog is in use, you must use the CAT option to indicate that another catalog, in this case the master catalog, should be used.

dlbl infile f dsn test.input (cat cat3 sys101

identifies an input file cataloged in the user catalog TESTCAT, which was identified with a ddname of CAT3 on the DLBL command.

The selection of a VSAM catalog for AMSERV jobs and VSAM programs running in CMS is summarized in Figure 6 on page 120.
Usage Notes for OS VSAM Users:

1. You may use the DLBL command to identify all access method services input and output files, and to identify all VSAM input and output files referenced in programs. For all other file definitions, including OS or CMS disk files referenced in programs that use VSAM data management, you must use the FILEDEF command.

File definition statements, either DLBL or FILEDEF, are not always required by VSAM. For more information on file definition requirements, see VSE/VSAM Programmer's Reference.

2. A DLBL ddname may have a maximum of seven characters. If you have ddnames in your programs that are eight characters long, only the first seven characters are processed when the programs are executed in CMS. If you have two ddnames with the same first seven characters and you attempt to execute this program in CMS, you will receive an open error when the second file is opened. You should recompile these programs providing unique seven-character ddnames.

3. If you release a disk for which you have a DLBL definition in effect, you should clear the DLBL definition before you execute a VSAM program or an AMSERV command. CMS checks that all disks for
which there are DLBL definitions are accessed, and issues error message DMSSTT069E if any are not.

4. The DLBL command does not support the DISP option. DISP is used in VSE/VSAM to specify the disposition of a reusable file. Therefore, in CMS, only the default values are available. For more information on the DISP option, refer to the VSE/VSAM Programmer's Reference.

**Specifying VSAM Extent Information:** You may specify extent information when you use the access method services control statements DEFINE SPACE, DEFINE MASTERCATALOG, DEFINE USERCATALOG, DEFINE CLUSTER (UNIQUE); or when you use the IMPORT or IMPORTTRA functions for a unique file. Space allocation is made only for primary allocation amounts.

In many situations, VSE/VSAM does not require EXTENT information. See VSE/VSAM Programmer's Reference for information on when EXTENT information is required.

When you enter the EXTENT option of the DLBL command, you are prompted to enter the disk extents for the specified file. You must enter extent information in accordance with the following rules:

- For count-key-data devices, you must specify the starting track number and number of tracks for each extent, as follows:

  19 38

  This extent allocates 38 tracks, beginning with the 19th track, on a 3330 device.

- For fixed-block devices, you must specify the starting block number and the number of blocks for each extent. The following example allocates 200 blocks, starting at block number 352, on a fixed-block device.

  352 200

  Because VSAM rounds the starting block to the next highest cylinder boundary, it is advisable to specify the starting block on a cylinder boundary.

- All count-key-data extents must begin and end on cylinder boundaries, regardless of whether the AMSERV file contains extent information in terms of cylinders, tracks, or records.

- Multiple extent entries may be entered on a single line separated by commas or on different lines. Commas at the end of a line are ignored.

- Multiple extents for the same volume must be entered in numerically ascending order; for example:

  20 400, 600 80

  These extents are valid for a 2314 device.
• When you enter multivolume extents, you must specify the mode letter for extents on additional disks; extents for each disk must be entered consecutively. For example:

dlbl file1 b (extent

DMSDLB331R Enter extent specifications:
100 60, 400 80, 60 40 d
200 100 c
400 100 c
(null line)

specifies extents on disks accessed at modes B, C, and D. Since B is the mode specified on the DLBL command line, it does not need to be re-specified along with the extent information.

• A DASD volume must be mounted and accessed for each mode referenced in an extent.

When you are finished entering extent information, you must enter a null line to terminate the DLBL command sequence. If you do not, an error may result and you will have to reenter the entire DLBL command. If you make any error entering the extents, you must reenter all the extent information.

The DLBL command does not check the extents to see if they are on cylinder boundaries or that they are entered in the proper sequence. If you do not enter them correctly, the access method services DEFINE function terminates with an error.

CMS assigns sequence numbers to the extents according to the order in which they were entered. These sequence numbers are listed when you use the LISTDS command with the EXTENT option.

**Identifying Multivolume VSAM Extents:** When you want to execute a program or use access method services to reference an existing multivolume VSAM data set, you may use the MULT option on the DLBL command that identifies the file.

In many situations, VSE/VSAM does not require this information. See VSE/VSAM Programmer's Reference for information on when this type of EXTENT information is required.

When you use the MULT option, you are prompted to enter additional disk mode letters, as follows:

dlbl infile c (mult

DMSDLB330R Enter volume specifications:
 d, e, f
g
=null line)

The above example identifies a file that has extents on disks accessed at modes C, D, E, F, and G. The rules for entering multiple extents are:
• All disks must be mounted and accessed when you issue the DLBL command.

• You must not repeat the mode letter of the disk that is entered on the DLBL command line (C in the above example).

• If you enter more than one mode letter on a line, they must be separated by commas; trailing commas on a line are ignored.

• A maximum of nine disks may be specified; you do not need to specify them in alphabetical order.

You must enter a null line to terminate the command when you are finished entering extents; if not, an error may result and you must re-enter the entire command sequence.

Using VSAM Catalogs: There are two special ddnames you must use to identify a VSAM master catalog and job catalog:

IJSYSCCT identifies the master catalog, both when you initially define it (using AMSERV) and when you begin a terminal session. You should use the PERM option when you define it.

IJSYSUC identifies a job catalog to be used for subsequent AMSERV jobs or VSAM programs.

Only one VSAM catalog is ever searched when a VSAM function is performed. If a job catalog is defined, you may override it by using the CAT option on the DLBL command for a data set. The following DLBL command sequence illustrates the use of catalogs:

dlbl ijsysct c dsn mastcat (perm
identifies the master catalog, MASTCAT, for the terminal session.

dlbl ijsysuc d dsn mycat (perm
identifies the job (user) catalog, MYCAT, for the terminal session.

dlbl intestl e dsn test.case (vsam
identifies a VSAM file to be used in a program. It is cataloged in the job catalog, MYCAT.

dlbl cat3 dsn testcat (cat ijsysct
identifies an additional user catalog, which has an entry in the master catalog. Since a job catalog is in use, you must use the CAT option to indicate that another catalog, in this case the master catalog, should be used.

dlbl infile e dsn test.input (cat cat3
identifies an input file cataloged in the user catalog TESTCAT, which was identified with a ddname of CAT3 on the DLBL command.
The selection of a VSAM catalog for AMSERV jobs and VSAM programs running in CMS is summarized in Figure 6 on page 120.

Responses:

If the DLBL command is issued with no operands, the current DLBL definitions are displayed at your terminal:

```
ddnamel devicel [fnl fnl fnl [datasetnamel]]
  :  :  :  :  :
ddnamen devicen [fnn ftn fmn [datasetnamen]]
```

DMSDLB220R Enter data set name:

This message is displayed when you use the DSN ? form of the DLBL command. Enter the exact DOS or OS data set name.

DMSDLB320I Maximum number of disk entries recorded

This message indicates that nine volumes have been specified for a VSAM data set, which is the maximum allowed under CMS.

DMSDLB321I Maximum number of extents recorded

This message indicates that 16 extents on a single disk or minidisk have been specified for a VSAM data space, catalog, or unique data set. This is the maximum number of extents allowed on a minidisk or disk.

DMSDLB322I DDNAME ddname not found; no CLEAR executed

This message indicates that the clear function was not performed because no DLBL definition is in effect for the ddname.

DMSDLB323I {Job|Master} catalog DLBL cleared

This message indicates that either the master catalog or job catalog has been cleared as a result of a clear request.

You also receive this message if you issue a DLBL * CLEAR command, and any DLBL definition is in effect for IJSYSCT or IJSYSUC that was not entered with the PERM option.

DMSDLB330R Enter volume specifications:

This message prompts you to enter volume specifications for existing multivolume VSAM files. (See “Identifying Multivolume VSAM Extents” in the appropriate usage section.)
DMSDLB331R Enter extent specifications:

This message prompts you to enter the data set extent or extents of a new VSAM data space, catalog or unique data set. (See “Specifying VSAM Extent Information” in the appropriate usage section.)

Messages and Return Codes:

DMSDLB001E No filename specified [RC = 24]
DMSDLB003E Invalid option: option [RC = 24]
DMSDLB005E No option specified [RC = 24]
DMSDLB023E No filetype specified [RC = 24]
DMSDLB048E Invalid mode mode [RC = 24]
DMSDLB050E Parameter missing after DDNAME [RC = 24]
DMSDLB065E option option specified twice [RC = 24]
DMSDLB066E option1 and option2 are conflicting options [RC = 24]
DMSDLB069E Disk mode not accessed [RC = 36]
DMSDLB070E Invalid parameter parameter [RC = 24]
DMSDLB086E Invalid DDNAME ddname [RC = 24]
DMSDLB109S Virtual storage capacity exceeded [RC = 104]
DMSDLB221E Invalid data set name [RC = 24]
DMSDLB301E SYSaaa not assigned for disk fm [RC = 36]
DMSDLB302E No SYSXXX operand entered [RC = 24]
DMSDLB304E Invalid operand value value [RC = 24]
DMSDLB305E Incomplete extent range [RC = 24]
DMSDLB306E SYSaaa not assigned for IGNORE [RC = 36]
DMSDLB307E Catalog DDNAME ddname not found [RC = 24]
DMSDLB308E mode disk in [non-]CMS format; invalid for [non-]CMS dataset [RC = 24]
DOSLIB

Use the DOSLIB command to delete, compact, or list information about the executable phases in a CMS/DOS phase library.

The format of the DOSLIB command is:

<table>
<thead>
<tr>
<th>DOSLIB</th>
<th>{DEL libname phasename1 [...phasenamen]}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{COMP libname}</td>
</tr>
<tr>
<td></td>
<td>{MAP libname [ (options...[]) ] }</td>
</tr>
<tr>
<td>Options:</td>
<td>[TERM] [DISK] [PRINT]</td>
</tr>
</tbody>
</table>

where:

DEL

   deletes phases from a CMS/DOS phase library. The library is not erased when the last phase is deleted from the library.

COMP

   compacts a CMS/DOS phase library.

MAP

   lists certain information about the phases of a DOSLIB. Available information provided is phase name, size, and relative location in the library.

libname

   is the filename of a CMS/DOS phase library. The filetype must be DOSLIB.

phasename1...phasenamen

   is the name of one or more phases that exist in the CMS/DOS phase library.
MAP Options:

The following options specify the output device for the MAP function. If more than one option is specified, only the first option is used.

**TERM**

displays the MAP output at the terminal.

**DISK**

writes the MAP output to a CMS disk file with the file identifier of 'libname MAP A5'. If a file with that name already exists, the old file is erased.

**PRINT**

spools the MAP output to the virtual printer.

Usage Notes:

1. The CMS/DOS environment does not have to be active when you issue the DOSLIB command.

2. Phases may only be added to a DOSLIB by the CMS/DOS linkage editor as a result of the DOSLKED command.

3. In order to fetch a program phase from a DOSLIB for execution, you must issue the GLOBAL command to identify the DOSLIB. When a FETCH command or dynamic fetch from a program is issued, all current DOSLIBs are searched for the specified phases.

4. If DOSLIBs are very large, or there are many of them to search, program execution is slowed down accordingly. To avoid excessive execution time, you should keep your DOSLIBs small and issue a GLOBAL command specifying only those libraries that you need.

Example:

To compact MYLIB DOSLIB, you would issued the command:

```
doslib comp mylib
```

Responses:

When you use the TERM option on the DOSLIB MAP command line, the following is displayed:

<table>
<thead>
<tr>
<th>PHASE</th>
<th>INDEX</th>
<th>BLOCKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>name1</td>
<td>loc</td>
<td>size</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>
DOSLIB

Messages and Return Codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSDSL002E</td>
<td>File fn DOSLIB not found [RC = 28]</td>
</tr>
<tr>
<td>DMSDSL003E</td>
<td>Invalid option: option [RC = 24]</td>
</tr>
<tr>
<td>DMSDSL013W</td>
<td>Phase phase not found in library libname [RC = 4]</td>
</tr>
<tr>
<td>DMSDSL014E</td>
<td>Invalid function function [RC = 24]</td>
</tr>
<tr>
<td>DMSDSL037E</td>
<td>Disk mode[(vdev)] is accessed as read/only [RC = 36]</td>
</tr>
<tr>
<td>DMSDSL046E</td>
<td>No library name specified [RC = 24]</td>
</tr>
<tr>
<td>DMSDSL047E</td>
<td>No function specified [RC = 24]</td>
</tr>
<tr>
<td>DMSDSL069E</td>
<td>Disk mode not accessed [RC = 36]</td>
</tr>
<tr>
<td>DMSDSL070E</td>
<td>Invalid parameter parameter [RC = 24]</td>
</tr>
<tr>
<td>DMSDSL098E</td>
<td>No phase name specified [RC = 24]</td>
</tr>
<tr>
<td>DMSDSL104S</td>
<td>Error nn reading file fn ft fm from disk [RC = 100]</td>
</tr>
<tr>
<td>DMSDSL105S</td>
<td>Error nn writing file fn ft fm on disk [RC = 100]</td>
</tr>
<tr>
<td>DMSDSL213W</td>
<td>Library fn ft fm not created [RC = 4]</td>
</tr>
</tbody>
</table>
Use the DOSLKED command in CMS/DOS to link-edit TEXT files from CMS disks or object modules from VSE private or system relocatable libraries and place them in executable form in a CMS phase library (DOSLIB).

The format of the DOSLKED command is:

\[
\text{DOSLKED} \quad fn \quad [\text{libname}] \quad [(\text{options...} [\text{)])} \quad \text{Options: } \quad \begin{array}{l}
\text{DISK} \\
\text{PRINT} \\
\text{TERM}
\end{array}
\]

where:

\( fn \)

specifies the name of the source file or module to be link-edited. CMS searches for:

1. A CMS file with a filetype of DOSLNK
2. A module in a private relocatable library (if IJSYSRL has been defined)
3. A CMS file with a filetype of TEXT
4. A module in the system relocatable library (if a mode was specified on the SET DOS ON command line)

\( \text{libname} \)

designates the name of the DOSLIB where the link-edited phase is to be written. The filetype is DOSLIB. If libname is not specified, the default is fn. The output filemode of the DOSLIB is determined as follows:

- If libname DOSLIB exists on a read/write disk, that filemode is used and the output is appended to it.
- If fn DOSLNK exists on a read/write disk, libname DOSLIB is written to that disk.
**DOSLKED**

- If fn DOSLNK exists on a read-only extension of a read/write disk, libname DOSLIB is written to the parent disk.

- If none of the above apply, libname DOSLIB is written to your A-disk.

**Options:**

Only one of the following options should be specified. If more than one is specified, only the first entry is used.

**DISK**

writes the linkage editor map produced by the DOSLKED command on your A-disk into a file with the filename of fn and a filetype of MAP. This is the default option.

**PRINT**

spools the linkage editor map to the virtual printer.

**TERM**

displays the linkage editor map at your terminal.

*Note:* All error messages are sent to the terminal as well as to the specified device.

**Usage Notes:**

1. You can create a CMS file with a filetype of DOSLNK to contain linkage editor control statements and, optionally, CMS text files.

2. If you want to link-edit a module from a private relocatable library, you must issue an ASSGN command for the logical unit SYSRLB and enter a DLBL command using a ddname of IJSYSRL to identify the library:

   ```
   assgn sysrlb c
dlbl ijsysrl c dsn reloc lib (sysrlb
   ```

   If you have defined a private relocatable library but do not want it to be searched, enter:

   ```
   assgn sysrlb ign
   ```

   to temporarily bypass it.

3. CMS TEXT files may also contain linkage editor control statements INCLUDE, PHASE, and ENTRY. The ACTION statement is ignored when a TEXT file is link-edited.

4. To access modules on a VSE system residence volume, you must have specified the mode letter of the system residence on the SET DOS ON command line:
set dos on z

5. The search order that CMS uses to locate object modules to be link-edited is:

   a. The specified object module on the VSE private relocatable library, if one is available
   b. CMS disks for a file with the specified filename and with a filetype of TEXT
   c. The specified object module on the VSE system relocatable library, if it is available

6. When a phase is added to an existing DOSLIB, it is always written at the end of the library. If a phase that is being added has the same name as an existing phase, the DOSLIB directory is updated to point to the new phase. The old phase is not deleted, however; you should issue the DOSLIB command with the COMP option to compress the space.

   If you run out of space in a DOSLIB while you are executing the DOSLKED command, you should reissue the DOSLKED command specifying a different DOSLIB, or compress the DOSLIB before attempting to reissue the DOSLKED command.

7. Prior to performing a DOSLKED on a TEXT file having multiple phase cards following the TEXT END cards, rename the filetype of TEXT to a filetype DOSLNK.

8. If the input to the DOS linkage editor contains a text file produced by punching a member from a TXTLIB, the last two records are LDT records. When using OS linkage edit, these are recognized as end cards. However, when you are using the DOSLKED command, delete these two cards prior to issuing the command. Failure to do so may cause unpredictable results.

**Linkage Editor Control Statements:** The CMS/DOS linkage editor recognizes and supports the VSE linkage editor control statements ACTION, PHASE, ENTRY, and INCLUDE. The CMS/DOS linkage editor ignores:

- The SVA operand of the PHASE statement
- The F + address form for specifying origin on the PHASE statement
- The BG, Fn, and SMAP operands of the ACTION statement

The S-form of specifying the origin on the PHASE statement corresponds to the CMS user area under CMS/DOS. If a default PHASE statement is required, the origin is assumed to be S. The PBDY operand of the PHASE statement indicates that the phase is link-edited on a 4K page boundary under CMS/DOS as opposed to a 2K page boundary for VSE.
In VSE, an ACTION, CLEAR control statement clears the unused portion of the core image library to binary zeros. In VSE, the core image library has a defined size, while in CMS/DOS the CMS phase library varies in size, depending on the number of phases cataloged. Therefore, in CMS/DOS an ACTION CLEAR control statement clears the current buffers to binary zeros before loading them; CMS/DOS cannot clear the entire unused portion of the CMS phase library because that portion varies as phases are added to and deleted from the CMS phase library. In CMS/DOS if you want your phases cleared you must issue an ACTION CLEAR control statement each time you add a phase to the CMS phase library.

**Linkage Editor Card Types:** The input to the linkage editor can consist of six card types, produced by a language translator or a programmer. These cards appear in the following order:

<table>
<thead>
<tr>
<th>Card Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESD</td>
<td>External symbol dictionary</td>
</tr>
<tr>
<td>SYM</td>
<td>Ignored by linkage editor</td>
</tr>
<tr>
<td>TXT</td>
<td>Text</td>
</tr>
<tr>
<td>RLD</td>
<td>Relocation list dictionary</td>
</tr>
<tr>
<td>REP</td>
<td>Replacement of text made by the programmer</td>
</tr>
<tr>
<td>END</td>
<td>End of module</td>
</tr>
</tbody>
</table>

CMS/DOS supports these six card types in the same manner that VSE does.

**Example:**

To link-edit the TEST TEXT file and place it in the TESTLIB DOSLIB, enter the command:

```
doslked test testlib
```

**Responses:**

When you use the TERM option of the DOSLKED command, the linkage editor map is displayed at the terminal.

21011 INVALID OPERATION IN CONTROL STATEMENT

This message indicates that a blank card was encountered in the process of link-editing a relocatable module. This message also appears in the MAP file. The invalid card is ignored and processing continues.
Messages and Return Codes:

- **DMSDLK001E**: No filename specified [RC = 24]
- **DMSDLK003E**: Invalid option: *option* [RC = 24]
- **DMSDLK006E**: No read/write disk accessed [RC = 36]
- **DMSDLK007E**: File *fn ft fm* is not fixed, 80-character records [RC = 32]
- **DMSDLK070E**: Invalid parameter *parameter* [RC = 24]
- **DMSDLK099E**: CMS/DOS environment not active [RC = 40]
- **DMSDLK104S**: Error *nn* reading file *fn ft fm* from disk [RC = 100]
- **DMSDLK105S**: Error *nn* writing file *fn ft fm* on disk [RC = 100]
- **DMSDLK210E**: Library *fn ft* is on a read/only disk [RC = 36]
- **DMSDLK245S**: Error *nnn* on printer [RC = 100]
DROPBUF

Use the DROPBUF command to eliminate only the most recently created program stack buffer or a specified program stack buffer and all buffers created after it.

The format of the DROPBUF command is:

```
DROPBUF  n
```

where:

- `n` indicates the number of the first program stack buffer you want to drop. CMS drops the indicated buffer and all buffers created after it. If `n` is not specified, only the most recently created buffer is dropped.

Usage Note:

Note that you can specify a number with DROPBUF. For example, if you issue:

```
DROPBUF  4
```

CMS eliminates program stack buffer 4 and all program stack buffers created after it. Thus, if there were presently six program stack buffers, CMS would eliminate program stack buffers 6, 5, and 4. If you issued DROPBUF without specifying `n`, only program stack buffer 6 would be eliminated.

Responses:

None.

Return Codes:

If an error occurs in DROPBUF processing, Register 15 contains one of the following nonzero return codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Invalid number specified</td>
</tr>
<tr>
<td>2</td>
<td>Specified buffer does not exist</td>
</tr>
</tbody>
</table>
DROP WINDOW

Use the DROP WINDOW command to move a window down in the order of displayed windows.

The format of the DROP WINDOW command is:

```
DROP WINDOW
{ wname  } [ n ]
 WM
```

**where:**

- `wname` is the name of the window to be dropped.
- `=` indicates that the topmost window is dropped.
- `WM` hides the WM window so that it is no longer visible on the screen and commands cannot be entered from it. You are returned to the environment you were in before entering `POP WINDOW WM`. The `n` and `*` have no effect when specified with `WM`.
- `n` is the number of positions the window is moved down. An “*” positions the window behind all other windows. If this operand is not specified, “*” is assumed.

**Usage Notes:**

1. If the window is hidden (see HIDE WINDOW), then DROP WINDOW has no effect.

2. When using full-screen CMS, you cannot drop a window that is showing the virtual screen indicated in the status area message. For example, if the CMS window is showing the CMS virtual screen, and the status area message instructs you to “Scroll for more information in vscreen CMS,” you cannot drop the CMS window. You can drop any other window.

3. In the WM window, the PA2 and CLEAR keys scroll the topmost window forward. When there is no more data in the window to scroll, you automatically exit the WM environment.
DROP WINDOW

Responses:

None.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSDRP386E  Missing operand(s) [RC = 24]
- DMSDRP388E  Invalid keyword: keyword [RC = 24]
- DMSDRP389E  Invalid operand: operand [RC = 24]
- DMSDRP391E  Unexpected operand(s): operand [RC = 24]
- DMSDRP921E  Window $username$ is not defined [RC = 28]
- DMSDRP929E  Window $username$ is not connected to a virtual screen [RC = 36]
Use the DSERV command in CMS/DOS to obtain information that is contained in VSE private or system libraries.

The format of the DSERV command is:

```
DSERV
  CD [PHASE {name [nn]} [d2...dn] [options...[ ]]]
  RD
  SD
  PD
  TD
  ALL

Options:
  [DISK]
  [TERM]
  [SORT]
```

where:

- **CD**
- **RD**
- **SD**
- **PD**
- **TD**
- **ALL**

specifies that information concerning one or more types of directories is to be displayed or printed. The directory types that can be specified are: CD (core image library), RD (relocatable library), SD (source statement library), PD (procedure library), TD (transient directory), and ALL (all directories).

There is no default value. The private libraries take precedence over system libraries.

**PHASE name**

specifies the name of the phase to be listed. If the phasename ends with an asterisk, all phases that start with the letters preceding the asterisk are listed. This operand is valid only for CD.
**DSERV**

\( nn \)

is the displacement within the phase where the version and level are to be found (the default is 12).

\( [d2...dn] \)

indicates additional libraries whose directories are to be listed. (See Usage Note 1.)

**Options:**

**DISK**

writes the output on your CMS A-disk to a file named DSERV MAP A5. This is the default value if TERM or PRINT is not specified.

**TERM**

displays the output at your terminal.

**PRINT**

spools the output to the system printer.

**SORT**

sorts the entries for each library alphanumerically. Otherwise, the order is the order in which the entries were cataloged, except in the case of the Core Image Library, which is always written alphanumerically by VSE for performance reasons.

**Usage Notes:**

1. You may specify more than one directory on DSERV command line; for example:

```
dserv rd sd cd phase $$bopen (term
```

displays the directories of the relocatable and source statement libraries, as well as the entry for the phase $$BOPEN from the core image directory.

You can specify only one phasename or phasename* at a time. However, if you specify more than one PHASE operand, only the last one entered is listed. For example, if you enter:

```
dserv cd phase cor* phase idc*
```

the file DSERV MAP contains a list of all phases that begin with the characters IDC. The first phasename specification is ignored.

2. If you want to obtain information from the directories of private source statement library directories, relocatable library directories, or core image library directories, the libraries must be assigned and identified (via ASSGN and DLBL commands) when the DSERV command is issued. Otherwise, the system library directories are used. System
directories are made available when you specify a mode letter on the
SET DOS ON command line.

3. The current assignments for logical units are ignored by the DSERV
command; output is directed only to the output device indicated by the
option list.

**Example:**

To display at your terminal an alphamerical list of procedures cataloged on
the system procedure library, you would issue:

dserv pd (sort term

**Responses:**

When you use the TERM option of the DSERV command, the contents of
the specified directory are displayed at your terminal.

**Messages and Return Codes:**

- DMSDSV003E Invalid option: option [RC = 24]
- DMSDSV021W No transient directory [RC = 4]
- DMSDSV022W No core image directory [RC = 4]
- DMSDSV023W No relocatable directory [RC = 4]
- DMSDSV024W No procedure directory [RC = 4]
- DMSDSV025W No source statement directory [RC = 4]
- DMSDSV026W phase not in library [RC = 4]
- DMSDSV027E Invalid device devtype [for SYSaaa] [RC = 24]
- DMSDSV027W No private core image library [RC = 4]
- DMSDSV028W No {private|system} transient directory entries [RC = 4]
- DMSDSV047E Invalid parameter parameter [RC = 24]
- DMSDSV095E Invalid address vstor [RC = 100]
- DMSDSV099E CMS/DOS environment not active [RC = 40]
- DMSDSV106S Error nn writing file fn ft fm on disk [RC = 100]
- DMSDSV248S Error nnn on printer [RC = 100]
- DMSDSV411S Input error code nn on SYSaaa [RC = rc]
Use the EDIT command to invoke the VM/SP System Product editor in CMS editor (EDIT) migration mode. Use the editor to create, modify, and manipulate CMS disk files. In EDIT migration mode, you may execute both EDIT and XEDIT subcommands. For complete details on EDIT migration mode, refer to the publication VM/SP System Product Editor Command and Macro Reference.

To invoke only the CMS editor, refer to the “Usage Note” below.

Once the CMS editor has been invoked, you may only execute EDIT subcommands and EDIT macro requests, and enter data lines into the disk file. A limited number of CMS commands may be executed in the CMS subset mode. Enter CMS subset mode from the edit environment by issuing the EDIT subcommand, CMS.

You can return control to the CMS environment by issuing the EDIT subcommands FILE or QUIT.

For complete details on the EDIT subcommand formats and usage, see “Appendix C. EDIT Subcommands and Macros.”

The format of the EDIT command is:

```
EDIT fn ft [fm] [ (options...[)])]
```

Options: [ LRECL nn ] [ NODISP ]

**where:**

- **fn ft**
  - is the filename and filetype of the file to be created or edited. If a file with the specified filename and filetype does not exist, the CMS editor assumes that you want to create a new file, and after you issue the INPUT subcommand, all data lines you enter become input to the file. If a file with the specified filename and filetype exists, you may issue EDIT subcommands to modify the specified file.

- **fm**
  - is the filemode of the file to be edited, indicating the disk on which the file resides. The editor determines the filemode of the edited file as follows:
Editing existing files: If the file does not reside on your A-disk or its extensions, you must specify fm.

When you specify fm, the specified disk and its extensions are searched. If a file is found on a read-only extension, the filemode of the parent disk is saved; when you issue a FILE or SAVE subcommand, the modified file is written to the parent disk.

If you specify fm as an asterisk (*) all accessed disks are searched for the specified file.

Creating new files: If you do not specify fm, the new file is written on your A-disk when you issue the FILE or SAVE subcommands.

Options:

LRECL nn
is the record length of the file to be created or edited. Use this option to override the default values supplied by the editor, which are determined as follows:

Editing Existing Files: Existing record length is kept regardless of format. If the file has variable-length records and the existing record length is less than the default record length, the default record length is used.

Creating New Files: All new files have a record length of 80, with the following exceptions:

<table>
<thead>
<tr>
<th>Filetype</th>
<th>LRECL</th>
</tr>
</thead>
<tbody>
<tr>
<td>LISTING</td>
<td>121</td>
</tr>
<tr>
<td>SCRIPT,VSBDATA</td>
<td>132</td>
</tr>
<tr>
<td>FREEFORT</td>
<td>81</td>
</tr>
</tbody>
</table>

The maximum record length supported by the editor is 160 characters.

NODISP
forces a 3270 display terminal into line (typewriter) mode. When the NODISP option is in effect, all subcommands that control the display as a 3270 terminal such as SCROLL, SCROLLUP, and FORMAT (and CHANGE with no operands) are made invalid for the edit session.

Note: It is recommended that the NODISP option always be used when editing on a 3066.
**Usage Note:**

When you issue the EDIT command, an EXEC named EDIT EXEC S2 is executed. This EXEC invokes the VM/SP System Product editor in EDIT migration mode.

If you want to invoke only the CMS editor on a permanent basis, your system programmer must rename this EXEC. Then, when you issue the EDIT command, the EXEC will not execute and the CMS editor will be invoked.

If you want to invoke the CMS editor only for a particular edit session, specify OLD on the EDIT command line. CMS passes the OLD parameter to EDIT EXEC S2 and only the CMS editor is invoked. Note that the old editor has not been enhanced for VM/SP and will not be enhanced for future releases; specifically the old editor will not include any support for new display devices.

**Example:**

If you want to create a new file on your A-disk called OVERTIME DATA, you would enter:

```
edit overtime data a
```

**Responses:**

**NEW FILE:**

The specified file does not exist.

**EDIT:**

The edit environment is entered. You may issue any valid EDIT subcommand or macro request.

**INPUT:**

The input environment is entered by issuing the EDIT subcommands REPLACE or INPUT with no operands. All subsequent input lines are accepted as input to the file.
## Messages and Return Codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Message Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSEDI003E</td>
<td>INVALID OPTION option [RC = 24]</td>
</tr>
<tr>
<td>DMSEDI024E</td>
<td>FILE fn ft fm ALREADY EXISTS [RC = 28]</td>
</tr>
<tr>
<td>DMSEDI029E</td>
<td>INVALID PARAMETER parameter IN THE ‘LRECL’ OPTION FIELD [RC = 24]</td>
</tr>
<tr>
<td>DMSEDI044E</td>
<td>RECORD LENGTH EXCEEDS ALLOWABLE MAXIMUM [RC = 32]</td>
</tr>
<tr>
<td>DMSEDI048E</td>
<td>INVALID MODE mode [RC = 24]</td>
</tr>
<tr>
<td>DMSEDI054E</td>
<td>INCOMPLETE FILEID SPECIFIED [RC = 24]</td>
</tr>
<tr>
<td>DMSEDI069E</td>
<td>DISK mode NOT ACCESSED [RC = 36]</td>
</tr>
<tr>
<td>DMSEDI076E</td>
<td>ACTUAL RECORD LENGTH EXCEEDS THAT SPECIFIED [RC = 40]</td>
</tr>
<tr>
<td>DMSEDI104S</td>
<td>ERROR nn READING FILE fn ft fm FROM DISK [RC = 100]</td>
</tr>
<tr>
<td>DMSEDI105S</td>
<td>ERROR nn WRITING FILE fn ft fm ON DISK [RC = 100]</td>
</tr>
<tr>
<td>DMSEDI117S</td>
<td>ERROR WRITING TO DISPLAY TERMINAL [RC = 100]</td>
</tr>
<tr>
<td>DMSEDI132S</td>
<td>FILE fn ft fm TOO LARGE [RC = 88]</td>
</tr>
<tr>
<td>DMSEDI143S</td>
<td>UNABLE TO LOAD MODULE [RC = 40]</td>
</tr>
<tr>
<td>DMSEDI144S</td>
<td>REQUESTED FILE IS IN ACTIVE STATUS</td>
</tr>
<tr>
<td>DMSEDI151E</td>
<td>3278 MOD5 DISPLAY TERMINAL NOT SUPPORTED BY OLD CMS EDITOR</td>
</tr>
<tr>
<td>DMSEDX069E</td>
<td>DISK mode NOT ACCESSED [RC = 36]</td>
</tr>
<tr>
<td>DMSEDX109S</td>
<td>VIRTUAL STORAGE CAPACITY EXCEEDED [RC = 104]</td>
</tr>
</tbody>
</table>
Use the ERASE command to delete one or more CMS files from a read/write disk.

The format of the ERASE command is:

\[
\text{ERASE} \ [fn] \ [ft] \ [fm] \ [\{\text{options...} \ [)\}]
\]

where:

\(fn\) is the filename of the file(s) to be erased. An asterisk coded in this position indicates that all filenames are to be used. \(fn\) must be specified, either with a name or an asterisk.

\(ft\) is the filetype of the file(s) to be erased. An asterisk coded in this position indicates that all filetypes are to be used. This field must be specified, either with a name or an asterisk.

\(fm\) is the filemode of the files to be erased. If this field is omitted, only the A-disk is searched. An asterisk coded in this position indicates that files with the specified filename and/or filetype are to be erased from all read/write disks.

Options:

\textbf{Type} displays at the terminal the file identifier of each file erased.

\textbf{Notype} file identifiers are not displayed at the terminal.
Usage Notes:

1. If you specify an asterisk for both filename and filetype you must specify both a filemode letter and number; for example:

   ```
   erase * * a5
   ```

2. To erase all files on a particular disk, you can use the FORMAT command to reformat it, or you can access the disk using the ACCESS command with the ERASE option.

3. If an asterisk is entered as the filemode, then either the filename or the filetype or both must be specified by name.

Example:

To erase all the files on your A-disk with a filetype of DATA, you would enter:

```
erase * data a (type
```

The file identifiers of the files erased are displayed at the terminal.

Responses:

If you specify the TYPE option, the file identification of each file erased is displayed. For example:

```
erase oldfile temp (type
```

results in the display:

```
OLDFILE TEMP A1
Ready;
```

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSERS002E  File [/n [/f [/fm/]] not found [RC = 28]
- DMSERS003E  Invalid option: option [RC = 24]
- DMSERS037E  Disk mode[(vdev)] is accessed as read/only [RC = 36]
- DMSERS046E  Invalid mode mode [RC = 24]
- DMSERS054E  Incomplete fileid specified [RC = 24]
- DMSERS069E  Disk mode not accessed [RC = 36]
- DMSERS076E  Invalid parameter parameter [RC = 24]
- DMSERS071E  Erase * * [fm*] not allowed [RC = 24]
- DMSERS109T  Virtual storage capacity exceeded
Note: You can invoke the ERASE command from the terminal, from an EXEC file, or as a function from a program. No error messages are issued if ERASE is invoked:

- As a function from a program
- From a CMS EXEC file that has the &CONTROL NOMSG option in effect
- From an EXEC2 EXEC file and CMDCALL is not in effect
- From a System Product Interpreter EXEC with ADDRESS COMMAND in effect.
ESERV

Use the ESERV EXEC procedure in CMS/DOS to copy edited VSE macros from system or private source statement E sublibraries to CMS disk files, or to list de-edited macros.

The format of the ESERV command is:

```
   ESERV  fn
```

where:

fn

specifies the filename of the CMS file that contains the ESERV control statements; it must have a filetype of ESERV. The logical unit SYSIPT must be assigned to the disk on which the ESERV file resides. fn is also the filename of the Listing and Macro files produced by the ESERV program.

Usage Notes:

1. The input file can contain any or all of the ESERV control statements as defined in Guide to the DOS/VSE Assembler.

2. You must have a read/write A-disk accessed when you use the ESERV command.

3. To copy macros from the system source statement library, you must have entered the CMS/DOS environment specifying the mode letter of the VSE system residence. To copy from a private source statement library, you must assign the logical unit SYSSLB and issue a DLBL command for the ddname IJSYSSL.

4. The output of the ESERV program is directed (as in VSE/AF) to devices assigned to the logical units SYSLST and/or SYSPCH. If either SYSLST or SYSPCH is not assigned, the following files are created:

```
   Unit  Output File
       SYSLST  fn LISTING mode
       SYSPCH  fn MACRO mode
```

where mode is the mode letter of the disk on which the source file, fn ESERV resides. If fn ESERV is on a read-only disk, the files are written to your A-disk.
You can override default assignments made by the ESERV EXEC as follows:

- If you assign SYSIPT to TAPE or READER, the source statements are read from that device.

- If you assign SYSLST or SYSPCH to another device, the SYSLST or SYSPCH files are written to that device.

5. The ESERV EXEC procedure clears all DLBL definitions, except those entered with the PERM option.

6. If you want to use the ESERV command in an EXEC procedure, you must use the EXEC command (because ESERV is also an EXEC).

7. When you use the ESERV control statements PUNCH or DSPCH, the ESERV program may generate CATAL.S, END, or */ records in the output file. When you add a MACRO file containing these statements to a CMS macro library using the MACLIB command, the statements are ignored and are not read into the MACLIB member.

8. Any DISKS accessed with a mode letter of “R” or “T” should be in read/only mode when an ESERV is running on them, otherwise message DMSDLB301E may occur.

9. If you want to issue ESERV from an EXEC program, you should precede it with the EXEC command; that is, specify

   `exec eserv`

Responses:

None. The CMS ready message indicates that the ESERV program completed execution successfully. You may examine the SYSLST output to verify the results of the ESERV program execution.

Messages and Return Codes:

- **DMSERV001E** NO FILENAME SPECIFIED. [RC = 24]
- **DMSERV002E** FILE fn ESERV NOT FOUND. [RC = 28]
- **DMSERV006E** NO READ / WRITE DISK ACCESSED. [RC = 36]
- **DMSERV070E** INVALID ARGUMENT `argument` [RC = 24]
- **DMSERV099E** CMS/DOS ENVIRONMENT NOT ACTIVE. [RC = 40]
- **DMSERV027E** INVALID DEVICE `devtype` FOR `SYSaaa` [RC = 24]
- **DMSERV037E** DISK `mode` IS READ ONLY. [RC = 36]

*Note:* The ESERV EXEC calls other CMS commands to perform certain functions, and so you may, on occasion, receive error messages that occur as a result of those commands.
Non-CMS error messages produced by the VSE ESERV program are described in the *Guide to the DOS/VSE Assembler.*
Use the EXEC command to execute one or more CMS commands or EXEC control statements contained in a specified System Product interpreter, CMS EXEC or EXEC 2 file.

The format of the EXEC command is:

\[
\text{[EXec]} \quad fn \ [\text{args...}]
\]

**where:**

- **[EXec]** indicates that the EXEC command may be omitted if you are executing the EXEC procedure from the CMS command environment and have not issued the command `SET IMPEX OFF.

- **fn** is the filename of a file containing one or more CMS commands and/or EXEC control statements to be executed. The filetype of the file must be EXEC. The file can have either fixed- or variable-length records with a logical record length not exceeding 130 characters. A text editor or a user program can create EXEC files. EXEC files a CMS editor creates have, by default, variable-length, 80-character records.

- **args** are any arguments you wish to pass to the EXEC. The CMS EXEC processor assigns arguments to special variables &1 through &30 in the order in which they appear in the argument list. The EXEC 2 processor assigns arguments to special variables starting with special variable &1. With the System Product interpreter and the EXEC 2 processors, the number of arguments is not limited. However, the number of bytes of data you can pass in the argument list is limited. The limit is the maximum number of bytes that can fit in a line: 130 bytes if the command is entered from a terminal, 255 bytes if the command is issued from a program. Arguments passed to the System Product interpreter are handled differently than they are in EXEC or EXEC 2. See the *VM/SP System Product Interpreter Reference* for details.

For information about the System Product interpreter, see the:

*VM/SP System Product Interpreter User's Guide* and the

*VM/SP System Product Interpreter Reference.*

See the *VM/SP EXEC 2 Reference* for information about EXEC 2.
“Appendix E. CMS EXEC Control Statements” contains complete descriptions of EXEC control statements, special variables, and built-in functions.

Example:

If the implied EXEC function is set to OFF (QUERY IMPEX to find out the setting), and if you want to execute your TPHONE EXEC with a nickname, you would enter:

```
exec tphone rick
```

Responses:

The amount of information displayed during the execution of a CMS EXEC depends on the setting of the &CONTROL control statement. By default, &CONTROL displays all CMS commands, responses, and error messages. In addition, it displays nonzero return codes from CMS in the format:

```
+++ R(nnnnn) +++
```

where nnnnn is the return code from the CMS command.

For details, see the description of the &CONTROL control statement in “Appendix F. EXEC Control Statements.”

The amount of information displayed during the execution of a System Product interpreter file depends on whether tracing is set on. See the VM/SP System Product Interpreter Reference for details.

The amount of information displayed during the execution of an EXEC 2 file depends on the setting of the &TRACE control statement. See VM/SP EXEC 2 Reference for details.

Return codes for error messages from CP commands directly correspond to the message number. For example, if you issued a CP LINK command with an incorrect userid, you receive error message DMKLNK053E userid not in CP directory. When issued from a CMS EXEC program, the same CP LINK command would have a return code of 53.

Messages and Return Codes:

```
DMSEXCCO01E  No filename specified
```

If the EXEC interpreter finds an error, it displays the message:

```
DMSEXCT072E  Error in EXEC file fn, line nnn: message
```
The possible errors, and the associated return codes, are:

<table>
<thead>
<tr>
<th>Description</th>
<th>Return Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE NOT FOUND</td>
<td>801</td>
</tr>
<tr>
<td>&amp;SKIP OR &amp;GOTO ERROR</td>
<td>802</td>
</tr>
<tr>
<td>BAD FILE FORMAT</td>
<td>803</td>
</tr>
<tr>
<td>TOO MANY ARGUMENTS</td>
<td>804</td>
</tr>
<tr>
<td>MAX DEPTH OF LOOP NESTING EXCEEDED</td>
<td>805</td>
</tr>
<tr>
<td>ERROR READING FILE</td>
<td>806</td>
</tr>
<tr>
<td>INVALID SYNTAX</td>
<td>807</td>
</tr>
<tr>
<td>INVALID FORM OF CONDITION</td>
<td>808</td>
</tr>
<tr>
<td>INVALID ASSIGNMENT</td>
<td>809</td>
</tr>
<tr>
<td>MISUSE OF SPECIAL VARIABLE</td>
<td>810</td>
</tr>
<tr>
<td>ERROR IN &amp;ERROR ACTION</td>
<td>811</td>
</tr>
<tr>
<td>CONVERSION ERROR</td>
<td>812</td>
</tr>
<tr>
<td>TOO MANY TOKENS IN STATEMENT</td>
<td>813</td>
</tr>
<tr>
<td>MISUSE OF BUILT-IN FUNCTION</td>
<td>814</td>
</tr>
<tr>
<td>EOF FOUND IN LOOP</td>
<td>815</td>
</tr>
<tr>
<td>INVALID CONTROL WORD</td>
<td>816</td>
</tr>
<tr>
<td>EXEC ARITHMETIC UNDERFLOW</td>
<td>817</td>
</tr>
<tr>
<td>EXEC ARITHMETIC OVERFLOW</td>
<td>818</td>
</tr>
<tr>
<td>SPECIAL CHARACTER IN VARIABLE SYMBOL</td>
<td>819</td>
</tr>
</tbody>
</table>

If the EXEC 2 interpreter finds an error, it displays the message:

DMSEXEO85E   Error in fn ft fm, line nnn - message

The possible errors and the associated return codes are:

<table>
<thead>
<tr>
<th>Description</th>
<th>Return Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE NOT FOUND</td>
<td>10001</td>
</tr>
<tr>
<td>WRONG FILE FORMAT</td>
<td>10002</td>
</tr>
<tr>
<td>WORD TOO LONG</td>
<td>10003</td>
</tr>
<tr>
<td>STATEMENT TOO LONG</td>
<td>10004</td>
</tr>
<tr>
<td>INVALID CONTROL WORD</td>
<td>10005</td>
</tr>
<tr>
<td>LABEL NOT FOUND</td>
<td>10006</td>
</tr>
<tr>
<td>INVALID VARIABLE NAME</td>
<td>10007</td>
</tr>
<tr>
<td>INVALID FORM OF CONDITION</td>
<td>10008</td>
</tr>
<tr>
<td>INVALID ASSIGNMENT</td>
<td>10009</td>
</tr>
<tr>
<td>MISSING ARGUMENT</td>
<td>10010</td>
</tr>
<tr>
<td>INVALID ARGUMENT</td>
<td>10011</td>
</tr>
<tr>
<td>CONVERSION ERROR</td>
<td>10012</td>
</tr>
<tr>
<td>NUMERIC OVERFLOW</td>
<td>10013</td>
</tr>
<tr>
<td>INVALID FUNCTION NAME</td>
<td>10014</td>
</tr>
<tr>
<td>END OF FILE FOUND IN LOOP</td>
<td>10015</td>
</tr>
<tr>
<td>DIVISION BY ZERO</td>
<td>10016</td>
</tr>
<tr>
<td>INVALID LOOP CONDITION</td>
<td>10017</td>
</tr>
<tr>
<td>ERROR RETURN DURING &amp;ERROR ACTION</td>
<td>10019</td>
</tr>
<tr>
<td>ASSIGNMENT TO UNSET ARGUMENT</td>
<td>10020</td>
</tr>
<tr>
<td>STATEMENT OUT OF CONTEXT</td>
<td>10021</td>
</tr>
<tr>
<td>INSUFFICIENT STORAGE AVAILABLE</td>
<td>10097</td>
</tr>
<tr>
<td>FILE READ ERROR nnn</td>
<td>10098</td>
</tr>
</tbody>
</table>
TRACE ERROR nrn  10099

DMSEXE175E Invalid EXEC command  RC=10096
DMSEXE255T Insufficient storage for Exec interpreter
   RC=10000

For the System Product interpreter, the possible errors and associated return codes are:

<table>
<thead>
<tr>
<th>Description</th>
<th>Return Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program is unreadable</td>
<td>20003</td>
</tr>
<tr>
<td>Program interrupted</td>
<td>20004</td>
</tr>
<tr>
<td>Machine storage exhausted</td>
<td>20005</td>
</tr>
<tr>
<td>Unmatched '/*' or quote</td>
<td>20006</td>
</tr>
<tr>
<td>WHEN or OTHERWISE expected</td>
<td>20007</td>
</tr>
<tr>
<td>Unexpected THEN or ELSE</td>
<td>20008</td>
</tr>
<tr>
<td>Unexpected WHEN or OTHERWISE</td>
<td>20009</td>
</tr>
<tr>
<td>Unexpected or unmatched END</td>
<td>20010</td>
</tr>
<tr>
<td>Control Stack Full</td>
<td>20011</td>
</tr>
<tr>
<td>Clause &gt; 500 characters</td>
<td>20012</td>
</tr>
<tr>
<td>Invalid character in data</td>
<td>20013</td>
</tr>
<tr>
<td>Incomplete DO/SELECT/IF</td>
<td>20014</td>
</tr>
<tr>
<td>Invalid Hex constant</td>
<td>20015</td>
</tr>
<tr>
<td>Label not found</td>
<td>20016</td>
</tr>
<tr>
<td>Unexpected PROCEDURE</td>
<td>20017</td>
</tr>
<tr>
<td>THEN expected</td>
<td>20018</td>
</tr>
<tr>
<td>String or symbol expected</td>
<td>20019</td>
</tr>
<tr>
<td>Symbol expected</td>
<td>20020</td>
</tr>
<tr>
<td>Invalid data on end of clause</td>
<td>20021</td>
</tr>
<tr>
<td>Invalid character string</td>
<td>20022</td>
</tr>
<tr>
<td>Invalid TRACE request</td>
<td>20024</td>
</tr>
<tr>
<td>Invalid sub-keyword found</td>
<td>20025</td>
</tr>
<tr>
<td>Invalid whole number</td>
<td>20026</td>
</tr>
<tr>
<td>Invalid DO syntax</td>
<td>20027</td>
</tr>
<tr>
<td>Invalid LEAVE or ITERATE</td>
<td>20028</td>
</tr>
<tr>
<td>Environment name too long</td>
<td>20029</td>
</tr>
<tr>
<td>Name or String &gt; 250 chars</td>
<td>20030</td>
</tr>
<tr>
<td>Name starts with numeric or &quot;.&quot;</td>
<td>20031</td>
</tr>
<tr>
<td>Invalid use of stem</td>
<td>20032</td>
</tr>
<tr>
<td>Invalid expression result</td>
<td>20033</td>
</tr>
<tr>
<td>Logical value not 0 or 1</td>
<td>20034</td>
</tr>
<tr>
<td>Invalid expression</td>
<td>20035</td>
</tr>
<tr>
<td>Unmatched &quot;(&quot; in expression</td>
<td>20036</td>
</tr>
<tr>
<td>Unexpected &quot;,&quot; or &quot;)&quot;</td>
<td>20037</td>
</tr>
<tr>
<td>Invalid template or pattern</td>
<td>20038</td>
</tr>
<tr>
<td>Evaluation stack overflow</td>
<td>20039</td>
</tr>
<tr>
<td>Incorrect call to routine</td>
<td>20040</td>
</tr>
<tr>
<td>Bad arithmetic conversion</td>
<td>20041</td>
</tr>
<tr>
<td>Arithmetic overflow/underflow</td>
<td>20042</td>
</tr>
<tr>
<td>Routine not found</td>
<td>20043</td>
</tr>
<tr>
<td>Function did not return data</td>
<td>20044</td>
</tr>
<tr>
<td>No data on function RETURN</td>
<td>20045</td>
</tr>
<tr>
<td>Failure in system service</td>
<td>20048</td>
</tr>
</tbody>
</table>
Interpreter failure 20049

DMSREX255T Insufficient storage for Exec interpreter
RC=20096 [RC=10096]
Use the EXECDROP command to remove the specified EXEC(s) or System Product Editor macro(s) from storage or to discontinue use of the specified EXEC(s) or Editor Macro(s) in an Installation Discontiguous Shared Segment.

The format of the EXECDROP command is:

```
EXECDrop

execlname

exectype

[options...[]]
```

**where:**

**execname**

is the name of the storage-resident EXEC(s) to be purged. If an asterisk (*) is coded in this field, all execnames are used.

**exectype**

is the type of the storage-resident EXEC(s) to be purged. If an asterisk (*) is coded in this field, all exectypes are used. The default is an asterisk (*).

* specified alone indicates that all storage-resident EXECs are purged.

**Options:**

**User**

indicates that the storage for the EXEC(s) was allocated from user free storage. Only the EXECs that satisfy the execname and/or exectype qualifications and that also have the USER attribute are dropped. If neither USER, SYSTEM, nor SHARED is specified, then all EXECs with the USER and SYSTEM attributes that satisfy the execname and/or exectype qualifications are dropped.
**EXECDROP**

**SYstem**
indicates that the storage for the EXEC(s) was allocated from nucleus free storage. Only the EXECs that satisfy the execname and/or exectype qualifications and that also have the SYSTEM attribute are dropped. If neither USER, SYSTEM, nor SHARED is specified, then all EXECs with the USER and SYSTEM attributes that satisfy the execname and/or exectype qualifications are dropped.

**SHared**
indicates that the EXEC(s) was located in an Installation Discontiguous Shared Segment (DCSS). Only the EXECs that satisfy the execname and/or exectype qualifications and that also have the SHARED attribute are dropped for the duration of your CMS session. If neither USER, SYSTEM, or SHARED is specified, then all EXECs with the USER and SYSTEM attribute that satisfy the execname and/or exectype qualifications are dropped.

SHARED EXECs can only be dropped when SET INSTSEG is ON. Once a SHARED EXEC is dropped, you can only reload it when you IPL CMS. To discontinue use of all SHARED EXECs temporarily, use the SET INSTSEG OFF command.

**Examples:**

To drop all storage-resident EXECs, specify:

```
execdrop *
```

To purge all storage-resident EXECs with exectypes of XEDIT that were EXECLOADed into user free storage, specify the following:

```
execdrop * xedit (user
```

**Responses:**

None.
Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSEXD003E Invalid option: option [RC = 24]
- DMSEXD042E No execid specified [RC = 24]
- DMSEXD070E Invalid parameter parameter [RC = 24]
- DMSEXD109S Virtual storage capacity exceeded [RC = 104]
- DMSEXD415E Invalid character char in execid execname exectype [RC = 20]
- DMSEXD416W There are no execname exectype {system|or|user|or|shared} EXECs storage resident [RC = 28]
- DMSEXD418W Drop pending for execname exectype [RC = 4]
Use the EXECIO command to:

- Read lines from disk or virtual reader to the program stack or a variable.
- Write lines from the program stack or a variable to a CMS disk file or to a virtual spool device (punch or printer).
- Cause execution of CP commands and recover resulting output.

In some cases output data to be written may be supplied directly on the EXECIO command line.

The information immediately following is reference level information about EXECIO format and operands. Following this reference information you can find extended descriptive and use information. If you are not familiar with EXECIO, you should review the complete command description before attempting to use it. Also, to get full benefit from EXECIO you should be familiar with use of EXECs under the System Product Interpreter or EXEC 2 (Refer to the VM/SP System Product Interpreter Reference or to the VM/SP EXEC 2 Reference).

In the following descriptions, “relative line number” means the number of lines processed to satisfy an EXECIO operation; “absolute line number” means the number of the line relative to the top of the file.
The format of the EXECIO command is:

<table>
<thead>
<tr>
<th>EXECIO</th>
<th>DISKR fn ft [linenum]</th>
<th>([FINIs] options [a] [b]) [ ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARD</td>
<td>(</td>
<td>options [a] [b])</td>
</tr>
<tr>
<td>CP</td>
<td>(</td>
<td>options [a] [b] [d] [e])</td>
</tr>
<tr>
<td>DISKW fn ft fm [linenum]</td>
<td>([FINIs] options [b] [c] [d])</td>
<td></td>
</tr>
<tr>
<td></td>
<td>recfm [lrecl]</td>
<td></td>
</tr>
<tr>
<td>PUNCH</td>
<td>(</td>
<td>options [b] [c] [d])</td>
</tr>
<tr>
<td>PRINT</td>
<td>[ ] [ ] [ ] [ ] [ ]</td>
<td></td>
</tr>
<tr>
<td>EMSG</td>
<td>(</td>
<td>options [b] [c] [d])</td>
</tr>
</tbody>
</table>

Option formats:

(a)  
[FInd /chars /] [Zone n1 n2] [LIFO] [SKip]
[LOCate /chars /]
[Avoid /chars /]

(b)  
[Margins n1 n2] [STRIP] [NOTYPE] [STEM x xxxn]

(c)  
[CAse U]
[M]

(d)  
[STring x x x ...]

(e)  
[BUffer length]

**Note:** Parsing of the EXECIO command differs from that of other CMS commands in that it involves handling of strings that may contain embedded blanks, parentheses, other special characters, and words of more that eight characters. Therefore, if a right parenthesis is used to mark the end of an EXECIO option, it must be preceded by at least one blank character. A right parenthesis cannot be used to mark the end of the STRING option.
where:

lines

is the number of source lines processed. This can be any non-negative integer. With the VAR option, the number of lines must be 1. An asterisk (*) indicates that the operation is to terminate when:

1. a null (0-length) line is read during an output operation;

2. an end-of-file condition is detected during an input operation.

Specification of *, together with the STRING option, is valid only with the CP operand. Using the * and STRING combination with any other operand causes an error message to be issued. Also the combination of the * and the VAR options is not allowed. If “lines” is specified as zero (0), no I/O operation is performed other than FINIS, when it is specified as an option.

DISKR

is used to read a specified number of lines from the CMS file “fn ft [fm]” to the program stack FIFO (first-in first-out) or to an EXEC 2 or System Product interpreter variable if the STEM or VAR options are specified.

CARD

is used to read a specified number of lines from the virtual reader to the program stack (FIFO) or to an EXEC 2 or System Product interpreter variable if the STEM or VAR options are specified.

CP

causes output resulting from a CP command to be placed on the program stack (FIFO) or to an EXEC 2 or System Product interpreter variable if the STEM or VAR options are specified. You may specify which CP command is to be issued via:

1. the STRING option on the EXECIO command line;

2. the next line from the program stack.

Keep in mind that all characters of CP commands must be in upper case.

DISKW

is used to write a specified number of lines from the program stack or from an EXEC 2 or System Product interpreter variable if the STEM or VAR options are specified to a new or existing CMS file “fn ft fm.”

Inserting a line into a variable length CMS file can cause truncation of the portion of the file following the inserted line. See the extended DISKW operand description.
**EXECIO**

*recfm*
*lrecl*

Define the record format and record length for any new file created as a result of a DISKW operation. The default value for recfm is V (variable), in which case "lrecl" has no meaning. If you specify F (fixed) for recfm, the default lrecl value is 80. The maximum lrecl value that may be specified is 255 unless the VAR or STEM option is used to bypass the use of the program stack, in which case the maximum is that which is defined for the type of file in use (800 byte or EDF file). If recfm or lrecl is specified for the DISKW operation, then a linenum value must be specified explicitly.

**PUNCH**

Is used to transfer a specified number of lines from the program stack or from an EXEC 2 or System Product interpreter variable if the STEM or VAR options are specified to the virtual punch.

**PRINT**

Is used to transfer a specified number of lines from the program stack or from an EXEC 2 or System Product interpreter variable if the STEM or VAR options are specified, to the virtual printer using the PRINTL macro.

**CC**

Is used with the PRINT operand to specify carriage control for each line transferred to the virtual printer. Using the CC operand, you can supply carriage control code explicitly, or, by specifying DATA, indicate that the carriage control character is the first byte of each line.

**code**

Is the character (ASA or machine code) that defines carriage control. A blank code (the default value) cannot be specified on the command line.

**DATA**

Specifies that the first byte of each line sent to the virtual printer is a carriage control character.

**EMSG**

Causes a message to be displayed. The text of the message may be:

1. the character string specified on the STRING option;
2. the next available line(s) from the program stack;
3. the information from a STEM or VAR variable.

Messages are edited according to the current CP SET EMSG settings.

**fn**

Is the filename of the file.
**EXECIO**

\[
\text{ft}
\]

is the filetype of the file.

\[
\text{fm}
\]

is the filemode of the file. When filemode is specified, that disk and its extensions are searched. If filemode is not specified, or is specified as an asterisk (*), all accessed disks are searched for the specified file.

\[
\text{linenum}
\]

is the absolute line number within the specified file where a DISKR or DISKW operation is to begin. If a value is zero or not specified for newly opened files, reading begins at the first line and writing begins at the last line. For other files, reading or writing begins at the line following the one at which the previous operation ended. Because EXEC processors manipulate EXECs that are currently executing, a read or write to a currently running EXEC should explicitly specify the linenum operand. Failure to do so may cause the first line to be read or written each time. If recfm or lrecl is specified for the DISKW operation, then a linenum value must be specified explicitly.

**FINIs**

causes the specified file to be closed following completion of a DISKR or DISKW operation.

**Option A**

**FInd**

is used to write to the program stack LIFO (last-in first-out) or FIFO (first-in first-out) to an EXEC 2 or System Product interpreter array.

1. the contents of that line;
2. the line number of the first occurrence of a line (or zone portion of that line) that begins with the characters specified between delimiters. For DISKR operations both the relative and absolute line numbers are written. Otherwise, only the relative line number is written.

If you wish to search only a portion of each line, use the ZONE option, explained below. If you wish to write only a portion of any line matching the search argument to the program stack or a variable, use the MARGINS option, also explained below.

**LOCate**

is like the FIND option explained above, except that the object characters may occur any place within a line (or zone portion of that line).

**Avoid**

is like the LOCATE option explained above, except that the search is for a line (or zone portion of that line) that does not contain the specified characters.
Zone
is used to restrict the portion of the input lines searched as a result of the FIND, LOCATE, or AVOID options. The search is between columns n1 and n2 (inclusive), if specified. The default values are column 1 through the end of the line (*). The limits of values that may be specified for n1 or n2 are 1 through 2^31-1.

LIFO
FIFO
defines the order in which lines are written to the program stack. Generally, the default order is FIFO (first-in first-out). The exceptions are operations that put line numbers on the program stack as a result of a search operation (FIND, LOCATE, or AVOID). These operations default to LIFO (last-in first-out). The use of VAR or STEM is not valid with this option.

SKip
allows a read function (DISKR, CARD, CP) to occur without writing any information to the program stack.

Option B

Margins
specifies that only a portion (columns n1 through n2 inclusive) of affected lines is to be processed (from the stack or a variable). The default values are column 1 through the end of each line (*). The limits of values that may be specified for n1 or n2 are 1 through 2^31-1.

STRIP
specifies that trailing blank characters are to be removed from any output lines or lines returned.

NOTYPE
suppresses the display of message DMSEIO632E at the virtual console.

STEM xxxx
indicates that the variables xxxx are to be used to supply input data for output-type operations (PUNCH, PRINT, EMSG, and DISKW) or that variables xxxx are the destination for output for the input-type operations (CARD, DISKR, and CP). The variables used are a concatenation of the specified stem and a number that corresponds to the number of lines of output. The special variable xxxx0 is set to the number of lines of information returned for the input-type operations. STEM can be used with the STRING operation only with the CP operation. The maximum length variable name with the STEM option is 240 bytes.

VAR xxxx
indicates that the variable xxxx is to be used to supply input data for output-type operations (PUNCH, PRINT, EMSG, and DISKW) or that variable xxxx is the destination for output for the input-type operations (CARD, DISKR, and CP). If VAR is specified, then the
number of lines must be 1. VAR cannot be used with the FIFO or LIFO options, nor with the FIND, AVOID, and LOCATE options because they cause more than one line of output to be written. VAR can be used with the STRING option only with the CP operation. The maximum length variable name is 250 characters.

Option C

CAse
causes data read from the program stack, from a STEM, or from a variable to be:

1. translated to uppercase if U is specified;
2. not translated (mixed) if M is specified.

M (mixed) is the default value.

Option D

STring
is used to supply output data explicitly on the EXECIO command line. Any characters following the STRING keyword are treated as string data, not additional EXECIO operands. Therefore, STRING, if specified, must be the final option on the command line.

Option E

BUFFer length
specifies the length, in characters (bytes), of the CP command response expected from a CP operation. The limits of values that may be specified for length are 1 through 2^31-1. If this option is not specified, up to 8192 characters of the response are returned.

Extended Descriptions and Use Information

General comments:

EXECIO commands are normally issued as statements from System Product Interpreter or EXEC 2 EXECs. Because some EXECIO option values can exceed eight characters, an extended parameter list is required for EXECIO execution when these options are specified. Otherwise, an error message results. Both the System Product Interpreter and EXEC 2 supply a tokenized parameter list and, if necessary, an extended parameter list.

You should keep in mind that when a CMS operation completes and the READY message (Ready;) displays, CMS closes all files. Any subsequent EXECIO read operation will begin at file line one unless a "linenum" value is specified. Any subsequent EXECIO write operation will begin at the end of the file unless a "linenum" value is specified. Therefore, when possible,
it is a good idea to specify a “linenum” value on the EXECIO command line.

For write operations, data to be written is normally taken from the program stack. However, data to be written may be supplied via the STRING option or via the VAR or STEM options (in all cases the EXEC in question must be a System Product interpreter EXEC or an EXEC 2 EXEC).

Program stack:

The program stack is a buffer area, expanded as necessary from available free storage. Data flow into and out of the program stack is:

1. normally FIFO (first-in first-out) for read or write operations;

2. LIFO (last-in first-out) for read options, such as FIND or LOCATE, that result in a line number being stacked.

A successful search (LOCATE, FIND, etc.) operation results in two lines being written (LIFO) to the program stack:

1. the contents of the line that satisfied the search argument;

2. the relative line number (number of lines read to obtain a match for the search argument), and for a DISKR operation only, the absolute line number (position from the top of the file).

Stacked line number values may be used on subsequent EXECIO operations for “lines” or “linenum” operands.

The CMS SENTRIES command results in a return code equal to the number of lines in the program stack. Thus, the difference between SENTRIES return codes, one before and one after an EXECIO operation, is the number of lines stacked as a result of that operation.

If the EXEC is coded in the REXX language, then the QUEUEDO() built-in function can be used to return the same information as the CMS SENTRIES command. Refer to the VM/SP System Product Interpreter Reference for information on the QUEUEDO() built-in function.

For stack operations, the maximum length that EXECIO processes is limited to 255 bytes (the maximum line length that can be stacked).

Setting Variables Directly:

The VAR and STEM options allow EXEC 2 and System Product interpreter variable(s) to be set directly, bypassing the use of the stack. Data flow in and out of the variable(s) is:

- always FIFO (first-in first-out) for read or write operations using STEM variables.
• FIFO for read options, such as FIND or LOCATE, that result in a line number being returned.

A successful search (LOCATE, FIND, etc.) operation results in two lines being assigned (FIFO) to EXEC variables if the STEM option has been used:

1. the contents of the line that satisfies the search argument;

2. the relative line number (number of lines read to obtain a match for the search argument), and for a DISKR operation only, the absolute line number (position from the top of the file).

Returned line number values may be used on subsequent EXECIO operations for “lines” or “linenum” operands.

If the STEM option was specified, then variable xxxx0 contains the number of lines of data returned.

For VAR and STEM operations called from the System Product Editor, the maximum that EXECIO processes is 2^31-1; if called from an EXEC2 program, the maximum value is 255.

**Closing files and virtual devices:**

EXECIO (DISKR or DISKW) operations do not close referenced files when the operation terminates unless the FINIS operand is specified on the command line. (Some CMS commands, such as ERASE and STATE, close the referenced file, then execute.) If you choose, you may close these files manually by invoking the CMS FINIS command. There is considerable system overhead associated with the execution of FINIS. Therefore, if multiple references are to be made to a given file, it should be closed only when necessary.

If successive EXECIO commands reference a particular internal area of a CMS file, it is probably more efficient to let the file remain open until the last of these commands is issued. If so, each operation begins at the file line following the last line processed. This eliminates much of the need for calculating the “linenum” value.

EXECIO does not close virtual spool devices. Therefore, to cause any spooled EXECIO output to be processed you must close the corresponding device. For example:

```cp close printer```

If an input spool file is read with the EXECIO CARD operation and the read is not completed (that is, the virtual machine does not get a last-card indication), you must issue a CP CLOSE READER command to be able to read that file again (or to read any other file). The file is purged unless you specify HOLD when you close a reader file. Refer to the CP CLOSE command in the VMjSP CP Command Reference.
If you have specified the PRINT or PUNCH operand and you try to write a line that is longer than the virtual device (PRINTER or PUNCH) allows, you will get error message DMSEIO632E.

"lines" operand:

For a DISKW, PUNCH, PRINT, or EMSG operation (if the STEM option had not been specified), if the "lines" operand exceeds the number of lines on the program stack, reading continues through the terminal input buffer. If the "lines" operand is still not satisfied, a VM READ is issued to the terminal. At that point, you must enter the balance of the lines (the number specified in the "lines" operand) from the terminal. Entering a blank character (null line) does not terminate the EXECIO operation; it writes a blank character to the output device. When the "lines" operand has been satisfied, the EXEC from which EXECIO was issued continues to execute.

If * (to end of file) is specified for "lines" on an output operation, and you want the operation to terminate at any given line in the program stack or a STEM array, you must make sure that line is null. Reading a null line terminates any of the four output operations if * is specified for the "lines" operand.

For input operations, the number of lines written to the program stack or the STEM array does not necessarily equal the number specified by "lines." For example, an end-of-file or a satisfied search condition terminates a read operation, even if the specified number of lines has not been written to the program stack or the STEM array. When a search argument (FIND, LOCATE, AVOID option) is satisfied, and no SKIP option is specified, and the default stacking order (LIFO) is used, the line at the top (first line out) of the stack or the STEM array contains the number of operations required to satisfy the search. The next line contains the line that satisfied the search. If the search argument is not satisfied, a return code of 2 is given.

When * is specified for "lines" on a read operation, the operation is terminated at end-of-file. A return code of 0 is given because the * is an explicit request to read to end-of-file.

When a search argument (FIND, LOCATE, and AVOID options) is not satisfied and an end-of-file situation occurs for the EXECIO CARD operation, the reader file is purged unless a CP SPOOL READER HOLD was previously specified. For more information on how spool file are processed, refer to the CP CLOSE and CP SPOOL commands in the VM/SP CP Command Reference.
**DISKR operation:**

The first line read on a DISKR operation may be:

1. the first line of the specified file;
2. specified using the "linenum" operand;
3. determined by the results of a previous operation.

The DISKR operation may be used to simply read a specified number of lines from a specified file and write them to the program stack or a variable. For example, suppose file MYFILE DATA contains:

```
The number one color is red
The number two color is yellow
The number three color is green
The number four color is blue
The number five color is black
```

The command:

```
EXECIO 2 DISKR MYFILE DATA * 1
```

writes to the program stack (FIFO) two lines beginning with line one, like this:

```
| The number one color is red | <-next line read
| The number two color is yellow |
|/ |
```

However, a little more complex version of this command:

```
EXECIO 2 DISKR MYFILE DATA * 3 (LIFO MARGINS 5 14
```

would have resulted in this program stack:

```
| number fou | <-next line read
| number thr |
|/ |
```

Note the use of * as a filemode operand on the command lines just above to serve as a place holder. The command:

```
EXECIO 2 DISKR MYFILE DATA * 1 (STEM X.
```

assigns to variables X.1 and X.2 one line each, beginning with line one, like this:

```
| The number one color is red | X.1
| The number two color is yellow | X.2
|/ |
```

The X.0 variable contains the number of lines (in this example, 2).
When a line satisfies the LOCATE, FIND, or AVOID option for a DISKR operation, EXECIO writes that line to the program stack (LIFO) or a variable (FIFO) and in an additional stack line or variable, writes the relative (number of lines read to satisfy the search) and absolute (position from the top of the file) line numbers.

**CP operand:**

When a search argument is required, the CP operand uses the FIND, LOCATE, and AVOID options to process output resulting from the associated CP command. Each line that satisfies the search criteria is written to the program stack or a variable. When data exceeds 8192 characters, it is truncated and an error code is returned. If you specify the BUFFER option, data is truncated after the number of characters specified in length or 8192 is reached, whichever is greater. The number of read operations required to match the search argument is written to the next stack line.

If you do not supply the CP command to be issued via the STRING option, the next line in the program stack is treated as that command. If there are no lines in the program stack, the next line in the console input buffer is treated as the CP command. If there are no lines in the console input buffer, then a VM READ is issued to the terminal. A null line terminates the operation.

Keep in mind that all characters of CP commands must be uppercase.

ZONE and MARGINS options do not affect the reading of the CP command; however, they do affect the portions of the lines processed as a result of the command execution.

**DISKW operand:**

The DISKW operand causes the next lines from the program stack to be written to a CMS file. The point at which writing begins in an existing file on a DISKW operation may:

1. follow the last file line (default "linenum" when writing to a newly opened file, for example);

2. be specified using the "linenum" operand;

3. be determined by the results of a previous operation.

For example, suppose you want to write 10 lines from the program stack to the end of an existing A-disk file, BUCKET STACK A. Your EXEC file statement to do this would be:

```
EXECIO 10 DISKW BUCKET STACK A
```

Now, take a slightly more complex requirement. Using stack lines down to the first null line, create a new A-disk file, BASKET STAX A, then close
the file after it is written. Also, make the file fixed length format with a record length of 60. The EXECIO command to do this is:

EXECIO * DISKW BASKET STAX A 1 F 60 (FINIS

A word of caution about using the linenum operand to insert lines in the middle of CMS variable length files. Because of the way CMS handles these files, any variable length line inserted must be equal in length to the line it displaces. Otherwise, for disks formatted in:

1. 512-, 1K-, 2K-, or 4K-byte blocks, all file lines following the one inserted are truncated;
2. 800-byte blocks, the file remains unchanged and CMS issues message 105S (nn = 15).

For example, assume a disk format in 2K-byte blocks. The variable format file WORDS LEARNING A is:

A is for apple
C is for cake
C is for candy
D is for dog

execution of:

EXECIO 1 DISKW WORDS LEARNING A 2 (STRING B is for butterfly

produces a file that contains only:

A is for apple
B is for butterfly

Because "B is for butterfly" contains more characters than the line it writes over, "C is for cake," all lines following it are truncated. However, slightly modifying the command to:

EXECIO 1 DISKW WORDS LEARNING A 2 (STRING B is for baby

results in:

A is for apple
B is for baby
C is for candy
D is for dog

To prevent truncation when inserting records in a variable-length file, you can use fixed-format files.

**recfm**

**lrecl operands:**

The default value for recfm is V (variable), in which case "lrecl" has no meaning. If you specify F (fixed) for recfm, the default lrecl value is 80. The maximum lrecl value that you may specify is 255 unless the VAR or STEM option is used to bypass the use of the program stack, in which case
the maximum is that which is defined for the type of file in use (800 byte or EDF file).

When lines are written to an existing file, the record format and record length of that file apply. Specifying recfm or lrecl values on the EXECIO command line that conflict with those of the existing file causes an error message to be issued.

**CC operand:**

When you specify CC together with the DATA operand, be sure the first character of each line to be sent to the virtual printer may be removed and interpreted as carriage control for that line.

You may use ASA or machine code characters with the CC operand to specify carriage control. For example, CC 0 causes space two lines before printing. You can find information about these codes under the PRINTL macro description in *VM/SP CMS Macros and Functions Reference.*

If your virtual printer has a FCB containing a channel 9 or 12, you should not print from the stack, but instead use a variable. When the channel 9 or 12 is sensed, the write operation terminates after carriage spacing but before writing the line. The line has been purged from the stack and EXECIO returns a return code of 102 or 103. You must modify the carriage control character in your data to take whatever action is appropriate, that is, skip to channel or print and not space.

**EMSG operand:**

Lines to be displayed by EMSG should have the format:

```
xxxmmmnnns
```

where:

- **xxxmmm** is the issuing module name
- **nnn** is the message number
- **s** indicates the message type (E - error, I - informational, W - warning etc.)

The current settings of the CP SET EMSG command control the displayed lines. These settings, combined with message length, can cause messages to be abbreviated or not displayed at all.

**linenum operand:**

When a linenum value (default 0) is not specified on the EXECIO command line, the number of the next file line available for reading or writing depends on results of previous operations that referenced that file. For example, consider the two EXECIO DISKR operations just below. By looking at the first of these commands you can see:
1. Four lines are to be read from MYFILE DATA, starting at line 1;

2. Because FINIS is not specified on the command line, MYFILE DATA remains open after the first read operation. Because the first command reads 4 lines, the subsequent read operation will begin at line 5.

   EXECIO 4 DISKR MYFILE DATA * 1

   EXECIO 3 DISKR MYFILE DATA (FINIS

Because the second EXECIO command specifies no linenum operand, reading of the specified 3 lines begins at line 5.

Two situations that would cause the second EXECIO command to not begin execution at line 5 are:

1. a program other than EXECIO accessing MYFILE DATA after the first and before the second EXECIO command is executed;

2. a CMS operation completing such that the CMS READY message (Ready;) is displayed. In that case CMS closes associated files. Therefore, subsequent operations using these files would begin at line 1.

The FINIS operand causes MYFILE DATA to close. Therefore, any subsequent DISKR operation using a default linenum value would begin reading at line 1.

**FIND**

**LOCATE**

**AVOID options:**

The delimiter pair for the specified character string need not be / /. They may be any character not included in the string. For example:

   EXECIO * DISKR MYFILE DATES (LOCATE $12/25/81$

**FIFO**

**LIFO options:**

Most EXECIO operations that write to the program stack default to FIFO, first line written to the stack will be the first read out. The exceptions (LIFO) are operations involving a search (LOCATE, FIND, and AVOID options). These operations result in the relative line number (number of lines read to satisfy the search) being stacked. For DISKR operations the absolute line number (position from the top of file) is also stacked on the same line. It is necessary to have these numbers at the top of the stack so that they are immediately accessible to a subsequent EXECIO command.
**SKIP option:**

On EXECIO read operations the SKIP operand prevents input lines from being written to the program stack or a variable. For example, you might want to put on the program stack all lines of MYFILE DATA that follow the line containing “4120 Rock Road.” First, to search through the file for the line after which reading to the program stack is to begin, issue:

```
EXECIO * DISKR MYFILE DATA * 1 (LOCATE /4120 Rock Road/ SKIP
```

The SKIP option prevents the line being searched for, together with the line number, from being written to the program stack. Then, to write to the program stack the next line through the end of file, issue:

```
EXECIO * DISKR MYFILE DATA
```

Keep in mind that accessing MYFILE DATA by another program or causing a CMS READY message to be displayed prior to issuing the second EXECIO command would change the point at which the second command begins reading. When possible, you should specify the linenum operand explicitly.

Another use of the SKIP option might be the execution of a CP command via the CP operand to obtain a return code without displaying the resulting messages or writing them to the program stack or a variable. For example:

```
EXECIO * CP (SKIP STRING Q userid
```

The userid must be uppercase.

As an alternative, specifying 0 for the “lines” operand value with the CP operand also causes results not to be displayed or written to the program stack.

**STEM option:**

On EXECIO operations, the STEM option allows an array of EXEC 2 or System Product interpreter variables to be set directly, bypassing the use of the stack. For example, if you want the first 3 lines of MYFILE DATA to be assigned to System Product interpreter variables X.1, X.2, and X.3, issue:

```
'EXECIO 3 DISKR MYFILE DATA * 1 (STEM X. '
```

Variable X.1 now contains the first line of MYFILE DATA, variable X.2 contains the second line, and variable X.3 contains the third line. Variable X.0 contains the number of lines (in this example, 3).

For System Product Interpreter variables, the variable name should be enclosed in quotes and must be in uppercase. For EXEC 2 variables, you omit the ‘&’, quotes are not necessary, and mixed case may be used.

**Note:** Doing a DISKR operation using the STEM option from an EXEC 2 EXEC may cause truncation to 255 bytes and a return code of 1.
For the STEM option, the maximum length variable name is 240 bytes.

**VAR option:**

On EXECIO operations, the VAR option allows an EXEC 2 or System Product interpreter variable to be set directly, bypassing the use of the stack. For example, if you want the second line of MYFILE DATA to be assigned to an EXEC 2 variable named X, issue:

EXECIO 1 DISKR MYFILE DATA * 2 (VAR X

Variable X now contains the second line of MYFILE DATA.

For System Product Interpreter variables, the variable name should be enclosed in quotes and must be in uppercase. For EXEC 2 variables, you omit the ‘&’, quotes are not necessary, and mixed case may be used.

**Note:** Doing a DISKR operation using the VAR option from an EXEC 2 EXEC may cause truncation to 255 bytes and a return code of 1.

For the VAR option, the maximum length variable name is 250 bytes. See the VM/SP CMS User's Guide for an example of using EXECIO within System Product Interpreter programs.

**Messages and Return Codes:**

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSEIO621E</td>
<td>Bad Plist: Device and lines arguments are required [RC = 24]</td>
</tr>
<tr>
<td>DMSEIO621E</td>
<td>Bad Plist: Invalid value value for number of lines [RC = 24]</td>
</tr>
<tr>
<td>DMSEIO621E</td>
<td>Bad Plist: Missing DEVICE argument [RC = 24]</td>
</tr>
<tr>
<td>DMSEIO621E</td>
<td>Bad Plist: Invalid DEVICE argument argument [RC = 24]</td>
</tr>
<tr>
<td>DMSEIO621E</td>
<td>Bad Plist: Disk argument argument is missing [RC = 24]</td>
</tr>
<tr>
<td>DMSEIO621E</td>
<td>Bad Plist: Invalid character in file identifier [RC = 24]</td>
</tr>
<tr>
<td>DMSEIO621E</td>
<td>Bad Plist: Input file fileid does not exist [RC = 24]</td>
</tr>
<tr>
<td>DMSEIO621E</td>
<td>Bad Plist: Invalid value value for disk file line number [RC = 24]</td>
</tr>
<tr>
<td>DMSEIO621E</td>
<td>Bad Plist: Disk filemode required for DISKW [RC = 24]</td>
</tr>
<tr>
<td>DMSEIO621E</td>
<td>Bad Plist: Invalid record format recfm-- Must be either F or V [RC = 24]</td>
</tr>
<tr>
<td>DMSEIO621E</td>
<td>Bad Plist: Invalid record length argument irel [RC = 24]</td>
</tr>
<tr>
<td>DMSEIO621E</td>
<td>Bad Plist: File format specified recfm does not agree with existing file format recfm [RC = 24]</td>
</tr>
<tr>
<td>DMSEIO621E</td>
<td>Bad Plist: File irecl specified irecl does not agree with existing file irecl irecl [RC = 24]</td>
</tr>
<tr>
<td>DMSEIO621E</td>
<td>Bad Plist: Invalid positional argument argument [RC = 24]</td>
</tr>
<tr>
<td>DMSEIO621E</td>
<td>Bad Plist: EXECIO options only allowed with extended plist [RC = 24]</td>
</tr>
<tr>
<td>DMSEIO621E</td>
<td>Bad Plist: Unknown option name name [RC = 24]</td>
</tr>
<tr>
<td>DMSEIO621E</td>
<td>Bad Plist: Value missing after option option [RC = 24]</td>
</tr>
<tr>
<td>DMSEIO621E</td>
<td>Bad Plist: Value value not valid for option option [RC = 24]</td>
</tr>
</tbody>
</table>
DMSEIO621E  Bad Plist: option option is not valid with option option
            [RC = 24]
DMSEIO621E  Bad Plist: option option not valid with operation operation
            [RC = 24]
DMSEIO621E  Bad Plist: STRING option with LINES=* is valid only for
            CP operation [RC = 24]
DMSEIO621E  Bad Plist: VAR option with LINES>1 is invalid [RC = 24]
DMSEIO621E  Bad Plist: Invalid mode mode [RC = 24]
DMSEIO621E  Bad Plist: Invalid EXEC variable name [RC = 24]
DMSEIO621E  Bad Plist: NAMEFIND must be invoked as a nucleus
            extension [RC = 24]
DMSEIO621E  Bad Plist: QUERY must be invoked as a nucleus extension
            [RC = 24]
DMSEIO622E  Insufficient free storage for EXECIO [RC = rc]
DMSEIO631E  function can only be executed from an EXEC-2 or REXX
            EXEC [RC = rc]
DMSEIO632E  I/O error in EXECIO: rc = nnnn from command command
            [RC = rc]

Return Code Definitions:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Finished correctly</td>
</tr>
<tr>
<td>1</td>
<td>Truncated</td>
</tr>
<tr>
<td>2</td>
<td>EOF before specified number of lines were read</td>
</tr>
<tr>
<td>3</td>
<td>Count ran out without successful pattern match</td>
</tr>
<tr>
<td>24</td>
<td>Bad PLIST</td>
</tr>
<tr>
<td>41</td>
<td>Insufficient free storage to load EXECIO</td>
</tr>
<tr>
<td>1nn</td>
<td>100 + return code from I/O operation (if nonzero)</td>
</tr>
<tr>
<td>2008</td>
<td>Variable name supplied on STEM or VAR option was invalid.</td>
</tr>
<tr>
<td>x1nnn</td>
<td>1000 + return code from CP command (if nonzero), where x is 0, 1, 2, or 3, as described above</td>
</tr>
</tbody>
</table>

Note: For more information on the EXECCOMM function and the associated return codes, refer to the VM/SP System Product Interpreter Reference.
EXECLOAD

Use the EXECLOAD command to load an EXEC or System Product Editor macro into storage and prepare it for execution.

The format of the EXECLOAD command is:

```
EXECLOAD | EXLoad | [fn ft][fm [execname [exectype]]] [options...[]]
```

**Options:**

- **User**
- **System**
- **Push**

**where:**

- **fn** is the filename of the EXEC to be loaded.
- **ft** is the filetype of the EXEC to be loaded.
- **fm** is the filemode of the EXEC to be loaded. The default for filemode is an asterisk (*), which means the first file in the disk search order that satisfies the filename and filetype qualifications is loaded. In order to assign an execname and exectype, you must also specify the filename, filetype and filemode.

**execname**

is the name to be assigned to the loaded EXEC. The default is ‘=’, which means the EXEC’s present filename is to be used.

**exectype**

is the type to be assigned to the loaded EXEC. The default is ‘-‘, which means the EXEC’s present filetype is to be used.
Options:

User
specifies that the storage for the loaded EXEC is allocated from user free storage. This is the default.

SYstem
specifies that the storage for the loaded EXEC is allocated from nucleus free storage.

Push
specifies that the EXEC is loaded whether an EXEC by the same name already exists in storage. This loaded EXEC does not replace the existing EXEC. Subsequent invocation of this execname and exectype executes the most recently loaded version. Also, a subsequent EXECDROP of this execname and exectype drops the most recently loaded version.

Usage Notes:

1. The filename and filetype can each be from one to eight characters. The valid characters are A-Z, a-z, 0-9, $, #, @, +, - (hyphen), : (colon), and _ (underscore). The execname and exectype of the execid may also be from one to eight characters. However, the execname and exectype are not limited to the filename and filetype character set. The only characters that are invalid within an execname or exectype are =, *, (, ), and X'FF'.

2. If SET INSTSEG is ON and you attempt to load an EXEC into storage with the same execname and exectype as a SHARED EXEC, you must specify the PUSH option.

3. To list the EXECs in storage and in an Installation Discontiguous Shared Segment (DCSS), use the EXECMAP command. To remove an EXEC from storage or to discontinue use of an EXEC in an Installation DCSS, use the EXECDROP command. Use the SET INSTSEG OFF command to discontinue use of all SHARED EXECs temporarily. To determine the status of a specific EXEC, use the EXECSTAT command.

4. To list the EXECs that are currently storage-resident, use the EXECMAP command. To purge a storage-resident EXEC, use the EXECDROP command. To determine the status of a specific EXEC, use the EXECSTAT command.
EXECLOAD

Examples:

The following command:
execload tphone exec a (user
loads the TPHONE EXEC from your A-disk into user free storage and assigns it the same name.

Specifying the following:
execload tphone exec a = xedit (system
loads the TPHONE EXEC A into nucleus free storage and assigns to it the name TPHONE XEDIT.

Responses:

None.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSEXL003E Invalid option: option [RC = 24]
DMSEXL042E No fileid specified [RC = 24]
DMSEXL054E Incomplete fileid specified [RC = 24]
DMSEXL062E Invalid character * in fileid [RC = 20]
DMSEXL070E Invalid parameter parameter [RC = 24]
DMSEXL104S Error nn reading file fn ft fm from disk [RC = 100]
DMSEXL108S Virtual storage capacity exceeded [RC = 104]
DMSEXL414E Execid execname exectype already in storage [RC = 1]
DMSEXL415E Invalid character char in execid execname exectype [RC = 20]
DMSEXL417E Only EXEC-2 and REXX EXECs are supported as storage resident EXECs [RC = 4]
EXECMAP

Use the EXECMAP command to list EXECs and System Product Editor macros in storage and in an Installation Discontiguous Shared Segment (DCSS).

The format of the EXECMAP command is:

```
EXECMap
EXMap
[execname [exectype]] [options...[]]
```

<table>
<thead>
<tr>
<th>Options:</th>
<th>User</th>
<th>SYstem</th>
<th>SHared</th>
<th>STACK</th>
<th>FIFO</th>
<th>FIFO</th>
<th>LIFO</th>
</tr>
</thead>
</table>

**where:**

*execname* is the name of the storage-resident EXEC(s) to be listed. If an asterisk (*) is coded in this field, all execnames are used.

*exectype* is the type of the storage-resident EXEC(s) to be listed. If an asterisk (*) is coded in this field, all exectypes are used. The default is an asterisk (*).

If no operands are specified, the list contains all of the EXECs and Editor Macros in storage and in the Installation DCSS. (Issuing EXECMAP with no operands is like issuing “EXECMAP * *”.)

**Options:**

*User* indicates that the storage for the EXEC(s) was allocated from user free storage when EXECLOADED. Only the EXECs that satisfy the execname and/or exectype qualifications and that also have the USER attribute are listed. If neither USER, SYSTEM, nor SHARED is specified, then all EXECs that satisfy the execname and/or exectype qualifications are listed. SHARED EXECs are not listed if SET INSTSEG is OFF.
EXECMAP

SYstem
indicates that the storage for the EXEC(s) was allocated from nucleus free storage when EXECLOADed. Only the EXECs that satisfy the execname and/or exectype qualifications and that also have the SYSTEM attribute are listed. If neither USER, SYSTEM, nor SHARED is specified, then all EXECs that satisfy the execname and/or exectype qualifications are listed. SHARED EXECs are not listed if SET INSTSEG is OFF.

SHared
indicates that the EXEC(s) was located in a Discontiguous Shared Segment (DCSS). Only the EXECs that satisfy the execname and/or exectype qualifications and that also have the SHARED attribute are listed. If neither USER, SYSTEM, nor SHARED is specified, then all EXECs that satisfy the execname and/or exectype qualifications are listed. SHARED EXECs are not listed if SET INSTSEG is OFF.

STACK [FIFO] [LIFO]
specifies that the listed information should be placed in the program stack (for use by an EXEC or other program) instead of being displayed at the terminal. The information is stacked either FIFO (first in first out) or LIFO (last in first out). The default order is FIFO.

FIFO
specifies that the information should be placed in the program stack rather than displayed at the terminal. The information is stacked FIFO. The options STACK, STACK FIFO, and FIFO are all equivalent.

LIFO
specifies that the information should be placed in the program stack rather than displayed at the terminal. The information is stacked LIFO. This option is equivalent to STACK LIFO.

Usage Notes:

1. When the STACK option is specified, a line is stacked for each storage-resident EXEC that satisfies the command request. The FIFO or LIFO options dictate the order of the list. The order is relative to the EXECs-in-STORAGE (EXIS) chain.

2. Use the EXECDROP command to delete a storage-resident EXEC or to discontinue use of an EXEC in a Discontiguous Shared Segment (DCSS). To discontinue use of all SHARED EXECs temporarily, use the SET INSTSEG OFF command. Use the EXECLOAD command to load an EXEC into storage. Use the EXECSTAT command to determine the status of storage-resident EXECs.
Examples:

To list all storage-resident EXECs with an exectype of XEDIT, then specify:

```
execmap * xedit
```

The following command:

```
execmap travel exec (fifo
```

stacks the map output FIFO for the storage-resident TRAVEL EXEC.

Responses:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Usage</th>
<th>Records</th>
<th>Bytes</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>execname</td>
<td>exectype</td>
<td>usage</td>
<td>records</td>
<td>bytes</td>
<td>user</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSEXM003E Invalid option: option [RC = 24]
- DMSEXM070E Invalid parameter parameter [RC = 24]
- DMSEXM108S Virtual storage capacity exceeded [RC = 104]
- DMSEXM415E Invalid character char in execid execname exectype [RC = 20]
- DMSEXM416W There are no execname exectype [SYSTEM][or]USER[or]SHARED] EXECs storage resident [RC = 28]
Use the EXECOS command to reset the OS and VSAM environments under CMS without returning to the interactive environment. The EXECOS command can be invoked by specifying EXECOS without parameters or by preceding any CMS command with EXECOS.

The format of the EXECOS command is:

```
EXECOS [cmd [operand1 [operand2 ... operandn]]]
```

where:

`[cmd [operand1 [operand2 ... operandn]]]` is a CMS command. If EXECOS precedes a CMS command, first the CMS command is processed, and then the EXECOS performs the OS reset function. The return code is that of the CMS command that was processed. A return code of -3 may indicate that the EXECOS command preceded an unknown CMS command. If EXECOS is specified without parameters, the OS environment is reset and the return code is zero.

Usage Notes:

1. The EXECOS command is primarily intended for use in an EXEC 2 or a System Product interpreter EXEC that either invokes several OS programs sequentially or invokes the same OS program repetitively.

   Note: The cleanup of an OS or VSAM environment, if needed, is performed after each command has been processed from a CMS EXEC or after processing a command entered at your terminal. Therefore, in these two cases, the use of the EXECOS command is unnecessary.

2. The EXECOS command clears the following:

   - STAE exits
   - STIMER exits
   - SPIE exits
   - STAX exits
   - TXTLIB directories
   - MACLIB pointers
   - SSTAT extensions
   - LINKLISTS (LINKSTRT and LINKLAST)
   - OS environment flags (OSSFLAGS)

   If VSAM is running, VSAM cleanup is also done.
If the Vector Facility is in use, it is disabled to force a vector reset.

3. When processing in the VSAM environment, issue EXECOS prior to issuing CMS commands or user applications that run in the user area. Failure to do so could result in unpredictable errors and the overlay of VSAM control blocks.

4. When you request a reset of the OS environment after the execution of a CMS EXEC, the EXECOS command should precede the CMS EXEC command. For example:

```plaintext
&TRACE ALL
EXECOS EXEC VMFASM DMSSEB DMSSP
&EXIT
```

5. Whenever the CMS command, CP, must be used within an EXEC or EXEC 2 EXEC to transmit a CP command to the VM/SP control program environment, a request for a reset of the OS environment after the execution of the CP command requires that the EXECOS command precede the CMS CP command. For example:

```plaintext
&TRACE ALL
EXECOS CP QUERY DASD
&EXIT
```

Responses:

None.

Messages and Return Codes:

None.
EXECSTAT

Use the EXECSTAT command to verify the existence of an EXEC or System Product Editor macro in storage and on disk. The status is returned as a return code in register 15.

The format of the EXECSTAT command is:

```
EXECStat EXStat [ execname ] [ exectype ]
```

where:

*execname*  
is the name of the storage-resident EXEC(s) whose existence is to be verified. If an asterisk (*) is coded in this field, all storage resident execnames will be used.

*exectype*  
is the type of the storage-resident EXEC(s) whose existence is to be verified. If an asterisk (*) is coded in this field, all exectypes are used.

Usage Notes:

1. Specifying an asterisk (*) for both the execname and the exectype will verify the existence of any storage-resident EXECs. When you specify 'EXECSTAT * *', the return codes have the following meanings:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>There are storage-resident EXECs.</td>
</tr>
<tr>
<td>4</td>
<td>There are no storage-resident EXECs.</td>
</tr>
</tbody>
</table>

The contents of register 1 should be disregarded in this case.

2. To list the EXECs in storage and in an Installation Discontiguous Shared Segment (DCSS), use the EXECMAP command. To remove an EXEC from storage or to discontinue use of an EXEC in an Installation DCSS, use the EXECDROP command. Use the SET INSTSEG OFF command to discontinue use of all SHARED EXECs temporarily. To load an EXEC into storage, use the EXECLOAD command.
### Examples:

To see if there are any storage-resident EXECs with the execname FILELIST, you would enter:

```
execstat filelist *
```

To see if there are any storage-resident EXECs with an exectype of XEDIT, you would enter:

```
execstat * XEDIT
```

### Responses:

The return codes and their meanings are:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>EXEC will execute from storage and register 1 contains pointer to the fileblock.</td>
</tr>
<tr>
<td>4</td>
<td>EXEC will execute from dasd and register 1 contains pointer to the FST.</td>
</tr>
<tr>
<td>8</td>
<td>EXEC will not execute from storage and does not exist on dasd.</td>
</tr>
</tbody>
</table>

### Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

- **DMSEXQ003E**: Invalid option: `option` [RC = 24]
- **DMSEXQ042E**: No execid specified [RC = 24]
- **DMSEXQ054E**: Incomplete execid specified [RC = 24]
- **DMSEXQ070E**: Invalid parameter `parameter` [RC = 24]
- **DMSEXQ109S**: Virtual storage capacity exceeded [RC = 104]
- **DMSEXQ415E**: Invalid character `char` in execid `execname exectype` [RC = 20]
EXECUPDT

Use the EXECUPDT command to apply updates to a System Product interpreter source program and create an executable version of the program. This command can only be used with System Product interpreter programs. See VM/SP CMS for System Programming for information on how to set up a program for updating. See the VM/SP System Product Editor Command and Macro Reference for information about editing files in update mode with the System Product editor.

EXECUPDT creates a variable-format, executable program from a fixed-format, 80-column, sequenced source file. The filetype of the source file must begin with a dollar sign ($), for example $EXEC or $XEDIT. You do not enter this dollar sign when you enter the filetype on the EXECUPDT command. EXECUPDT updates files in the same manner as the UPDATE command and accepts all UPDATE command options. In addition to the UPDATE command options, EXECUPDT has options that allow you to request removal of Support Identification (SID) codes from the file, include a log of applied updates with the file, or remove comments from the file to improve performance.

The format of the EXECUPDT command is:

```
EXECUPDT fn [ft ft fm] [options...()]]
```

where:

```
fn  ft fm
```

is the file identifier of the source input file. The file must consist of 80-character fixed-format records with sequence numbers in positions 73 through 80 or 76 through 80. If the filetype or filemode is omitted, EXEC and * are assumed, respectively.
Options:

CTL
specifies that a file named "fn1" with a filetype of CNTRL is an update control file that controls the application of multiple update files to the source input file.

HISTORY
specifies that a file named 'fn' with a filetype of UPDATES is to be appended to the updated source file as a System Product interpreter comment. This file, 'fn UPDATES', is created by the UPDATE command.

NOHISTORY
specifies that the UPDATES file created by the UPDATE command should not be appended to the updated source file. This is the default.

COMPRESS
specifies that comment lines in the source file that start in column 1 and begin with /*! should be removed from the updated source file to improve performance. This is the default.

Note: You identify those comments that are extraneous (for instance, a prolog) by putting an exclamation point (!) after the /* that begins the comment. Remember that you must leave a comment as the first statement so VM/SP can identify it as a System Product interpreter program.

NOCOMPRESS
specifies that comments are not to be removed from the updated source file.

COMMENTS
specifies that comments are not to be removed from the source file except those removed by the COMPRESS option. This is the default.

NOCOMMENTS
specifies that all comments and leading blanks are to be removed from the source file. One comment line containing the execname and exectype is inserted at the beginning of the file. Do not specify NOCOMMENTS with the COMPRESS option.

Note: If you use NOCOMMENTS on an EXEC that uses data stored within comments, the updated file may not execute properly.

ETMODE
specifies that the source file contains DBCS characters and that shift-in and shift-out characters should be paired while comments are being removed. ETMODE only functions when specified with NOCOMMENTS.
EXECUPDT

SID
specifies that columns 63 to 71 of the input file contain an SID Code which is to be removed from the updated source file.

NOSID
specifies that columns 63 to 71 of the input file do not contain an SID Code. Only the sequence numbers in columns 73 to 80 of the input file are to be removed from the updated source file. This is the default.

NOUPdate
specifies that no update files are to be applied before building the updated source file. Specify this option when converting a source file which does not yet have updates. You can also use the CTL option to convert a file with no updates. The EXECUPDT command ignores the HISTORY option if NOUPDATE is specified.

Usage Notes:

1. You may also use any other options accepted by the UPDATE command. See the description of the CMS UPDATE command for a complete list of available options.

2. If the SID option is specified, everything in columns 63-80 is truncated. Thus, any data in columns 63 and beyond is lost.

3. If you want to issue EXECUPDT from an EXEC program, you should precede it with the EXEC command; that is, specify
exec execupdt

Example:

If you want to create an executable version of the source file MYFILE $XEDIT and remove comments from the file, you would enter:
execupdt myfile xedit a (comp

Messages and Return Codes:

DMSWUP002E File fn ft fm not found [RC = 28]
DMSWUP054E Incomplete fileid specified [RC = 24]
DMSWUP419E fn ft has an error with quote/comment nesting. A quote is|A comment is/n comments are open at the end of the program. [RC = 8]
DMSWUP637E Missing value for the CTL option [RC = 24]
DMSWUP649E Extraneous parameter parameter [RC = 24]
DMSWUP671E Error updating fn ft fm; rc = nn from XEDIT [RC = 100]
DMSUPD001E No filename specified [RC = 24]
DMSUPD002E File [fn ft [fm]] not found [RC = 28]
DMSUPD003E Invalid option: option [RC = 24]
<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSUPD007E</td>
<td>File fn ft fm is not fixed, 80-character records [RC = 32]</td>
</tr>
<tr>
<td>DMSUPD010W</td>
<td>Premature EOF on file fn ft fm—sequence number seqno not found [RC = 12]</td>
</tr>
<tr>
<td>DMSUPD024E</td>
<td>File fn ft fm already exists [RC = 28]</td>
</tr>
<tr>
<td>DMSUPD048E</td>
<td>Invalid mode mode [RC = 24]</td>
</tr>
<tr>
<td>DMSUPD065E</td>
<td>option option specified twice [RC = 24]</td>
</tr>
<tr>
<td>DMSUPD066E</td>
<td>option1 and option2 are conflicting options [RC = 24]</td>
</tr>
<tr>
<td>DMSUPD069E</td>
<td>Disk mode not accessed [RC = 36]</td>
</tr>
<tr>
<td>DMSUPD070E</td>
<td>Invalid parameter parameter [RC = 24]</td>
</tr>
<tr>
<td>DMSUPD104S</td>
<td>Error nn reading file fn ft fm from disk [RC = 100]</td>
</tr>
<tr>
<td>DMSUPD105S</td>
<td>Error nn writing file fn ft fm on disk [RC = 100]</td>
</tr>
<tr>
<td>DMSUPD174W</td>
<td>Sequence error introduced in output file: seqno1 to seqno2 seqno1 to seqno2 [RC = 8]</td>
</tr>
<tr>
<td>DMSUPD176W</td>
<td>Sequencing overflow following sequence number seqno [RC = 8]</td>
</tr>
<tr>
<td>DMSUPD179E</td>
<td>Missing or duplicate MACS card in control file fn ft fm [RC = 32]</td>
</tr>
<tr>
<td>DMSUPD181E</td>
<td>No update files were found [RC = 40]</td>
</tr>
<tr>
<td>DMSUPD182W</td>
<td>Sequence increment is zero [RC = 8]</td>
</tr>
<tr>
<td>DMSUPD183E</td>
<td>Invalid {CONTROL\AUX} file control card [RC = 32]</td>
</tr>
<tr>
<td>DMSUPD184W</td>
<td>./S not first card in update file—ignored [RC = 12]</td>
</tr>
<tr>
<td>DMSUPD185W</td>
<td>Invalid character in sequence field seqno [RC = 12]</td>
</tr>
<tr>
<td>DMSUPD186W</td>
<td>Sequence number seqno not found [RC = 12]</td>
</tr>
<tr>
<td>DMSUPD187E</td>
<td>Option STK invalid without CTL [RC = 24]</td>
</tr>
<tr>
<td>DMSUPD207W</td>
<td>Invalid update file control card [RC = 12]</td>
</tr>
<tr>
<td>DMSUPD210W</td>
<td>Input file sequence error: seqno1 to seqno2 [RC = 4]</td>
</tr>
<tr>
<td>DMSUPD299E</td>
<td>Insufficient storage to complete update [RC = 41]</td>
</tr>
<tr>
<td>DMSUPD300E</td>
<td>Insufficient storage to begin update [RC = 41]</td>
</tr>
<tr>
<td>DMSUPD361E</td>
<td>Disk mode is not a CMS disk [RC = 36]</td>
</tr>
</tbody>
</table>
FETCH

Use the FETCH command in CMS/DOS to load an executable phase into storage for execution.

The format of the FETCH command is:

```
FETCH phasename [(options...[])]
```

| Options: | START | COMP | ORIGIN hexloc |

**where:**

*phasename*  
is the name of the phase to be loaded into virtual storage. CMS searches for the phase:

- In a VSE private core image library, if IJSYSCL has been defined.
- In CMS DOSLIBs that have been identified with the GLOBAL command.
- In the VSE system core image library, if you specified the mode letter of the VSE system residence on the SET DOS ON command line.

**Options:**

**START**  
specifies that once the phase is loaded into storage, execution should begin immediately.

**COMP**  
specifies that when the phase is to be executed, register 1 should contain the address of its entry point. (See Usage Note 5.)

**ORIGIN hexloc**  
fetches the program and loads it at the location specified by hexloc; this location must be in the CMS user area below the start of the CMS nucleus. The location, hexloc, is a hexadecimal number of up to eight characters. (See Usage Note 6.)
Usage Notes:

1. If you do not use the START option, FETCH displays a message at your terminal indicating the name of the phase and the storage location of its entry point. At this time, you can set address instruction stops for testing. To continue, issue the START command to initiate execution of the phase just loaded.

2. The fetch routine is also invoked by supervisor call (SVC) instructions 1, 2, 4, or 65. The search order for executable phases is the same as listed above.

3. If you want to fetch a phase from a private core image library, you must issue an ASSGN command for the logical unit SYSCLB and define the library in a DLBL command using the ddname IJSYSCL. For example:

   assgn sysclb c
dlbl ijsyscl c dsn core image lib (sysclb perm

4. Phase fetched from VSE core image libraries must have been link-edited with ACTION REL.

5. CMS uses the COMP option when it fetches the DOS PL/I compiler because that compiler expects register 1 to contain its entry point address. This option is not required when you issue the FETCH command to load your own programs.

   When CMS starts executing a phase that has COMP specified, the

   DMSLIO740I Execution begins ...

   message is not displayed.

6. The ORIGIN option is used by the VSAMGEN installation EXEC procedure to load nonsharable modules on a segment boundary. It is not required when you issue the FETCH command to load your own programs, unless you want to load them at a location other than 20000.

7. The FETCH command should only be used with the START command to execute a VSE program. It should not be used with GENMOD to attempt to create an executable CMS module file.

8. Multiphase program support is different in CMS/DOS than in VSE. The core image directory is not searched for multiphase programs. Thus the value of HIPROG in BGCOM reflects only the ending address of the longest phase loaded, not that of the phase in the library that has the highest ending address.
FETCH

Example:

To load the executable phase MYFILE into your CMS virtual machine and immediately begin execution of the program, you would enter:

fetch myfile (start

Responses:

DMSFET710I Phase phase entry point at location hexloc

This message is issued when the START option is not specified. It indicates the virtual storage address at which the phase was loaded.

DMSLO740I Execution begins ...

This message is issued when the START option is specified; it indicates that program execution has begun.

Messages and Return Codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSFCH016E</td>
<td>No private core image library found [RC = 28]</td>
</tr>
<tr>
<td>DMSFCH104S</td>
<td>Error nn reading file fn ft fm from disk [RC = 100]</td>
</tr>
<tr>
<td>DMSFCH109S</td>
<td>Virtual storage capacity exceeded [RC = 104]</td>
</tr>
<tr>
<td>DMSFCH113S</td>
<td>Disk(vdev) not attached [RC = 100]</td>
</tr>
<tr>
<td>DMSFCH115E</td>
<td>Phase load point less than vstor [RC = 40]</td>
</tr>
<tr>
<td>DMSFCH411S</td>
<td>Input error code nn on SYSaax [RC = rc]</td>
</tr>
<tr>
<td>DMSFCH623S</td>
<td>Phase cannot be loaded at location hexloc--this area is available for system use only [RC = 88]</td>
</tr>
<tr>
<td>DMSFCH777S</td>
<td>DOS partition too small to accommodate fetch request [RC = 104]</td>
</tr>
<tr>
<td>DMSFET003E</td>
<td>Invalid option: option [RC = 24]</td>
</tr>
<tr>
<td>DMSFET004E</td>
<td>Phase phase not found [RC = 28]</td>
</tr>
<tr>
<td>DMSFET029E</td>
<td>Invalid parameter parameter in the option option field [RC = 24]</td>
</tr>
<tr>
<td>DMSFET070E</td>
<td>Invalid parameter parameter [RC = 24]</td>
</tr>
<tr>
<td>DMSFET098E</td>
<td>No phase name specified [RC = 24]</td>
</tr>
<tr>
<td>DMSFET099E</td>
<td>CMS/DOS environment not active [RC = 40]</td>
</tr>
<tr>
<td>DMSFET623S</td>
<td>Phase cannot be loaded at location hexloc--this area is available for system use only [RC = 88]</td>
</tr>
</tbody>
</table>
FILEDEF

Use the FILEDEF command to establish data definitions for OS ddnames, to define files to be copied with the MOVEFILE command, or to override default file definitions made by the assembler and the OS language processors.

The format of the FILEDEF command is:

<table>
<thead>
<tr>
<th>FILEDEF</th>
<th>Terminal PRINTER</th>
<th>[optionA optionD [ ) ] ]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[optionA OPTCD J [ ) ] ]</td>
<td></td>
</tr>
<tr>
<td>PRINTER</td>
<td>PUNCH READER</td>
<td>[optionA ]</td>
</tr>
<tr>
<td></td>
<td>[optionA ]</td>
<td></td>
</tr>
<tr>
<td>DISK</td>
<td>[fn ft fm] [(optionA optionB [ ) ] ]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[FILE ddname A1]</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td>DISK [fn ft fm] [DSN ? DSN qual1 qual2 ...]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[FILE ddname A1]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[DSN qual1.qual2 ...]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[optionA optionB [ ) ] ]</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td>DISK vaddr</td>
<td></td>
</tr>
<tr>
<td>DUMMY</td>
<td>[optionA ]</td>
<td></td>
</tr>
<tr>
<td>TAP n</td>
<td>LABOFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLP [n]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SL [n] [VOLID valid] [DISP MOD] [OptionG]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SUL [n] [VOLID valid]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NL [n]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NSL filename</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[(optionA optionC optionE optionF [ ) ] ]</td>
<td></td>
</tr>
<tr>
<td>GRAF cuu</td>
<td>([PERM] [CHANGE NOCHANGE [ ) ] ]</td>
<td></td>
</tr>
<tr>
<td>CLEAR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FILEDEF

<table>
<thead>
<tr>
<th>Option A</th>
<th>Option B</th>
<th>Option C</th>
<th>Option D</th>
<th>Option E</th>
<th>Option F</th>
<th>Option G</th>
</tr>
</thead>
<tbody>
<tr>
<td>[PERM]</td>
<td>[KEYLEN nnn]</td>
<td>[7TRACK]</td>
<td>[UPCASE]</td>
<td>[LEAVE]</td>
<td>[ALT]</td>
<td>[SYSPARM]</td>
</tr>
<tr>
<td>[CHANGE NOCHANGE]</td>
<td>[XTENT nnnn]</td>
<td>[TRITCH a]</td>
<td>[LOWCASE]</td>
<td>[NOEOV]</td>
<td>[TAPn]</td>
<td>[string]</td>
</tr>
<tr>
<td>[RECFM a]</td>
<td>[LIMCT nnn]</td>
<td>[DEN den]</td>
<td></td>
<td></td>
<td>[cuu]</td>
<td></td>
</tr>
<tr>
<td>[LRECL nnnn]</td>
<td>[OPTCD a]</td>
<td></td>
<td>[UPCASE]</td>
<td></td>
<td>[ALT]</td>
<td></td>
</tr>
<tr>
<td>[BLOCK nnnn]</td>
<td>[DISP MOD]</td>
<td></td>
<td>[LOWCASE]</td>
<td></td>
<td>[TAPn]</td>
<td></td>
</tr>
<tr>
<td>[BLKSIZE nnnn]</td>
<td></td>
<td></td>
<td></td>
<td>[LEAVE]</td>
<td>[cuu]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td></td>
<td></td>
<td>[NOEOV]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[SYSPARM]</td>
<td></td>
<td></td>
<td>(??)</td>
</tr>
</tbody>
</table>
where:

\[\{ \text{ddname} \} \]

is the name by which the file is referred to in your program. The ddname may be from one to eight alphameric characters, but the first character must be alphabetic or national. If a number \( nn \) is specified, it is translated to a FORTRAN data definition name of \( \text{FT}nn\text{FOOl} \). An asterisk (*) may be specified with the CLEAR operand to indicate that all file definitions not entered with the PERM option should be cleared.

**Devices:**

**Terminal**

is your terminal (terminal I/O must not be blocked). Terminal input will be truncated to the console input buffer length of 130 characters.

**PRinter**

is the spooled printer.

**PUnch**

is the spooled punch.

**Reader**

is the spooled card reader (card reader I/O must not be blocked).

**DISK**

specifies that the virtual I/O device is a disk. As shown in the format, you can choose one of two forms for specifying the DISK operand. Both forms are described in “Using the FILEDEF DISK Operand.”

**DUMMY**

indicates that no real I/O takes place for a data set.

**TAP \([n]\)**

is a magnetic tape. The following symbolic names for a tape drive are supported and represent these virtual control units:

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>Virtual Address</th>
<th>Symbolic Name</th>
<th>Virtual Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAP0</td>
<td>180</td>
<td>TAP8</td>
<td>288</td>
</tr>
<tr>
<td>TAP1</td>
<td>181</td>
<td>TAP9</td>
<td>289</td>
</tr>
<tr>
<td>TAP2</td>
<td>182</td>
<td>TAPA</td>
<td>28A</td>
</tr>
<tr>
<td>TAP3</td>
<td>183</td>
<td>TAPB</td>
<td>28B</td>
</tr>
<tr>
<td>TAP4</td>
<td>184</td>
<td>TAPC</td>
<td>28C</td>
</tr>
<tr>
<td>TAP5</td>
<td>185</td>
<td>TAPD</td>
<td>28D</td>
</tr>
<tr>
<td>TAP6</td>
<td>186</td>
<td>TAPE</td>
<td>28E</td>
</tr>
<tr>
<td>TAP7</td>
<td>187</td>
<td>TAPF</td>
<td>28F</td>
</tr>
</tbody>
</table>
If \( n \) is not specified, `FILEDEF` uses the existing TAPn device for the specified ddname. TAP defaults to TAP2 if there is no existing definition for the specified ddname, or if the existing device was not TAPn. TAP0 through TAP7 are at one virtual control unit and TAP8 through TAPF are at another virtual control unit. You can also specify the type of label processing you want on your tape. Specifying label processing is discussed in “Using the `FILEDEF` TAPn operand.”

**GRAF**

specifies that the virtual I/O device is a Graphic Display.

\( cuu \)

is the virtual device address of the attached graphic display.

**CLEAR**

removes any existing definition for the specified ddname. Clearing a ddname before defining it ensures that a file definition does not exist and that any options previously defined with the ddname no longer have effect.

**Options:**

Whenever an invalid option is specified for a particular device type, an error message is issued. Figure 7 on page 197 shows valid options for each device type.
<table>
<thead>
<tr>
<th>Options</th>
<th>OPERANDS READER, PUNCH, PRINTER</th>
<th>TERMINAL</th>
<th>TAPln</th>
<th>DISK DUMMY1</th>
<th>GRAF</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALT</td>
<td></td>
<td></td>
<td></td>
<td>X4</td>
<td></td>
</tr>
<tr>
<td>BLOCK, BLKSIZE</td>
<td>X</td>
<td>X</td>
<td>X0</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>CHANGE, NOCHANGE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CONCAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>DEN</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISP MOD</td>
<td></td>
<td></td>
<td>X4</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>DSORG</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>KEYLEN</td>
<td></td>
<td></td>
<td></td>
<td>X3</td>
<td></td>
</tr>
<tr>
<td>LEAVE</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>LIMCT</td>
<td></td>
<td></td>
<td></td>
<td>X3</td>
<td></td>
</tr>
<tr>
<td>LOWCASE, UPCASE</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>LRECL</td>
<td>X</td>
<td>X</td>
<td>X6</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MEMBER</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>NOEOV</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>OPTCD</td>
<td></td>
<td></td>
<td>X5</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>PERM</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SYSPARM</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>RECFM</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>TRTCH</td>
<td></td>
<td></td>
<td></td>
<td>X3</td>
<td></td>
</tr>
<tr>
<td>XTENT</td>
<td></td>
<td></td>
<td></td>
<td>X2</td>
<td></td>
</tr>
<tr>
<td>7TRACK, 9TRACK, 18TRACK</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7. Valid File Characteristics for Each Device Type of the FILEDEF Command

---

1. No options may be necessary but all disk options are accepted.
2. This option is meaningful only for BDAM files.
3. This option is for 7-track tapes only.
4. This option is for SL tapes only.
5. This option is for Printer only.
6. This should be used for OS compatibility of output.
PERM
retains the current definition until it either is explicitly cleared or is changed with a new FILEDEF command with the CHANGE option. If PERM is not specified, the definition is cleared when a FILEDEF * CLEAR command is executed.

CHANGE
merges the file definitions whenever a file definition already exists for a ddname and a new FILEDEF command specifying the same ddname is issued; the options associated with the two definitions are merged. Options from the original definition remain in effect unless duplicated in the new definition. New options are added to the option list.

NOCHANGE
retains the current file definition, if one exists, for the specified ddname. With this option, the system stops further processing (error checking, scanning, etc.) of the new FILEDEF command if a file definition exists for the specified ddname.

RECFM a
is the record format of the file, where "a" can be one of the following:

<table>
<thead>
<tr>
<th>a</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>Fixed length</td>
</tr>
<tr>
<td>FB</td>
<td>Fixed blocked</td>
</tr>
<tr>
<td>V</td>
<td>Variable length</td>
</tr>
<tr>
<td>VB</td>
<td>Variable blocked</td>
</tr>
<tr>
<td>U</td>
<td>Undefined</td>
</tr>
<tr>
<td>FS,FBS</td>
<td>Fixed length, standard blocks</td>
</tr>
<tr>
<td>VS,VBS</td>
<td>Variable length, spanned records</td>
</tr>
</tbody>
</table>

The values below, A and M, are carriage control characters. They may be combined with any of the above values for "a." For example, specifying FBA gives fixed block records with ASA control characters. Specifying VBM gives variable blocked records with machine codes.

A8    ASA control characters
M9    Machine codes

LRECL nnnnn
is the logical record length (nnnnn) of the file, in bytes. LRECL should not exceed 32760 bytes for fixed length records or 32756 for variable length records because of OS restrictions.

---

7  FB and VB should not be used with TERMINAL or READER devices.

8  A may be used with TERMINAL devices, but simulation is limited to the characters "blank" (single space), "0," (double space), "," (triple space), and "1" (page eject, simulated as double space). All other control characters are treated as a single space.

9  M should not be used with TERMINAL devices.
**FILEDEF**

**BLOCK nnnnn**

**BLKSIZE nnnnn**

is the logical block size (nnnnn) of the file, in bytes. BLOCK should not exceed 32760 bytes because of OS restrictions. If both BLOCK and BLKSIZE options are specified, the value of nnnnn for BLOCK is used and BLKSIZE is ignored.

**KEYLEN nnn**

is the size (nnn) of the key (in bytes). The maximum value accepted is 256.

**XTENT nnnnn**

is the number of records (nnnnn) in the extent for the file. The default is 50. The maximum value is 16,777,215.

**LIMCT nnn**

is the maximum number of extra tracks or blocks (nnn) to be searched. The maximum value is 256.

**OPTCD a**

is the direct access search processing desired. The variable “a” may be any combination of up to three of the following: (A and R are mutually exclusive.)

<table>
<thead>
<tr>
<th>Code</th>
<th>DASD Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Actual device addressing</td>
</tr>
<tr>
<td>E</td>
<td>Extended search</td>
</tr>
<tr>
<td>F</td>
<td>Feedback addressing</td>
</tr>
<tr>
<td>R</td>
<td>Relative block addressing</td>
</tr>
</tbody>
</table>

**OPTCD J**

is valid only for the Printer. When the virtual printer is a 3800, ‘J’ indicates to QSAM and BSAM that the output line contains a TRC (Table Reference Character) byte.

**Note:** The KEYLEN, XTENT, LIMCT, and OPTCD options should only be used with BDAM, QSAM, or BSAM files.

**DISP MOD**

positions the read/write pointer after the last record in the disk file. This option should only be used for adding records to the end of a file. When adding records to the end of a file, the file must be on a disk accessed as read/write. If a disk is an extension of another disk, the extension is automatically read/only and you cannot write to it. DISP MOD may be used to add records to the end of the tape file only for standard label tapes.

**MEMBER membername**

allows you to specify the name of a member of an OS partitioned data set; membername is the name of the PDS member.
CONCAT
allows you to assign the same ddname to two or more OS libraries so that you can refer to them in a single GLOBAL command. You may concatenate libraries with filetypes of MACLIB and LOADLIB.

You cannot issue multiple FILEDEF commands with the CONCAT option for LOADLIBS and MACLIBs that have the same filenames but are on different disks. Only the information for the last FILEDEF is retained.

Any file format options you specify in the first FILEDEF command line remain in effect for subsequently concatenated libraries. For a detailed description of concatenated macro libraries, see VM/SP CMS for System Programming.

DSORG \{ PS \  
  PO \  
  DA \}

is the data set organization: physical sequential (PS), partitioned (PO), or direct access (DA).

\{ 7TRACK  
  9TRACK  
  18TRACK \}

is the tape setting. The tape device mode is not checked or set by filedef. Use the TAPE command MODESET option to set the mode of a tape.

TRTCH \(a\)

is the tape recording technique for 7-track tapes. Use the following chart to determine the value of “\(a\)” for 7-track tapes.

<table>
<thead>
<tr>
<th>(a)</th>
<th>Parity</th>
<th>Converter</th>
<th>Translator</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>odd</td>
<td>off</td>
<td>off</td>
</tr>
<tr>
<td>OC</td>
<td>odd</td>
<td>on</td>
<td>off</td>
</tr>
<tr>
<td>OT</td>
<td>odd</td>
<td>off</td>
<td>on</td>
</tr>
<tr>
<td>E</td>
<td>even</td>
<td>off</td>
<td>off</td>
</tr>
<tr>
<td>ET</td>
<td>even</td>
<td>off</td>
<td>on</td>
</tr>
</tbody>
</table>

The default value of TRTCH is OC.

DEN \(den\)

is tape density: \(den\) can be 200, 556, 800, 1600, 6250, or 38K BPI (bytes per inch). If 200 or 556 are specified, 7TRACK is assumed. If 800, 1600, or 6250 are specified 9TRACK is assumed. If 38K is specified, 18TRACK is assumed. If the DEN option is not specified for dual-density 1600/6250 BPI tape drives, density will be forced to 6250 BPI whether or not a TAPE command has been issued. For dual-density 800/1600 tape drives, the density defaults to 1600 BPI. The following densities are allowed for the given track sizes.
FILEDEF

7track
   200, 556, 800

9track
   800, 1600, 6250

18track
   38K

UPCASE
   translates all terminal input data to uppercase.

LOWCASE
   retains all terminal input data as typed in.

LEAVE
   is only valid for TAPn files that are SUL or SL (standard label). With
   this option selected, the tape is not moved before label processing. If
   LEAVE is not specified, tapes with files specified as SL or SUL are
   rewound and then positioned before the files are processed.

NOEOV
   is only valid for TAPn files. With NOEOV selected, there is no
   automatic limited end-of-volume processing when end of tape is sensed
   on output. Under OS simulation for standard labelled tapes, if
   NOEOV is specified, it is ignored during end-of-volume processing.
   See the section “CMS Tape Label Processing” in the VM/SP CMS

ALT \{ TAPn \}
   cuu
   is only valid for TAPn files that are SL (standard label). This option
   specifies an alternate tape drive to be used when an EOV condition
   occurs on the primary tape drive. It allows the next volume of a
   multi-volume tape file to be mounted on an alternate drive at the same
   time as the first volume. Processing switches between drives at each
   EOV condition and continues until an EOT condition is encountered.

   The TAPn specifies the symbolic tape identification. The cuu
   specifies the virtual device address of the tape. The following
   symbolic names and virtual device addresses are supported:
Both the primary and alternate tape drives must have compatible characteristics; that is, they must have an equal track setting and must record at the same density.

Usage Notes:

1. If you do not issue a FILEDEF command for an OS input or output file, CMS uses the ddname on the DCB macro to issue the following default file definition:

   FILEDEF ddname DISK FILE ddname A1

   See “Usage Notes” under the discussion of the ASSEMBLE command for information on the default file definitions made by the assembler.

2. To identify VSE files for VSE program execution or to identify VSAM data sets for either OS or VSE program execution, you must use the DLBL command.

3. A file definition established with the FILEDEF command remains in effect until explicitly changed or cleared. The system clears file definitions under the following circumstances:

   • When the assembler or any of the language processors are invoked. (Note that FILEDEF definitions entered with the PERM option are not cleared.)

   • When a program abends or when you issue the Immediate command HX to halt command or program execution.

4. The FILEDEF command does not supply default values for LRECL and BLKSIZE. As under OS, if DCB information is unavailable when a file is opened, an open error is issued for the file. The following chart summarizes the results at OPEN time of specifying LRECL and BLKSIZE options.
### Table: BLKSIZE and LRECL Results

<table>
<thead>
<tr>
<th>BLKSIZE Specified</th>
<th>LRECL Specified</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Specified</td>
<td>Not Specified</td>
<td>If the input file exists on disk, the item length (or item length + 4 for variable length records) becomes the BLKSIZE.</td>
</tr>
<tr>
<td>Specified</td>
<td>Not Specified</td>
<td>LRECL = BLKSIZE (or LRECL = BLKSIZE-4, for variable-length records).</td>
</tr>
<tr>
<td>Not Specified</td>
<td>Specified</td>
<td>BLKSIZE = LRECL (or BLKSIZE = LRECL + 4, for variable-length records).</td>
</tr>
<tr>
<td>Specified</td>
<td>Specified</td>
<td>The values specified are used.</td>
</tr>
</tbody>
</table>

If V or VB is specified for RECFM, LRECL must be at least 4 bytes less than BLKSIZE and LRECL must be at least 4 bytes greater than the largest record of the file. If VS or VBS is specified for RECFM, LRECL can exceed the specified BLKSIZE, but LRECL should not exceed a maximum value of 32756 because of OS restrictions.

VSE sequential (SAM) files do not contain BLKSIZE, LRECL, or RECFM specifications. These options must be specified by a FILEDEF command or DCB statement if OS macros are used to access VSE files. Otherwise the defaults, BLKSIZE = 32760 and RECFM = U, are assumed. LRECL is not used for RECFM = U files.

5. When copying a variable length data set (RECFM = V or VB) from an OS disk to a CMS disk, the logical record length (LRECL) of the file that is created on the CMS disk is equal to the size of the largest record in the data set being copied. If the file that is being created has a filemode of 4, the logical record length will be equal to the LRECL of the largest record plus 8 bytes. The actual LRECL of the new file can be determined by using the CMS LISTFILE command.

6. There is an auxiliary processing option for FILEDEF that is only valid when FILEDEF is executed by an internal program call: this option cannot be entered as a terminal command. The option, AUXPROC addr, allows an auxiliary processing routine to receive control during I/O operations. For details on how to use this option of the FILEDEF command, see the manual, VM/SP CMS for System Programming.

7. If a FILEDEF command is issued with a DDNAME that matches a current DDNAME defined by a previous FILEDEF command and the devices are the same, the filename, filetype, filemode, and options previously specified remain in effect, unless re-specified by the new FILEDEF command. If the devices are not the same, all previous specifications are removed.

8. CMS supports one virtual reader at address OOC, one virtual punch at address OOD, and one virtual printer at address OOE. When you issue a CMS command or execute a program that uses one of these unit record devices, the device must be attached at the virtual address indicated.
If a program has two or more data control blocks (DCBs) with different ddnames open for the same unit record devices, records from different files may be mixed together into one file. Separate files are not maintained.

9. To copy data between shared (e.g., 3420) and non-shared (e.g., 3480) tape subsystems, assign each tape device to a different virtual control unit. For example:

```
FILEDEF IN TAPO (18TRACK) *TAPO is assigned to 180
FILEDEF OUT TAP8 (9TRACK) *TAP8 is assigned to 288
MOVEFILE IN OUT
```

or...

```
FILEDEF IN TAPA (18TRACK) *TAPA is assigned to 28A
FILEDEF OUT TAP1 (9TRACK) *TAP1 is assigned to 181
MOVEFILE IN OUT
```

10. If the FILEDEF command is entered with no operands, a list of current definitions is displayed.

11. FILEDEF uses the extended plist for processing the DSN qual1 [qual2...qualn] parameter. If you are calling FILEDEF from an assembler language program and using DSN qual1 [qual2...qualn], you should supply an extended plist. VM/SP CMS for System Programming has more information on how an assembler language program can supply an extended plist.

**Using the FILEDEF DISK Operand**

There are three general forms for specifying the DISK operand in a FILEDEF command.

1. If you specify the first form:

```
FILEDEF ddname DISK fn ft [fm]
```

fn and ft (filename and filetype) are assumed to be a CMS fileid. If fm is the filemode of an OS disk, fn and ft are assumed to be the only two qualifiers of an OS data set name. If fm is specified as an asterisk, (*) then all accessed disks are searched.

You cannot use this form unless the OS data set name or VSE fileid conforms to the OS naming convention (1- to 8-byte qualifiers separated by periods, to a maximum of 44 characters, including periods). Also, the data set name can have only two qualifiers; otherwise, you must use the DSN ? or DSN qual1... form. For example, if the OS data set name or VSE fileid is TEST.SAMPLE.MAY, you enter:

```
FILEDEF MINE B1 DSN TEST SAMPLE MAY
```

-- or --

```
FILEDEF MINE B1 DSN TEST.SAMPLE.MAY
```
-- or --

FILEDEF MINE B1 DSN ?
TEST.SAMPLE.MAY

If the OS data set name or VSE fileid is TEST.SAMPLE, then you may enter:

FILEDEF MINE DISK TEST SAMPLE B1

2. The second form of the DISK operand is used only with OS data sets and VSE files:

FILEDEF ddname {DISK fn ft fm | DSN ?
FILE ddname | A1 | DSN qual1 [qual2...]}

This form allows you to enter OS and VSE file identifications that do not conform to OS data set naming conventions. The DSN operand corresponds to the DSN parameter on the OS DD (data definition) statement. There are many ways you can specify this form:

• FILEDEF ddname DISK fn ft fm DSN qual1 [qual2...]

-- or --

• FILEDEF ddname DISK fn ft fm DSN qual1 [.qual2...]

The above forms of the FILEDEF command associate the CMS filename and filetype you specify with the OS data set name or VSE fileid specified following the DSN operand. Once it is defined, you can refer to the OS data set name or VSE fileid by using the CMS filename and filetype. If you omit DISK, filename, filetype, and filemode, the default values are FILE ddname A1.

• FILEDEF ddname DSN ?

This form of the FILEDEF command allows you to specify the OS data set name or VSE fileid interactively. Using this form, you can enter an OS data set name or VSE fileid containing embedded special characters such as blanks. If you use this form, the default filename and filetype for your file, FILE ddname, is the CMS filename and filetype associated with the OS data set name or VSE fileid. The filemode for this form is always the default, A1.

To use the interactive DSN operand, you key in DSN ?; CMS then requests that you enter the OS data set name or DOS fileid exactly as it appears in the data set or file. Do not omit the periods that separate the qualifiers of an OS data set name, but do not insert periods where they do not appear.
FILEDEF

qual1[.qual2...]  
where qual1.qual2... are the qualifiers of the OS data set name or VSE fileid. When you use this form, you must code the periods separating the qualifiers.

- FILEDEF ddname mode DSN qual1 [qual2...]
  - or -

- FILEDEF ddname mode DSN qual1 [.qual2...]

This form allows you to specify the OS data set name or VSE fileid explicitly. The default value for the filename and filetype is FILE dname. When you use this form, you can use periods to separate the qualifiers. If the command is entered with a blank separating the qualifiers, FILEDEF replaces them with periods. For example, for an OS data set or VSE file named MY.FILE.IN, you enter:

FILEDEF ddname B1 DSN MY.FILE.IN
  - or -

FILEDEF ddname B1 DSN MY.FILE.IN

3. FILEDEF ddname DISK vaddr

This form associates a “ddname” with a virtual minidisk address. The “ddname” is the name of the virtual minidisk referred to in your program. It is intended to be used for a minidisk for which a RESERVE command has been issued. This form cannot be used as a regular FILEDEF for OS simulation. An open issued for such a dname would fail. The “vaddr” is the virtual address of the device referred to in your program by “ddname.”

Using the FILEDEF TAPn Operand

When you define a tape file with the FILEDEF command, you can specify the type of label processing to be done for the file. You do this by specifying a second operand after the word TAPn. The operands that you may specify and their meanings are:

LABOFF indicates that there is no CMS tape label processing for this tape file. LABOFF is the default. The tape is not positioned if this operand is specified.

BLP indicates that the system is to bypass label processing but that the tape is to be positioned before the file is processed.

SL indicates that you are using IBM standard labels.
SUL indicates that you are using standard user labels (not processed for MOVEFILE).

NL indicates that your tape has no IBM standard labels. (Do not use this operand if your tape has a VOL1 label. A file on it will not be opened.)

NSL indicates that you are using nonstandard labels.

For the operands BLP, SL, and SUL:

n indicates the position of the file on a multifile volume. When n is not specified, the default is 1.

For SL and SUL files:

**valid** specifies a 1- to 6-character volume serial number to be verified by reading the VOL1 label on the tape. If not specified in FILEDEF, valid may be specified on a LABELDEF command. If specified on both commands, the more recent specification is used. **VOLID** is only valid for SL or SUL tape files. If **VOLID** is not specified, the volume label on the tape is not checked.

For SL files:

**DISP MOD** The **DISP MOD** option may be used to add records to tape files only for standard label tapes:

```
FILEDEF file a tap1 sl (disp mod
```

when the file is opened (output), the tape is positioned at the end of the file, ready to add new records.

**SYSPARM** passes the address of the character value **string** to DMSTVI (an interface routine). The maximum length of the string is determined by the total length of command. The command cannot exceed 130 characters unless the command is issued from a program and an extended plist is used. In this case, the string can be up to 65,535 bytes. If longer, it is truncated. The string cannot contain blanks or parentheses.

If you want to enter a string with blanks or parentheses, use the **SYSPARM (?)** format. When you enter this format, you are prompted with

```
ENTER SYSPARM:
```

You can enter up to 130 characters. If longer, the string is truncated. You can omit the string's right parenthesis if **SYSPARM** is the last option specified.
For the NSL operand:

**filename** is required for NSL files. It is the filename of a file that contains a routine for processing nonstandard labels. The filename must be that of a TEXT or MODULE file. If you have both a MODULE and TEXT file with this name, the MODULE file is used. MODULE files must be created so that they start at an address that does not allow them to overlay a user program if they are to be used for NSL routines. See the section “Tape Labels in CMS” in the *VM/SP CMS User's Guide* for details on writing routines to process nonstandard labels.

Unless LRECL is specified on the output FILEDEF, the EOV2/EOF2 label contains the record length of the last record written for record format V or VB tape files. Files with a record format F or FB contain zero unless LRECL is specified. LRECL is important for OS compatibility. You can define a file on tap2 with standard labels by using the following command:

```plaintext
filedef filea tap2 sl valid dept10
```

When this tape file is opened, CMS checks to see that it has a VOL1 label with a volume serial number of dept10.

To specify the second file on the same tape, use

```plaintext
filedef filea tap2 sl 2 valid dept10
```

The same file could be defined as having no labels by using

```plaintext
filedef filea tap2 blp 2
filedef filea tap2 nl 2
```

If you use the above specification, your tape must not contain IBM standard labels. NL causes CMS to read your tape when you try to open a file on it and checks to see if the tape contains a VOL1 label as its first record. If a VOL1 label is there, CMS does not open your tape file.

If you specify

```plaintext
filedef filea tap2 blp 2
```

CMS positions the tape to the second file, but does not check to see if the tape has a label.

**Note:** If you mount a blank tape and specify NL, the tape will run off the end of the reel. Write a tape mark to prevent this from occurring.

To define a tape file with nonstandard labels, use the following command:

```plaintext
filedef filea tap2 nsl nonstd
```

The routine NONSTD must exist as a TEXT or MODULE file and be able to process the particular nonstandard labels you are using for your tapes.
If you defined filea with no label parameter at all, for example,

filedef filea tap2

there is no label processing or positioning before the data in filea is processed.

When you use the options DEN, TRTCH, 7TRACK, 9TRACK, or 18TRACK to set the mode of an output file, if the type of label processing is anything other than LABOFF (the default), the tape will be written at the current mode of the tape drive and not the mode specified in the FILEDEF command. This is due to a hardware restriction which allows the mode of a tape drive to be reset only when the tape is at load point. If LABOFF is used, the tape will still be at the load point when the first record of the file is written, so the mode will be reset. See the "CMS TAPE Command" Usage Notes for more information.

Read the section “Tape Labels in CMS” in the VM/SP CMS User’s Guide before you write programs that handle labeled tapes.

Use the LEAVE and NOEOV options for tape files only.

LEAVE indicates that a tape containing standard-label files is not to be moved before label processing. Using this option prevents CMS from rewinding the tape and checking the VOL1 label as it otherwise does for SL and SUL files. The command:

filedef fileb tap1 sl (leave

defines a tape file on tap1 but tells CMS not to position the tape before processing the labels for fileb. Note that you must position the tape properly yourself before using the LEAVE option. LEAVE may be used with SL, SUL, and BLP. However, it has no effect if used with NL. NL tapes are always rewound and positioned before a file on them is opened (even if you specify LEAVE).

Use the LEAVE option with multifile volumes where rewinding and repositioning a tape before processing each file is inefficient. You must not move the tape between files if you use this option. Note that for BLP files you can obtain the effect of LEAVE by defining the file as LABOFF rather than BLP.

Using NOEOV, CMS does not do any end-of-tape processing on output. If this option is not specified, CMS writes a tape mark after it encounters EOT on output and, for SL and SUL files, also writes an EOV1 label and another tape mark after the first tape mark. The tape is then rewound and unloaded. NOEOV suppresses this limited EOV processing. In OS simulation, if you specify the NOEOV option, it is ignored during end-of-volume processing.
Using the FILEDEF ALT Operand

Use the ALT option for standard label tape files only. You can use alternate tape drive support as follows:

filedef filea tap1 sl (alt tap2

where tap2 is the alternate drive where the second volume of the tape file is mounted.

Responses:

If FILEDEF is entered with no operands and there are no filedefs in effect, the message:

DMSFLD324I No user defined FILEDEFs in effect

is displayed.

If FILEDEF is entered with no operands and there are filedefs in effect, a list of current definitions is displayed. For example:

ddnamel devicel [filenamel filetype1 filemodel [datasetname]]
  . . . . . . .
ddnameN deviceN [filenamexN filetypeN filemodeN [datasetname]]

DMSFLD069I Disk mode not accessed

The specified disk is not accessed; the file definition remains in effect. You should access the disk before you attempt to read or write the file.

DMSFLD220R Enter data set name:

A FILEDEF command with the DSN ? operand was entered. Enter the exact OS or VSE file identification, including embedded periods and blanks.

DMSFLD704I Invalid CLEAR request

A CLEAR request was entered for a file definition that does not exist; no action is taken.

DMSSTT228I User labels bypassed on data set data set name

This message is displayed when you issue a FILEDEF command for an OS data set that contains user labels. The message is displayed the first time you issue the FILEDEF command after accessing the disk on which the data set resides.
Messages and Return Codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSFLD003E</td>
<td>Invalid option: <em>option</em> [RC = 24]</td>
</tr>
<tr>
<td>DMSFLD023E</td>
<td>No filetype specified [RC = 24]</td>
</tr>
<tr>
<td>DMSFLD027E</td>
<td>Invalid device <em>devtype</em> [for SYSaaa] [RC = 24]</td>
</tr>
<tr>
<td>DMSFLD029E</td>
<td>Invalid parameter <em>parameter</em> in the option <em>option</em> field [RC = 24]</td>
</tr>
<tr>
<td>DMSFLD035E</td>
<td>Invalid tape mode [RC = 24]</td>
</tr>
<tr>
<td>DMSFLD050E</td>
<td>Parameter missing after DDNAME [RC = 24]</td>
</tr>
<tr>
<td>DMSFLD065E</td>
<td><em>option</em> option specified twice [RC = 24]</td>
</tr>
<tr>
<td>DMSFLD066E</td>
<td><em>option1</em> and <em>option2</em> are conflicting options [RC = 24]</td>
</tr>
<tr>
<td>DMSFLD070E</td>
<td>Invalid parameter <em>parameter</em> [RC = 24]</td>
</tr>
<tr>
<td>DMSFLD109S</td>
<td>Virtual storage capacity exceeded [RC = 104]</td>
</tr>
<tr>
<td>DMSFLD221E</td>
<td>Invalid data set name [RC = 24]</td>
</tr>
<tr>
<td>DMSFLD224E</td>
<td>Fileid already in use [RC = 24]</td>
</tr>
<tr>
<td>DMSFLD420E</td>
<td>NSL exit filename missing or invalid [RC = 24]</td>
</tr>
<tr>
<td>DMSFLD699E</td>
<td>No filetype specified or <em>vdev</em> is an invalid disk address [RC = 24]</td>
</tr>
<tr>
<td>DMSFLD735E</td>
<td>Primary and alternate tape drives are identical. [RC = 24]</td>
</tr>
<tr>
<td>DMSFLO447E</td>
<td>Invalid sysparm information [RC = 24]</td>
</tr>
</tbody>
</table>
Use the FILELIST command to display a list of information about CMS files residing on accessed disks. In the FILELIST environment, information that is normally provided by the LISTFILE command (with the DATE option) is displayed under the control of the System Product Editor. You can use XEDIT subcommands to manipulate the list itself. You can also issue CMS commands against the files directly from the displayed list.

The format of the FILELIST command is:

```
```

*where:*

- **fn**
  is the filename of the file(s) for which information is to be collected. If an asterisk (*) is coded in this field, all filenames are used.

  Certain special characters can be used as part of the filename to request that the list contain a specific subset of files. See the usage note, “Pattern Matching,” for more information on using these characters.

- **ft**
  is the filetype of the file(s) for which information is to be collected. If an asterisk is coded in this field, all filetypes are used.

  Certain special characters can be used as part of the filetype to request that the list contain a specific subset of files. See the usage note, “Pattern Matching,” for more information on using these characters.

- **fm**
  is the filemode of the file(s) for which information is to be collected. If this field is omitted, only the A-disk is searched. If an asterisk is coded, all accessed disks are searched.

  If no operands are specified, the list contains all the files on your A-disk. (Issuing FILELIST with no operands is like issuing “filelist * * a.”)
FILELIST

Options:

Append
specifies that the list of files should be appended to the existing list. This option is meaningful only when issued from within the FILELIST environment. If issued outside of FILELIST, it results in an error condition.

Filelist
specifies that fn ft fm is a file that already contains a list of files, produced by an earlier invocation of FILELIST or LISTFILE (using the EXEC option). Information about each file in this list is displayed.

If this option is specified, no special characters used for pattern matching may appear in fn ft or fm. For information on pattern matching, see the usage note, “Pattern Matching,” below.

For information on creating and saving a list of files, see the usage note, “Saving a List of Files,” below.

Nofilelist
specifies that fn ft fm is not a list of files.

PROFile fn
specifies the name of an XEDIT macro to be executed when XEDIT is invoked by the FILELIST command. If not specified, a macro named PROFFLST XEDIT is invoked. For more information on the PROFFLST macro, see the usage note, “Default PF Key Settings,” below.

Usage Notes:

1. Tailoring the FILELIST Command Options
You can use the DEFAULTS command to set up options and/or override command defaults for FILELIST. However, the options you specify in the command line when entering the FILELIST command override those specified in the DEFAULTS command. This allows you to customize the defaults of the FILELIST command, yet override them when you desire. Refer to the DEFAULTS command description for more information.

2. Pattern Matching
You can use two special characters in the fn, ft, and fm operands to request that the list of files contain a specific subset of files. The special characters are * (asterisk) and % (percent), where:

* represents any number of character(s). As many asterisks as required can appear anywhere in a filename or filetype. However, the total number of characters, including the asterisks, may not exceed eight. (Only one asterisk may be used for a filemode.)
For example, if you enter:

```
filelist *d* *file*
```

you are requesting that the list contain all files on your A-disk whose filename contains “d” and whose filetype contains “file.” The list might contain the following files:

```
YOURDATA  AFILE1  A
HISDATA   AFILE2  A
ADOG      IDOGFILE A
DATA      FILE1   A
```

The wildcard character % means any single character, but any character will do. As many percent symbols as necessary may appear anywhere in a filename or filetype. For example, if you enter:

```
filelist %%% stock
```

you are requesting that the list contain all files on your A-disk whose filename is three characters in length and whose filetype is “stock.” The list might contain the following files:

```
THE STOCK A
HIS STOCK A
HER STOCK A
```

3. Format of the List

When you invoke the FILELIST command you are placed in the XEDIT environment, editing a file “userid FILELIST A0.” A sample FILELIST screen is shown in the “Examples” section. Each line in this file contains:

- a command area
- filename, filetype, filemode
- format and logical record length of the file
- number of records and number of blocks in the file
- date and time the file was last written on the disk

The full power of XEDIT is available to you while you issue commands against the list of files. For example, you may want to use XEDIT subcommands to scroll through the list of files, locate a particular file, etc.

However, some XEDIT subcommands are inappropriate in this environment. Subcommands that alter the format or the contents of “userid FILELIST” (for example, SET TRUNC, SET FTYPE, or SET LINEND) may cause unpredictable results.
4. Saving a List of Files

You can save a list of files created by the FILELIST command simply by filing it, that is, issuing FILE or SAVE from the command line. Remember that the list is a file, whose filename is your userid and whose filetype is FILELIST. If you issue FILE or SAVE, the file "userid FILELIST" is kept until the next time you issue FILE or SAVE from the list.

You can also save a particular list of files by filing it under a different fileid. One way to do this is to issue the XEDIT subcommand FILE from the command line, specifying a different filename and/or filetype. For example, you could issue "FILE MY FILES." Another way is to issue FILE from the command line, and then to use the CMS command RENAME.

Saving a list of files is useful when you want to send multiple files using the SENDFILE command. The list of files that you saved can be specified in the SENDFILE command issued with the FILELIST option. With this method, you can send multiple files by issuing the SENDFILE command only once. The only file identifier you have to keep track of is that of the list. For information on sending a list of files, see the SENDFILE command (the description of the Filelist option).

5. Issuing Commands From the List

On a full screen display, you can issue commands directly from the line on which a file is displayed. You do this by moving the cursor to the line that describes the file to be used by the command, typing the command in the space provided to the left of the filename, and then pressing the ENTER key to execute the command.

If a command is longer than the command space provided on the screen, just continue typing over the information in the line. You may type over the entire line displayed, up to column 79.

The ENTER key is set to EXECUTE, which is described below, under "Special Commands." When you press the ENTER key, all commands typed on one screen are executed, and the screen is restored to its previous state. However, the list is updated to reflect the current status of the files (see "Responses").

You may want to enter commands from the FILELIST command line before executing commands that are typed on the list. To do this, move the cursor to the command line by using the PF12 key (instead of the ENTER key). After typing a command on the command line and pressing ENTER, you can use PF12 to move the cursor back to its previous position on the list.

Another way to issue commands that make use of the files displayed is to issue EXECUTE from the FILELIST command line. A complete description of EXECUTE follows, in the section "Special Commands."
6. Default Key Settings

The PROFLST XEDIT macro is executed when the FILELIST command is invoked, unless you specified a different macro as an option in the FILELIST command. It sets the keys to the following values:

**ENTER**
Execute command(s) typed on file line(s) or on the command line. (The ENTER key is set by the XEDIT subcommand, SET ENTER IGNORE MACRO EXECUTE).

**PF 1 Help**
Display FILELIST command description.

**PF 2 Refresh**
Update the list to indicate new files, erased files, etc., using the same parameters as those specified when FILELIST was invoked.

**PF 3 Quit**
Exit from FILELIST.

**PF 4 Sort**
by filetype, filename.

**PF 5 Sort**
by date and time, newest to oldest.

**PF 6 Sort**
by size, largest first.

**PF 7 Backward**
Scroll back one screen.

**PF 8 Forward**
Scroll forward one screen.

**PF 9 Fl/n**
Issue the command FILELIST /n ** at the cursor, so that a list is displayed, containing all files that have the filename that is displayed on the line containing the cursor (all filetypes and filemodes).

**PF 10**
Not assigned.

**PF 11 XEDIT**
Edit the file where the cursor is placed.
PF 12 Cursor

If the cursor is in the file area, move it to the command line. If the cursor is on the command line, move it back to its previous location in the file (or to the current line).

Note: On a terminal equipped with 24 PF keys, PF keys 13 to 24 are assigned the same values as PF keys 1 to 12 as discussed here.

Some XEDIT subcommands are stacked by the FILELIST command (for example, SET TRUNC, SET LRECL, and SET VERIFY). In order to override these settings in a profile, these SET subcommands must be stacked FIFO.

In addition to setting the above PF keys, the PROFFLST XEDIT macro sets synonyms that you can use to sort your FILELIST files. The synonyms are:

SNAME Sorts the list alphabetically by filename, filetype, and filemode.

STYPE Sorts the list alphabetically by filetype, filename, and filemode.

SMODE Sorts the list by filemode, filename, and filetype.

SRECF Sorts the list by record format, filename, filetype, and filemode.

SLREC Sorts the list by logical record length and then by size (greatest to least).

SSIZE Sorts the list by number of blocks and number of records (greatest to least).

SDATE Sorts the list by year, month, day, and time (most recent to oldest).

7. If you want to issue FILELIST from an EXEC program, you should precede it with the EXEC command; that is, specify

execute filelist

Responses:

When a command is executed, one of the following symbols is displayed in the “Cmd” space to the left of the file for which the command was executed.

* Means the command was executed successfully (RC = 0).

*n Is the return code from the command executed (RC = n).
**?** Means that the command was an unknown CP/CMS command (RC = -3).

**!** Means that the command was not valid in CMS subset. For a list of commands valid in CMS subset mode, see the *VM/SP CMS User’s Guide*.

The following responses can also appear directly on the FILELIST screen:

*    fname  ftype  fmode  ** Not found. **
*    No files match the search criteria: fname ftype fmode
*    fname  ftype  fmode  ** Discarded or renamed **
*    fname  ftype  fmode  ** Fileid is in Mixed Case.
    Invalid for EXECUTE *
*    fname  ftype  fmode  has been discarded.
File  fname  ftype  fmode  has been discarded.

**Messages and Return Codes:**

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSWFL002E  File fn ft fm not found [RC = 28]
DMSWFL054E  Incomplete fileid specified [RC = 24]
DMSWFL651E  APPEND must be issued from RDRLIST or FILELIST
    [RC = 40]
DMSWFL680E  Invalid fileid specified with FILELIST option [RC = 20]

**Special Commands Used in the FILELIST Environment**

Two commands, EXECUTE and DISCARD, make use of the list of files displayed by the FILELIST command. EXECUTE can be used only in the FILELIST, MACLIST, and RDRLIST command environments, while DISCARD can be used only in the FILELIST, MACLIST, RDRLIST, and PEEK command environments.

**EXECUTE**

Use EXECUTE (an XEDIT macro) to issue CP/CMS commands (or EXECs) that make use of files displayed by FILELIST.

EXECUTE may be used in two ways. First, on a display terminal, the command(s) to be executed can be typed directly on the FILELIST screen and “EXECUTE” entered either on the command line or from a PF key or by pressing the ENTER key. Second, the command to be executed can be typed in the command line at the bottom of the screen, following “EXECUTE” (as one of its operands). The command is then executed against one or more files in the list, beginning with the current line of the list.
The format of the EXECUTE macro is:

```
EXECUTE [Cursor|lines] [command]
```

where:

Cursor
means that a command is to be executed against the line that contains
the cursor. The command can either be typed on the line that
describes the file, or it can be typed as an operand of EXECUTE. The
CURSOR operand is valid only on display terminals and is
particularly useful when assigned to a PF key. For example, if
EXECUTE CURSOR XEDIT is assigned to a PF key, you can place
the cursor on the line describing the file you want to edit and then
press the PF key.

lines
is the number of lines in the file the command is to be executed for,
starting with the current line of the list. If a command is specified,
the default is one (1). You can specify an asterisk (*), which means
“execute this command on all lines, from the current line to the end of
the file.”

command
is a CMS or CP command (or any program or EXEC) that makes use
of files in the list. You can either type out the command operands, or
you can use the symbols described below to represent the filename,
filetype, and/or filemode. (See the usage note, “Using Symbols as Part
of a Command.”)

Usage Notes:

1. Entering Commands on a Full Screen Display

You can type commands that operate on files in the list directly on the
lines of the FILELIST display. When you enter EXECUTE (either from
the command line or by pressing the ENTER key), all commands typed
on the lines in the file displayed on the current screen are executed.
The filename, filetype, and filemode are appended automatically to the
command, unless you typed one of the symbols described below (in
usage note 3).

Note that when a command is typed on the FILELIST or RDRLIST
screen, EXECUTE rebuilds the line and compares it with the line
displayed on the screen. The line is scanned from right to left, and the
first character that is different signals the end of the command.
Therefore, if the file information has been changed (as the result of a
previous command), but this information has not yet been updated (by
pressing PF2 to refresh the screen), EXECUTE will incorrectly
interpret the information on the screen. An example follows.
2. Entering Commands on the Command Line

Another way to issue commands that make use of the files displayed is to move the current line to the first (or only) file you want the command to use, and then to issue EXECUTE (in the form “EXECUTE lines command”) from the XEDIT command line. This method may be used on both display and typewriter terminals.

For example:

First move the current line (by using XEDIT subcommands like UP or DOWN) to the first file you want to use in the command. On a full screen display, the current line is the first file on the screen. Then (in the XEDIT command line) you type:

\[\text{execute n xedit}\]

where “n” is the number of files to be edited, starting with the current line. (You can use any command, not just XEDIT.)

\[\text{Note:}\ \text{You can use XEDIT synonyms or macros to make issuing common commands easier. For example, you might want to set up a command “EX” to be a synonym for “EXECUTE 1 XEDIT.”}\]

3. Using Symbols as Part of a Command

Symbols can be used to represent operands in the command to be executed. They can be used in the commands typed on the screen, or as part of the command in EXECUTE (on the command line). Symbols are needed if the command to be executed has operands or options that follow the fileid. Examples of using symbols are in the “Examples” section, below.
The following symbols can be used:

/ means the filename filetype filemode displayed on the line.

/n means the filename displayed on the line.

/t means the filetype displayed on the line.

/m means the filemode displayed on the line.

/o means execute the line as is, and omit appending anything.

Any combinations of symbols can be used. For example:

/n /t means: filename followed by filetype.

/nt means: filename followed by filetype.

/tn means: filetype followed by filename.

/ntm is equivalent to / alone.

/nnt means filename followed by filename and filetype.

Note: If the symbol '/' appears in a command or in its operands, it must be issued from the command line, and not as part of an EXECUTE command.

4. Special Symbols Used Alone

The following special symbols can be typed alone on the lines of the FILELIST display. They have the following meanings:

= means execute the previous command for this file. Commands are executed starting at the top of the screen. For example, suppose you enter the DISCARD command on the top line. You can then type an equal sign on any other line(s). Those files preceded by equal signs are discarded when the EXECUTE command is entered (from the command line or by pressing the ENTER key).

? means display the last command executed. The command is displayed on the line in which the ? is entered.

/ means make this line the current line. (On the FILELIST screen, the current line is the first file on the screen.)

5. Files with mixed-case fileids can only be manipulated by user-written EXECs and modules. Files will have to be renamed in order to be used by EXECUTE.
Messages and Return Codes for EXECUTE:

- DMSWEX526E Option CURSOR valid in display mode only [RC = 3]
- DMSWEX543E Invalid number: number [RC = 5]
- DMSWEX561E Cursor is not on a valid data field [RC = 1 or 3]
- DMSWEX651E EXECUTE must be issued from RDRLIST, FILELIST or MACLIST [RC = 40]
- DMSWEX654E Invalid symbol symbol; {/O must be specified alone|invalid character char following / symbol} [RC = 24]

On a typewriter terminal only:

Executing: command
+++E(nn)+++ 

DISCARD

Use the DISCARD command to erase from disk a file that is displayed in the list. (DISCARD is equivalent to the CMS command ERASE.) DISCARD can either be typed in the command area of the line that describes the file you want discarded, or it can be entered from the command line (at the bottom of the screen).

The format of the DISCARD command as used in the FILELIST environment is:

| DISCARD | [fn ft fm] |

where:

fn ft fm

is the file identifier of the file to be erased. If DISCARD is typed on the line that describes the file to be discarded, no file identifier should be specified. The filename, filetype, and filemode are appended automatically.

Messages and Return Codes for DISCARD:

The possible error messages for specifying a command format incorrectly are listed on page 24.

- DMSWDC651E DISCARD must be issued from FILELIST, RDRLIST, MACLIST or PEEK [RC = 40]
- DMSWDC653E Error executing command; rc = nn [RC = nn]
Examples:

The following FILELIST screen was created by issuing the FILELIST command with no operands, which is equivalent to FILELIST ** A. Note that the files are sorted by date and time, newest to oldest.

![Sample FILELIST Screen](image)

Figure 8. Sample FILELIST Screen

**Examples of Using Symbols:** The following examples show how symbols can be used to represent operands in a command. The values substituted for the symbols and the resulting command are shown. In each case, the command can be entered in either of the following ways:

- typed in the “Cmd” area of the screen. The command is executed either by entering EXECUTE on the XEDIT command line and then pressing ENTER, or simply by pressing ENTER.

- entered from the XEDIT command line, as an operand of EXECUTE (in the form “EXECUTE lines command”).

If a symbol is not specified, the filename, filetype, and filemode are appended automatically to the command.

<table>
<thead>
<tr>
<th>FILEID</th>
<th>COMMAND</th>
<th>RESULTING COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>pizza toppings a</td>
<td>discard</td>
<td>discard pizza toppings a</td>
</tr>
<tr>
<td>cookie assemble a</td>
<td>assemble /n</td>
<td>assemble cookie</td>
</tr>
<tr>
<td>jelly beans a</td>
<td>copy /= flavors=</td>
<td>copy jelly beans a jelly flavors a</td>
</tr>
<tr>
<td>dieting tips a</td>
<td>copy /=nt b</td>
<td>copy dieting tips a dieting tips b</td>
</tr>
</tbody>
</table>

Chapter 2. CMS Commands 223
FINIS

Use the FINIS command to close one or more files.

The format of the FINIS command is:

```
FINIS        fn  ft  [fm]
          *  *  [*]
```

where:

fn
is the filename of the file to be closed. If you code an asterisk (*) in
this field, all filenames are closed.

ft
is the filetype of the file to be closed. If you code an asterisk (*) in
this field, all filetypes are closed.

fm
is the filemode of the file to be closed. If you code an asterisk (*) in
this field, all disks are searched for the specified file. If this field is
omitted, A1 is assumed.

Usage Note:

Use FINIS when your program does not close a file during its execution.
CMS commands close files automatically at the end of their execution. (An
"EXEC" file is considered to be a single CMS command, independent of its
content.)

Responses:

None.

Messages and Return Codes:

If an error occurs, register 15 contains the following error code:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>File not open</td>
</tr>
</tbody>
</table>
Use the FORMAT command to:

- Initialize a virtual disk (minidisk) for use with CMS files
- Count or reset the number of cylinders on a virtual disk
- Write a label on a virtual disk

The format of the FORMAT command is:

```
FORMAT cuu mode [nocy] [oblk] [(options...[ ])]
```

**Options:**

<table>
<thead>
<tr>
<th>Blksizes</th>
<th>512</th>
<th>800</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>1K</th>
<th>2K</th>
<th>4K</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Noerase]</td>
<td>[Label]</td>
<td>[Recomp]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**where:**

- **cuu**
  - is the virtual device address of the virtual disk to be formatted.
  - Valid addresses are 001 through 5FF for a virtual machine in basic control mode and 001 through FFF for a virtual machine in extended control mode.

- **mode**
  - is the filemode letter to be assigned to the specified device address.
  - Valid filemode letters are A through Z. This field must be specified. If any other disk is accessed at this mode, it is released.

- **nocy**
  - is the number of cylinders to be made available for use. All available cylinders on the disk are used if the number specified exceeds the actual number available.
FORMAT

\textit{noblk} is the number of FB-512 blocks to be made available for use. If the number specified exceeds the actual number of blocks on the disk, then all the blocks on the disk are made available for use.

\textbf{Options:}

\textbf{Blksize} specifies the physical DASD block size of the CMS minidisk. The block sizes 1024, 2048, and 4096 may alternately be specified as 1K, 2K, and 4K, respectively. For FB-512 devices, only block sizes 512, 1024, 2048, and 4096 are supported; for CKD (count key data) devices, all block sizes are supported.

The \texttt{BLKSIZE} option defaults to a block size that optimizes the I/O and data storage for the particular device. CKD devices default as follows:

<table>
<thead>
<tr>
<th>DASD</th>
<th>Default Blocksize</th>
</tr>
</thead>
<tbody>
<tr>
<td>2314</td>
<td>1024</td>
</tr>
<tr>
<td>3340</td>
<td>1024</td>
</tr>
<tr>
<td>3330</td>
<td>2048</td>
</tr>
<tr>
<td>3350</td>
<td>2048</td>
</tr>
<tr>
<td>3375</td>
<td>4096</td>
</tr>
<tr>
<td>3380</td>
<td>4096</td>
</tr>
</tbody>
</table>

FB-512 devices, such as 3310 and 3370, default to a block size of 1024. For more information on choosing an appropriate blocksize, see Usage Note 7 on page 228.

\textbf{Noerase} specifies for FB-512 devices that the permanently formatted FB-512 blocks are not to be cleared to zeros. If not specified, the FB-512 blocks will be cleared. For non-FB-512 devices, this option is ignored.

\textbf{Label} writes a label on the disk without formatting the disk. The CMS disk label is written on cylinder 0, track 0, record 3 of the virtual disk or block 1 of an FB-512 device. A prompting message requests a six-character disk label (fewer than six characters are left-justified and blanks padded).

\textbf{Recomp} changes the number of cylinders or FB-512 blocks on the disk that are available to the user. This number becomes the actual number of minidisk cylinders or FB-512 blocks, or the number specified by \texttt{nocyl/noblk}, whichever is less. If \texttt{nocyl} is not specified and the disk is formatted in 800-byte blocks, all cylinders are used. If the disk is formatted in 512-, 1K-, 2K-, or 4K-byte blocks, the maximum number of cylinders or FB-512 blocks last formatted on the disk is made available to the user.
Usage Notes:

1. You can use the FORMAT command with any virtual 3310, 3330, 3340, 3350, 3370, 3375, 3380, or 2314 device. The speed matching buffer feature (Feature #6550) for the 3380 supports the use of extended count-key-data channel programs.

   Note: The speed matching buffer is not supported for 3380 Models AD4/BD4 or AE4/BE4.

2. When you do not specify either the RECOMP or LABEL option, the disk area is initialized by writing a device-dependent number of records (containing binary zeros) on each track. Any previous data on the disk is erased. A read after write check is made as the disk is formatted. For example:

   format 191 a 25

   initializes 25 cylinders of the disk located at virtual address 191 in CMS format. The command:

   format 192 b 25 (recomp)

   changes the number of cylinders available at virtual address 192 to 25 cylinders, but does not erase any existing CMS files. To change only the label on a disk, you can enter:

   format 193 c (label)

   Respond to the prompting message with a six-character label.

3. If you want to format a minidisk for VSAM files, you must use the Device Support Facility. If you want to format an entire disk, you may use any OS or DOS disk initialization program.

4. Because the FORMAT command requires heavy processor utilization and is heavily I/O bound, system performance may be degraded if there are many users on the system when you use FORMAT.

5. When formatting FB-512 devices, enough blocks of the minidisk area must be formatted to support the CMS disk structure, or message DMS216E will be displayed, and the FORMAT request will be terminated. The number of FB-512 blocks which must be formatted for minidisks of 512-, 1K-, 2K-, and 4K-byte CMS blocksize is 6, 12, 24, and 48, respectively.

6. If the FORMAT command with the RECOMP option fails and CMS issues message DMSFOR214W, "CANNOT RECOMPUTE WITHOUT LOSS OF DATA. NO CHANGE.," query your A-disk to determine the number of unallocated cylinders. If the number of cylinders seems adequate, it is possible that some of the allocated space is at the end of the disk, and is thus not available to the FORMAT command. Issue the command:
COPY * * A == (REP

followed by the FORMAT command with the RECOMP option.

7. Choosing an appropriate BLKSIZE to format a disk depends upon its intended use. A 4K BLKSIZE will optimize the I/O if the disk is to contain large files with no missing records (dense). A BLKSIZE of 1K is more appropriate when creating many small files or sparse files. For example, PL/I regional files are sparse and they may allocate more space on a 4K disk than on a 1K disk, therefore, the smaller BLKSIZE is preferable.

The larger the block size of the disk, the greater the amount of storage required for input/output buffers. Each buffer that the system needs must be a contiguous block of system keyed storage. The size of this area of storage being the block size of the disk. Programs that dynamically allocate storage based upon machine size may use up all of the available storage. This may not allow the system enough storage to allocate buffers for its use. Consequently, a program needing a 4K disk that uses all of the available storage may be unable to get I/O buffers if they are not already allocated. For more information on CMS storage management, refer to the VM/SP System Logic and Problem Determination Guide Volume 2 - CMS.

8. Because the CMS file system uses a five level tree structure when using the 512-byte block size, the maximum number of data blocks for a variable format file is about 15 times less than the actual limit (2 to the power of 31 minus 1).

9. A CMS nucleus cannot be saved on a CKD device formatted with 512-byte block size.

Responses:

DMSFOR603R Format will erase all files on disk mode(vdev). Do you wish to continue? Enter 1 (YES) or 0 (NO).

To reply yes, enter 1 or 'YES'. To reply no, enter 0 or 'NO'. If you respond 'YES', you must only enter the character string 'YES'. You have indicated that a disk area is to be initialized; all existing files are erased. If the character string contains leading or trailing blanks, such as 'YES' or 'YES ', the response is processed as a 'NO' response. Responding 'NO', pressing the ENTER key, or entering a character string other than 'YES' cancels execution of the FORMAT command.
DMSFOR605R Enter disk label:

You have requested that a label be written on the disk. Enter a one- to six-character label.

DMSFOR705I Disk remains unchanged

The response to message DMSFOR603R was other than 'YES'.

DMSFOR732I nnnn {cylinders|FB-512 blocks} formatted on mode(vdev)

The format operation is complete.

DMSFOR733I Formatting disk mode

The disk represented by mode letter 'mode' is being formatted.

LABEL CUU M STAT CYL TYPE BLKSIZE FILES BLKS USED-(%) BLKS LEFT BLK TOTAL
label cuu m R/W nnnn type blkszize nnnnn nnnn- % nnn nnnnnnn

This message provides the status of a disk when you use the RECOMP option. The response is the same as when you issue the QUERY command with the DISK operand.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSFOR003E Invalid option: option [RC = 24]
DMSFOR005E No BLKSIZE specified [RC = 24]
DMSFOR017E Invalid device address vdev [RC = 24]
DMSFOR028E No device specified [RC = 24]
DMSFOR037E Disk mode(vdev) is accessed as read/only [RC = 36]
DMSFOR048E Invalid mode mode [RC = 24]
DMSFOR069E Disk mode not accessed [RC = 36]
DMSFOR070E Invalid parameter parameter [RC = 24]
DMSFOR113S Device vdev not attached [RC = 100]
DMSFOR114S Device vdev is an unsupported device type, or requested
BLKSIZE is not supported for the device [RC = 88]
DMSFOR125S Permanent unit check on disk mode(vdev) [RC = 100]
DMSFOR126S Error {reading|writing} label on disk mode(vdev) [RC = 100]
DMSFOR214W Cannot recompute without loss of data; no change [RC = 8]
DMSFOR216E Insufficient blocks on disk to support CMS disk structure [RC = 100]
Use the GENDIRT command to fill in a CMS auxiliary directory. The auxiliary directory contains the name and location of modules that would otherwise significantly increase the size of the resident directory, thus increasing search time and storage requirements. By using GENDIRT to fill in an auxiliary directory, the file entries for the given command are loaded only when the command is invoked.

The format of the GENDIRT command is:

```
GENDIRT  directoryname  [targetmode|sourcemode]
```

where:

directoryname
  is the entry point of the auxiliary directory.

targetmode
  is the filemode letter of the disk containing the modules referred to in the directory. The letter is the filemode of the disk containing the modules at execution time, not the filemode of the disk at creation of the directory. At directory creation time, all modules named in the directory being created must be on either the A-disk or a read-only extension; that is, not all disks are searched. The default value for targetmode is S (system disk). It is your responsibility to determine the usefulness of this operand at your installation, and to inform all users whose programs are in auxiliary directories exactly what filemode to specify on the ACCESS command.

sourcemode
  is the mode of the disk that contains the modules or files when the GENDIRT command is issued. If not specified, 'A' is the default.

Note: For information on creating auxiliary directories and for further requirements for using the targetmode option, see the VM/SP CMS for System Programming.

Messages and Return Codes:

- DMSGND002W  File fn ft [fm] not found [RC = 4 or 28]
- DMSGND021E  Entry point name not found [RC = 40]
- DMSGND022E  No directory name specified [RC = 24 or 28]
- DMSGND070E  Invalid parameter parameter [RC = 24]
GENMOD

Use the GENMOD command to generate a (MODULE) file on a CMS disk.

The format of the GENMOD command is:

\[
\text{Genmod} \quad [fn \quad [\text{MODULE} \quad [fm \quad […]]] \quad \text{[options...[ ]]}]
\]

Options:

\[
\begin{align*}
\text{Options:} & \quad [\text{FROM } entry1] \quad [\text{TO } entry2] \\
\text{MAP} & \quad \text{NOMAP} \\
\text{STR} & \quad \text{NOSTR} \\
\text{OS} & \quad \text{DOS} \\
\text{SYSTEM} & \quad \text{ALL}
\end{align*}
\]

where:

\[fn\]

is the filename of the MODULE file being created. If \( fn \) is not specified, the file created has a filename equal to that of the first entry point in the LOAD MAP.

\[fm\]

is the filemode of the MODULE file being created. If \( fm \) is not specified, A1 is assumed.

Options:

If conflicting options are specified, the last one entered is used.

\textbf{FROM } entry1

specifies an entry point or a control section name that represents the starting virtual storage location from which the nonrelocatable copy is generated.

\textbf{TO } entry2

specifies an entry point or a control section name that represents the ending virtual storage location from which the nonrelocatable copy is generated.
**GENMOD**

**MAP**
copies system loader table entries for the generated module into a map record which is included at the end of the MODULE file. The record can contain as many as 3276 map entries. The MODMAP command can be issued to display the module map.

**NOMAP**
specifies that a module map is not to be contained in the MODULE file.

*Note:* If a module is generated with the NOMAP option, that module cannot later be loaded and started with the CMS LOADMOD and START commands. When NOMAP is specified, the information produced is not sufficient for the START command to execute properly. However, a module generated with the NOMAP option can later be invoked as a command; that is, it can be invoked if its filename is entered.

**STR**
invokes the CMS storage initialization routine when the MODULE file is subsequently loaded (using the LOADMOD command). This routine frees any storage remaining from a previous program. STR is the default setting if the MODULE is to be loaded at the beginning of available user storage.

If you have issued CMS SET RELPAGE ON, STR causes CMS storage initialization to release the remaining pages of storage.

*Note:* If a program running in the user area calls a transient routine that was generated with the STR option, the user area storage pointers will be reset. This reset condition could cause errors upon return to the original program (for example, when OS GETMAIN/FREEMAIN macros are issued in the user program).

**NOSTR**
indicates that, when the MODULE is loaded, free storage pointers are not reset for any storage currently in use. NOSTR is the default setting if the MODULE file is to be loaded at a location other than the default load address.

**OS**
indicates that the program may contain OS macros and, therefore, should be executed only when CMS/DOS is not active.

**DOS**
indicates that the program contains VSE macros; CMS/DOS must be active (that is, SET DOS ON must have been previously invoked) in order for this program to execute. (See Usage Note 2).
**ALL**

indicates that the program:

- Contains CMS macros and must be capable of running regardless of whether CMS/DOS is active or not
- Contains no VSE or OS macros
- Preserves and resets the DOS flag in the CMS nucleus
- Does its own setting of the DOS flags

*Note:* The ALL option is primarily for use by CMS system programmers. CMS system routines are aware of which environment is active and will preserve and reset the DOS flag in the CMS nucleus.

**SYSTEM**

indicates that when the MODULE file is subsequently loaded, it is to have a storage protect key of zero.

**Usage Notes:**

1. The GENMOD command is usually invoked following the LOAD command, and possibly the INCLUDE command. For example, the sequence:

   ```
   load myprog
   genmod testprog
   ```

   loads the file MYPROG TEXT into virtual storage and creates a nonrelocatable load module named TESTPROG MODULE. TESTPROG may now be invoked as a user-written command from the CMS environment.

2. The execution of MODULE files created from VSE programs is not supported and may give unpredictable results. GENMOD is intended for use with the LOAD command, not the FETCH command. Storage initialization for FETCH is different from that for LOAD.

3. Before the file is written, undefined symbols are set to location zero and the common reference control section is initialized. The undefined symbols are not retained as unresolved symbols in the MODULE file. Therefore, once the MODULE file is generated, those references cannot be resolved and may cause unpredictable results during execution.

4. If you load a program into the user area you should issue the GENMOD command with the STR option. Be careful if the program uses OS GETMAIN or FREEMAIN macros because your program, plus the amount of storage obtained via GETMAIN, cannot exceed two pages (8192 bytes). It is recommended that you do not use GETMAIN macros in programs that execute in the user area.
5. A transient module (loaded with the ORIGIN TRANS option) that was generated with the SYSTEM option is written on disk as a fixed-length record with a maximum length of 8192 bytes. The relocation and history information for these files cannot be saved.

6. If you are using FORTRAN under CMS, compiling with the RENT option, and have named the main program the same as fn, then you must use the FROM option specifying entry1 the same as fn, preceded by an “at” sign (@). For example, you would enter:

```
genmod mnprog (from @mnprog
```

7. If FROM is not specified on the GENMOD command, the starting virtual storage location (entry point) of the module is either the address of fn (if it is an external name) or the entry point determined according to the hierarchy discussed in Usage Note 4 of the LOAD command. This is not necessarily the lowest address loaded. If you have any external references before your START or CSECT instructions, you must specify the ‘FROM entry1’ operand on the GENMOD command to load your program properly.

8. If you are using PL/I under CMS, use FROM PLISTART as an option to avoid unpredictable results.

9. The GENMOD command can be used to create a relocatable CMS MODULE file. Relocation is performed only when using the NUCXLOAD command. When the GENMOD command generates a MODULE file, one record of this MODULE file can contain relocation information. This depends on the previous LOAD or INCLUDE command that was issued. If the RLDSAVE option was specified on the LOAD or INCLUDE command, the MODULE file may contain the relocation information. If the RLDSAVE option was not specified on the LOAD or INCLUDE command, the relocation information for the file cannot be saved. See the LOAD or INCLUDE commands for more information.

10. The FROM or TO options should not be specified when module relocation information is being saved (using RLDSAVE option of the LOAD and INCLUDE commands). If specified, the results will be unpredictable.

11. If the HIST option was specified on the LOAD or INCLUDE command, the module file may contain history information. See the LOAD or INCLUDE commands for more information.
Example:

To generate the module file for the MYMOD MODULE that has already been loaded into storage by the LOAD command, you would enter:

```
genmod mymod
```

Responses:

None.

Messages and Return Codes:

- **DMSMOD003E** Invalid option: option [RC = 24]
- **DMSMOD005E** No option specified [RC = 24]
- **DMSMOD018E** No load map available [RC = 40]
- **DMSMOD021E** Entry point name not found [RC = 40]
- **DMSMOD032E** Invalid filetype ft [RC = 24]
- **DMSMOD037E** Disk mode[(vdev)] is accessed as read/only [RC = 36]
- **DMSMOD040E** No files loaded [RC = 40]
- **DMSMOD069E** Disk mode not accessed [RC = 36]
- **DMSMOD070E** Invalid parameter parameter [RC = 24]
- **DMSMOD084E** Invalid use of FROM and TO options [RC = 24]
- **DMSMOD105S** Error nn writing file fn ft fm on disk [RC = 100]
GENMSG

Use the GENMSG command to convert a message repository file into an internal form. Each record is read from the input file, the syntax is checked, and it is placed in an output file in a form the message processor can use.

The format of the GENMSG command is:

<table>
<thead>
<tr>
<th>GENMSG</th>
<th>fn ft fm applid [langid] [(options ... [ ]) ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Options:</td>
<td></td>
</tr>
<tr>
<td>[CP]</td>
<td>DbcD DbcD List Nolist Xref NOxref</td>
</tr>
<tr>
<td>[Object]</td>
<td>[Margin nn] [Margin 72]</td>
</tr>
</tbody>
</table>

**where:**

*fn ft fm*  
is the fileid of the external repository to be converted. This file must be in fixed-record format, and have a LRECL of 80.

*applid*  
specifies the application for which this repository is intended. This application identifier is 3 characters long.

*langid*  
specifies the language whose repository should be used. The *langid* is 1 to 5 characters long. The default value is the language currently in use.

**Options:**

*CP*  
notes that the input file contains messages for CP. The output repository file is modified accordingly.
**Dbcs**  
specifies that the input file contains Double-Byte Character Set (DBCS) characters as part of its message text.

**NODbcs**  
specifies that the input file does not contain Double-Byte Character Set (DBCS) characters.

**List**  
creates a listing file from the compilation. This file has the same filename as the input file, and a filetype of LISTING.

**NOList**  
does not create a listing file from the compilation.

**Xref**  
creates a cross-reference of message text and offset within the repository and place it in the listing file.

**NOXref**  
does not create a cross-reference.

**Object**  
creates an object deck (repository in internal form). This deck has the same file name as the input file, and a file type of TXT followed by the language name.

**NOObject**  
does not create an object deck.

**Margin nn**  
shows that message source is taken from columns 1-nn in the input deck. The default is MARGIN 72.

---

**Usage Notes:**

1. To learn how to make your own message repository refer to the publication, *VM/SP CMS for System Programming*.

2. If you wish to change a given CMS message, build your own message repository and include the number of the CMS message that you want to override. Load your repository as a user repository using the SET LANGUAGE command. See the example below.

   Unpredictable results may occur if the source message repository for CMS (DMSMES REPOS) is altered, recompiled, and used to build CMS.

3. GENMSG creates two output files:
Example:

Assume you are working in an English language application called MYAPPL1 (applid = MYA). You already have created a small message repository for MYAPPL1, and it has a fileid MYOWN MESSAGES A. This repository contains a message with two formats and a message with five formats. To compile your repository, you enter:

```
genmsg myown messages a mya (margin 63
```
The following LISTING file is created by the message compiler:

<table>
<thead>
<tr>
<th>GENMSG</th>
<th>Version 1 Release 1.0</th>
<th>Page 1</th>
<th>Time 17:48:50 Date 85.248</th>
</tr>
</thead>
</table>

Options used: MARGIN 63
Substitution character is &
Number of message number characters to display is 3

<table>
<thead>
<tr>
<th>GENMSG</th>
<th>Version 1 Release 1.0</th>
<th>Page 2</th>
<th>Time 17:48:50 Date 85.248</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>MESSAGE</th>
<th>NUM</th>
<th>FMT</th>
<th>LINE</th>
<th>SEV</th>
<th>TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0001</td>
<td>01</td>
<td>01</td>
<td>E</td>
<td>No filename specified</td>
</tr>
<tr>
<td></td>
<td>0001</td>
<td>02</td>
<td>01</td>
<td>E</td>
<td>No &amp;1 names specified</td>
</tr>
<tr>
<td></td>
<td>0002</td>
<td>01</td>
<td>01</td>
<td>E</td>
<td>File &amp;1 &amp;2 &amp;3 not found</td>
</tr>
<tr>
<td></td>
<td>0002</td>
<td>02</td>
<td>01</td>
<td>E</td>
<td>&amp;1 file &amp;2 not found</td>
</tr>
<tr>
<td></td>
<td>0002</td>
<td>03</td>
<td>01</td>
<td>E</td>
<td>Dataset not found</td>
</tr>
<tr>
<td></td>
<td>0002</td>
<td>04</td>
<td>01</td>
<td>E</td>
<td>File(s) &amp;1 not found</td>
</tr>
<tr>
<td></td>
<td>0002</td>
<td>05</td>
<td>01</td>
<td>E</td>
<td>Note &amp;1 not found</td>
</tr>
</tbody>
</table>

Total Messages Informational Warning Error Severe Terminating
2 2 0 0 0 0

The text deck is 000000F8 bytes in length
Return code was 0

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSMGC002E File fn ft fm not found [RC=28]
DMSMGC003E Invalid option: option [RC=16]
DMSMGC006E No read/write disk accessed [RC=36]
DMSMGC029E Invalid parameter parameter in the option option field [RC=24]
DMSMGC049E Invalid line number nn [RC=8]
DMSMGC054E Incomplete fileid specified [RC=24]
DMSMGC065E option option specified twice [RC=24]
DMSMGC066E option1 and option2 are conflicting options [RC=24]
DMSMGC073E Unable to open file ddname [RC=16]
DMSMGC104S Error nn reading file fn ft fm from disk [RC=100]
DMSMGC105S Error nn reading file fn ft fm to disk [RC=100]
DMSMGC109S Virtual storage capacity exceeded [RC=104]
DMSMGC147E Message not in ascending sequence [RC=8]
DMSMGC580E Invalid string : unmatched shift-out (SO) and shift-in (SI) [RC=5]
DMSMGC580E  Invalid string: invalid double-byte character(s)  [RC=5]
DMSMGC766I  Substitution character is char
DMSMGC767I  Number of message number characters to display is nn
DMSMGC768W  Invalid substitution character value char [RC=4]
DMSMGC769W  Invalid number of message characters value value [RC=4]
DMSMGC770E  Invalid application ID applid  [RC=16]
DMSMGC771E  Invalid message number  [RC=8]
DMSMGC772E  Invalid format number  [RC=8]
DMSMGC773E  Duplicate message ID id  [RC=4]
DMSMGC774E  Line numbers for messages are not consecutive  [RC=8]
DMSMGC775W  Text too long - 240 characters is the maximum allowed  [RC=4]
DMSMGC776I  Options used: list
GET VSCREEN

Use the GET VSCREEN command to write lines from a CMS file to the specified virtual screen.

The format of the GET VSCREEN command is:

```
GET VSCREEN  [vname fn ft [fm [fromrec [numrec ]]]]
```

**where:**

- **vname**
  is the name of the virtual screen to be updated with the data in the specified CMS file.

- **fn**
  is the filename of the file.

- **ft**
  is the filetype of the file.

- **fm**
  is the filemode of the file. If you do not specify *fm* or if * is specified, then all accessed disks are searched.

- **fromrec**
  is the starting record of the file that is moved into the virtual screen. GET VSCREEN starts with the first record in the file by default.

- **numrec**
  is the number of records that are read from the file and moved into the virtual screen. All the records are read by default. An * means that records are processed until the end of the file is reached.

**Usage Notes:**

GET VSCREEN queues each record of a CMS file to the virtual screen. The information is appended to the end of the virtual screen when the virtual screen is updated. See WRITE VSCREEN for more information on queues.
GET VSCREEN

Responses:

None.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSGET069E Disk mode not accessed [RC = 36]
DMSGET386E Missing operand(s) [RC = 24]
DMSGET388E Invalid keyword: keyword [RC = 24]
DMSGET389E Invalid operand: operand [RC = 24]
DMSGET391E Unexpected operand(s): operand [RC = 24]
DMSGET921E Virtual screen vname is not defined [RC = 28]
DMSSTT048E Invalid mode mode [RC = 24]
DMSSTT062E Invalid character char in fileid [fn ft fm] [RC = 20]
DMSWVL002E File fn ft fm not found [RC = 28]
DMSWVL104S Error nn reading file fn ft fm from disk [RC = 100]
DMSWVL109S Virtual storage capacity exceeded [RC = 104]
DMSWVL156E Record nnn not found -- file fn ft fm has only nnn records [RC = 32]
DMSWVL928E Command is not valid for virtual screen vname [RC = 12]
Use the GLOBAL command to identify which CMS, CMS/DOS, or OS libraries are to be searched for macros, copy files, subroutines, VSE executable phases, or OS load modules when processing subsequent CMS commands.

The format of the GLOBAL command is:

```
GLOBAL [MACLIB TXTLIB DOSLIB LOADLIB] libname1...libname63
```

where:

**MACLIB** precedes the specification of macro libraries that are to be searched for macros and copy files during the execution of language processor commands. The macro libraries may be CMS files or OS data sets. If you specify an OS data set, a FILEDEF command must be issued for the data set before you issue the GLOBAL command.

**TXTLIB** precedes the specification of text libraries to be searched for missing subroutines when the LOAD or INCLUDE command is issued, or when a dynamic load occurs (that is, when an OS SVC 8 is issued).

*Note:* Subroutines that are called by dynamic load should (1) contain only VCONs that are resolved within the same text library member or (2) be resident in storage throughout the processing of the original CMS LOAD or INCLUDE command. Otherwise, the entry point is unpredictable.

**DOSLIB** precedes the specification of DOS simulated core image libraries (that is, CMS/DOS phase libraries) to be searched for missing phases. This operand does not apply to system or private core image libraries residing on DOS disks. DOSLIB can be specified regardless of whether the CMS/DOS environment is active or not.

**LOADLIB** precedes the specification of load module libraries to be searched for a module that the OSRUN command or the LINK, LOAD, ATTACH, or XCTL macros refer to. The libraries can be CMS LOADLIBS or OS...
GLOBAL

module libraries. If you specify an OS data set, issue a FILEDEF command for the data set before you issue the GLOBAL command.

libname1...

are the filenames of up to 63 libraries of the specified filetype (MACLIB, TXTLIB, DOSLIB, or LOADLIB). The libraries are searched in the order in which they are named. The library list is subject to other system limits, such as command line length. This command supercedes any previous GLOBAL command for the specified filetype. If no filenames are specified, the command cancels any previous GLOBAL command for this filetype.

Usage Notes:

1. A GLOBAL command remains in effect for an entire CMS session unless it is explicitly canceled or reissued. If a program failure forces you to IPL CMS again, you must reissue the GLOBAL command.

2. There are no default libraries; if you wish to use the same libraries during every terminal session, place the GLOBAL command(s) in your PROFILE EXEC.

3. If you want to use an OS library during the execution of a language processor, you can issue a GLOBAL command to access the library, as long as you have defined the library via the FILEDEF command. If you want to use that library for more than one job, however, you should use the PERM option on the FILEDEF command, since the language processors clear nonpermanent file definitions.

4. To find out what libraries have been specified, issue the QUERY command with the MACLIB, TXTLIB, DOSLIB, LOADLIB, or LIBRARY operands. (The LIBRARY operand requests a display of all libraries.)

5. For information on creating and/or manipulating CMS libraries, see the discussion of the MACLIB, TXTLIB, DOSLIB, and LOADLIB commands.

Example:

To specify that you want the CMSLIB, OSMACRO, and OSMACROI macro libraries searched when processing CMS commands, you would enter:

global maclib cmslib osmacro osmacro1
Responses:

None.

Messages and Return Codes:

DMSG002W  File fn ft [fm] not found [RC = 4 or 28]
DMSG014E  Invalid function function [RC = 24]
DMSG047E  No function specified [RC = 24]
DMSG056E  File fn ft [fm] contains invalid record formats [RC = 32]
DMSG104S  Error nn reading file fn ft fm from disk [RC = 100]
DMSG108S  More than nn libraries specified [RC = 88]
DMSG109S  Virtual storage capacity exceeded [RC = 104]
DMSSTT062E Invalid character char in fileid fn ft [RC = 20]
The GLOBALV (GLOBAL Variables) command addresses two primary needs: 1) the need for several EXECs to share a common set of values; 2) the need to retain those values, either temporarily or permanently, for subsequent use.

Sharing:

Values are often given names, describing what they represent, for easy reference. Although the values often vary, their names usually do not. The GLOBALV command processor builds and maintains group(s) of named, variable values in free storage for shared use by EXECs. EXECs "share" a value by referring to it by a common name. When requested, GLOBALV retrieves a variable(s) from the group(s) and places it in the program stack for subsequent use by the requesting EXEC.

GLOBALV supports use of more than one group. This allows for grouping distinct variables that are either related or often used together, which facilitates both more efficient retrieval and more selective use. The "global variable group(s)," built by GLOBALV from a set of CMS GLOBALV type files on the user's A-disk and extensions, exist throughout an IPL, unless explicitly purged or re-initialized.

Retaining:

When variables are defined or changed, the user decides whether the variables or changes are to last:

1. For the current IPL only
2. Throughout an entire session (normally, from LOGON to LOGOFF)
3. Permanently, i.e. across sessions

Variables defined for the current IPL only are retained in storage. Those required longer than a single IPL are retained in CMS files on the user's A-disk from where they are put in storage. The CMS filenames are SESSION GLOBALV (for values required throughout the session), and LASTING GLOBALV (for values that are to last permanently). These two files and a third A-disk file (INITIAL GLOBALV) are the source from which the GLOBALV command processor creates and initializes the variable(s) in storage. The INITIAL file is normally created by the user as an alternative way of defining a large number of variables for an IPL.

The CMS GLOBALV disk files may be of fixed or variable format. Fixed format facilitates creation of files by users (via editing). It accommodates variables whose names and values do not exceed eight bytes each. The GLOBALV command processor uses variable format which allows for varying length variable names and values. In addition, variable format
includes a special field which, when used, identifies the group name into which the variable will be grouped. (See “Usage Note 1.”)

The GLOBALV command processor manages requests to define or set (SET...) variables both in storage and in the LASTING and SESSION GLOBALV files on the user's A-disk.

The format of the GLOBALV command is:

```
GLOBALV

INIT
SELECT [group
          UNNAMED] [group
          UNNAMED]

SET
SETS {name1 [value1] [name2 value2] ...}
SETL
SETLS
SETSL
SETLP
SETPL

LIST [name1 [name2 ]...]
STACK
STACKR {name1 [name2 ]...}

SELECT [group
          UNNAMED] [group
          UNNAMED] [group
          UNNAMED] [group
          UNNAMED]

PUT
PUTS {name1 [name2 ]...}
PUTP

GET [name1 [name2 ]...]
PURGE

Note: Although this command (except for the GET and PUT options) may be used in CMS EXECs, it is designed for use with System Product Interpreter or EXEC 2 EXECs. Restrictions and precautions are listed in “Usage Notes for CMS EXECs.”
```
GLOBALV

where:

INIT

allocates and initializes global variable(s) in free storage from data in the LASTING, SESSION, and INITIAL GLOBALV files on the issuer's A-disk and extensions. Not all files need be present. It also performs any needed cleanup (to eliminate multiple and null entries) in the LASTING GLOBALV file.

If the records in the GLOBALV file(s) contain no group name, for grouping the variables, (as with fixed format records) GLOBALV's INIT function allocates only one global variable group, UNNAMED, in free storage. Otherwise, (variable format) GLOBALV INIT will allocate as many unique global variable groups in free storage as identified in the GLOBALV files.

GLOBALV INIT initializes free storage with variables defined in the LASTING, SESSION, and INITIAL GLOBALV files respectively. If any variables are defined more than once within the LASTING file or within the SESSION file, the value defined last in the file is the one used to initialize storage. If a variable of the same name is defined in both the LASTING and SESSION files, the value assigned in the SESSION file will override the value assigned in the LASTING file when the storage is initialized. (See "Usage Notes 2 and 3.")

After initializing free-storage from the LASTING GLOBALV file, the file is re-written to eliminate multiple definitions for any variable names and any null (zero length) value assignments.

The global variable(s) in free storage are required by all other GLOBALV functions. Therefore, GLOBALV INIT is performed automatically if not explicitly requested prior to other GLOBALV requests.

SELECT

identifies the global variable group which is the subject of this or subsequent SELECT sub-functions. If no sub-function is specified, the GLOBALV command processor interprets the command as a request to set the default group for subsequent SELECT sub-functions. The default is set to the group indicated by "group" or to UNNAMED if no group is specified. A GLOBALV SELECT command that does specify a sub-function affects only the group specified in the command. It has no effect on setting or resetting the default group.

The SELECT phrase (SELECT group or SELECT UNNAMED) is optional preceding all forms of the SELECT sub-functions, SET, PUT, GET, LIST, and STACK. If the SELECT phrase is not used, the sub-function affects the default group, described above. (See "Examples" for uses of GLOBALV SELECT.)
SELECT Sub-functions:

SET
SET Sets
SETP

assigns the value “value1” to the variable “name1,” the value “value2” to the variable “name2,” etc. Since SET fields are delimited by blanks, the values cannot contain any blanks. (Use the SETL sub-function, described next, for such values.) If no value is specified, the value is assumed to be null.

SET adds the assignment(s) in the selected or default global variable group in storage. If the variable name is already defined, its value is replaced by the one specified in the command. SETS adds/replaces the assignment(s) in the selected or default group and appends it to the SESSION GLOBALV disk file. SETP adds/replaces the assignment(s) in the selected or default group and appends it to the LASTING GLOBALV disk file.

CMS EXEC users, refer to “Usage Notes for CMS EXECs.”

SETL
SETLS
SETSL
SETLP
SETPL

assigns the specified literal value, which may contain blanks, to the variable name. The first blank following the name delimits the name from the value field and is not part of the value. All characters following this blank (including any other blanks) are part of the value. If no value is specified, the value is assumed to be null.

SETL adds the assignment in the selected or default global variable group in storage. If the variable name is already defined, its value replaced by the one specified in the command. SETLS adds/replaces the assignment in the selected or default group and appends it to the SESSION GLOBALV disk file. SETSL is equivalent to SETLS. SETLP adds/replaces the assignment in the selected or default group and appends it to the LASTING GLOBALV disk file. SETPL is equivalent to SETLP.

CMS EXEC users, refer to “Usage Notes for CMS EXECs.”

LIST

displays a list of the specified variable name(s) and their associated value(s). If no name is specified, all variables in the selected or default group are listed.

STACK

places the value(s) associated with the specified variable name(s), from the selected or default group, LIFO in the program stack. When multiple variables are named in a single stack request, the values are stacked LIFO in the program stack such that the variable named first
GLOBALV

in the command is the first retrieved from the stack. Refer to Example 2 under "EXAMPLES." If a variable is not found in the group, a null (zero length) line is stacked. The command has no effect of the variable name is omitted.

This stacking technique requires that the System Product Interpreter EXEC issue a separate "PULL" or PARSE PULL control statement to read each value from the stack.

STACKR
places a "&READ n" control statement and, for each variable name specified, a "&name = &LITERAL OF value" assignment statement LIFO in the program stack such that "&READ n" is the first retrievable line. In the &READ control statement, "n" is the number of subsequent assignment statements and, in the assignment statement, "value" is the value associated with the specified variable name in the selected or default group. When multiple variables are named in a single STACKR request, the values are stacked LIFO in the program stack such that the "&READ n" is the first retrievable line from the stack, and the first named variable assignment statement is the next retrievable line from the stack, etc. Refer to Example 1 in the “Examples” section. The command has no effect if the variable name is omitted.

This stacking technique requires only a single &READ control statement to read all the variables named on the GLOBALV command from the stack. The STACKR option only applies to EXEC and EXEC 2 EXECs.

CMS EXEC users, refer to “Usage Notes for CMS EXECs.”

PUT
PUTS
PUTP
determines the value of the EXEC 2 or System Product Interpreter variable(s) specified in 'name1', 'name2', etc. and assigns that value as a global value in the selected or default global variable group. If a global value already exists with the name specified, it is replaced with the specified name's value.

PUT adds or replaces the assignment(s) in the selected or default global variable group in storage. PUTS does the same, but appends the group to the SESSION GLOBALV disk file. PUTP does the same, but appends the group to the LASTING GLOBALV disk file.

Note: The PUT and GET sub-functions can only be used from an EXEC 2 or a System Product Interpreter EXEC, and are subject to the rules of the interpreter being used. Refer to the usage notes for additional details.
GET
assigns values from the specified or default global variable group to
the specified EXEC 2 or System Product Interpreter variable names. If
no names are specified, GET does nothing.

Note: The PUT and GET sub-functions can only be used from an
EXEC 2 or a System Product Interpreter EXEC, and are subject to the
rules of the interpreter being used. Refer to the usage notes for
additional details.

PURGE
causes the variables in storage to be cleared. The group itself is not
purged.

CAUTION
If the SELECT phrase is not included with the
PURGE sub-function the result will be a GLOBALV
PURGE of all global variable(s) in storage.

GRPLIST
displays a list of all existing global variable groups.

GRPSTACK
places the names of all existing variable groups, line by line, in the
program stack such that these items will be the first retrievable from
the stack. A null line, used as a delimiter, indicates the end of the
stacked group names.

PURGE
causes all global variable(s) in free storage to be released.

Usage Notes:

1. The CMS GLOBALV disk files may be of fixed or variable format.
Fixed format records are 16 bytes in length and consist of two eight-byte
fields that contain a variable name, followed by its assigned value.
Variable format records may be up to 520 bytes in length and consist of
the following five fields (Because f1 and f2 can not be more than X'FF'
or 255, any variable name or variable value that exceeds this maximum
length will be truncated to its first 255 bytes.):

<table>
<thead>
<tr>
<th>group name</th>
<th>f1</th>
<th>variable name</th>
<th>f2</th>
<th>variable value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8</td>
<td>9</td>
<td>n</td>
<td>n+1</td>
</tr>
</tbody>
</table>

Note that “all” includes those groups created by use of the DEFAULTS and
EXECUTE commands.
GLOBALV

**group name**
identifies the group for grouping the variable (from GLOBALV [SELECT group\|UNNAMED] SET...).

**f1**
defines the actual length of the variable name field immediately following.

**variable name**
identifies the name by which this shared value will be commonly referenced.

**f2**
defines the actual length of the variable value field immediately following.

**variable value**
specifies the actual value assigned to the named variable.

Use fixed format when editing (creating or updating) files. Variable format records would be difficult to edit because changes in the variable name or variable value fields must also be reflected in their respective length fields. Although not impossible, this further editing is awkward and likely to be overlooked, increasing the chance of errors in those fields.

To establish the initial set of lasting variables, the user may edit them into a fixed format LASTING file. Note that whenever the GLOBALV command processor rewrites this file, during initialization, it will use variable format.

Probably the easiest way to create GLOBALV file(s) is to let the GLOBALV command processor do the work. Create an EXEC file containing the appropriate GLOBALV ... SETS and/or SETL commands. Then when the EXEC is invoked, the GLOBALV command processor will build the file(s) as it executes the commands.

2. The SESSION file is not erased by the GLOBALV command processor. This is the responsibility of the user. The length of a session is thus determined by the frequency with which the user erases the SESSION GLOBALV file. To make the duration of a session the time between CMS IPLs, the user might choose to include an ERASE SESSION GLOBALV command in the PROFILE EXEC. To make a session last for all IPLs of CMS during one day, erase the SESSION GLOBALV file whenever the date changes.

The SESSION GLOBALV file also is never cleaned up (to eliminate multiple and null entries) by the GLOBALV command processor, as the LASTING GLOBALV file is at each initialization. Without this automatic cleanup, the SESSION GLOBALV file continues to grow longer with each SETS and SETSL command.
GLOBALV

3. If the file is present during initialization of the global variable(s) in storage, its variables take precedence over LASTING variables of the same name. For variables of the same name defined within a file or in more than one file, the order of precedence, is:

SESSION - last in file is used
LASTING - last in file is used
INITIAL - first in file is used

So, for example, if a variable were defined for a given group several times in each file, and all files were present at initialization, the value used in the storage would be that defined last in the SESSION GLOBALV file.

4. The GLOBALV function SETL does not preserve lower-case argument values when called from the following:

- from the command line.
- from an XEDIT macro without an &COMMAND preceding it.
- from a System Product Interpreter EXEC with the ADDRESS CMS command.

5. The PUT and GET sub-functions use the EXECCOM facility of the interpreter currently executing, EXEC 2 or the System Product Interpreter (REXX), to set and retrieve variables.

For EXEC 2, names passed through EXECCOM are taken exactly as specified, and embedded ampersands (&) do not cause multiple substitution.

For REXX, no substitution or case translation takes place. Simple symbols must be valid REXX variable names, that is, in uppercase, and not starting with a number or a period. However, in compound symbols, any characters (including lowercase, blanks, etc.) are permitted following a valid REXX stem.

Usage Notes for CMS EXECs:

1. When defining values using GLOBALV’s SELECT sub-function, SET..., be aware that values (tokens) larger than eight characters will be truncated to eight characters by the CMS EXEC processor.

2. Avoid use of GLOBALV’s SELECT sub-function, SETL... . It requires an extended parameter list, such as is provided by EXEC 2. Use in CMS EXECs causes an error from the GLOBALV command processor.

3. Avoid use of GLOBALV’s SELECT sub-function, STACKR. The literal assignment statement it generates is not in a format the CMS EXEC processor recognizes. The CMS EXEC command processor will generate the following error message:
Examples:

These examples illustrate the use and effect of several, consecutive GLOBALV SELECT commands.

Example 1:

GLOBALV SET DEPT 222
(SELECT phrase is omitted.)
Effect: The value "222" is assigned to variable name "DEPT" in the default global variable group "UNNAMED".  

GLOBALV SELECT TABA
Effect: The default global variable group for subsequent SELECT sub-functions is set to "TABA"

GLOBALV SET EMP 8888 MONTH MAY
Effect: The value "8888" is assigned to the variable name "EMP" and the value "MAY" is assigned to the variable name "MONTH" in the default group "TABA".

GLOBALV SELECT UNNAMED SET YEAR 1982
Effect: The value "1982" is assigned to the variable name "YEAR" in the global variable group "UNNAMED". The default setting is not changed.

GLOBALV SETS YEAR 1981
Effect: The value "1981" is assigned to the variable name "YEAR" in the default global variable group "TABA" and the assignment is entered into the SESSION GLOBALV disk file.

GLOBALV SELECT TABA
Effect: The default global variable group for subsequent SELECT sub-functions is set to "TABA"

GLOBALV SETS YEAR 1981
Effect: The value "1981" is assigned to the variable name "YEAR" in the default global variable group "TABA" and the assignment is entered into the SESSION GLOBALV disk file.
GLOBALV STACK YEAR DEPT

Effect: Places the value associated with variable name "YEAR" from group "TABA" onto the stack. Since the variable "DEPT" is not defined in global variable group "TABA", a null line is stacked.

<table>
<thead>
<tr>
<th>Stack Before After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next line to read:</td>
</tr>
<tr>
<td>ABC 1981</td>
</tr>
<tr>
<td>XYZ (null line)</td>
</tr>
<tr>
<td>ABC XYZ</td>
</tr>
</tbody>
</table>

GLOBALV SELECT UNNAMED STACKR YEAR DEPT

Effect: Places a "&READ 002" control statement, and two literal assignment statements, defining the variable name "YEAR" and the variable name "DEPT" with their associated values from global variable group "UNNAMED", onto the stack.

<table>
<thead>
<tr>
<th>Stack After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next line to read:</td>
</tr>
<tr>
<td>&amp;READ 002</td>
</tr>
<tr>
<td>&amp;YEAR = LITERAL OF 1982</td>
</tr>
<tr>
<td>&amp;DEPT = LITERAL OF 222</td>
</tr>
<tr>
<td>1981</td>
</tr>
<tr>
<td>(null line)</td>
</tr>
<tr>
<td>ABC</td>
</tr>
<tr>
<td>XYZ</td>
</tr>
</tbody>
</table>

Example 2:

The effect of the following request, which names 3 variables:

GLOBALV SELECT TABA STACK EMP MONTH YEAR

<table>
<thead>
<tr>
<th>Stack After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next line to read:</td>
</tr>
<tr>
<td>8888 MAY 1981</td>
</tr>
</tbody>
</table>

Whereas, the effect of 3 consecutive STACK requests, naming a single variable each (the same 3 variables as the multiple request above):

GLOBALV SELECT TABA
GLOBALV STACK EMP
GLOBALV STACK MONTH
GLOBALV STACK YEAR

<table>
<thead>
<tr>
<th>Stack After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next line to read:</td>
</tr>
<tr>
<td>1981 MAY 8888</td>
</tr>
</tbody>
</table>

Responses:

GLOBALV ... LIST results in a display of the requested list.

GLOBALV GRPLIST results in a display of the requested list.
GLOBALV

Messages and Return Codes:

DMSGLO047E  No function specified [RC = 24]
DMSGLO104S  Error nn reading file fn GLOBALV A from disk [RC = 100]
DMSGLO108S  Virtual storage capacity exceeded [RC = 40041]
DMSGLO618E  NUCEXT failed [RC = nn]
DMSGLO622E  Insufficient free storage; no table made [RC = rc]
DMSGLO628E  Invalid GLOBALV function function [RC = 4]
DMSGLO631E  function can only be executed from an EXEC-2 or REXX
             EXEC [or as a CMS command] [RC = rc]
DMSGLO649E  Extraneous parameter parameter [RC = 24]

Error Codes:

GLOBALV error codes consist of two 3-character fields. The first field
corresponds to errors encountered during the GLOBALV INIT function; the
second corresponds to errors encountered during other GLOBALV
functions.

Code    Meaning

000 ...  Function completed successfully.
001      Truncation to the maximum length of 255 bytes occurred for
         variable name and/or variable value.
004      Invalid function/sub-function; or invalid environment for use
         of function/sub-function.
008      Error return from ATTN. Stacking suspended.
012      No free storage available to define (SET ..) additional variables.
         Processing suspended at point of error.
024      No function specified on GLOBALV command.
1nn      I/O error appending newly defined variable(s) to LASTING or
         SESSION GLOBALV file on the user's A-disk. The assignment
         was, however, added to the appropriate global variable group
         in storage. Refer to FSWRITE macro for meaning of "nn."
5nn      EXECCOMM failed with return code "nn." For the meaning of
         "nn," refer to the EXECCOMM facility in the VM/SP System
         Product Interpreter Reference or the VM/SP EXEC 2 Reference,
         depending on the EXEC environment in which the failure
         occurred.
-5nn     EXECCOMM failed with return code "-nn." For the meaning of
         "-nn," refer to the EXECCOMM facility in the VM/SP System
         Product Interpreter Reference or the VM/SP EXEC 2 Reference,
         depending on the EXEC environment in which the failure
         occurred.
1nn 000  I/O error reading GLOBALV type files from user's A-disk. No
         global variable(s) in storage created. Refer to FSREAD macro
         for meaning of "nn."
2nn ...  I/O error rewriting LASTING GLOBALV file into a temporary
         file. Global variables in storage are created, but rewrite of
         the LASTING file was suspended. The original LASTING file
remains intact on the user's A-disk. Refer to FSWRITE macro for meaning of "nn."

Function completed successfully.

I/O error appending newly defined variable(s) to LASTING or SESSION GLOBALV file on the user's A-disk. The assignment was, however, added to the appropriate global variable in storage. Refer to FSWRITE macro for meaning of "nn."

Error occurred renaming the temporary LASTING GLOBALV file to become the new LASTING file. Global variables in storage are created. The original LASTING file was destroyed, but TEMPFILE GLOBALV contains its corresponding variables. Refer to RENAME command for meaning of "nn."

Function completed successfully.

Error return from ATTN. Stacking suspended.

Error occurred when GLOBALV attempted to allocate storage for a workarea. GLOBALV initialization functions could not proceed.
HELP

Issue the HELP command to use the CMS HELP facility. HELP files can contain three layers of information: BRIEF, DETAIL, and RELATED. BRIEF provides quick, concise help; DETAIL provides complete information about the command; and RELATED provides related information about the task.

HELP files are available for CMS, CP, GCS, TSAF, IPCS, and SRPI commands; for EDIT, XEDIT, and DEBUG subcommands; for SQL/DS commands; for EXEC and EXEC 2 control statements; and for Restructured Extended Executor (REXX) language instructions.

The format of the HELP command is:

```
HELP [taskname | menu-name | component-name] [cmd-name]
```

where:

- **Help**, specified without any parameters, displays a task menu if you are using the VM/SP HELP files. If you are using files other than the VM/SP HELP files and if the HELP HELPMENU file has been created, then you will get the HELP HELPMENU file.

- **TASKs** displays the main TASK menu and is an easy way to begin using HELP. This menu leads to other task panels, short explanations for getting message information or for using the HELP facility, and the
major MENU panels. TASK is the default if no parameters are
specified in the HELP command and if a user-created file HELP
HELPMENU does not exist.

Help
displays a BRIEF HELP for the HELP command with a command
description, its format, an example, and, if applicable, a message
indicating that additional help is available. To receive at your
terminal the information you’re now reading, enter

    help help (detail)

taskname TASKs
displays the specified TASK file. TASK panel selections may include
other TASK menus, a MENU panel, or a command HELP file. The
TASK panels are organized in a branching structure of general to
specific tasks. For example:

<table>
<thead>
<tr>
<th>Taskname</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISK</td>
<td>Disk operations</td>
</tr>
<tr>
<td>MANAGE</td>
<td>File management</td>
</tr>
<tr>
<td>INQUIRE</td>
<td>System information</td>
</tr>
<tr>
<td>QPRINTER</td>
<td>CP QUERY PRINTER command</td>
</tr>
<tr>
<td>TAPEDUMP</td>
<td>CMS TAPE DUMP command</td>
</tr>
<tr>
<td>WINDOW</td>
<td>Window manipulation</td>
</tr>
<tr>
<td>BORDER</td>
<td>Windowing border commands</td>
</tr>
</tbody>
</table>

The general TASKs files contain task menus that lead you to the
specific task that you want information about. For example, entering:

    help disk tasks

displays a task menu listing the specific tasks concerning disks. You
can enter any of the specific TASKs file names (one that have a
filetype of HELPTASK) as well as the general ones.

In addition to the supplied tasknames, installations may create their
own taskname files.

menuname MENU
displays a selection of HELP files. (They may be command or other
menus.) The menuname is usually the name of a component. For
example, if you want to display the menu of CMS commands, you
would enter HELP CMS MENU.

component-name command-name
displays the HELP file for the specified command-name. The
command-name can be the name of a command, a subcommand, or a
statement. The component-name is the name of the component (or
grouping) about which you want information. The HELP facility has
the following major components:
Component  |  Description
---------  |  --------------------------
CMS        |  Conversational Monitor System commands
CP         |  Control Program commands
DEBUG      |  DEBUG subcommands
EDIT       |  EDIT subcommands
EXEC       |  EXEC statements
EXEC2      |  EXEC 2 statements
XEDIT      |  XEDIT subcommands
REXX       |  System Product Interpreter Statements
IPCS       |  Interactive Problem Control System commands
SQLDS      |  SQL/Data System Program Product (5748-XXJ) (only if you have this installed on your system.)
TSAF       |  Transparent Services Access Facility commands
CMSSET     |  CMS SET commands
CMSQUERY   |  CMS QUERY commands
CPSET      |  CP SET commands
CPQUERY    |  CP QUERY commands
SET        |  XEDIT SET subcommands
QUERY      |  XEDIT QUERY subcommands
PREFIX     |  XEDIT prefix commands

In addition to these components, installations may create their own components.

If a component is not specified, then the search order for HELP files is:

1. CMS
2. CP
3. MENU
4. TASK
5. MSG

Therefore, if you specify

```
help command-name
```

HELP searches for the following:

1. CMS command-name
2. CP command-name
3. command-name MENU
4. command-name TASK
5. MSG command-name
MESSAGE message-id  
MSG  
displays the HELP file for a message. 'message-id' is the 7, 8, 10, or 11-character message-id you specify to display the HELP file for a message. Specify the message-id in one of four forms:  

xxxnnnt  
xxxnnnnt  
xxxxmmnnnt  
xxxxmmmmmnt  

where:  

xxx  
indicates the component (for example, DMS for CMS messages, DMK for CP messages).  

mmm  
is the module identifier.  

nnn or nnnn  
is the message number.  

t  
is the message type.  

For example, for information on a message, you can specify either the 7- or 8-character message-id:  

DMS250S  
DMS0250S  

or the 10- or 11-character message-id that also identifies the issuing module:  

DMSHLP250S  
DMSHLP0250S  

Options:  

HELP files may contain different layers of information. You can get to these different layers by using various options. When you specify any of the options listed below, those sections of the HELP file are made available to you. Therefore, requested options determine what portion of the file is displayed. However, not all of the following option sections are available in all HELP files.  

BRIEF, DETAIL, and RELATED are layering options, which allow you to specify the level of information. BRIEF, the default, provides concise information to the inexperienced user. DETAIL provides more complete information to the experienced user. RELATED provides the user with additional information about a task.  

BRIEF, DETAIL, and RELATED are also conflicting options. You should specify only one of these options in the command string. If you specify
HELP

more than one of these options at a time, the last option you enter will be in effect.

DETAIL consists of its own subsetting options, which allow you to choose a specific part or parts of the help file. They include ALL, DESCript, FORMat, PARMs, NOTES, OPTions, and ERRors. ALL, the default, gives you all of the above-listed subsetting options. You can specify the other options in any combination.

The last set of options is referred to as other options. They include SCReen, NOScreen, EXTend, TYPe, and NOType.

OptionA, Layering Options:

BRIef

 displays a concise description of the specified command, its basic syntax (command without options), an example, and, if applicable, a message telling the user that either ALL or RELATED information is available.

DETAIL

 displays the detail subset of the HELP information for the specified command. The DETAIL subset can consist of any or all of the following help file sections:

 ALL
 DESCript
 FORMat
 PARMs
 OPTions
 NOTES
 ERRors.

If you specify the DETAIL subset on the command line, it will be in effect for only the current HELP request. If you specify the DETAIL subset by using the DEFAULTS command, it will be in effect for future HELP requests. The default for the detail subset is ALL. If this default has not been changed and if the DETAIL option is specified,

 help erase (detail

displays all the HELP information for the ERASE command, excluding the BRIEF and RELATED information. If the default has been changed, for example, to NOTES, and if the detail option is specified,

 help erase (detail

displays the Usage Notes information for the ERASE command.
HELP

| RELated                | displays information in a task panel from which you can select help on related topics. For example, |
|                       | help sendfile (related |
|                       | displays a task panel from which you can select help on commands related to sendfile: |
|                       | NAMES |
|                       | NOTE |
|                       | RDRLIST |
|                       | RECEIVE |
|                       | SET MSG |
|                       | TELL |

OptionB, Subsetting Options:

ALL displays all of the six HELP file sections listed below.

DESCript displays a general description of the specified HELP file.

FORMAT displays the format section (the syntax).

PARMs displays the parameter section (explanation of the operands).

OPTIONS displays the options section (a list of available options with a brief description).

NOTEs displays the usage notes and example sections.

ERRORs displays the error message and response sections.

Note: This section of the HELP files contains command responses and a reference to the VM/SP Messages and Codes manual. However, you can tailor your own HELP files to include the ERRORS section. Consult "Tailoring the HELP Facility" in the CMS User's Guide.
HELP

OptionC, Other Options:

SCReen
specifies that the entire screen is used to display the HELP file. While viewing the help file you can use PF keys and some System Product Editor commands to manipulate the display. This option is ignored on a line-oriented terminal.

NOScreen
specifies that the file is typed out on the screen.

TYPE
allows error message DMSHLL254E to be issued.

NOType
suppresses the printing of error message DMSHLL254E. This option allows you to change the message's text and placement.

EXTend
uses the HELP search order when you issue the HELP command and the component is specified. If the file is not found in the specified component, the HELP search order is used to locate the HELP file.

Usage Notes:

1. HELP is always displayed in a window named HELPWIN that is showing virtual screen HELPWIN. When displaying BRIef HELP using full-screen CMS, the CMS window is scrolled up, if required, so that you see both your work entry and the HELP information at the same time.

2. You can use the DEFAULTS command to set the HELP options. The initial HELP options have been preset to BRIef, ALL, and SCReen. However, the options you specify on the command line when entering the HELP command override those specified in the DEFAULTS command. This allows you to customize the defaults of the HELP command, yet override them when you desire. For more information, refer to the DEFAULTS command.

3. The HELP disk is specified at system generation time by the system support personnel. If the disk isn't already accessed, HELP accesses the disk containing the system HELP files. The HELP disk is accessed using the last available filemode and remains accessed after HELP has completed processing.

4. For commands or statement names containing special characters, use the special character. If, for example, you wanted to display the HELP file for the EXEC statement &ARG, you would issue HELP EXEC &ARG. For more information, see "Tailoring the HELP Facility," in the CMS User's Guide.
5. If you enter ‘CP TERM SCROLL nnn’ on a line-oriented terminal, it allows you to specify the number of lines to be scrolled on the display screen. For normal frame by frame scrolling, specify ‘nnn’ to equal the number of data lines on the screen. Specifying ‘CONT’ instead, causes continuous scrolling to the end of a file.

6. If you request HELP on a line-oriented terminal by issuing the command ‘HELP XEDIT MENU’ or on a display terminal by issuing ‘HELP XEDIT MENU (NOSCREEN)’, you get the following:

For information on one of the following commands, if its name is preceded by an ‘*’, enter ‘HELP name MENU’, or if preceded by a ‘:’, enter ‘HELP name TASK’. Otherwise, enter ‘HELP XEDIT name’.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:EDIT</td>
<td>Add</td>
</tr>
<tr>
<td>COMMAND</td>
<td>Append</td>
</tr>
<tr>
<td>CMS</td>
<td>CDelete</td>
</tr>
<tr>
<td>CLocate</td>
<td>CFirst</td>
</tr>
<tr>
<td>COverlay</td>
<td>CRepatch</td>
</tr>
<tr>
<td>EMSG</td>
<td>EXPand</td>
</tr>
<tr>
<td>GET</td>
<td>Help</td>
</tr>
<tr>
<td>Locate</td>
<td>LOMercas</td>
</tr>
<tr>
<td>MSG</td>
<td>Next</td>
</tr>
<tr>
<td>PREFIX</td>
<td>PREServe</td>
</tr>
<tr>
<td>READ</td>
<td>RECover</td>
</tr>
<tr>
<td>RESTore</td>
<td>RGTLEFT</td>
</tr>
<tr>
<td>SI</td>
<td>SORT</td>
</tr>
<tr>
<td>TOP</td>
<td>TRANSfer</td>
</tr>
<tr>
<td>VMFOPT</td>
<td></td>
</tr>
</tbody>
</table>

If you request help by issuing ‘HELP TASK’ on a line-oriented terminal or by issuing ‘HELP TASK (NOSCREEN)’ on a display terminal, you see the following:

Chapter 2. CMS Commands 265
For information on one of the following topics, enter HELP followed by the request information for that topic.

<table>
<thead>
<tr>
<th>Request</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>TASK TASK</td>
<td>TASKS - Helps if you don’t know VM commands.</td>
</tr>
<tr>
<td>TASK TASK</td>
<td>MESSAGE - Good choice for beginners.</td>
</tr>
<tr>
<td>MSG HELP</td>
<td>HELP - Explains how to get help for VM messages.</td>
</tr>
<tr>
<td>HELPINFO TASK</td>
<td>MENUS - Lists the HELP component MENUSs.</td>
</tr>
<tr>
<td>MENUS MENUS</td>
<td>COMMANDS - Lists VM commands that you can use.</td>
</tr>
<tr>
<td>COMMANDS MENUS</td>
<td>CMS - Shows only CMS commands.</td>
</tr>
<tr>
<td>CMS MENU</td>
<td>CP - Show only CP commands.</td>
</tr>
<tr>
<td>CP MENU</td>
<td>XEDIT - Lists System Product Editor items.</td>
</tr>
<tr>
<td>XEDIT MENU</td>
<td>REXX - Helps you use the REXX language.</td>
</tr>
<tr>
<td>REXX MENU</td>
<td>DEBUG - Helps you debug programs.</td>
</tr>
<tr>
<td>DEBUG MENU</td>
<td>SQLDS - Shows SQL/Data System items.</td>
</tr>
<tr>
<td>SQLDS MENU</td>
<td>SRPI - Lists the SRPI commands.</td>
</tr>
<tr>
<td>SRPI TASK</td>
<td></td>
</tr>
</tbody>
</table>

7. If you are viewing a command HELP file, you can issue the MOREHELP command from the command line. The default for MOREHELP is to display the DETAIL layer of that HELP file. DETAIL is determined by your setting of the DEFAULTS options. The options include ALL, DESCript, FORMat, PARMs, OPTions, NOTEs, and ERRors. You can specify an option(s) on the command line to view a certain section(s) of the HELP file. For example, if you are viewing a HELP file and decide that you want to see the format section for that file, you can enter the following on the command line:

   morehelp (format

The format section for that file will be displayed.

8. HELP loads the HELPXED XEDIT macro into user storage (using the EXECLOAD command) if the macro has not already been loaded. Loading this file into storage improves the performance of the HELP command. If you occasionally use HELP, you may want to EXECDCROP the HELPXED XEDIT macro after using HELP to release the storage.

9. Any data not within a conditional section (.cs on/.cs off) is uncontrolled data; all uncontrolled data defaults to DETAIL.

10. If a HELP file does not contain any of the requested sections, then the user receives a warning message and alternate information. For more information, see the VM/SP CMS User's Guide.

11. Some CMS commands are inappropriate to the BRIEF HELP environment. They are:
Examples:

Following are some examples of HELP requests.

- To request a HELP file for message DMSHLP002E, issue either:
  
  help dms002E
  or
  help dmshlp002E
- To request a menu of CP commands, issue:
  
  help cp menu
- To request a HELP file for the XEDIT LOCATE subcommand, issue:
  
  help xedit locate
- To request a description for the CMS TAPE command, issue either:
  
  help cms tape (desc
  or
  help tape (desc
- If you are viewing the HELP file for the CMS PRINT command and decide you want to get help on the CMS COPYFILE command, issue:
  
  help cms copyfile
- To request a display of RELATED information of the HELP file for the CMS SENDFILE command, issue:
  
  help cms sendfile (related
- If you are editing a file and want help on the COPYFILE command, and you are not sure whether COPYFILE is an Editor, CP, or CMS command, enter:
HELP

help copyfile (extend)

- To request a HELP file for the CMS SET APL command, you may enter:

  help cmsset apl

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSHEL241I  Press PF10 for detail information.
DMSHEL241I  Press PF11 to get related information.
DMSHEL241I  Press PF10 for detail information; PF11 to get related information.
DMSHEL242I  This HELP file fn ft has not been converted to the current release format or contains an invalid format word. For information, refer to the CMS User’s Guide.
DMSHEL244W  Requested HELP section unavailable; option option assumed.
DMSHEL254E  HELP cannot find the information you requested. If not misspelled, please enter HELP for menu assistance or HELP HELP for the HELP command. [RC = 28]
DMSHEL529E  Subcommand is only valid in display mode
DMSHEL529E  SET commandword subcommand is only valid in editing mode
DMSHEL545E  Missing operand(s)
DMSHEL561E  Cursor is not on a valid data field
DMSHEL586E  String not found
DMSHEL657E  Undefined PFkey/PAkey.
DMSHLC003E  Invalid option: option [RC = 24]
DMSHLL242I  This HELP file fnname fnftype has not been converted to the current release format or contains an invalid format word. For information, refer to the CMS User’s Guide.
DMSHLL243I  RELATED information is not available.
DMSHLL244W  Requested HELP subset information is not available; the option option has been assumed.
DMSHLL254E  HELP cannot find the information you requested. If not misspelled, please enter HELP for menu assistance or HELP HELP for the HELP command. [RC = 28]
DMSHLL355I  For related information on this subject, enter MOREHELP (RELATED).
DMSHLL356I  For more detail on this subject, enter MOREHELP.
DMSHLL639E  Error in routine routine; return code was xx
DMSHLLZ242I  This HELP file fnname fnftype has not been converted to the current release format or contains an invalid format word. For information, refer to the CMS User’s Guide.
HELPCONV

Use the HELPCONV command to convert a specified file into a formatted HELP file, leaving the .CS, .CM, and .MT control words in the file. The output file has the filetype $HLPxxxx, where xxxx is the name of the component to which the file belongs.

The format of the HELPCONV command is:

```
HELPCONV filename filetype [filemode]
```

where:

filename
is the filename of the HELP file to be converted.

filetype
is the filetype of the HELP file to be converted.

filemode
is the filemode of the HELP file to be converted. If the filemode is omitted or if an asterisk (*) is coded, all accessed disks are searched.

Usage Notes:

1. After using the HELPCONV command on a specified Help file, you will have two versions of the Help file:

   - filename HELPxxxx A1 The original file
   - filename $HLPxxxx A1 The converted file

   You should verify that the $HLPxxxx file is formatted the way you want it to appear when displayed by HELP.

   To use the converted file in the HELP command, you must rename the converted file so that its filetype is HELPcomponent. For example, if you enter

   helpconv link helpcp

   you create a converted file with a fileid of LINK $HLPCP A1. To have the HELP command call the new help file, rename the file so that the filetype is HELPCP:

   rename link $hlpcp a link helpcp a
Note that the converted file has the same name as the original file. Therefore, to rename the converted file, you must first rename the original file, move it to another read/write disk, or copy the formatted file to another read/write disk, specifying the filetype as HELPcomponent.

2. To use Help format words other than .CS, .CM, and .MT (for instance, .BX, .IN, etc.) in your own Help files, it’s still necessary to use the HELPConv command. It is also critical in these situations to ensure the proper formatting of MENU and TASK files, that is, that the file begins with a .FO OFF format word. For further information on how to process these files, please see “Tailoring the HELP Facility” in the VM/SP CMS User’s Guide.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSHLB251E</td>
<td>HELP processing error code nnn-description [RC = 12]</td>
</tr>
<tr>
<td>DMSHLI002E</td>
<td>File not found [RC = 28]</td>
</tr>
<tr>
<td>DMSHLI104S</td>
<td>Error nn reading file fn ft fm from disk [RC = 104]</td>
</tr>
<tr>
<td>DMSHLI109S</td>
<td>Virtual storage capacity exceeded [RC = 104]</td>
</tr>
<tr>
<td>DMSHLS109S</td>
<td>Virtual storage capacity exceeded [RC = 104]</td>
</tr>
<tr>
<td>DMSHLS250S</td>
<td>I/O error or device error [RC = 256]</td>
</tr>
<tr>
<td>DMSHLP251E</td>
<td>HELP processing error code nnn-description [RC = 12]</td>
</tr>
<tr>
<td>DMSHLS907E</td>
<td>I/O error on file fn ft fm [RC = 256]</td>
</tr>
</tbody>
</table>

Code Description

801 Numeric format word parameter is outside valid range.
802 Format word parameter should be a number.
803 Invalid format word.
804 Format word parameter missing.
805 Invalid format word parameter.
806 Undent greater than indent.
807 Excessive or negative space count generated.
HIDE WINDOW

Use the HIDE WINDOW command to prevent the specified window from being displayed and to connect the window to a virtual screen. The SHOW WINDOW command will redisplay the window.

The format of the HIDE WINDOW command is:

```
HIDE WINDOW [wname [ON vname [line col]]]
```

where:

- **wname**
  - is the name of the window to be hidden. An "=" indicates that the topmost window is hidden. If wname is not specified, "=" is assumed.

- **vname**
  - is the name of the virtual screen to which the window will be connected.

- **line**
  - is the virtual screen line number where the upper left corner of the window is placed.

- **col**
  - is the column number of the virtual screen where the upper left corner of the window is placed.

Usage Notes:

1. Multiple windows may be connected to a single virtual screen.

2. If the window is already connected to a virtual screen when you issue the HIDE WINDOW command, you do not have to specify the virtual screen information.

3. When you specify a virtual screen name, line, and column, the line and column values must be less than or equal to the corresponding virtual screen dimensions. The minimum line and column value is 1. If line and column are not specified, the default is 1 for both.

   If the specified line is past the current bottom of the virtual screen, the window is connected to the bottom of the virtual screen.
4. When you hide a window, the window is removed from the list of displayed windows. Therefore, issuing a command like POP WINDOW or DROP WINDOW that manipulates the order of displayed windows has no effect on hidden windows.

5. You must always have a window showing the CMS virtual screen when using full-screen CMS. If you hide all the windows showing the CMS virtual screen, the CMS window is automatically shown at the top of the CMS virtual screen. The CMS window is on top of all other windows, including the STATUS window. You should then issue the POP WINDOW STATUS command.

Responses:

None.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSHID386E Missing operand(s) [RC = 24]
- DMSHID388E Invalid keyword: keyword [RC = 24]
- DMSHID389E Invalid operand: operand [RC = 24]
- DMSHID391E Unexpected operand(s): operand [RC = 24]
- DMSHID921E Window wname is not defined [RC = 28]
- DMSWMM921E Virtual screen vname is not defined [RC = 28]
- DMSWMM921E Window wname is not defined [RC = 28]
- DMSWMM923E Specified location is outside the virtual screen [RC = 32]
- DMSWMM929E Window wname is not connected to a virtual screen [RC = 36]
IDENTIFY

Use the IDENTIFY command to display or stack the following information: your userid and node; the userid of the RSCS virtual machine; the date, time, time zone, and day of the week.

The format of the IDENTIFY command is:

```
IDENTIFY [ ( options...[ ) ] ]
```

Options:

<table>
<thead>
<tr>
<th>Options</th>
<th>STACK</th>
<th>FIFO</th>
<th>LIFO</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>STACK</td>
<td>FIFO</td>
<td>LIFO</td>
<td>TYPE</td>
</tr>
<tr>
<td></td>
<td>FIFO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LIFO</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

STACK [ FIFO ]

specifies that the information should be placed in the program stack rather than displayed at the terminal. The information is stacked either FIFO (first in first out) or LIFO (last in first out). The default order is FIFO.

FIFO

specifies that the information should be placed in the program stack rather than displayed at the terminal. The information is stacked FIFO. The options STACK, STACK FIFO, and FIFO are all equivalent.

LIFO

specifies that the information should be placed in the program stack rather than displayed at the terminal. The information is stacked LIFO. This option is equivalent to STACK LIFO.

TYPE

specifies that the information should be displayed at the terminal. This is the default option.
IDENTIFY

Example:

To display the userid information at your terminal, you would enter the following:

identify (type

Responses:

The following information is displayed or stacked:

```
userid AT node VIA rscsid date time zone day
```

*where:*

userid is the userid of your virtual machine.

node is the RSCS node of your computer.

rscsid is the userid of the RSCS virtual machine.

date is the local date, in the form mm/dd/yy.

time is the local time, in the form hh:mm:ss.

zone is the local time zone.

day is the day of the week.

*Implementation notes:*

The userid and node are from the CP QUERY USERID command. The date, time, and zone are from the CP QUERY TIME command.

The CP QUERY CPUID command is used to retrieve the CPU serial number. (CP QUERY CPUID returns a 16-digit processor identification; however, IDENTIFY only uses digits three through eight.) This number is then looked up in the file SYSTEM NETID *. That file will have one or more lines of the form:

```
cpu-serial-number node rscsid
```

If there is a conflict in nodes between the SYSTEM NETID file and CP QUERY USERID, the node in SYSTEM NETID takes precedence. If there is no record with a matching serial number, or if the file is not found, the rscsid is set to *.
**Important Note:**

The person responsible for the CMS system at an installation is responsible for creating the SYSTEM NETID file. This file should have a filemode of S2.

**Messages and Return Codes:**

Additional error messages for specifying a command format incorrectly are listed on page 24.

- **DMSIDE003E**  Invalid option: option [RC = 24]
- **DMSIDE056E**  File fn ft [fm] contains invalid record formats [RC = 32]
- **DMSIDE070E**  Invalid parameter parameter [RC = 24]
- **DMSIDE104S**  Error nn reading file fn ft fm from disk [RC = 100]
Use the IMMCMD command to establish or cancel Immediate commands from within an EXEC. IMMCMD determines whether a particular Immediate command has been established or if it has been issued by the terminal user.

The format of the IMMCMD command is:

```
IMMCMD [SET CLEAR QUERY STATUS] name
```

**where:**

**SET**
establishes an Immediate command. If an Immediate command with the same name already exists, it is overridden in a stack-like manner.

**CLEAR**
clears an Immediate command. Any previously overridden Immediate command with the same name is reinstated by this action.

**QUERY**
indicates whether the Immediate command has been established. A return code of 0 indicates that the command has been established. A return code of 44 indicates that the command has not been established.

**STATUS**
indicates whether the command has been issued by the terminal user. The only output from the STATUS operand is a return code of 0 or 1. A return code of 0 indicates that the Immediate command has not been entered from the terminal since the last invocation of the IMMCMD SET or IMMCMD STATUS for that Immediate command. A return code of 1 indicates that the Immediate command has been issued since the last invocation of IMMCMD SET or IMMCMD STATUS for that Immediate command.

**name**
the 1 to 8 character name of the Immediate command that is established (SET), cancelled (CLEAR) or inquired about (STATUS, QUERY). This operand is always required.
Usage Notes:

1. The IMMCMD command should be used only from EXEC files (CMS EXEC, EXEC 2, or System Product Interpreter).

2. All Immediate commands established by the IMMCMD command can be explicitly cancelled. If these Immediate commands are not explicitly cancelled by the IMMCMD command, they are cancelled automatically. They are cancelled either by returning to the CMS command environment (if not in CMS subset) or by entry to the CMS abend routine. User exit routines cannot be used with Immediate commands that are established by the IMMCMD command.

3. Since no exit routine can be given control when an Immediate command (declared via IMMCMD) has been issued from the terminal, the EXEC writer must use the STATUS operand of IMMCMD. A return code of 1 from IMMCMD STATUS informs the EXEC writer that a particular Immediate command has been issued by the terminal user. The EXEC writer can then take appropriate processing action if the Immediate command has been issued.

For a general discussion on Immediate commands, see the VM/SP CMS User’s Guide. For information on creating Immediate commands, see VM/SP CMS for System Programming.

Example:

To clear the HM immediate command, you would enter into your EXEC (written in the REXX language):

'immcmd clear hm'

Messages and Return Codes:

- DMSIMM014E Invalid function function [RC = 24]
- DMSIMM047E No function specified [RC = 24]
- DMSIMM070E Invalid parameter parameter [RC = 24]
- DMSIMM109S Virtual storage capacity exceeded [RC = 104]
- DMSIMM261E No immediate command name was specified [RC = 24]
- DMSIMM262E Immediate command command not found [RC = 44]
- DMSIMM263E Specified immediate command is a nucleus extension and cannot be cleared [RC = 48]
INCLUDE

Use the INCLUDE command to read one or more TEXT files (containing relocatable object code) from disk and to load them into virtual storage, establishing the proper linkages between the files. A LOAD command must have been previously issued for the INCLUDE command to produce desirable results. For information on the CMS loader and the handling of unresolved references, see the description of the LOAD command.

The format of the INCLUDE command is:

```
INCLUDE fn... [ (options...] ) ]
```

```
<table>
<thead>
<tr>
<th>Options:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEAR</td>
</tr>
<tr>
<td>NOCLEAR</td>
</tr>
<tr>
<td>RESET</td>
</tr>
<tr>
<td>entry</td>
</tr>
<tr>
<td>ORIGIN</td>
</tr>
<tr>
<td>hextol</td>
</tr>
<tr>
<td>TRANS</td>
</tr>
<tr>
<td>MAP</td>
</tr>
<tr>
<td>NOMAP</td>
</tr>
<tr>
<td>TYPE</td>
</tr>
<tr>
<td>NOTYPE</td>
</tr>
<tr>
<td>INV</td>
</tr>
<tr>
<td>NOINV</td>
</tr>
<tr>
<td>REP</td>
</tr>
<tr>
<td>NOREP</td>
</tr>
<tr>
<td>AUTO</td>
</tr>
<tr>
<td>NOAUTO</td>
</tr>
<tr>
<td>LIBE</td>
</tr>
<tr>
<td>NOLIBE</td>
</tr>
<tr>
<td>[START]</td>
</tr>
<tr>
<td>[SAME]</td>
</tr>
<tr>
<td>DUP</td>
</tr>
<tr>
<td>[NODUP]</td>
</tr>
<tr>
<td>[RLDsave]</td>
</tr>
<tr>
<td>HIST</td>
</tr>
<tr>
<td>NOHIST</td>
</tr>
</tbody>
</table>
```

where:

fn...

are the names of the files to be loaded into storage. Files must have a filetype of TEXT and consist of relocatable object code such as that produced by the OS language processor. If a GLOBAL TXTLIB command has identified one or more TXTLIBs, fn may indicate the name of a TXTLIB member.

Options:

If options were specified with a previous LOAD or INCLUDE command, these options (with the exception of CLEAR, NODUP, and ORIGIN) remain set if SAME is specified when INCLUDE is issued. Otherwise, the options assume their default settings. If conflicting options are specified, the last one entered is in effect.
CLEAR
clears the load area in storage to binary zeros before the files are loaded.

NOCLEAR
does not clear the load area before loading.

RESET \{ entry \}
resets the execution starting point previously set by a LOAD or INCLUDE command. If entry is specified, the starting execution address is reset to the specified location. If an asterisk (*) is specified or if the RESET option is omitted, the loader input is searched for control statements. The entry point is selected from the last ENTRY statement encountered or from an assembler- or compiler-produced END statement. If none is found, a default entry point is selected as follows: if an asterisk was specified, the first byte of the first control section loaded by the INCLUDE command becomes the default entry point; if the RESET option was omitted, the entry point defaults to the execution starting point previously set by a LOAD or INCLUDE command.

ORIGIN \{ hexloc \}
TRANS
begins loading the program at the location specified by hexloc; this location must be in the CMS transient area or in the user area below the start of the CMS nucleus. The variable, hexloc, is a hexadecimal number of up to six characters. If this option is not specified, loading begins at the next available storage location. INCLUDE does not overlay any previously loaded files unless this option is specified and the address given indicates a location within a previously loaded object module. TRANS indicates that the file is loaded into the transient area.

MAP
adds information to the load map.

NOMAP
does not add any information to the load map.

TYPE
displays the load map of the files at the terminal, as well as writing it on the A-disk. This option is valid only if MAP is specified or implied.

NOTYPE
does not display the load map at the terminal.

INV
writes invalid card images in the LOAD MAP file.
**INCLUDE**

**NOINV**

does not write invalid card images in the LOAD MAP file.

**REP**

writes Replace (REP) statement images in the LOAD MAP file. See the explanation of the CMS LOAD command for a description of the Replace (REP) statement.

**NOREP**

suppresses the writing of Replace (REP) statements in the LOAD MAP file.

**AUTO**

searches your disks for TEXT files to resolve undefined references.

**NOMALS**

suppresses automatic searching for TEXT files.

**LIBE**

searches the text libraries defined by the GLOBAL command for missing subroutines.

**NOLIBE**

does not search any text libraries for unresolved references.

**START**

begins execution after loading is completed.

**SAME**

retains the same options (except ORIGIN, NODUP, and CLEAR) that were used by a previous INCLUDE or LOAD command. Otherwise, the default setting of unspecified options is assumed. If other options are specified with SAME, they override previously specified options. (See Usage Note 1.)

**DUP**

displays warning messages at your virtual console when a duplicate CSECT is encountered during processing. The duplicate CSECT is not loaded.

**NODUP**

does not display warning messages at your virtual console when duplicate CSECTs are encountered during processing. The duplicate CSECT is not loaded.

**RLDsave**

instructs the CMS loader to save the relocation information from the text files. This option is used with the CMS GENMOD command to generate relocatable format CMS module files. These modules may be used with the CMS NUCXLOAD command.
**HIST**

saves the history information (comments) from text files. This information is later included in the module generated by a GENMOD command. The history information from the specified text files is added to history information requested from other text files for this module (using the HIST option on the LOAD command and other INCLUDE commands).

**NOHIST**

does not save history information from text files.

**Usage Notes:**

1. If you specify several nondefault options on the LOAD command and you want those options to remain in effect, use the SAME option when you issue the INCLUDE command; for example:

   `include main subi data (reset main map start)`

   brings the files named MAIN TEXT, SUBI TEXT, and DATA TEXT into virtual storage and appends them to previously loaded files. Information about these loaded files is added to the LOAD MAP file. Execution begins at entry point MAIN.

   `load myprog (nomap nolibe norep)`
   `include mysub (map same)`

   During execution of the LOAD command, the file named MYPROG TEXT is brought into real storage. The following options are in effect: NOMAP, NOLIBE, NOREP, NOTYPE, INV, and AUTO. During execution of the INCLUDE command, the file named MYSUB TEXT is appended to MYPROG TEXT. The following options are in effect:

   `MAP, NOLIBE, NOREP, NOTYPE, INV, AUTO`

2. When the INCLUDE command is issued, the loader tables are not reset.

3. When the RLDsave information is specified, the CMS LOADER can save the relocation information for up to 16,384 address constants.

4. When you specify the HIST option, up to 819 comment records can be saved from text files. These records are included later in the module generated by a GENMOD command. When this limit is exceeded, a message is displayed and a warning is placed in the last record of the history data to indicate this condition. Any history information exceeding 819 records is not included.

5. For additional information on the CMS loader, see the discussion of the LOAD command, or consult **VM/SP CMS User's Guide**.
Responses:

DMSLIO740I Execution begins ...

START was specified with INCLUDE and the loaded program has begun execution. Any further responses are from the program.

INVALID CARD - xxx...xxx

INV was specified with LOAD and an invalid card has been found. The message and the contents of the invalid card (xxx...xxx) are listed in the LOAD MAP file. The invalid card is ignored and loading continues.

Messages and Return Codes:

DMSLIO001E No filename specified [RC = 24]
DMSLIO002E File([s]) fn TXTLIB not found [RC = 28]
DMSLIO003E Invalid option: option [RC = 24]
DMSLIO005E No option specified [RC = 24]
DMSLIO021E Entry point name not found [RC = 40]
DMSLIO029E Invalid parameter parameter in the option option field [RC = 24]
DMSLIO055E No entry point defined [RC = 40]
DMSLIO056E File fn ft [fm] contains invalid record formats [RC = 32]
DMSLIO099E CMS/DOS environment not active [RC = 40]
DMSLIO104S Error nn reading file fn ft fm from disk [RC = 100]
DMSLIO105S Error nn writing file fn ft fm on disk [RC = 100]
DMSLIO108S Virtual storage capacity exceeded [RC = 104]
DMSLIO116S Loader table overflow [RC = 104]
DMSLIO168S Pseudo register table overflow [RC = 104]
DMSLIO169S ESDID table overflow [RC = 104]
DMSLIO201W The following names are undefined: namelist [RC = 4]
DMSLIO202W Duplicate identifier identifier [RC = 4]
DMSLIO203W SET LOCATION COUNTER name undefined [RC = 4]
DMSLIO206W Pseudo register alignment error [RC = 4]
DMSLIO623S Module cannot be loaded at location location--this area is available for system use only [RC = 88]
DMSLIO625S There are too many items that require relocation to save all of the RLD information [RC = 104]
DMSLIO749W There are too many comments in text files to be included in the module. [RC = 4]
DMSLIO907T I/O error on file fn ft fm [RC = 256]
LABELDEF

Use the LABELDEF command to specify standard HDR1 and EOF1 tape label description information for CMS, CMS/DOS, and OS simulation. This command is required for CMS/DOS and CMS tape label processing. It is optional for OS simulation. However, it is needed if you want to specify a filename to be checked or the exact data to be written in any field of an output HDR1 and EOF1 label.

The format of the LABELDEF command is:

```
<table>
<thead>
<tr>
<th>Labeldef</th>
</tr>
</thead>
<tbody>
<tr>
<td>fn</td>
</tr>
<tr>
<td>[ CLEAR ]</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>[ FID {?</td>
</tr>
<tr>
<td>[ FSEQ fseq ] [ GENN genn ] [ GENV genu ]</td>
</tr>
<tr>
<td>[ CRDTE yyddd ] [ EXDTE yyddd ] [ SEC {0</td>
</tr>
<tr>
<td>[ (options...[ )]) ]</td>
</tr>
</tbody>
</table>
```

Options: [ PERM ] [ CHANGE ] [ NOCHANGE ]

* may be specified only with CLEAR. It clears all existing label definitions.

**filename**

is one of the following:

- ddname for FILEDEF files (OS simulation). When using FORTRAN, you must explicitly specify ddname as FTnnF001.
- filename in DTFMT macro (CMS/DOS simulation).
- labeldefid specified in the TAPEMAC or TAPPDS command or in the LABID field of the TAPESL macro (can be 1-8 characters).
LABELDEF

CLEAR
removes a label definition.

LABELDEF filename CLEAR clears only the label definition for that filename.

LABELDEF * CLEAR removes all existing label definitions unless specified as PERM.

FID \{ ? \}
{fid}
supplies the file (data set for OS) identifier in the tape label. Use the FID ? form if the identifier exceeds 8 characters (up to a maximum of 44) or the identifier contains special characters. The system responds by prompting you to supply the information. If the file identifier does not exceed 8 characters, enter the fileid directly (FID fid). If the file identifier exceeds 17, only the rightmost 17 characters are moved into the HDR1 record when a file is written to tape, and only the rightmost 17 are displayed for a LABELDEF query.

VOLID \{ \}
{volid}
{SCRATCH}
supplies the volume serial number(s) (1-6 alphanumeric characters). If there is only one volid or if you are not using OS simulation, you can enter the volid directly (VOLID volid).

If you use OS simulation and have multiple volids to process, use the VOLID ? form. The system responds by prompting you to supply the volume serial numbers needed to process the file.

For non-OS simulation, more than one volid may be specified; however, only the first volid will be used.

Specifying a volid of SCRATCH results in no serial number checking for that tape and for any subsequent tapes to be mounted for the current file. When specifying multiple volids, SCRATCH must be the last volid specified. If you specify a VOLID of SCRATCH and you are a non-OS simulation user, a tape with the VOLID of 'SCRATC' is searched for.

VOLSEQ volseq
is the volume sequence number (1-4 numeric characters). Under OS-simulation the volume sequence number is ignored and is set to 0001.

FSEQ fseq
is the file (data set for OS) sequence number in the label (1-4 numeric characters).
**GENN** *genn*
is the generation number (1-4 numeric characters).

**GENV** *genv*
is the generation version (1-2 numeric characters).

**CRDTE** *yyddd*
is the creation date.

**EXDTE** *yyddd*
is the expiration date.

**SEC**
specifies security classification (0, 1, or 3). See the IBM publication *OS/VS Tape Labels*, for the meaning of security classification on tape files. Note that this number has no effect on how the file is processed. It is used only for checking or writing purposes.

**Options:**

**PERM**
retains the current definition until it either is explicitly cleared or is changed by a new LABELDEF command with the CHANGE option. If PERM is not specified, the definition is cleared when a LABELDEF * CLEAR command is executed.

**CHANGE**
merges the label definitions whenever a label definition already exists for a filename and a new LABELDEF command specifying the same filename is issued. In this situation, the options associated with the two definitions are merged. Options from the original definition remain in effect unless duplicated in the new definition. New options are added to the option list.

**NOCHANGE**
retains the current label definition, if one exists, for the specified filename.

The following default values are used in output labels when a value is not explicitly specified:

**FID**
For OS simulation, fid is the ddname specified in the FILEDEF command for the file.

For CMS/DOS, fid is the DTFMT symbolic name.

For the CMS TAPESL macro, fid is the LABELDEF specified in the LABID parameter.
LABELDEF

VOLID

is CMS001. For OS simulation, the actual VOLID from the tape mounted is used if processing an “SL” tape file.

FSEQ

is 0001.

VOLSEQ

is 0001.

GENN

is blanks.

GENV

is blanks.

CRDTE

is the date when the label is written.

EXDTE

is the date when the label is written.

SEC

is 0.

Usage Notes:

1. To check a field in an input label, specify it on your LABELDEF command for the label. If you do not specify a value for a particular field, this field is not checked at all for input. For output, any field you specify is written in the label exactly as you specify it on the LABELDEF command. If you do not specify a field for output, the default value for that field is written in the label.

If you write the following LABELDEF command,

```
labeldef filex fid master fseq 2 exdte 78285
```

and use the statement for an input file, only the file identifier, file sequence number, and expiration date in HDR1 labels are checked. Error messages are issued when there fields in the tape label do not match those specified in the LABELDEF statement. If you use the same statement for an output file, the fields leave the following values:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileid</td>
<td>MASTER</td>
</tr>
<tr>
<td>file sequence number</td>
<td>0002</td>
</tr>
<tr>
<td>volume sequence number</td>
<td>0001</td>
</tr>
<tr>
<td>creation date</td>
<td>date when label is written</td>
</tr>
<tr>
<td>expiration date</td>
<td>78285</td>
</tr>
<tr>
<td>security</td>
<td>0</td>
</tr>
<tr>
<td>volume serial number</td>
<td>CMS001</td>
</tr>
<tr>
<td>generation number</td>
<td>blank</td>
</tr>
<tr>
<td>generation version</td>
<td>blank</td>
</tr>
</tbody>
</table>
2. If you issue LABELDEF without any operands, a list of all LABELDEFs currently in effect is displayed on your terminal. For an OS simulation user, if SCRATCH was entered at command time and the file has not been opened, then all the VOLIDs and SCRATCH will be displayed. If SCRATCH was entered at command time and the file has been opened, then all the VOLIDs and the VOLIDs of all the scratch tapes will be displayed following SCRATC or SCRATCH.

3. For OS simulation, a LABELDEF statement may be used as well as a FILEDEF statement for a file. Use of a LABELDEF statement is optional in this case. The statements

```
labeldef filez fid payroll fseq 2 exdte 78300
filedef filez tap1 sl volid vol4
```

define filez as a labeled tape file on tape 181. The volume serial is VOL4, the fileid is PAYROLL, and the file sequence number is 0002. Expiration date is day 300 in 1978. If you only use the FILEDEF command, you have only defined the VOLID (volume serial number).

4. For CMS and CMS/DOS, a LABELDEF command is required. The command

```
labeldef file14 volid supvol volseq 3
```

defines a tape label with a volume serial of SUPVOL and a volume sequence number of 0003. This LABELDEF statement could be used by a CMS/DOS program containing a DTFMT macro with the form

```
FILE14 DTFMT ...FILABL=STD...
```

or by a CMS program with a TAPESL macro similar to the following:

```
TAPESL HOUT,181,LABID=FILE14
```

A CMS TAPEMAC command could use the same LABELDEF as follows:

```
tapemac maclib sl file14
```

In all three preceding examples, the LABELDEF statement must be issued before the program or command is executed.

5. See the section “Tape Labels in CMS” in the VM/SP CMS User’s Guide for more details on CMS tape label processing.
LABELDEF

Messages and Return Codes:

- DMSLBD003E  Invalid option: option [RC = 24]
- DMSLBD029E  Invalid parameter parameter in the option option field [RC = 24]
- DMSLBD065E  option option specified twice [RC = 24]
- DMSLBD066E  option1 and option2 are conflicting options [RC = 24]
- DMSLBD070E  Invalid parameter parameter [RC = 24]
- DMSLBD109S  Virtual storage capacity exceeded [RC = 104]
- DMSLBD220R  Enter data set name:
- DMSLBD221E  Invalid data set name [RC = 24]
- DMSLBD324I  No user defined LABELDEFs in effect
- DMSLBD438E  Valid valid is a duplicate entry [RC = 24]
- DMSLBD439E  Valid valid is an invalid entry [RC = 24]
- DMSLBD441R  Enter valid information
- DMSLBD442E  SCRATCH may only be used as the last valid for the file [RC = 24]
- DMSLBD704I  Invalid CLEAR request
LISTDS

Use the LISTDS command to list, at your terminal, information about the data sets or files residing on accessed OS or DOS disks. In addition, use LISTDS to display extent or free space information when you want to allocate space for VSAM files.

The format of the LISTDS command is:

```
LISTDS [dsname] [fm] [options... [ ) ]]
```

**Options:**

- `EXTENT`
- `FORMAT`
- `FREE`
- `PDS`

**where:**

? indicates that you want to enter interactively the OS data set name, VSE fileid, or VSAM data space name. When you enter a question mark (?), CMS prompts you to enter the OS data set name, DOS fileid, or VSAM data space name exactly as it appears on the disk. This form allows you to enter names that contain embedded blanks or hyphens.

dname is the OS data set name or VSE fileid or VSAM data space name. It takes the form:

```
qual1 [qual2...qualn]
```

where blanks separate the qualifiers, the dataset name used will be the concatenation of the qualifiers with periods. (See Usage Note 1.)

fm is the filemode of the disk to be searched for the specified file. If a dname is not specified, a list of all the files or data sets on the specified disk is displayed.

* indicates that you want all of your accessed DOS or OS disks searched for the specified data set or file. If a dname is not specified, a list of all files on all accessed OS and DOS disks is displayed. If a dname is
LISTDS

specified, CMS stops searching all of your accessed DOS or OS disks as soon as it finds the first copy of the specified data set or file.

Options:

The FREE and EXTENT options are mutually exclusive; the FORMAT and PDS options cannot be specified with either FREE or EXTENT.

FREE
requests a display of all free space extents on a specific minidisk or on all accessed DOS and OS disks. If you enter the FREE option, you cannot specify a dsname.

EXTENT
EX
requests a display of allocated extents for a single file or for an entire disk or minidisk. If a dsname is specified, only the extents for that particular file or data set are listed. If fm is specified as *, all disks are searched for extents occupied by that file, but only the extents for the first file or data set encountered are displayed.

If a dsname is not specified, then a list of all currently allocated extents on the specified disk, or on all disks, is displayed.

FORMAT
FO
requests a display of the date, disk label, filemode, and data set name for an OS data set as well as RECFM, LRECL, BLKSIZE, and DSORG information. For a VSE file, LISTDS displays the date, disk label, filemode, and fileid, but gives no information about the RECFM, LRECL, and BLKSIZE (two blanks appear for each); DSORG is always PS.

PDS
displays the member names of referenced OS partitioned data sets.

Usage Notes:

1. If you want to enter an OS or VSE file identification on the LISTDS command line, it may consist of qualifiers separated by periods or blanks. For example, the file TEST.INPUT.SOURCE.D could be listed as follows:

   listds test input source d
or

   listds test.input.source.d
Or, you can enter the name interactively, as follows:

```
listds ? *
```

DMSLDS220R Enter data set name:
test.input.source.d

Note that when the data set name is entered interactively, it must be entered in its exact form; when entered on the LISTDS command line, the periods may be omitted.

You must use the interactive form to enter a VSE fileid that contains embedded blanks.

2. When using access method services, use the FREE option to determine what free space is available for allocation by VSAM. For example:

```
listds * (free
```

requests a display of unallocated extents on all accessed OS or DOS disks. You can then use the EXTENT option on the DLBL command when you define the file for AMSERV.

3. Full disk displays using the FREE option will display free alternate tracks as well as free space extents.

4. Since CMS does not support ISAM files, LISTDS lists extent and free information on ISAM files, but ignores format 2 DSCB's.

5. Since CMS does not support track overflow, LISTDS will not read beyond a track if DCB=RECFM=T is specified for the OS VTOC.

6. LISTDS uses the extended plist for processing the dsname parameter. If you are calling LISTDS from an assembler language program and using dsname, you should supply an extended plist. VM/SP CMS for System Programming has more information on how an assembler language program can supply an extended plist.

Responses:

DMSLDS220R Enter data set name:

This message prompts you to enter the data set name when you use the ? operand on the LISTDS command. Enter the file identification in its exact form. A sample sequence might be:

```
listds ? c
```

DMSLDS220R Enter data set name:

my.file.test
FM DATA SET NAME
C MY.FILE.TEST
Ready;
The response shown above following the entry of the data set name is the same as the response given when you enter a data set name on the LISTDS command line.

DMSLDS229I  No members found

This message is displayed when you use the PDS option and the data set has no members.

DMSLDS233I  No free space available on fm disk

This message is displayed when you use the FREE option and there is no free space available on the specified disk.

Responses to the EXTENT Option: A sample response to the EXTENT option is shown below. The headers and the type of information supplied are the same when you request information for a specific file only, or for all disks.

```
lstds g (extent

EXTENT INFORMATION FOR 'VTOC' ON 'G' DISK:
SEQ TYPE  CYL-HD(RELTRK) TO CYL-HD(RELTRK)  TRACKS
 000  VTOC  0099 00  1881 0099 18 1899 19

EXTENT INFORMATION FOR 'PRIVAT.CORE.IMAGE.LIB' ON 'G' DISK:
SEQ TYPE  CYL-HD(RELTRK) TO CYL-HD(RELTRK)  TRACKS
 000  DATA  0000 01  1 0049 18 949 949

EXTENT INFORMATION FOR 'SYSTEM.WORK.FILE.NO.6' ON 'G' DISK:
SEQ TYPE  CYL-HD(RELTRK) TO CYL-HD(RELTRK)  TRACKS
 000  DATA  0050 00  950 0051 18 987 38

EXTENT INFORMATION FOR 'COBOL TEST PROGRAM' ON 'G' DISK:
SEQ TYPE  CYL-HD(RELTRK) TO CYL-HD(RELTRK)  TRACKS
 000  DATA  0052 02  990 0054 01 1027 38

EXTENT INFORMATION FOR 'DKSQ01A' ON 'G' DISK:
SEQ TYPE  CYL-HD(RELTRK) TO CYL-HD(RELTRK)  TRACKS
 000  DATA  0080 01  1521 0081 00 1539 19

or for a fixed-block device:

EXTENT INFORMATION FOR 'DSQ01A' ON 'G' DISK:
SEQ TYPE  BLOCKNO TO BLOCKNO  BLOCKS
 000  DATA  000500  000550  51

where:

SEQ

indicates the sequence number assigned this extent when the extents were defined via the DLBL command. CMS assigns the sequence numbers for VSAM data sets; the first extent set has a sequence of 000, the second extent has a sequence of 001, and so on.
```
TYPE can have the following designations:

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>Data area extent</td>
</tr>
<tr>
<td>VTOC</td>
<td>VTOC extent of the disk</td>
</tr>
<tr>
<td>SPLIT</td>
<td>Split cylinder extent</td>
</tr>
<tr>
<td>LABEL</td>
<td>User label extent</td>
</tr>
<tr>
<td>INDEX</td>
<td>ISAM index area extent</td>
</tr>
<tr>
<td>OVFLO</td>
<td>ISAM independent overflow area extent</td>
</tr>
<tr>
<td>MODEL</td>
<td>Model data set label in the VTOC. Does not define an extent</td>
</tr>
</tbody>
</table>

CYL-HD(RELTRK) TO CYL-HD(RELTRK)
indicates the cylinder, head, and relative track numbers of the start and end tracks of this extent.

TRACKS
indicates the number of tracks in the extent.

BLOCKNO TO BLOCKNO
indicates the relative block numbers of the start and end of the extent.

BLOCKS
indicates the number of blocks in the extent.

Response to the FREE Option: A sample response to the FREE option is shown below. The same headers and type of information is shown when you request free information for all accessed disks.

listds g (free
FREESPACE EXTENTS FOR 'G' DISK:
CYL-HD(RELTRK) TO CYL-HD(RELTRK) TRACKS
0052 00 988 0052 01 989 2
0054 02 1028 0080 00 1520 493
0081 01 1540 0098 18 1880 341

or for a fixed-block device:

listds g (free
FREESPACE EXTENTS FOR 'G' DISK:
FB/E BLOCKNO TO FB/E BLOCKNO BLOCKS
000501 001330 830
010310 029610 19301
068990 069990 1001

Chapter 2. CMS Commands 293
LISTDS

where:

CYL-HD(RELTRK) TO CYL-HD(RELTRK)
indicates the cylinder, head and relative track numbers of the
starting and ending track in the free extent.

TRACKS
indicates the total number of free tracks in the extent.

BLOCKNO TO BLOCKNO
indicates the relative block number of the start and end of
extents that are free on the fixed-block device.

BLOCKS
indicates the total number of blocks contained in each extent.

Response to the FORMAT and PDS Options:

If you enter the FORMAT and PDS options, you receive information similar
to the following:

```plaintext
listds d (fo pd)
```

<table>
<thead>
<tr>
<th>RECFM</th>
<th>LRECL</th>
<th>BLKSI</th>
<th>DSORG</th>
<th>DATE</th>
<th>LABEL</th>
<th>FM</th>
<th>DATA SET NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB</td>
<td>80</td>
<td>800</td>
<td>PO</td>
<td>01/31/75</td>
<td>OSSYS1</td>
<td>D</td>
<td>SYS1.MACLIB</td>
</tr>
</tbody>
</table>

MEMBER NAMES:
ABEND  ATTACH  BLDL  BSP     CLOSE  DCB  DETACH  DEVTYPE
FIND   PUT      READ   WRITE  XDAP

<table>
<thead>
<tr>
<th>RECFM</th>
<th>LRECL</th>
<th>BLKSI</th>
<th>DSORG</th>
<th>DATE</th>
<th>LABEL</th>
<th>FM</th>
<th>DATA SET NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>80</td>
<td>80</td>
<td>PS</td>
<td>01/10/75</td>
<td>OSSYS1</td>
<td>D</td>
<td>SAMPLE</td>
</tr>
</tbody>
</table>

Messages and Return Codes:

- **DMSLDS002E** Dataset not found [RC = 28]
- **DMSLDS003E** Invalid option: `option` [RC = 24]
- **DMSLDS048E** Invalid mode `mode` [RC = 24]
- **DMSLDS099E** Disk `mode` not accessed [RC = 36]
- **DMSLDS221E** Invalid data set name [RC = 24]
- **DMSLDS222E** I/O error reading `data set name` from `{fm|OS|DOS} disk` [RC = 28]
- **DMSLDS223E** No filemode specified [RC = 24]
- **DMSLDS226E** No dataset name allowed with FREE option [RC = 24]
- **DMSLDS227W** Invalid extent found for `data set name` on `fm` disk [RC = 4]
- **DMSLDS231E** I/O error reading VTOC from `{fm|OS|DOS} disk` [RC = 28]
- **DMSLDS333E** `nnn`K partition too large for this virtual machine [RC = 24]
LISTFILE

Use the LISTFILE command to obtain specified information about CMS files residing on accessed disks.

The format of the LISTFILE command is:

```
[ fn [ ft [ fm ] ] ] [ ( options... [ ) ] ]
```

### Options:

- **Header**
- **NOHeader**
- **Exec** [Trace] [ARGS]
- **APpend** [ARGS]
- **STACK** [FIFO | LIFO]
- **FIFO**
- **LIFO**
- **XEDIT**
- **FName**
- **FType**
- **FMode** [ % x ]
- **Format**
- **Alloc**
- **Date**
- **Label**
- **Blocks**

### where:

- **fn** is the filename of the files for which information is to be collected. If an asterisk is coded in this field, all filenames are used.

In addition, certain special characters (* and %) can be used as part of the filename to request that the list contain a specific subset of files. See the usage note, “Pattern Matching,” for information on using these special characters.

- **ft** is the filetype of the files for which information is to be collected. If an asterisk is coded in this field, all filetypes are used.

In addition, certain special characters (* and %) can be used as part of the filetype to request that the list contain a specific subset of files. See the usage note, “Pattern Matching,” for information on using these special characters.

- **fm** is the filemode of the files for which information is to be collected. If this field is omitted, only the A-disk is searched. If an asterisk is coded, all accessed disks are searched.
LISTFILE

Output Format Options:

**Header**
includes column headings in the listing. HEADER is the default if any of the supplemental information options (FORMAT, ALLOCATE, DATE, or LABEL) are specified. The format of the heading is:

FILENAME FILETYPE FM FORMAT LRECL RECS BLOCKS DATE TIME LABEL

**NOHeader**
does not include column headings in the list. NOHEADER is the default if only filename, filetype, or filemode information is requested.

Output Disposition Options:

**Exec**
creates a CMS EXEC file of 80- or 88-character records (one record for each of the files that satisfies the given file identifier) on your A-disk. An 80-character record file is created unless you specify the LABEL option, in which case an 88-character record file is created. If a CMS EXEC already exists, it is replaced. The header is not included in the file.

**Trace**
causes the EXEC 2 statement &TRACE OFF to be written as the first record of the CMS EXEC file, which is created when the EXEC option is specified. With this option, no statements issued from the CMS EXEC file are traced. For more information on the &TRACE statement, see the VM/SP EXEC 2 Reference. The TRACE option implies the EXEC option.

**ARGS**
causes EXEC 2 dummy arguments &3 through &15 to be appended to each line in the CMS EXEC file (following the fileid of each file). Each record of the CMS EXEC file has the form:

&1 &2 fileid &3 &4 &5 &6 ...&15

Specifying this option allows you to pass up to 15 arguments to the CMS EXEC file. The ARGs option does not imply the EXEC option and therefore must be specified in conjunction with EXEC, TRACE, or APPEND.

**APPend**
creates a CMS EXEC and appends it to the existing CMS EXEC file. If no CMS EXEC file exists, one is created.
LISTFILE

STACK [FIFO LIFO]

specifies that the information should be placed in the program stack (for use by an EXEC or other program) instead of being displayed at the terminal. The information is stacked either FIFO (first in first out) or LIFO (last in first out). The default order is FIFO.

FIFO

specifies that the information should be placed in the program stack rather than displayed at the terminal. The information is stacked FIFO. The options STACK, STACK FIFO, and FIFO are all equivalent.

LIFO

specifies that the information should be placed in the program stack rather than displayed at the terminal. The information is stacked LIFO. This option is equivalent to STACK LIFO.

XEDIT

specifies that the information should be placed in the file in the XEDIT ring currently being displayed. The edited file must either be fixed format with a logical record length (lrecl) of 108 or variable length format. The information replaces the current line and below until the END OF FILE is reached. Then, the remaining text (if any) is inserted before the END OF FILE. Each line is 108 characters and contains the information that is obtained with the LABEL option, plus the filename, filetype, and filemode appended to the end of the line. This option is only valid when LISTFILE is issued from the XEDIT environment. Refer to the section “XEDIT Interfaces to Access Files in Storage” in the VM/SP System Programmer’s Guide for more information on using this option.

Information Request Options:

Only one of these options need be specified. If one is specified, any options with a higher priority are also in effect. If none of the following options are specified, the default information request options are in effect.

Default Information Request Options:

FName

creates a list containing only filenames. Option priority is 7.

FType

creates a list containing only filenames and filetypes. Option priority is 6.

FMode

creates a list containing filenames, filetypes, and filemodes. Option priority is 5.
LISTFILE

Supplemental Information Options:

FOrmat
includes the record format and logical record length of each file in the list. Option priority is 4.

ALoc
includes the amount of disk space that CMS has allocated to the specified file in the list. The quantities given are the number of logical records in the file and the number of CMS data blocks. Option priority is 3.

Date
includes the date the file was last written in the list.

The form of the date is:
month/day/year  hour:minute
for 800-byte block disks, or:
month/day/year  hour:minute:second
for all other format sizes.

Option priority is 2.

Label
includes the label of the disk on which the file resides in the list.
Option priority is 1.

Other Options:

Blocks
causes the total number of CMS data blocks used by the files in the list to be displayed as the last line of the list, in the form BLOCKS n. It is displayed as a separate line. When BLOCKS is specified along with the STACK option, the line is displayed and not stacked.

%x
is used to change the place holding character from % to x, where x is any character, for this invocation of LISTFILE. For more information on using a place holding character, see the usage note, “Pattern Matching,” below.
Usage Notes:

1. Pattern Matching

If you enter the LISTFILE command with no operands, a list of all files on your A-disk is displayed at the terminal.

If you want information about a specific subset of your files, you can use two special characters in the fn and ft operands. (Only an asterisk may be specified for filemode.) The special characters are * (asterisk) and % (percent), where:

* represents any number of character(s). As many asterisks as required can appear anywhere in a filename or filetype. However, the total number of characters, including the asterisks, may not exceed eight. (Only one asterisk may be used for a filemode.)

For example, if you enter:

```
listfile *d* *file*
```

you are requesting that the list contain all files on your A-disk whose filename contains “d” and whose filetype contains “file.”

The list might contain the following files:

```
YOURDATA  AFILE1  A1
HISDATA   AFILE2  A1
ADOG      1DOGFILE A2
```

% is a place holding character that means a single character, but any character will do. As many percent symbols as necessary may appear anywhere in a filename or filetype. For example, if you enter:

```
listfile %%% stock
```

you are requesting that the list contain all files on your A-disk whose filename is three characters in length and whose filetype is “stock.” The list might contain the following files:

```
THE      STOCK  A1
HIS      STOCK  A1
HER      STOCK  A1
```

The default place holding character (%) can be changed by using the %x option. For example,

```
listfile $ script (%$
```

displays all SCRIPT files on the A-disk whose filename is one character in length.

2. If you request any additional information with the supplemental information options, that information is displayed along with the header.
3. When you use the EXEC or APPEND option, the CMS EXEC A1 that is created is in the format:

```bash
&1 &2 filename filetype filemode
```

where column 1 is blank.

If you specify the ARGS option with EXEC or APPEND, each line in the CMS EXEC is in the format:

```bash
&1 &2 filename filetype filemode &3 &4 &5 &6 ... &15
```

This allows you to pass up to 15 arguments to the EXEC. For example, if the following command is issued,

```bash
LISTFILE * * A (EXEC ARGS
```

a CMS EXEC file is created, with each record formatted as shown above. The following command

```bash
CMS TAPE DUMP ( WTM
```

causes the tape dumping command to be executed against each file in the CMS EXEC, with TAPE assigned to &1, DUMP to &2, ( to &3, and WTM to &4.

If you use any of the supplemental information options, that information is included in the EXEC file. For information on using CMS EXEC files, see the VM/SP CMS User's Guide.

4. You can invoke the LISTFILE command from the terminal, from an EXEC file, or as a function from a program. If LISTFILE is invoked as a function or from an EXEC file that has the &CONTROL NOMSG option in effect, the DMSLST002E ("File not found") error message is not issued.

5. To display only the files with a particular filemode number, specify the numeric portion of the filemode in the listfile command. For example, to display only the files with filetype 'EXEC' on your A2 disk:

```bash
Listfile * exec a2
```

The display might look like this:

```
ALPHA EXEC A2
SEND EXEC A2
TEMP EXEC A2
```

6. The options STACK, LIFO, and FIFO cause the requested information to be placed in the program stack. When the requested information is to be stacked, the options relating to the CMS EXEC (APPEND, EXEC, TRACE, and ARGS) and the options relating to the display format (HEADER, NOHEADER) should not be specified.
7. The XEDIT option is valid only when LISTFILE is issued from the XEDIT environment. The edited file must either be fixed format with a logical record length (lrecl) of 108 or variable length format. The information replaces the current line and below until the END OF FILE is reached. Then, the remaining text (if any) is inserted before the END OF FILE.

Responses:

Unless the EXEC, TRACE, APPEND, STACK, LIFO, or FIFO option is specified, the requested information is displayed at the terminal. Depending on the options specified, as discussed above, the information displayed is:

<table>
<thead>
<tr>
<th>FILENAME</th>
<th>FILETYPE</th>
<th>FM</th>
<th>FORMAT</th>
<th>LRECL</th>
<th>RECS</th>
<th>BLOCKS</th>
<th>DATE</th>
<th>TIME</th>
<th>LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>fn</td>
<td>ASSEMBLE</td>
<td>fm</td>
<td>V</td>
<td>lrecl</td>
<td>norecs</td>
<td>noblks</td>
<td>mm/dd/yy</td>
<td>hh:mm:ss</td>
<td>volid</td>
</tr>
</tbody>
</table>

*where:*

- fn is the filename of the file.
- ft is the filetype of the file.
- fm is the filemode of the file.
- F is the file format: F is fixed-length, V is variable-length.
- lrecl is the logical record length of the largest record in the file.
- norecs is the number of logical records in the file.
- noblks is the number of CMS data blocks that the file occupies on disk.
- mm/dd/yy is the date (month/day/year) that the file was last updated.
- hh:mm:ss is the time (hours:minutes:seconds) that the file was last updated.
- volid is the volume serial number of the virtual disk on which the file resides.

One entry is displayed for each file listed.
LISTFILE

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSLST002E  File(s) [fn [fn [fm]]] not found [RC = 28]
DMSLST003E  Invalid option: option [RC = 24]
DMSLST037E  Disk mode[(vdev)] is accessed as read/only [RC = 36]
DMSLST037E  Output disk mode[(vdev)] is accessed as read/only [RC = 36]
DMSLST048E  Invalid mode mode [RC = 24]
DMSLST066E  option1 and option2 are conflicting options [RC = 24]
DMSLST069E  Disk mode not accessed [RC = 36]
DMSLST070E  Invalid parameter parameter [RC = 24]
DMSLST105S  Error nn writing file fn ft fm on disk [RC = 100]
DMSLST688E  XEDIT option only valid from XEDIT environment [RC = 24]
DMSLST689E  File must be F-format 108 or V-format [RC = 24]
**LISTIO**

Use the LISTIO command in CMS/DOS to display a list of current assignments for system and/or programmer logical units in your virtual machine.

The format of the LISTIO command is:

```
LISTIO [SYS
PROG
SYS xxx
A
UA
ALL]
[(options...[])]

Options: [EXEC
APPEND]
[STAT]
```

where:

**SYS**
requests a list of the physical devices assigned to all system logical units.

**PROG**
requests a list of the physical devices assigned to programmer logical units SYS000 through SYS241.

**SYSxxx**
requests a display of the physical device assigned to the particular logical unit specified.

**A**
requests a list of only those logical units that have been assigned to physical devices.

**UA**
requests a list of only those logical units that have not been assigned to physical devices; that is, that are unassigned.

**ALL**
requests a list of the physical units assigned to all system and programmer logical units. If no operand is specified, ALL is the default.
LISTIO

Options:

The EXEC and APPEND options are mutually exclusive; if both are entered on the command line, the last one entered is in effect.

EXEC

erases the existing $LISTIO EXEC file, if one exists, and creates a new one.

APPEND

adds new entries to the end of an existing $LISTIO EXEC file. If no $LISTIO EXEC file exists, a new one is created.

STAT

lists the status (read-only or read/write) of all disk devices currently assigned.

Usage Notes:

1. Logical units are assigned and unassigned with the ASSGN command. For a list of logical units and valid device types, see the discussion of the ASSGN command.

2. The $LISTIO EXEC contains one record for each logical unit listed. The format is:

&1 &2 SYSxxx {device mode [status]}

where column 1 is blank.

Responses:

Depending on the operands specified, the following is displayed for each unit requested in the LISTIO command:

SYSxxx {device mode [status]}

where device is the device type (READER, PRINTER, PUNCH, TERMINAL, TAPn, IGN, or UA). If the device is a disk, the one-character mode letter is displayed. If the STAT option is specified, the status (R/O or R/W) is also displayed.
### Messages and Return Codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSLLU003E</td>
<td>Invalid option: <em>option</em> [RC = 24]</td>
</tr>
<tr>
<td>DMSLLU006E</td>
<td>No read/write A disk accessed [RC = 36]</td>
</tr>
<tr>
<td>DMSLLU070E</td>
<td>Invalid parameter <em>parameter</em> [RC = 24]</td>
</tr>
<tr>
<td>DMSLLU099E</td>
<td>CMS/DOS environment not active [RC = 40]</td>
</tr>
<tr>
<td>DMSLLU105S</td>
<td>Error <em>nn</em> writing file <em>fn ft fm</em> on disk [RC = 100]</td>
</tr>
<tr>
<td>DMSLLU303E</td>
<td>No SYSXXX satisfies request [RC = 28]</td>
</tr>
</tbody>
</table>
Use the LKED command to create a CMS LOADLIB or LOADLIB member.

The format of the LKED command is:

```
LKED  [name]  [(options...[])]
```

Options:

- `[NCAL]`  
- `[LET]`  
- `[ALIGN2]`  
- `[NE]`  
- `[OL]`  
- `[RENT]`  
- `[REUS]`  
- `[REFR]`  
- `[OVLY]`  
- `[XCAL]`  

- `[NAME membername]`  
- `[LIBE libraryname]`  
- `[XREF]`  
- `[MAP]`  
- `[TERM]`  
- `[PRINT]`  
- `[LIST]`  
- `[DISK]`  
- `[NOPRINT]`  

- `[SIZE value1 value2]`

**where:**

- `fname` specifies the filename of the object file to be processed. The file must have a filetype of TEXT and fixed-length, 80-character records.

**Options:**

If duplicate or conflicting linkage editor options are specified, the linkage editor resolves them according to normal procedures (refer to OS/VSE Linkage Editor and Loader, GC26-3813). If duplicate or conflicting CMS-related options are specified, the last one entered on the command line is in effect. The CMS-related options are: `TERM`, `NOTERM`, `PRINT`, `DISK`, `NOPRINT`, `NAME`, and `LIBE`.

- **NCAL** suppresses the automatic library call function of the linkage editor.

- **LET** suppresses marking of the load module “not executable” in the event of some linkage editor error condition.
ALIGN2
indicates that boundary alignment specified in the linkage editor input file is to be performed on the basis of 2048-byte boundaries. If this option is omitted, alignment is performed on the basis of 4096-byte boundaries.

NE
marks the load module output as “not to be edited” such that it cannot be processed again by the linkage editor.

OL
marks the load module output “only loadable.”

RENT
marks the load module reenterable.

REUS
marks the load module reusable.

REFR
marks the load module refreshable.

OVLY
processes an overlay structure.

XCAL
allows valid exclusive CALLs in the overlay structure.

NAME membername
is the member name to be used for the load module created. The member name specified here overrides the default name, but it cannot override a name specified via the linkage editor NAME control statement.

LIBE libraryname
is the filename of a LOADLIB file where the output load module is to be placed. The LOADLIB file specified here may also be used for auxiliary input to the linkage editor via the INCLUDE statement.

XREF
produces an external symbol cross-reference for the modules being processed.

MAP
produces only a module map for the processed module(s).

LIST
includes only linkage editor control messages in the printed output file.

TERM
displays any linkage editor diagnostic messages at the user terminal.
NOTERM
suppresses the displaying of diagnostic messages.

PRINT
spools the linkage editor printed output file to the printer.

DISK
stores the linkage editor output in a CMS disk file with a filetype of LKEDIT.

NOPRINT
produces no output file.

SIZE value1 value2
indicates the amount of virtual storage to be used by the linkage editor and specifies the portion of that storage to be reserved for the load module buffer. The SIZE parameters must lie within the following limits:

value1 64K to 9999K (or 65536 to 999999)
value2 6K to 100K (or 6144 to 102400)

If either of the SIZE parameters is omitted, the default values are 192K for value1 and 64K for value2. If either value is invalid, the OS Linkage Editor defaults to the 9999K for value1 and 100K for value2, which are the maximum values. Values greater than 999999 can be entered in the form nnnnK (with K equal to 1024). For example, enter 2000K instead of 2048000. Values accepted by the linkage editor are displayed in the output file.

Usage Notes:

1. Only a subset of the possible linkage editor control statements are meaningful in CMS. Since the CMS interface program cannot examine the input data for the LKED command, all of the control statements are allowed, even though several of them result in the creation of a load module file that cannot be used under CMS. For both command options and control statements, see the publication OS/VS Linkage Editor and Loader, GC26-3813.

2. When you use the linkage editor INCLUDE control statement to include a load module, the DDNAME referring to the module library must be other than SYSLMOD and it must have been previously defined by a FILEDEF. If you include a member of the LOADLIB that receives linkage editor output, you can enter statements in the following form:

filedef libdef disk mylib loadlib a (recfm u
lked fname (libe mylib)

Contents of file FNAME TEXT:

include libdef (libmemb1)
name libmemb2
3. The LKED command produces one temporary file:

\texttt{fname SYSUT1}

This file is temporarily created for each link-edit step; any existing file with the same file identifier is erased at the beginning of the link edit. This file is placed on the read/write disk with the most available space. Work space is automatically allocated as needed during the link edit and returned to available status when the link edit is complete. Insufficient space causes abnormal termination of the link edit.

4. The LKED command produces two permanent files:

\texttt{fname LOADLIB}
\texttt{fname LKEDIT}

The ‘\texttt{fname LOADLIB}’ file contains the load module(s) that the linkage editor created. This file is in CMS simulated partitioned data set format, as created by the CMS OS data management macros. The filename of the input file becomes the filename of the LOADLIB file, unless the LIBE option is specified. The filename of the input file also becomes the member name of the output load module, unless either the NAME option or a NAME control statement is used. One or more load modules may be created during a single LKED command execution if the NAME linkage editor control statement is used in the input file. When the NAME control statement is used, that name becomes the member name in the LOADLIB file. The replace option of the NAME statement determines whether existing members with the same name are replaced or retained.

The ‘\texttt{fname LKEDIT}’ file contains the printed output listing produced according to the XREF, MAP, or LIST options. This file is created on disk unless the PRINT or NOPRINT option is specified. The LOADLIB and LKEDIT files are placed on (1) the disk from which the input file was read, (2) the parent disk, or (3) the primary disk. Failure to obtain sufficient space for these files results in abnormal termination of the linkage editor. The specified LOADLIB must not exist on any read/only extension of the A-disk.

5. Because the LKED command uses the OS Linkage Editor for the actual link of the TEXT file to the LOADLIB as an executable module, you can override the file definitions with the CMS FILEDEF command prior to using the LKED command. Refer to the publication \textit{OS/VS Linkage Editor and Loader}, GC26-3813, for information regarding the file definitions.

The default FILEDEF commands issued by the LKED command for the ddnames presented to the Linkage Editor are:
FILEDEF SYSLIN DISK fname TEXT * (RECFM F BLOCK 80 NOCHANGE
FILEDEF SYSLMOD DISK fname LOADLIB A1 (RECFM U BLOCK 260 NOCHANGE

--or--

FILEDEF SYSLMOD DISK libname LOADLIB A1 (RECFM U BLOCK 260 NOCHANGE
FILEDEF SYSUT1 DISK fname SYSUT1 *
FILEDEF SYSPRINT DISK fname LKEDIT A1

--or--

FILEDEF SYSPRINT PRINTER

--or--

FILEDEF SYSPRINT DUMMY

At the completion of the LKED command, all FILEDEFs that do not
have the PERM option are erased.

6. The blocksize of the output LOADLIB is limited to 6K.

7. For information on listing, copying, and compressing a CMS LOADLIB,
refer to the LOADLIB command.

Messages and Return Codes:

DMSLKD001E No filename specified [RC = 24]
DMSLKD002E File fn ft fm not found [RC = 28]
DMSLKD004W Warning messages issued [RC = 4]
DMSLKD005E No option specified [RC = 24]
DMSLKD006E No read/write disk accessed [RC = 36]
DMSLKD007E File fn ft fm is not fixed, 80-character records [RC = 32]
DMSLKD008W Error messages issued [RC = 8]
DMSLKD012W Severe error messages issued [RC = 12]
DMSLKD016W Terminal error messages issued [RC = 16]
DMSLKD070E Invalid parameter parameter [RC = 24]
LOAD

Use the LOAD command to read one or more TEXT files (containing relocatable object code) from disk and to load them into virtual storage, establishing the proper linkages between the files.

The format of the LOAD command is:

```
LOAD fn ... [ (options...[ )]]
```

**Options:**

- CLEAR
- NOCLEAR
- RESET
- ORIGIN
- TRANS
- MAP
- TYPE
- INV
- REP
- AUTO
- NOMAP
- NOTYPE
- NOINV
- NOREP
- NOAUTO
- LIBE
- START
- NOLIBE
- DUP
- RLDsave
- NODUP
- HIST
- NOHIST

**where:**

*fn*...

specifies the names of the files to be loaded into storage. The files must have a filetype of TEXT and consist of relocatable object code such as that produced by the OS language processors. If a GLOBAL TXTLIB command has been issued, *fn* may indicate the name of a TXTLIB member.

**Options:**

If conflicting options are specified, the last one entered is in effect. Options may be overridden or added when you use the INCLUDE command to load additional TEXT files.

**CLEAR**

clears the load area in storage before the object files are loaded. Whole page frames are released; the remainder of storage that is not on a page boundary is set to binary zeros.

**NOCLEAR**

does not clear the load area before loading.
LOAD

RESET \{ entry \}

sets the starting location for the programs currently loaded. The operand, entry, must be an external name (for example, CSECT control section) or ENTRY) in the loaded programs. If RESET is not specified, the default entry point is used. (See Usage Note 4.) If * is entered the results are the same as if the RESET option were omitted.

Note: The RESET option should not be used when loading TEXT files created by any of the following OS/VS language processors under CMS: OS Code and Go FORTRAN, OS FORTRAN IV (G1), OS FORTRAN IV (H) Extended, OS/VS COBOL Compiler and Library, OS Full American National Standard COBOL Version 4 Compiler and Library.

ORIGIN \{ hexloc \}

loads the program beginning at the location specified by hexloc; this location must be in the CMS transient area or in the user area. The location, hexloc, is a hexadecimal number of up to six characters. If TRANS is specified, the file is loaded into the CMS nucleus transient area. If ORIGIN is not specified, loading begins at the first available storage location in the user program area.

If you specify ORIGIN TRANS, the name of the module currently in the transient area is updated. If you specify ORIGIN hexloc and hexloc is within the transient area, the name is not updated.

Note: Any program loaded into the transient area must have a starting address of X'E000'. See the discussion of the GENMOD command for information on loading programs in the transient area.

MAP

writes a load map on your A-disk, named LOAD MAP A5.

NOMAP

does not create the LOAD MAP file.

TYPE

displays the load map at your terminal and writes it on the A-disk. This option is valid only if the MAP option is in effect.

NOTYPE

does not display the load map at the terminal.

INV

includes invalid card images in the load map.

NOINV

does not include invalid card images in the load map.
LOAD

REP
includes Replace (REP) statements in the load map.

NOREP
does not include the Replace (REP) statements in the load map.

AUTO
searches your virtual disks for TEXT files to resolve undefined references.

NOAUTO
suppresses automatic searching for TEXT files.

LIBE
searches the text libraries for missing subroutines. If text libraries are to be searched for TEXT files, they must previously have been defined by a GLOBAL command.

NOLIBE
does not search the text libraries for unresolved references.

START
executes the program being loaded when loading is completed. LOAD does not normally begin execution of the loaded files. To begin execution immediately upon successful completion of loading, specify START. Execution begins at the default entry point. (See Usage Note 4.)

DUP
displays warning messages at your terminal when a CSECT having the same name as another CSECT that has already been loaded is encountered during processing. The CSECT with the duplicate name is not loaded. (See Usage Note 3.)

NODUP
does not display warning messages at your terminal when CSECTs having the same name (duplicate CSECTs) are encountered during processing. Only the first CSECT by a given name is loaded, the CSECT with the duplicate name is not loaded.

RLDsave
instructs the CMS LOADER to save the relocation information from the text files. This option is used with the CMS GENMOD command to generate relocatable format CMS module files. These modules may be used with the CMS NUCXLOAD command.

HIST
saves the history information (comments) from text files. This information is later included in the module generated by a GENMOD command. The history information requested from text files specified in subsequent INCLUDE commands is also included in the module when you issue the GENMOD command. Only history information
LOAD

Usage Notes:

1. Unless the NOMAP option is specified, you must have a read/write CMS A-disk accessed when you issue the LOAD command. The loader creates a load map on the A-disk each time the LOAD command is issued without the NOMAP option. A load map is a file that contains the location of control sections and entry points of files loaded into storage. This load map is named LOAD MAP A5. Each time LOAD is issued, a new LOAD MAP file replaces any previous LOAD MAP file.

   If invalid card images exist in the file or files that are being loaded, they are listed with the message INVALID CARD in the LOAD MAP file. To suppress this listing in the load map, use the NOINV option.

   If Replace (REP) statements exist in the file being loaded, they are included in the LOAD MAP file. To suppress this listing of REP statements, specify the NOREP option.

   If the ENTRY or LIBRARY control cards are encountered in the file, the load map contains an entry:

   CONTROL CARD- ...

   listing the card that was read.

   Mapping of any common areas that exist in the loaded files will occur when the program is prepared for execution by the START or GENMOD command or by the START option of the LOAD or INCLUDE command. An updated load map may be displayed before program execution if the START command is issued with the NO option to suppress execution.

2. CSECTs (control sections) having a duplicate name (duplicate CSECTs) are bypassed by the loader. Only the first CSECT encountered is physically loaded. The CSECTs with the duplicate names are not loaded. A warning message is displayed at your terminal if you specified the DUP option. If a section contains an ADCON that references a CSECT with a duplicate name that has not been loaded, that ADCON may be resolved incorrectly.

3. The loader selects the entry point for the loaded program according to the following hierarchy:

   • From the parameter list on the START command

   • From the last RESET operand in a LOAD or INCLUDE command

   The last LOAD command and its subsequent INCLUDE commands is included in the module by the GENMOD command.

NOHIST
does not save history information from text files.
LOAD

- From the last ENTRY statement in the input
- From the last LDT statement in the input
- From the first assembler- or compiler-produced END statement that specifies an entry point if no ENTRY statement is in the input
- From the first byte of the first control section of the loaded program if there is no ENTRY statement and no assembler- or compiler-produced END statement specifying an entry point

4. The LOAD command should not be used to execute programs containing DOS macros. To link-edit and execute programs in the CMS/DOS environment, use the DOSLKED and FETCH commands.

5. When the text to be loaded contains dummy sections or defines pseudo registers, their cumulative length must not exceed 32767 (X'7FFF'). If this limit is exceeded, the values stored by the CXD entry will not be accurate and the load map will not contain the correct cumulative length values.

6. Common sections and pseudo-register cumulative length files (CXDs) are not resolved until either a GENMOD command or START command is issued (or the START option of the LOAD or INCLUDE command). Be sure to resolve everything before performing functions such as a CP SAVESYS.

7. See Figure 9 for an illustration of the loader search order. The loader uses this search order to locate the filename on the LOAD and INCLUDE command lines, as well as in the handling of unresolved references.

8. When you specify the HIST option, up to 819 comment records can be saved from text files. These records are included later in the module generated by a GENMOD command. When this limit is exceeded, a message is displayed and a warning is placed in the last record of the history data to indicate this condition. Any history information exceeding 819 records is not included.
Use standard order of search to locate the TEXT files specified by fn...

Any unresolved references?

YES

Is NOAUTO specified?

NO

Use standard order of search to locate files with a filetype of TEXT and a filename corresponding to the unresolved reference...

Any unresolved references?

YES

Is NOLIBE specified?

NO

Search active text libraries (those that were previously specified by a GLOBAL command). Files are searched in the order they are entered in the command.

Search complete

Figure 9. Loader Search Order
9. The CMS loader also loads routines called dynamically by OS LINK, LOAD, and XCTL macros. Under certain circumstances, an incorrect entry point may be returned to the calling program. See the VM/SP CMS User's Guide for more information.

10. LOAD does not clear user storage unless the CLEAR option is specified.

11. When the RLDSAVE option is specified, the CMS LOADER can save the relocation information for up to 16,384 address constants.

Loader Control Statements

You can add loader control statements to TEXT files either by editing them or by punching real cards and adding them to a punched text deck before reading it into your virtual machine. The seven control cards recognized by the CMS loader are discussed below.

The ENTRY and LIBRARY cards, which are discussed first, are similar to the OS linkage editor control statements ENTRY and LIBRARY. The CMS ENTRY and LIBRARY statements must be entered beginning in column 1.

ENTRY Statement: The ENTRY statement specifies the first instruction to be executed. It can be placed before, between, or after object modules or other control statements. The format of the ENTRY statement is shown in Figure 10. The external name is the name of a control section or an entry name in the input deck. It must be the name of an instruction, not of data.

| ENTRY | external name |

Figure 10. ENTRY Statement Format

LIBRARY Statement: The LIBRARY statement can be used to specify the never-call function. The never-call function (indicated by an asterisk (*) as the first operand) specifies those external references that are not to be resolved by the automatic library call during any loader step. It is negated when a deck containing the external name referred to is included as part of the input to the loader. The format of the LIBRARY statement is shown in Figure 11. The external reference refers to an external reference that may be unresolved after input processing. It is not to be resolved. Multiple external references within the parentheses must be separated by commas.

The LIBRARY statement can be placed before, between, or after object decks or other control statements.

| LIBRARY | *(external reference) |

Figure 11. LIBRARY Statement Format

Loader Terminate (LDT) Statement: The LDT statement is used in a text library as the last record of a member. It indicates to the loader that all
records for that member were processed. The LDT statement can contain a
name to be used as the entry point for the loaded member. The LDT
statement has the format shown in Figure 12.

<table>
<thead>
<tr>
<th>Column</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X'02' (12-2-9 punch). Identifies this as a loader control statement.</td>
</tr>
<tr>
<td>2-4</td>
<td>LDT -- identifies type of statement.</td>
</tr>
<tr>
<td>5-16</td>
<td>Not used.</td>
</tr>
<tr>
<td>17-24</td>
<td>Blank or entry name (left-justified and padded with blanks to eight characters).</td>
</tr>
<tr>
<td>25</td>
<td>Blank.</td>
</tr>
<tr>
<td>26-33</td>
<td>May contain information specified on a SETSSI card processed by the TXTLIB command.</td>
</tr>
<tr>
<td>34-72</td>
<td>May be used for comments or left blank.</td>
</tr>
<tr>
<td>73-80</td>
<td>Not used by the loader. You may leave these columns blank or insert program identification for your own convenience.</td>
</tr>
</tbody>
</table>

Figure 12. LDT Statement Format

Include Control Section (ICS) Statement: The ICS statement changes the
length of a specified control section or defines a new control section. It
should be used only when REP statements cause a control section to be
increased in length. The format of an ICS statement is shown in Figure 13.
An ICS statement must be placed at the front of the file or TEXT file.

<table>
<thead>
<tr>
<th>Column</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X'02' (12-2-9 punch). Identifies this as a loader control statement.</td>
</tr>
<tr>
<td>2-4</td>
<td>ICS -- identifies the type of load statement.</td>
</tr>
<tr>
<td>5-15</td>
<td>Blank.</td>
</tr>
<tr>
<td>16</td>
<td>, (comma).</td>
</tr>
<tr>
<td>17-24</td>
<td>Control section name -- left-justified in these columns.</td>
</tr>
<tr>
<td>25-28</td>
<td>Hexadecimal length in bytes of the control section. This must not be less than the actual length of the previously specified control section. It must be right-justified in columns with unused leading columns filled with zeros.</td>
</tr>
<tr>
<td>29-72</td>
<td>May be used for comments or left blank.</td>
</tr>
</tbody>
</table>

Figure 13 (Part 1 of 2). ICS Statement Format
**Column** | **Contents**  
---|---  
73-80 | Not used by the loader. You may leave these columns blank or insert program identification for your own convenience.

Figure 13 (Part 2 of 2). ICS Statement Format

**Set Location Counter (SLC) Statement:** The SLC statement sets the location counter used with the loader. The file loaded after the SLC statement is placed in virtual storage beginning at the address set by this SLC statement. The SLC statement has the format shown in Figure 14. It sets the location counter in one of three ways:

1. With the absolute virtual address specified as a hexadecimal number in columns 7-12.

2. With the symbolic address already defined as a program name or entry point. This is specified by a symbolic name punched in columns 17-22.

3. If both a hexadecimal address and a symbolic name are specified, the absolute virtual address is converted to binary and added to the address assigned to the symbolic name; the resulting sum is the address to which the loader’s location counter is set. For example, if 0000F8 was specified in columns 7-12 of the SLC card image and GAMMA was specified in columns 17-22 and columns 23-24 are blank, where GAMMA has an assigned address of 006100 (hexadecimal), the absolute address in columns 7-12 is added to the address assigned to GAMMA giving a total of 0061F8. Thus, the location counter would be set to 0061F8.

<table>
<thead>
<tr>
<th>Column</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X'02' (12-2-9 punch). Identifies this as a loader control statement.</td>
</tr>
<tr>
<td>2-4</td>
<td>SLC -- identifies the type of load statement.</td>
</tr>
<tr>
<td>5-6</td>
<td>Blank.</td>
</tr>
<tr>
<td>7-12</td>
<td>Hexadecimal address to be added to the value of the symbol, if any, in columns 17-22. It must be right-justified in these columns, with unused leading columns filled with zeros.</td>
</tr>
<tr>
<td>13-16</td>
<td>Blank.</td>
</tr>
<tr>
<td>17-24</td>
<td>Symbolic name whose assigned location is used by the loader. Must be left-justified in these columns. If blank, the address in the absolute field is used.</td>
</tr>
<tr>
<td>25-72</td>
<td>May be used for comments or left blank.</td>
</tr>
</tbody>
</table>

Figure 14 (Part 1 of 2). SLC Statement Format
Figure 14 (Part 2 of 2). SLC Statement Format

Replace (REP) Statement: A REP statement allows instructions and constants to be changed and additions made. The REP statement must be punched in hexadecimal code. The format of a REP statement is shown in Figure 15.

<table>
<thead>
<tr>
<th>Column</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>73-80</td>
<td>Not used by the loader. You may leave these columns blank or insert program identification for your own convenience.</td>
</tr>
</tbody>
</table>

Figure 15. REP Statement Format

The data in columns 17-70 (excluding the commas) replaces what has already been loaded into virtual storage, beginning at the address specified in columns 7-12. REP statements are placed in the file either (1) immediately preceding the last statement (END statement) if the text deck does not contain relocatable data such as address constants, or (2) immediately preceding the first RLD (relocatable dictionary) statement if there is relocatable data in the text deck. If additions made by REP statements increase the length of a control section, an ICS statement, which defines the total length of the control section, must be placed at the front of the deck.
**Set Page Boundary (SPB) Statement:** An SPB statement instructs the loader to update the location counter to point to the next page boundary. The SPB statement has the format shown in Figure 16. This statement can be placed before, between, or after object modules or other control statements.

<table>
<thead>
<tr>
<th>Column</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X'02' (12-2-9 punch). Identifies this as a loader control statement.</td>
</tr>
<tr>
<td>2-4</td>
<td>SPB -- identifies the type of load statement.</td>
</tr>
<tr>
<td>5-72</td>
<td>May be used for comments or left blank.</td>
</tr>
<tr>
<td>73-80</td>
<td>Not used by the loader. This field may be left blank or program identification may be inserted.</td>
</tr>
</tbody>
</table>

Figure 16. SPB Statement Format

**Responses:**

DMSLI0740I Execution begins ...  

START was specified with LOAD and the loaded program starts execution. Any further responses are from the program.

INVALID CARD - xxx...xxx  

INV was specified with LOAD and an invalid statement was found. The message and the contents of the invalid statement (xxx...xxx) are listed in the file LOAD MAP. The invalid statement is ignored and loading continues.

**Messages and Return Codes:**

- DMSLGT002I File fn TXTLIB not found
- DMSLIO001E No filename specified [RC = 24]
- DMSLIO002E File[es] fn TXTLIB not found [RC = 28]
- DMSLIO003E Invalid option: option [RC = 24]
- DMSLIO005E No option specified [RC = 24]
- DMSLIO021E Entry point name not found [RC = 40]
- DMSLIO029E Invalid parameter parameter in the option option field [RC = 24]
- DMSLIO055E No entry point defined [RC = 40]
- DMSLIO066E File fn ft [fm] contains invalid record formats [RC = 32]
- DMSLIO099E CMS/DOS environment not active [RC = 40]
- DMSLIO104S Error nn reading file fn ft fm from disk [RC = 100]
- DMSLIO105S Error nn writing file fn ft fm on disk [RC = 100]
- DMSLIO109S Virtual storage capacity exceeded [RC = 104]
- DMSLIO116S Loader table overflow [RC = 104]
- DMSLIO168S Pseudo register table overflow [RC = 104]
- DMSLIO169S ESD data referenced by name card is missing [RC = 32]
<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSLIO201W</td>
<td>The following names are undefined: namelist [RC = 4]</td>
</tr>
<tr>
<td>DMSLIO202W</td>
<td>Duplicate identifier identifier [RC = 4]</td>
</tr>
<tr>
<td>DMSLIO203W</td>
<td>SET LOCATION COUNTER name undefined [RC = 4]</td>
</tr>
<tr>
<td>DMSLIO206W</td>
<td>Pseudo register alignment error [RC = 4]</td>
</tr>
<tr>
<td>DMSLIO623S</td>
<td>Module cannot be loaded at location location--this area is available for system use only [RC = 88]</td>
</tr>
<tr>
<td>DMSLIO625S</td>
<td>There are too many items that require relocation to save all of the RLD information [RC = 104]</td>
</tr>
<tr>
<td>DMSLIO749W</td>
<td>There are too many comments in text files to save all of the history information [RC = 4]</td>
</tr>
<tr>
<td>DMSLIO907T</td>
<td>I/O error on file fn ft fm [RC = 256]</td>
</tr>
<tr>
<td>DMSSTT062E</td>
<td>Invalid character char in fileid fn ft [RC = 20]</td>
</tr>
</tbody>
</table>
LOADLIB

Use the LOADLIB command to list, copy, or compress a CMS LOADLIB. CMS LOADLIBs can be merged, and specified members can optionally be selected or excluded during the merge.

The format of the LOADLIB command is:

```
LOADLIB [LIST [COMPRESS [fileid1]] [COPY [fileid2] [fileid3]] [(options)]]
```

Options:

- **TERM**
- **PRINT**
- **DISK**
- **REPLACE**
- **MODIFY**

SYSIN Control Statements (COPY function only):

- **SELECT**
- **EXCLUDE**

where:

**LIST**

lists by member name, the contents of the CMS LOADLIB specified by fileid1, and gives a hexadecimal representation of each member's size. Specifying a fileid1 that is a concatenated SYSUT1 results in an error message indicating that this is not supported.

**COMPRESS**

recreates a LOADLIB with the same name as the specified file (fileid1), and deletes all obsolete members from the new data set.

**COPY**

copies members of fileid1 into fileid2. If fileid2 already exists, MODIFY or REPLACE must be specified. If you specify MODIFY, existing members are not replaced in the output data set, but new members are added. If you specify REPLACE, existing members are replaced in the output data set and new members are added.

You must specify SYSIN control statements. If you do not specify SYSIN control statements in a SYSIN dataset (fileid3), you will be prompted for them at the terminal with the message: "ENTER:"
LOADLIB

Note: You may specify the LOADLIB function (LIST, COMPRESS, COPY) either on the command line or in the SYSIN data set (fileid3). If you specify the function in the SYSIN data set, you must issue the FILEDEF command for fileid1, fileid2 (if required), and fileid3 before you issue the LOADLIB command. However, if you specify the function on the command line, fileid1, and optionally, fileid2 and fileid3 may be specified either on the command line or defined via FILEDEF commands. Any FILEDEF commands issued by the user remain in effect after the command function completes. During subsequent use of LOADLIB functions, file definitions which have not been cleared or reissued may override the file identifiers entered in the LOADLIB command line.

fileid1

is the filename, filetype, and filemode of the input LOADLIB. This data set is referred to as the SYSUT1 data set. SYSUT1 is always required. An OS load library may not be specified as input.

fileid2

is the filename, filetype, and filemode of the output LOADLIB. This data set is referred to as the SYSUT2 data set. If the SYSUT2 data set already exists, either MODIFY or REPLACE must be specified. If a SYSUT2 data set is not specified, LOADLIB SYSUT2 A (or the filemode of the first available read/write disk) is the default. When the default SYSUT2 file is used and no errors occur, fileid1 is erased and the new file is renamed fileid1. SYSUT2 is ignored for the LIST or COMPRESS functions.

fileid3

is the filename, filetype, and filemode of the control data set. This data set is referred to as the SYSIN data set. If no SYSIN data set is specified, the user is prompted at the terminal to enter LOADLIB functions or SYSIN COPY control statements.

Options Entered in the Command Line:

TERM
directs printer output to the terminal. TERM is the default.

PRINT
directs printer output to the printer.

DISK
directs printer output to disk. The DISK option creates a file named LOADLIB LISTING *, where "*" is the filemode of the first available read/write disk.

REPLACE
replaces existing members of a data set and adds new members.

MODIFY
does not replace existing members of a data set; adds new members.
SYSIN Control Statements for the Copy Function:

SELECT
copies only selected members of a data set. Each member to be copied must be named in a separate line entry following the SELECT statement. Note that if you specify the SELECT statement, the LOADLIB command does not replace existing members of a data set. If you want to replace an existing member of a data set, you must specify (R) immediately following the member name.

EXCLUDE
copies a whole data set except for a few members. Each member to be excluded must be named in a separate line entry following the EXCLUDE statement. You can exclude up to 256 members for each COPY function.

Note: Indicate the end of control statements from the terminal by entering a null line; EOF serves this purpose in a SYSIN file. If you want to copy an entire data set, specify COPY and enter a null line at the terminal (or include a blank line in a SYSIN file). To avoid unexpected results, clear the file definitions used by the COPY function before specifying new file identifiers in subsequent LOADLIB commands.

Usage Notes:

1. If a LOADLIB COPY or COMPRESS into an existing LOADLIB results in a CMS ABEND001, check the integrity of the directory. If the file, $PDSTEMP LOADLIB, exists on your disk, do not erase it. The $PDSTEMP LOADLIB file contains the updated directory for the LOADLIB. Reissue another LOADLIB COPY or COMPRESS command where the modified output LOADLIB is the SYSUT1 data set and omit the SYSUT2 data set from the command input. If the command is successful, the LOADLIB's directory will be restored.

2. To select or exclude members of a data set, enter the LOADLIB COPY command and press ENTER. When you receive the status message "VM READ" or "Enter a command or press a PF or PA key," type in SELECT or EXCLUDE (or press ENTER to copy the entire data set). When you receive another status message "VM READ" or "Enter a command or press a PF or PA key," type the name of the first data set member to be selected or excluded and press ENTER. For each successive status message of "VM READ" or "Enter a command or press a PF or PA key," type the name of the dataset to be selected or excluded. When you have specified each data set memeber that you wish to select or exclude, enter a null line to end the command.

3. Although SELECT and EXCLUDE are both valid control statements for the LOADLIB COPY command, they must be used separately. Interchanging SELECT and EXCLUDE statements within the same command routine may lead to unpredictable results.
Responses:

MEMBER - member name HAS BEEN COPIED|EXCLUDED
ALIAS - alias name HAS BEEN COPIED|EXCLUDED
ALIAS - alias name NOT COPIED. MEMBER member name FOR THE ALIAS NOT FOUND. An EXCLUDE or SELECT statement controlled the COPY function, and the SYSUT2 data set did not contain the parent member for the alias. The member may not have been moved from the SYSUT1 data set because it was excluded, it was not specified in the SELECT list, or the member name did not precede the alias name in the SELECT list.
MEMBER - member name HAS BEEN REPLACED IN DATA SET
MEMBER - member name DOES NOT EXIST BUT HAS BEEN ADDED TO DATA SET. REPLACE was specified but the member was not in the output data set, therefore the member was added to the output data set.
MEMBER - member name COPY UNSUCCESSFUL
An error occurred while trying to add/replace the member in the output data set. (For example, if MODIFY was specified and the member already existed in the output data set.) The COPY continues with the next member to be copied.
MEMBER - member name NOT FOUND
The member requested was not found in the input data set.
MEMBER - member name NOT COPIED. WRONG LENGTH NOTE LIST FOUND.
MEMBER - member name NOT COPIED. NOTE LIST UPDATE ERROR.
USER TTR WAS NOT UPDATED
NOTE LIST TTR OR RECORD WAS NOT UPDATED

Messages and Return Codes:

DMSUTL003E Invalid option: option [RC=24]
DMSUTL014E Invalid function function [RC=24]
DMSUTL024E File fn ft fm already exists [RC=28]
DMSUTL032E Invalid filetype ft [RC=24]
DMSUTL037E Output disk mode is accessed as read/only [RC=36]
DMSUTL039E No entries in library fn ft fm [RC=32]
DMSUTL042E No fileid(s) specified [RC=24]
DMSUTL047E No function specified [RC=24]
DMSUTL054E Incomplete fileid specified [RC=24]
DMSUTL065E option option specified twice [RC=24]
DMSUTL066E option1 and option2 are conflicting options [RC=24]
DMSUTL069E Disk mode is not accessed [RC=36]
DMSUTL073E Unable to open file ddname [RC=28]
DMSUTL188W SYSUT2 header is invalid because of blocksize incompatibility; user action required [RC=4]
DMSUTL189E The LIST function of the LOADLIB command does not support concatenated SYSUT1 [RC=24]
DMSUTL901T Unexpected error at vstor1: plist function fn ft fm at vstor2, base vstor3, rc=nn [RC=256]
DMSUTL907T I/O error on file fn ft fm [RC=256]
LOADMOD

Use the LOADMOD command to load a MODULE file into storage. The file must be in nonrelocatable format as created by the GENMOD command.

The format of the LOADMOD command is:

```
LOADMod fn [MODULE [fm]]
```

where:

fn
is the filename of the file to be loaded into storage. The filetype must be MODULE.

fm
is the filemode of the module to be loaded. If not specified, or specified as an asterisk, all your disks are searched for the file.

Usage Notes:

1. You can use the LOADMOD command when you want to debug a CMS MODULE file. After the file is loaded, you may set address stops or breakpoints before you begin execution with the START command; for example:

```
loadmod progl
cp adstop 210ae
start
```

2. If a MODULE file was created using the DOS option of the GENMOD command, the CMS/DOS environment must be active when it is loaded. If it was created using the OS option (the default), the CMS/DOS environment must not be active when it is loaded.

3. MODULE files created with the ALL option, or with SYSTEM option and loaded into the transient area, may be loaded regardless of whether the CMS/DOS environment is active. If the LOADMOD command is called from a program, the loading is also done regardless of whether the CMS/DOS environment is active.

4. When in CMS SUBSET mode, a LOADMOD into the user area would result in return code 32. For more information on CMS SUBSET mode,
see the VM/SP CMS User’s Guide and VM/SP CMS for System Programming.

5. For detailed information on how the STR flag resets storage, see VM/SP CMS for System Programming.

Responses:

None.

Messages and Return Codes:

<table>
<thead>
<tr>
<th>Message Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSMOD001E</td>
<td>No filename specified [RC = 24]</td>
</tr>
<tr>
<td>DMSMOD002E</td>
<td>File([s]) fn [ft [fm]] not found [RC = 28]</td>
</tr>
<tr>
<td>DMSMOD018E</td>
<td>No load map available [RC = 40]</td>
</tr>
<tr>
<td>DMSMOD032E</td>
<td>Invalid filetype ft [RC = 24]</td>
</tr>
<tr>
<td>DMSMOD037E</td>
<td>Disk mode([vdev]) is accessed as read/only [RC = 36]</td>
</tr>
<tr>
<td>DMSMOD069E</td>
<td>Disk mode not accessed [RC = 36]</td>
</tr>
<tr>
<td>DMSMOD070E</td>
<td>Invalid parameter parameter [RC = 24]</td>
</tr>
<tr>
<td>DMSMOD104S</td>
<td>Error nn reading file fn ft fm from disk [RC = 100]</td>
</tr>
<tr>
<td>DMSMOD109S</td>
<td>Virtual storage capacity exceeded [RC = 104]</td>
</tr>
<tr>
<td>DMSMOD114E</td>
<td>fn ft fm not loaded; CMS/DOS environment [not] active [RC = 40 or -0005]</td>
</tr>
<tr>
<td>DMSMOD116S</td>
<td>Loader table overflow [RC = 104]</td>
</tr>
</tbody>
</table>
MACLIB

Use the MACLIB command to create and modify CMS macro libraries.

The format of the MACLIB command is:

<table>
<thead>
<tr>
<th>MAClib</th>
<th>GEN</th>
<th>ADD</th>
<th>REP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>libname</td>
<td>fn1</td>
<td>fn2...</td>
</tr>
</tbody>
</table>

| DEL | libname | membername1 | membername2... |
| COMP | libname |
| MAP | libname | membername1 | membername2... |

Options:

- DISK
- PRINT
- TERM
- STACK [FIFO, LIFO]
- FIFO
- LIFO
- XEDIT

where:

- GEN
  generates a CMS macro library.

- ADD
  adds members to an existing macro library. No checking is done for duplicate names, entry points, or CSECTS.

- REP
  replaces existing members in a macro library.

- DEL
  deletes members from a macro library. If more than one member exists with the same name, only the first entry is deleted.
MACLIB

COMP
  compacts a macro library.

MAP
  lists certain information about the members in a macro library.
  Available information includes member name, size, and location
  relative to the beginning of the library.

libname
  is the filename of a macro library. If the file already exists, it must
  have a filetype of MACLIB; if it is being created, it is given a filetype
  of MACLIB.

fn1 [fn2...]  
  are the names of the macro definition files to be used. A macro
  definition file must reside on a CMS disk and its filetype must be
  either MACRO or COPY. Each file may contain one or more macros
  and must contain fixed-length, 80-character records.

membername1[membername2...]  
  are the names of the macros that exist in a macro library.

MAP Options:

The following options specify where the output of the MAP function is sent.
Only one option may be specified. If more than one option is specified, only
the first one given is used.

DISK
  writes the MAP output on a CMS disk with the file identifier of
  "libname MAP A1." If a file with that name already exists, the old file
  is erased. If no option is specified, DISK is the default.

PRINT
  writes the file "libname MAP A1" to your A-disk and spools a copy to
  the virtual printer.

TERM
  displays the MAP output at the terminal.

STACK [FIFO]
  places the MAP output in the program stack. It can be stacked either
  FIFO (first in first out) or LIFO (last in first out). The default order is
  FIFO.

FIFO
  places the MAP output in the program stack. It is stacked FIFO. The
  options STACK, STACK FIFO, and FIFO are equivalent.
LIFO
places the MAP output in the program stack. It is stacked LIFO. This option is equivalent to STACK LIFO.

XEDIT
inserts the MAP output into the file being edited. Each map line is 130 characters and contains the following twice on the line: the member name, index, size, library filename, library filetype, and library filemode. The XEDIT option does not return the header record. This option is only valid when MACLIB MAP is issued from the XEDIT environment.

Usage Notes:

1. When a MACRO file is added to a MACLIB, the membername is taken from the macro prototype statement. If there is more than one macro definition in the file, each macro is written into a separate MACLIB member. If the filetype is COPY and the file contains more than one macro, each macro must be preceded by a control statement of the following format:

   *COPY membername

   The name on the control statement is the name of the macro when it is placed in the macro library. If there is only one macro in the COPY file and it is not preceded by a COPY control statement, its name (in the macro library) is the same as the filename of the COPY file. If there are several macro definitions in a COPY file and the first one is not preceded by a COPY control statement, the entire file is treated as one macro.

2. If any MACRO file contains invalid records between members, the MACLIB command displays an error message and terminates. Any members read before the invalid card is encountered are already in the MACLIB. The MACLIB command ignores CATALS, END, and /* records when it reads MACRO files created by the ESERV program.

3. If you want a macro library searched during an assembly or compilation, you must identify it using the GLOBAL command before you begin compiling.

4. The MACLIBs distributed with the CMS system are: CMSLIB, DMSSP, DOSMACRO, OSMACRO, OSMACRO1, OSVSAM, and TSOMAC.

5. The DISK, PRINT, or TERM options will erase the old MAP file, if one exists.

6. If you delete the last remaining member of a maclib, the maclib is erased.
7. If any accessed disk contains a MACRO file with the same filename as the COPY file on your A-disk, the MACLIB’s REP, ADD, and GEN functions use the MACRO version.

8. The XEDIT option is valid only when MACLIB MAP is issued from the XEDIT environment. The edited file must either be fixed format with a logical record length of 130 or variable length format. The information replaces the current line and below until the END OF FILE is reached. Then, the remaining text (if any) is inserted before the END OF FILE.

Responses:

When you enter the MACLIB MAP command with the TERM option, the names of the library members, their sizes, and their locations in the library are displayed.

MACRO INDEX SIZE
name  loc   size
    :   :    :
    :   :    :

Messages and Return Codes:

DMSLBM001E No filename specified [RC = 24]
DMSLBM002E File fn MACLIB not found [RC = 28]
DMSLBM002W File fn ft [fm] not found [RC = 4 or 28]
DMSLBM003E Invalid option: option [RC = 24]
DMSLBM013W Member membername not found in library libname [RC = 4]
DMSLBM014E Invalid function function [RC = 24]
DMSLBM037E Disk mode[dev] is accessed as read/only [RC = 36]
DMSLBM046E No library name specified [RC = 24]
DMSLBM047E No function specified [RC = 24]
DMSLBM056E File fn ft [fm] contains invalid record formats [RC = 32]
DMSLBM069E Disk mode not accessed [RC = 36]
DMSLBM070E Invalid parameter parameter [RC = 24]
DMSLBM104S Error nn reading file fn ft fm from disk [RC = 100]
DMSLBM105S Error nn writing file fn ft fm on disk [RC = 100]
DMSLBM109S Virtual storage capacity exceeded [RC = 104]
DMSLBM157S MACLIB limit exceeded[, last member added was membername] [RC = 88]
DMSLBM167S Previous MACLIB function not finished [RC = 88]
DMSLBM213W Library fn MACLIB fm not created, or erased if empty [RC = 4]
DMSLBM688E XEDIT option only valid from XEDIT environment [RC = 24]
DMSLBM689E File must be F-format 130 or V-format [RC = 24]
DMSLBM907T I/O error on file fn ft fm [RC = 256]
MACLIST

Use the MACLIST command to display a list of all members in a specified macro library. You can issue CMS commands against the members directly from the displayed list. In the MACLIST environment, information that is normally provided by the MACLIB command (with the MAP option) is displayed under the control of the System Product editor. You can use XEDIT subcommands to manipulate the list itself.

The format of the MACLIST command is:

```
MACLIST MList libname [(options [])]
```

Options: [Append] [Compact] [PROFile fn]

where:

- `libname` is the filename of the CMS maclib for which information is to be displayed. The file must have a filetype of MACLIB.

Options:

- **Append** specifies that the list of members in this library should be appended to the existing list. This option is only valid from the MACLIST environment.

- **Compact** specifies that the library is to be compacted upon completion of MACLIST using the MACLIB COMP command.

- **NOCompact** specifies that the library is not to be compacted upon completion of MACLIST. This is the default.

- **PROFile fn** specifies the name of an XEDIT macro to be executed when XEDIT is invoked by the MACLIST command. If not specified, a macro named PROFMLST XEDIT is invoked. For more information on the
Usage Notes:

1. Tailoring the MACLIST Command Options

You can use the DEFAULTS command to set up options and/or override command defaults for MACLIST. However, the options you specify in the command line when entering the MACLIST command override those specified in the DEFAULTS command. This allows you to customize the defaults of the MACLIST command, yet override them when you desire. Refer to the DEFAULTS command description for more information.

2. Format of the List

When you invoke the MACLIST command you are placed in the XEDIT environment, editing a file “userid MACLIST A0.” A sample MACLIST screen is shown in the “Examples” section. Each line in this file contains:

- a command area
- member name
- index
- number of records in the member
- library filename, library filetype, and library filemode.

The full power of XEDIT is available to you while you issue commands against the list of members. For example, you may want to use XEDIT subcommands to scroll through the list of members to locate a particular member, etc.

However, some XEDIT subcommands are inappropriate in this environment. Subcommands that alter the format or the contents of “userid MACLIST” (for example, SET TRUNC, SET FTYPE, or SET LINEND) may cause unpredictable results.

3. Issuing Commands From the List

On a full screen display, you can issue commands directly from the line on which a member name is displayed. You do this by moving the cursor to the line that describes the member to be used by the command, typing the command in the space provided to the left of the member name, and then pressing the ENTER key to execute the command.
If a command is longer than the command space provided on the screen, just continue typing over the information in the line. You may type over the entire line displayed, up to column 79.

The ENTER key is set to EXECUTE, which is described below, under “Special Commands.” When you press the ENTER key, all commands typed on one screen are executed, and the screen is restored to its previous state. However, the list is updated to reflect the current status of the members (see “Responses”).

You may want to enter commands from the MACLIST command line before executing commands that are typed on the list. To do this, move the cursor to the command line by using the PF12 key (instead of the ENTER key). After typing a command on the command line and pressing ENTER, you can use PF12 to move the cursor back to its previous position on the list.

Another way to issue commands that make use of the members displayed is to issue EXECUTE from the MACLIST command line. A complete description of EXECUTE follows, in the section “Special Commands.”

4. Default Key Settings

The PROFMLST XEDIT macro is executed when the MACLIST command is invoked, unless you specified a different macro as an option in the MACLIST command. It sets the keys to the following values:

ENTER Execute command(s) typed on member line(s) or on the command line. (The ENTER key is set by the XEDIT subcommand, SET ENTER IGNORE MACRO EXECUTE).

PF 1 Help Display MACLIST command description.

PF 2 Refresh Update the list to indicate new members, deleted members, etc., using the same parameters as those specified when MACLIST was invoked.

PF 3 Quit Exit from MACLIST.

PF 4 Sort(name) Sorts by member name.

PF 5 Sort(index) Sorts by index, largest first.

PF 6 Sort(size) Sorts by size, largest first.

PF 7 Backward Scroll back one screen.

PF 8 Forward Scroll forward one screen.
MACLIST

PF 9  F1/n  Issue the command FILELIST /n ** at the cursor, so that a list is displayed, containing all files that have a filename that is the same as the membername displayed on the line containing the cursor. (all filetypes and filemodes).

PF 10  Not assigned.

PF 11  XEDIT  Edit the member where the cursor is placed.

PF 12  Cursor  If the cursor is in the file area, move it to the command line. If the cursor is on the command line, move it back to its previous location in the file (or to the current line).

Note: On a terminal equipped with 24 PF keys, PF keys 13 to 24 are assigned the same values as PF keys 1 to 12 as discussed here.

In addition to setting the above PF keys, the PROFMLST XEDIT macro sets synonyms that you can use to sort your MACLIST list. The synonyms are:

SINDEX  Sorts the list by index (greatest to least) within a library.

SLIB  Sorts the list alphabetically by library fileid, then by membername and index.

SNAME  Sorts the list alphabetically by member name and then by library fileid and index.

SSIZE  Sorts the list by member size (number of records, greatest to least).

5. If you want to issue MACLIST from an EXEC program, you should precede it with the EXEC command; that is, specify

   exec maclist

Responses:

When a command is executed, one of the following symbols is displayed in the "Cmd" space to the left of the file for which the command was executed.

*  Means the command executed successfully (RC = 0).

*n  Is the return code from the command executed (RC = n).

*?  Means that the command was an unknown CP/CMS command (RC = -3).

The following responses can also appear directly on the MACLIST screen:
* Library 'libname libtype libmode' not found. *
* membername has been discarded.
* membername ** Member is a duplicate entry. Invalid for EXECUTE **
* membername ** Member is no longer in the library. **
Member membername has been discarded.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSWML002E File fn MACLIB * not found
DMSWML651E APPEND must be issued from MACLIST
DMSWML668E The APPEND option must be specified alone

Special Commands Used in the MACLIST Environment

Two commands, EXECUTE and DISCARD, make use of the list of members displayed by the MACLIST command. EXECUTE can be used only in the FILELIST, MACLIST, and RDRLIST command environments, while DISCARD can be used only in the FILELIST, MACLIST, PEEK, and RDRLIST command environments.

Using EXECUTE in MACLIST

Use EXECUTE (an XEDIT macro) to issue CP/CMS commands (or EXECs) that make use of member names displayed by MACLIST.

EXECUTE may be used in two ways. First, on a display terminal, the command(s) to be executed can be typed directly on the MACLIST screen and “EXECUTE” entered either on the command line or from a PF key or by pressing the ENTER key. Second, the command to be executed can be typed in the command line at the bottom of the screen, following “EXECUTE” (as one of its operands). The command is then executed against one or more members names in the list, beginning with the current line of the list.

The format of the EXECUTE macro is:

```
EXECUTE [Cursor | lines] [command]
```

where:

Cursor

means that a command is to be executed against the line that contains the cursor. The command can either be typed on the line that describes the member, or it can be typed as an operand of EXECUTE. The CURSOR operand is valid only on display terminals and is
particularly useful when assigned to a PF key. For example, if EXECUTE CURSOR XEDIT is assigned to a PF key, you can place the cursor on the line describing the member you want to edit and then press the PF key.

**lines**

is the number of lines in the file the command is to be executed for, starting with the current line of the list. If a command is specified, the default is one (1). You can specify an asterisk (*), which means “execute this command on all lines, from the current line to the end of the file.”

**command**

is a CMS or CP command (or any program or EXEC) that makes use of member names in the list. You can either type out the command operands, or you can use the symbols described below to represent the membername or library filename, filetype and/or filemode. (See the usage note, “Using Symbols as Part of a Command.”)

### Usage Notes:

1. **Entering Commands on a Full Screen Display**

   You can type commands that operate on member names in the list directly on the lines of the MACLIST display. When you enter EXECUTE (either from the command line or by pressing the ENTER key), all commands typed on the lines in the file displayed on the current screen are executed.

   Note that when a command is typed on the MACLIST, FILELIST, or RDRLIST screen, EXECUTE rebuilds the line and compares it with the line displayed on the screen. The line is scanned from right to left, and the first character that is different signals the end of the command. Therefore, if the member information has been changed (as the result of a previous command), but this information has not yet been updated (by pressing PF2 to refresh the screen), EXECUTE will incorrectly interpret the information on the screen. An example follows.

If a MACLIST contains the following:

<table>
<thead>
<tr>
<th>Cmd</th>
<th>Member name</th>
<th>Index</th>
<th>Records</th>
<th>Library name</th>
<th>Library type</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLOWER</td>
<td>STOP</td>
<td>YIELD</td>
<td>435</td>
<td>607</td>
<td>135</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>5</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MYLIB</td>
<td>MYLIB</td>
<td>MYLIB</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MACLIB</td>
<td>MACLIB</td>
<td>MACLIB</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A1</td>
<td>A1</td>
<td>A1</td>
<td></td>
</tr>
</tbody>
</table>

The following command:

```
= = = > maclib add mylib swerve
```

will add SWERVE to MYLIB library. After entering the command the list remains unchanged.
### 2. Entering Commands on the Command Line

Another way to issue commands that make use of the member names displayed is to move the current line to the first (or only) member you want the command to use, and then to issue EXECUTE (in the form “EXECUTE lines command”) from the XEDIT command line. This method may be used on both display and typewriter terminals.

For example:

First move the current line (by using XEDIT subcommands like UP or DOWN) to the first member name you want to use in the command. On a full screen display, the current line is the first member name on the screen. Then (in the XEDIT command line) you type:

```plaintext
execute n xedit
```

where “n” is the number of members to be edited, starting with the current line. (You can use any command, not just XEDIT.)

**Note:** You can use XEDIT synonyms or macros to make issuing common commands easier. For example, you might want to set up a command “EX” to be a synonym for “EXECUTE 1 XEDIT.”

### 3. Using Symbols as Part of a Command

Symbols can be used to represent operands in the command to be executed. They can be used in the commands typed on the screen, or as part of the command in EXECUTE (on the command line). Symbols are needed if the command to be executed has operands or options that follow the fileid. Examples of using symbols are in the “Examples” section, below.

The following symbols can be used:

- `/` means the libname libtype libmode (MEMBER membername that is displayed on the line.)

---

### Table: MACLIST

<table>
<thead>
<tr>
<th>Cmd</th>
<th>Member name</th>
<th>Index</th>
<th>Records</th>
<th>Library name</th>
<th>Library type</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLOWER</td>
<td></td>
<td>435</td>
<td>5</td>
<td>MYLIB</td>
<td>MACLIB</td>
<td>A1</td>
</tr>
<tr>
<td>STOP</td>
<td></td>
<td>607</td>
<td>5</td>
<td>MYLIB</td>
<td>MACLIB</td>
<td>A1</td>
</tr>
<tr>
<td>YIELD</td>
<td></td>
<td>135</td>
<td>54</td>
<td>MYLIB</td>
<td>MACLIB</td>
<td>A1</td>
</tr>
</tbody>
</table>

Pressing PF2 updates the MACLIST and SWERVE now appears on the list.
/l means the library filename displayed on the line.
/t means the library filetype displayed on the line.
/m means the library filemode displayed on the line.
/n means the member name displayed on the line.
/o means execute the line as is, and do not append anything.

Any combinations of symbols can be used. For example:

/l/t means the library filename followed by library filetype.

/l/t means the library filename followed by library filetype.

/t/l means the library filetype followed by library filename.

/l/tm (MEMBER /n is equivalent to / alone.

After the command(s) have been executed, EXECUTE updates the status of the list with any changed information and uses asterisks and return codes to indicate command completion. See Responses above.

4. Special Symbols Used Alone

The following special symbols can be typed alone on the lines of the MACLIST display. They have the following meanings:

= means execute the previous command for this member. Commands are executed starting at the top of the screen. For example, suppose you enter the DISCARD command on the top line. You can then type an equal sign on any other line(s). Those member preceded by equal signs are discarded when the EXECUTE command is entered (from the command line or by pressing the ENTER key).

? means display the last command executed. The command is displayed on the line in which the ? is entered.

/ means make this line the current line. (On the MACLIST screen, the current line is the first member name on the screen.)

5. If a member is a duplicate name and it is not the first one found in the maclib, you cannot issue any CMS commands, XEDIT subcommands, or any special symbols (=, ?, and /) for that member.
Messages and Return Codes for EXECUTE:

DMSWEX526E  Option CURSOR valid in display mode only
DMSWEX543E  Invalid number: number
DMSWEX561E  Cursor is not on a valid data field
DMSWEX651E  EXECUTE must be issued from RDRLIST, FILELIST or MACLIST
DMSWEX654E  Invalid symbol symbol; {/0 must be specified alone|invalid character char following / symbol}

On a typewriter terminal only:

Executing: command
+ + + E(nn) + + +

Using DISCARD in MACLIST

Use the DISCARD command to remove a member from a library. (DISCARD is equivalent to the CMS command MACLIB DEL.) DISCARD can either be typed in the command area of the line that describes the member you want discarded, or it can be entered from the command line (at the bottom of the screen). A message will indicate that a member has been deleted.

The format of the DISCARD command as used in the MACLIST environment is:

| DISCARD     | [libname libtype libmode (MEMBER membername)] |

where:

libname libtype libmode (MEMBER membername) is the name of the member to be deleted. If DISCARD is typed on the line that describes the member to be discarded, no other identifier should be specified. The library name, library type, library mode and (MEMBER membername) are appended automatically.

Messages and Return Codes for DISCARD:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSWDC651E  DISCARD must be issued from FILELIST, RDRLIST, MACLIST or PEEK
DMSWDC653E  Error executing command; rc = nn
DMSWDC752E  Unable to delete member membername from fn ft fm
Examples:

The following MACLIST screen was created by issuing the MACLIST command as follows.

```
maclist mylib
```

Note that the members are sorted alphabetically by member name. Members with the same name are then sorted by index number (least to greatest).

![Sample MACLIST Screen](image)

Figure 17. Sample MACLIST Screen

**Examples of Using Symbols:**

The following examples show how symbols can be used to represent operands in a command. The values substituted for the symbols and the resulting command are shown. In each case, the command can be entered in either of the following ways:

- Typed in the “Cmd” area of the screen. The command is executed either by entering EXECUTE on the XEDIT command line and then pressing ENTER, or simply by pressing ENTER.

- Entered from the XEDIT command line, as an operand of EXECUTE (in the form “EXECUTE lines command”).

If a symbol is not specified, the libname, libtype, libmode, and (MEMBER membername are appended automatically to the command.
<table>
<thead>
<tr>
<th>COMMAND</th>
<th>RESULTING COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>discard /ltm (member yield</td>
<td>MACLIB DEL MYLIB (YIELD</td>
</tr>
<tr>
<td>listfile /lt *</td>
<td>LISTFILE MYLIB MACLIB *</td>
</tr>
<tr>
<td>copyfile /ltm /l oldlib /m</td>
<td>COPYFILE MYLIB MACLIB A1 MYLIB OLDLIB A1</td>
</tr>
</tbody>
</table>
MAKEBUF

MAKEBUF

Use the MAKEBUF command to create a new buffer within the program stack.

The format of the MAKEBUF command is:

| MAKEBUF |

Usage Notes:

1. When you issue a MAKEBUF command, CMS returns as a return code the number of the program stack buffer just created. If you issue a MAKEBUF command in an EXEC that has the &ERROR statement in effect, the MAKEBUF return code causes the &ERROR statement to execute.

2. Use the WAITRD function to read lines from the buffers the MAKEBUF command creates. WAITRD first reads lines from the most recently created buffer. When the most recent buffer is exhausted, WAITRD reads the next most recent buffer. When all program stack buffers are exhausted, WAITRD reads from the terminal input buffer.

Responses:

None

Messages and Return Codes:

None
MAXIMIZE WINDOW

Use the MAXIMIZE WINDOW command to expand a window to the physical screen size.

The format of the MAXIMIZE WINDOW command is:

```
MAXimize WINdow [wname = ]
```

where:

`wname`

is the name of the window to be expanded. An "=" indicates that the topmost window is expanded. If this operand is not specified, "=" is assumed.

Usage Notes:

1. The RESTORE WINDOW command returns the window to its size and location prior to the maximize.

2. A window can be maximized whether or not it is connected to a virtual screen and whether or not the window is displayed.

3. A maximized window is positioned at line 1, column 1 of the physical screen.

4. A variable size window that is maximized still retains its variable size properties. Thus, depending on how many lines exist in the virtual screen to which the window is connected, the window may appear to be less than full screen size when it is displayed on the physical screen.

For example, suppose a variable size window is connected to line one of a virtual screen that contains three data lines. When the window is maximized:

- it moves to line 1, column 1;
- its width is the width of the physical screen; and,
- it contains only three data lines.
5. When you maximize a window so that it fills the entire screen and covers all other windows, you may not be able to enter commands. The WM window is automatically displayed, and the WMPF keys and command line are available to manipulate the window. Use the RESTORE WINDOW command to restore the window and the DROP WINDOW command to exit the WM environment.

Responses:

None.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSMAX386E Missing operand(s) [RC = 24]
- DMSMAX388E Invalid keyword: keyword [RC = 24]
- DMSMAX391E Unexpected operand(s): operand [RC = 24]
- DMSMAX921E Window wname is not defined [RC = 28]
MINIMIZE WINDOW

Use the MINIMIZE WINDOW command to reduce the size of the window to one line.

The format of the MINIMIZE WINDOW command is:

```
MINimize WINDOW [wname = ]
```

where:

`wname` is the name of the window to be reduced. An "=" indicates that the topmost window will be reduced. If this operand is not specified, "=" is assumed.

Usage Notes:

The RESTORE WINDOW command returns the window to its size and location prior to issuing the MINIMIZE command.

Responses:

None.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSMIN386E  Missing operand(s) [RC = 24]
- DMSMIN388E  Invalid keyword: `keyword` [RC = 24]
- DMSMIN391E  Unexpected operand(s): `operand` [RC = 24]
- DMSMIN921E  Window `wname` is not defined [RC = 28]
MODMAP

MODMAP

Use the MODMAP command to display the load map associated with the specified MODULE file.

The format of the MODMAP command is:

```
MODmap  fn
```

where:

- `fn` is the filename of the MODULE file whose load map is to be displayed. The filetype of the file must be MODULE; all of your accessed disks are searched for the specified file.

Usage Notes:

1. You cannot issue a MODMAP command for modules that are CMS transient area modules or that have been created with the NOMAP option of the GENMOD command.

2. When in CMS SUBSET mode, a MODMAP that requires a LOADMOD into the user area will receive a return code 32 from LOADMOD.

Responses:

The load map associated with the file is displayed at the terminal, in the format:

```
name  location
   .    .
   .    .
```

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSMDP001E No filename specified [RC = 24]
- DMSMDP070E Invalid parameter parameter [RC = 24]
MOREHELP

Use the MOREHELP command to obtain either additional or related information about the last valid HELP command you issued. This command is particularly helpful if you use a linemode terminal or if you cannot use a PF key to obtain DETAIL or RELATED information.

The format of the MOREHELP command is:

```
MOREhelp [ ( [optionA] [optionB] [ ] ) ]
```

<table>
<thead>
<tr>
<th>OptionA:</th>
<th>DETail</th>
<th>BRIef</th>
<th>RELated</th>
</tr>
</thead>
<tbody>
<tr>
<td>OptionB:</td>
<td>ALL</td>
<td>DESCript</td>
<td>FORMAT</td>
</tr>
<tr>
<td></td>
<td>OPTIONS</td>
<td>NOTES</td>
<td>ERRors</td>
</tr>
</tbody>
</table>

where:

OptionA:

DETail
displays the DETAIL layer of the HELP file. The amount of DETAIL information displayed is determined by your DEFAULTS selection of the subsetting options. The subsetting options are ALL or DESCRIPT, FORMAT, PARMs, OPTIONS, NOTES, and ERRORS. The options DESCRIPT, FORMAT, PARMs, OPTIONS, NOTES, and ERRORS can be specified in any combination.

BRIef
displays the BRIEF layer of the HELP file. If the BRIEF layer is not available, then you will see the next available layer of information.

RELated
displays the RELATED layer of the HELP file, if available.
MOREHELP

OptionB:

The following options can be specified with the DETAIL operand.

ALL
  displays all the DETAIL information of the HELP file. This includes the DESCRIPT, FORMAT, PARMs, OPTIONS, NOTES, and ERRORS sections. It does not include the BRIef and the RELated sections.

DESCript
  displays the description information of the HELP file.

FORMat
  displays the format information (the syntax).

PARMs
  displays the parameter section (explanation of the operands).

OPTions
  displays the options section (a list of available options with a brief description).

NOTEs
  displays the usage notes and example sections.

ERRors
  displays the error messages and response sections.

Usage Notes:

1. The DETAIL option displays the specified subset information (as preset by the DEFAULTS command). The initial default is set to display all six subset sections. If the file does not contain the DETAIL subset you have specified, then a message and the next available layer of HELP information will be displayed.

2. Additional information is based upon the last valid HELP command you entered. If you have not entered a valid HELP command, message DMSMOR353E, "No previous HELP command has been entered," will be displayed.

3. RELATED HELP information is available for a limited number of HELP files. However, if you wish to tailor your own HELP files to include this option, you may do so. Refer to “Tailoring the HELP facility” in the VM/SP CMS User’s Guide.

4. If there is no RELATED HELP information available, you will see a warning message. No information will be displayed.

5. If you want to issue MOREHELP from an EXEC program, you should precede it with the EXEC command; that is, specify
exec morehelp

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSMOR353E  No previous HELP command has been entered. Please enter HELP MOREHELP for information on the MOREHELP command. [RC = 4]

DMSMOR354E  RELATED information is not available for the last HELP command entered. [RC = 32]

DMSMOR639E  Error in routine routine; return code was xx
Use the MOVEFILE command to move data from any device supported by VM/SP to any other device supported by VM/SP.

The format of the MOVEFILE command is:

```
MOVEfile [inddname outddname] [PDS [ ]]
```

where:

`inddname` is the ddname representing the input file definition. If ddname is not specified, the default input ddname, INMOVE, is used.

`outddname` is the ddname representing the output file definition. If ddname is not specified, the default output ddname, OUTMOVE, is used.

Option:

`PDS` moves each of the members of the CMS MACLIB or TXTLIB or of an OS partitioned data set into a separate CMS disk file. Each CMS file has a filename equal to the member name and a filetype equal to the filetype of the output file definition.

Usage Notes:

1. Use the FILEDEF command to provide file definitions for the ddnames used in the MOVEFILE command. If you use the ddnames INMOVE and OUTMOVE on the FILEDEF commands, then you need not specify them on the MOVEFILE command line. For example:

   ```
   filedef inmove disk sys1 maclib b (member stow
   filedef outmove disk stow macro
   movefile
   ```

   copies the member STOW from the OS partitioned data set SYS1.MACLIB into the CMS file STOW MACRO.

   If you enter:
filedef indd reader
filedef outdd printer
movefile indd outdd

A file is moved from your virtual reader to your virtual printer.

2. To copy an entire OS partitioned data set into individual CMS files, you could enter:

filedef test2 disk sys1 maclib b
filedef macro disk
movefile test2 macro (pds

These commands copy members from the OS partitioned data set SYS1.MACLIB or the CMS file SYS1.MACLIB into separate files, each with a filename equal to the membername and a filetype of MACRO. Note that the output ddname was not specified in full, so that CMS assigned the default file definition (FILE ddname).

3. You cannot copy VSAM data sets with the MOVEFILE command.

4. The MOVEFILE command does not support data containing spanned records. Use of spanned records results in the error message DMSSOP036E and an error code of 7.

5. To copy an entire partitioned data set into another partitioned data set, use the COPYFILE command. If an attempt is made to use the MOVEFILE command without the PDS option for a partitioned data set, only the first member is copied and an end-of-file condition results. The resultant output file will contain all input records, including the header, until the end of the first member.

6. When using the MOVEFILE command to move members from CMS maclibs, note that each member is followed by a // record, which is a maclib delimiter. You can edit the file to delete the // record.

7. If you use the MOVEFILE command and FILEDEF command with the options DISP MOD and REC FM FB to add a file to the end of an existing OS simulated file, the user should erase the end-of-file mark at the end of the existing file. The end-of-file mark will be present only if the last physical record written was a short block.

8. The following record formats are supported for DOS files on FBA devices: fixed, fixed blocked, variable, variable blocked, and undefined. The FILEDEF for the input file must specify at least the REC FM and BLOCK; for fixed block files the LRECL must also be specified. Do not issue "SET DOS ON". If you do, MOVEFILE will result in an error message.

9. When copying a variable length data set (RECFM = V or VB) from an OS disk to a CMS disk, the logical record length (LRECL) of the file that is created on the CMS disk is equal to the size of the largest record in the data set being copied. If the file that is being created has a filemode of 4, the logical record length will be equal to the LRECL of...
the largest record plus 8 bytes. The actual LRECL of the new file can be determined by using the CMS LISTFILE command.

10. For OS compatibility of the output tape labels, you must specify LRECL and BLOCK/BLKSIZE in your output FILEDEF.

11. Any attempt to move an empty OS data set that has not been closed will cause unpredictable results.

**Default Device Attributes:**

If a record format (RECFM), blocksize (BLOCK), and logical record length (LRECL) are specified on the FILEDEF command, these values are used in the data control block (DCB) defining the characteristics of the move operation. If the FILEDEF was issued without a record format or blocksize specified, these values are determined according to the defaults listed in Figure 18. If the blocksize was not specified, the default blocksize is used. If the logical record length was not specified, the default logical record length is determined as follows: for an F or U record format, the logical record length equals the blocksize; for a V record format, the logical record length equals the blocksize minus 4.

<table>
<thead>
<tr>
<th>Device</th>
<th>Input ddname</th>
<th>Output ddname</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RECFM</td>
<td>Blocksize</td>
</tr>
<tr>
<td>Card Reader</td>
<td>F</td>
<td>80</td>
</tr>
<tr>
<td>Card Punch</td>
<td>NA(^2)</td>
<td>NA(^2)</td>
</tr>
<tr>
<td>Printer</td>
<td>NA(^2)</td>
<td>NA(^2)</td>
</tr>
<tr>
<td>Terminal</td>
<td>U</td>
<td>130</td>
</tr>
<tr>
<td>Tape(^1)</td>
<td>U</td>
<td>3600</td>
</tr>
<tr>
<td>Disk file</td>
<td>RECFM of file</td>
<td>Blocksize of file</td>
</tr>
<tr>
<td>Dummy</td>
<td>NA(^2)</td>
<td>NA(^2)</td>
</tr>
</tbody>
</table>

\(^1\)If the default record format and blocksize are used in a tape-to-tape move operation and an input record is greater than 3600 bytes, it is truncated to 3600 bytes on the output tape.

\(^2\)Not applicable.

Figure 18. Default Device Attributes for MOVEFILE Command
Responses:

DMSMVE225I PDS member *membername* moved

The specified member of an OS partitioned data set was moved successfully to a CMS file. This response is issued for each member moved when you use the PDS option.

DMSMVE226I End of PDS move

The last member of the partitioned data set was moved successfully to a CMS file.

DMSMVE706I Terminal input; type null line for end of data

The input ddname in the MOVEFILE specified a device type of terminal. This message requests the input data; a null line terminates input.

DMSMVE708I Disk file **fn ddname** A1 assumed for DDNAME **ddname**

No file definition is in effect for a ddname specified on the MOVEFILE command. The MOVEFILE issues the default FILEDEF command:

```
FILEDEF ddname DISK FILE ddname A1
```

If file ddname does not exist for the input file, MOVEFILE terminates processing.

Messages and Return Codes:

- **DMSMVE002E** File(s) [fn [ft [fm]]) not found [RC = 28]
- **DMSMVE003E** Invalid option: option [RC = 24]
- **DMSMVE007E** Output disk model([odev]) is accessed as read/only [RC = 40]
- **DMSMVE041E** Input and output files are the same [RC = 40]
- **DMSMVE069E** Output disk mode is not accessed [RC = 36]
- **DMSMVE070E** Invalid parameter parameter [RC = 24]
- **DMSMVE075E** Device devtype invalid for {input|output} [RC = 40]
- **DMSMVE086E** Invalid DDNAME ddname [RC = 24]
- **DMSMVE127S** Unsupported device for DDNAME [RC = 100]
- **DMSMVE128S** I/O error on input after reading nnn records; input error code on DDNAME [RC = 100]
- **DMSMVE129S** I/O error on output writing record number nnnn; output error code on DDNAME [RC = 100]
- **DMSMVE130S** Blocksize on V format file ddname is less than 9 [RC = 88]
- **DMSMVE232E** Invalid RECFM-spanned records not supported [RC = 88]
- **DMSMVG089E** Open error code nn on {fn|SYSaaa|tapn} [RC = 36]
NAMEFIND

Use the NAMEFIND command to display information from a names file, or to place that information in the program stack (for use by an EXEC or other program).

A names file has a filetype of NAMES and must be in the format described in the usage note below, "Format of a Names File." A "userid NAMES" file is a special names file, used by the NAMES, NOTE, SENDFILE, RECEIVE, and TELL commands, that makes it easier for you to communicate with other computer users. You can use the NAMES command to create a "userid NAMES" file. NAMEFIND searches a "userid NAMES" file, unless a different filename is specified.

The format of the NAMEFIND command is:

```
NAMEFInd :tag value [:tag [value]]... [options...[)]
```

**Options:**

- STACK \[n \* 1\]
- FIFO \[n \* 1\]
- LIFO \[n \* 1\]
- TYPE \[n \* 1\]
- FILE \(fn\)
- LINenum
- START \(reenum\)
- SIZe \[n \* 8\]
- XEDIT

**where:**

- :tag is a tag in a names file. You can specify multiple tags in a NAMEFIND command. The maximum length of a tag is 255. For more information on tags, see the usage note, "Format of a Names File."

- value is the value of a tag in a names file. The maximum length of a value is 255.
Options:

STACK [n] [FIFO|LIFO]
means that information from the number of entries specified (n) that meet the search criteria is placed in the program stack, rather than being displayed at the terminal. The number (n) specified is the number of entries containing matching information. Valid values for n are 1 to 99999999 or *. If n is omitted, the default is one (1). If an asterisk (*) is specified, information from all the entries meeting the search criteria is stacked. If more than 8 digits are specified for n, the value is truncated on the right. The information is stacked either FIFO (first in first out) or LIFO (last in first out). The default order is FIFO.

FIFO n
specifies that the information is placed in the console stack. Valid values for n are 1 to 99999999 or *. If more than 8 digits are specified for n, the value is truncated on the right. The options STACK, STACK FIFO, and FIFO are all equivalent.

LIFO n
specifies that the information is placed in the console stack rather than being displayed at the terminal. The information is stacked LIFO (last in first out). Valid values for n are 1 to 99999999 or *. If more than 8 digits are specified for n, the value is truncated on the right. This option is equivalent to STACK LIFO.

TYPE n
means that information from the number of entries specified (n) that meet the search criteria is displayed at the terminal. The number (n) specified is the number of entries containing matching information. Valid values for n are 1 to 99999999 or *. If n is omitted, the default is one (1). If an asterisk (*) is specified, information from all the entries meeting the search criteria is displayed. If more than 8 digits are specified for n, the value is truncated on the right.

FILE fn
specifies a file whose filename is “fn” and whose filetype is “NAMES.” This option allows you to use NAMEFIND to search a names file whose filename is something other than your userid. If this option is not specified, the file “userid NAMES *” is searched.

LINenum
requests that the record number of the beginning of the entry be displayed or stacked. It is displayed or stacked before any of the other information. The record number returned has 9 digits and a value of 1 to 999999999.

START recnum
specifies that the search is to begin at the “recnum” record of the file. Valid values for recnum are 1 to 99999999 or *. If more than 8 digits are specified for n, the value is truncated on the right.
**NAMEFIND**

### Size n

Specifies the maximum size of a buffer where a names file is kept. The size of the buffer is $n$, where $n$ is in 1024-character units. Valid values for $n$ are 0 to 99999999 or *. If zero (0) is specified, no buffer is used, and the names file is read into storage each time NAMEFIND is invoked. If an asterisk (*) is specified, the buffer is as large as the names file requires. If more than 8 digits are specified for $n$, the value is truncated on the right. If no SIZE option is specified, SIZE 8 (8192 characters) is the default. This represents the maximum size of the buffer. (If the names file is smaller than 8192 characters, a smaller buffer is used.) This option improves the performance of NAMEFIND when a names file is large. For more information on its use, see the usage note, “Using the SIZE Option,” below.

### XEDIT

Specifies that the information should be read from the file in storage rather than from the disk. This option is only valid when NAMEFIND is issued from the XEDIT environment.

#### Usage Notes:

1. **Format of a Names File**

A names file is a collection of entries, with each entry identified by a “nickname.” A nickname tag plus a series of other tags with associated values make up an entry.

A special names file is one whose fileid is “userid NAMES,” which can be created using the NAMES command. A “userid NAMES” file contains entries for other computer users and entries for lists of users. An entry contains the information necessary to communicate with that person. Once you create a “userid NAMES” file, you can prepare notes for and send files and messages to other people just by using their “nicknames” as operands in the NOTE, SENDFILE, and TELL commands. The tags in each entry supply the additional information required to perform these functions.

You can add, remove, or change entries in the “userid NAMES” file either by using the NAMES command (which displays a menu), or by editing the “userid NAMES” file directly. (The NAMES command can be used only for a file whose fileid is “userid NAMES.”)

A sample “userid NAMES” file is shown below, in the “Examples” section.

**Format of Entries in a Names File:**

The format of data lines in a names file is as follows:
The value need not be on the same record as its tag and can continue onto the next record.

The only tag that is required is a :NICK tag:

:tag.value [:tag.value...]

This is the primary tag, one for each entry. It identifies the beginning of an entry and must be the first word on a line.

Any tags that follow relate to the preceding :NICK tag. The maximum number of tags with values for a given :NICK entry is 64. Therefore, between :NICK entries, you can have from zero to 63 tags.

In addition,

* An asterisk in column one begins a comment line.
  . An asterisk in column two begin a comment line.
  Blank lines are ignored.

## 2. How NAMEFIND Searches a Names File

When you issue a NAMEFIND command, each tag specified with a value is a search tag. NAMEFIND searches until all search tags are found in an entry. Each tag specified without a value is a "return" tag, whose value is returned. If no return tags are specified, the entire entry is displayed or stacked.

Given the "userid NAMES" file shown in the "Examples" section below, the command

NAMEFIND :NICK SNOW :NAME :PHONE

would display:

Snow White
ZZZ-ZZZZ

(:NICK SNOW is the search tag. :NAME and :PHONE are the return tags.)

If you specify two or more identical 'return' tags, a value for each of the tags is returned. A null value is returned to a 'return' tag if an
identical 'return' tag does not exist in the Names file. Given the
"userid Names" file used in the above example, the command:

```
NAMEFIND :NICK SNOW :NAME :NAME :NAME
```
displays:

```
Snow White
Ready;
```

(:NICK SNOW is the search tag and :NAME, :NAME, and :NAME are
identical return tags).

You can specify the tag “:LIST” to display all the names in a list. For
example, the command

```
NAMEFIND :NICK DWARFS :LIST
```
would display:

```
SNOOZY DUMMY BOSS SMILEY GROUCHY SNIFFLES WISTFUL
```

You could then issue NAMEFIND for each of the names in the list
shown above, specifying the return tag :USERID to retrieve the userid
of each person.

You need know the value of only one unique tag in an entry for that
entry to be located. The tags specified without values determine the
information that is displayed (or stacked). For example, the command

```
NAMEFIND :USERID QUEEN :NICK
```
would display:

```
WITCH
```

If duplicate entries exist in a names file, only the first is found, unless
an option value (n) greater than one is specified. If duplicate :NICK
entries are submitted from the NAMES menu (which is displayed with
the NAMES command), a warning message is displayed.

Case and multiple blanks are ignored during the search. Case and
multiple blanks in tag values are preserved when the values are
displayed or stacked.

3. Tags in a “userid NAMES” File

The CMS commands that reference a “userid NAMES” file are NOTE,
SENDFILE, TELL, and RECEIVE. These commands make use of the
tags described below. Fields that correspond to these tags appear on
the NAMES menu. You can also add other tags to the file (for example,
for use by other applications).
:NICK.nickname

This is the primary tag, one for each person or list in the file. It identifies the beginning of an entry and must be the first word on a line.

You should have a :NICK entry for yourself, because the tags that supply your address, phone number, etc., are used by the NOTE command to generate note headings.

All of the following tags relate to the preceding :NICK tag. (Not all tags are required for each entry; however, the CMS commands that reference the "userid NAMES" file make use of the following tags.)

:USERID.userid

specifies the userid of the preceding :NICK entry. This tag is required for communicating with this user via NOTE, SENDFILE, and TELL. If no :USERID tag is specified, the nickname is just an entry (for an address list, or perhaps a name of someone who does not use a computer).

:NODE.node

specifies the node of the preceding :NICK entry. If no node is specified, the default node is your node.

:NOTEBOOK.filename

is the name of a file whose filetype is NOTEBOOK, in which notes (prepared by the NOTE command) sent to or received from this person are kept. See the NOTE command for more information on keeping notes.

:NAME.name

is the person's real name.

:PHONE.phone number

is the person's phone number.
is the person’s postal address. Semicolons (;) in the tag’s value separate the lines of the address. They do not appear in the header of a note (prepared by the NOTE command).

is a list of names. If a name in the list is not a nickname in the “userid NAMES” file, it is assumed to be a userid on the sender’s computer. A name can also be specified as “userid AT node,” just as it can in the NOTE, SENDFILE, and TELL commands. The nickname specified on the associated :NICK tag can be the nickname for the whole list, or it can be the nickname for one user.

4. Using the SIZE Option

When NAMEFIND is invoked, the names file is read into a buffer in virtual storage. It is kept in this buffer instead of being read from disk each time NAMEFIND is invoked. In following invocations of NAMEFIND, characteristics of the NAMES file, such as creation date and time, are compared with those of the file in the buffer to determine if the file has been changed. If so, the changed file is read into the buffer. The CMS commands NOTE, RECEIVE, SENDFILE, and TELL all invoke the NAMEFIND command to search a names file. Having a names file kept in a buffer improves performance of these commands, particularly if the file is large.

When the XEDIT option is specified, the SIZE option is ignored because the file is already in storage.

If no SIZE option is specified, the default buffer size is SIZE 8 (6192 characters). If a names file is too large to fit in the buffer, the size of the buffer can be increased accordingly. Naturally, it can also be decreased to conserve virtual storage. However, if the names file is larger than the size (n) allocated for the buffer, NAMEFIND reads as much of the file as will fit into the buffer, and then reads the rest from disk. By specifying “NAMEFIND (SIZE *” (a good candidate for your PROFILE EXEC), the buffer uses as much storage as is needed to contain a names file, no more and no less.

5. The total length of all tags and values specified may not exceed 255.

6. NAMEFIND uses the extended plist for processing the :tag.value parameter. If you are calling NAMEFIND from an assembler language program and using :tag.value, you should supply an extended plist. VM/SP CMS for System Programming has more information on how an assembler language program can supply an extended plist.
NAMEFIND

Messages and Return Codes:

DMSNAM002E  Files [fn [ft [fm]]] not found [RC = 28]
DMSNAM003E  Invalid option: option [RC = 24]
DMSNAM029E  Invalid parameter parameter in the option option field [RC = 24]
DMSNAM104E  Error mn reading file fn ft fm from {disk|XEDIT} [RC = 100]
DMSNAM156E  Record nnn not found--the file fn ft fm has only nnn records [RC = 32]
DMSNAM618E  NUCEXT failed [RC = nnn]
DMSNAM621E  Bad plist: message [RC = 24]
DMSNAM622E  Insufficient free storage for NAMEFIND [RC = rc]
DMSNAM622W  Insufficient free storage for NAMEFIND buffer, processing continues
DMSNAM633E  Too many tags were encountered--maximum is 64 per line [RC = 88]
DMSNAM633W  Returned values were truncated [RC = 88]
DMSNAM634E  No value to search for was specified [RC = 24]
DMSNAM635I  No entries were found that matched your search criteria
DMSNAM637E  Missing value for the option option [RC = 24]
DMSNAM688E  XEDIT option only valid from XEDIT environment [RC = 24]

Examples:

The following is a sample “userid NAMES” file:

```
:nick.SNOW :userid.SNOWHITE :node.FOREST
:name.Snow White
:addr.Forest Primeval
:phone.ZZZ-ZZZZ
:nick.SNOOZY :userid.SNOOZY :node.COTTAGE
:name.I. M. Dozing
:addr.Dwarf Cottage;Forest
:phone.777-7777
:nick.DUMMY :userid.DUMMY :node.COTTAGE
:name.S. A. What
:addr.Dwarf Cottage;Forest
:phone.777-7777
:nick.BOSS :userid.BOSS :node.COTTAGE
:name.T.O.P. Banana
:addr.Dwarf Cottage;Forest
:phone.777-7777
:nick.SNIFFFES :userid.SNIFFFES :node.COTTAGE
:name.A. H. Choo
:addr.Dwarf Cottage;Forest
:phone.777-7777
:nick.GROUCHY :userid.GROUCHY :node.COTTAGE
:name.E. B. Scrooge
:addr.Dwarf Cottage;Forest
:phone.777-7777
:nick.SMILEY :userid.SMILEY :node.COTTAGE
:name.H. A. Haas
:addr.Dwarf Cottage;Forest
:phone.777-7777
:nick.WISTFUL :userid.WISTFUL :node.COTTAGE
:name.R. U. Shy
:addr.Dwarf Cottage;Forest
:phone.777-7777
:nick.WITCH :userid.WITCH :node.CASTLE
:name.Bad Queen
:addr.Vanity Lane;Mirror City
:phone.UGLY-1111
:nick.GORGEOUS :userid.PRINCE :node.ATLARGE
:name.Prince Charming
:notebook.PRIVATE
:phone.Area 111 111-1111
:nick.DWARFS
:list. SNOOZY DUMMY BOSS SMILEY GROUCHY SNIFFFES WISTFUL
```

Figure 19. Sample ‘userid NAMES’ File
NAMES

Use the NAMES command to display a menu from which you can create, change, and remove entries in a “userid NAMES” file. The menu can be used only on a display terminal.

The format of the NAMES command is:

```
NAMES [nickname]
```

**where:**

*nickname*

is the name assigned to an entry in a “userid NAMES” file. If you specify a nickname, the NAMES menu is displayed with all the information from that entry (if the entry exists) filled in on the menu. You can then examine or change the values in that entry. For example, you might want to update someone’s address or phone number.

If the entry does not exist, the menu is displayed with only the “nickname” field filled in (with the nickname you specified) truncated to 8 characters. You can then fill in the other fields to add a new entry to the NAMES file.

If you invoke NAMES without specifying a nickname, the menu is displayed with all fields left blank. You can then “fill in the blanks” on the menu to create a new entry, or you can scroll through the names file.

**Usage Notes:**

1. **What Is a “userid NAMES” File?**

A “userid NAMES” file (where “userid” is your userid) is a collection of information about other computer users with whom you communicate. An “entry” in this file is all the information associated with a particular nickname.

Having a “userid NAMES” file makes it easier for you to communicate with other users, because you can assign nicknames to them. You can then prepare notes for and send files and messages to other users by using their nicknames as operands in the NOTE, SENDFILE, and TELL commands.
You can also create an entry for a list of names. In this case, the nickname refers to all the users in the list. This makes it possible to send notes, files, or messages to everyone on the list by issuing the appropriate command only once.

2. Entering Information on the NAMES Menu

The NAMES menu helps you to create and edit a “userid NAMES” file. All of the information you type on one menu is an “entry” in the file. You fill in the fields on the menu and press a PF key to create, display, and/or change your names file. The PF key functions are described in the usage note, “PF Key Settings on the NAMES Menu.”

The following list describes the various fields on the menu and explains the information you type in. Refer to the sample menus in the “Examples” section, below, to see the location of the fields on the menu.

Nickname:

is any name you choose to represent a single user or a list of users. An example of each is shown in the “Examples” section, below. Once an entry is created, the nickname is the only piece of information you need to communicate with this user (using the NOTE, SENDFILE, or TELL commands).

You should create an entry for yourself, because the fields that contain your mailing address, phone number, etc., are used by the NOTE command to generate headings.

If the nickname contains any character that is also defined as one of the special logical characters, such as CHARDEL (character delete symbol), LINEDEL (line delete symbol), LINEND (line end symbol), ESCAPE (escape character), or TABCHAR (tab character), unpredictable results may occur for commands using the nickname. For further information, consult the “Logical Line Editing Symbols” section in the VM/SP CMS User’s Guide.

Userid:

is the userid of the person whose nickname you specified. You can leave this field blank if the nickname represents a list, that is, if the List of Names field is filled in. However, if the nickname represents a list and you also specify a userid, the note is also sent to this userid.

You can also leave this field blank if you want the entry to contain information about a person, but you do not intend to communicate with him via the computer. You might choose to do this if you’re using the NAMES file simply to compile an address list.
NAMES

Node:
is the node of the person whose nickname you specified. If not specified, the default node is the one on which this names file exists. You can leave this field blank if the nickname represents a list.

Notebook:
is the filename of a file whose filetype is NOTEBOOK, in which notes (prepared by the NOTE command) sent to or received from this person are to be kept. You can leave this field blank if you want all incoming and outgoing notes saved in the default notebook file, ALL NOTEBOOK.

Name:
is the name of the person whose nickname you specified. You can leave this field blank if the nickname represents a list.

Phone:
is the phone number(s) of the person whose nickname you specified. You can leave this field blank if the nickname represents a list.

Address:
is the address of the person whose nickname you specified. You can leave this field blank if the nickname represents a list.

List of names:
is the names of the people in a list, when the nickname represents the name of this list. The names of the people in the list can be specified in the following ways: as a nickname of an entry in the names file; as a userid of a user who shares your computer; or in the form “userid AT node.” Each time you send a note, a file, or a message to the nickname specified, it will go to everyone on this list. A sample entry for a list of names is shown in the “Examples” section, below.

Tag:
is an identifier of the nickname that you specified. The maximum length of a tag is 255. For more information on tags, see the NAMEFIND command.

Value:
is the value for the corresponding tag of the specified nickname. Values may have a maximum length of 255 characters.

Note: Use caution when entering a colon (:) in a tag field or tag value field. Colons are delimiters and if you enter them incorrectly it may cause unpredictable results.
3. PF Key Settings on the NAMES Menu

The PF key functions appear on the NAMES menu itself (see "Examples") and are summarized in the following list:

- **PF 1** Help: Display NAMES command description.
- **PF 2** Add: Add this entry to the NAMES file.
- **PF 3** Quit: Exit from menu.
- **PF 4** Clear: Clear input fields.
- **PF 5** Find: Locate in the file the first field that is filled in on the menu.
- **PF 6** Change: Change this entry.
- **PF 7** Previous: Display the previous entry.
- **PF 8** Next: Display the next entry.
- **PF 9** Not assigned.
- **PF 10** Delete: Delete this entry.
- **PF 11** Not assigned.
- **PF 12** Cursor: If cursor is on the menu, move it to the command line; if cursor is on the command line, move it back to its previous location on the menu.

**Note:** On a terminal equipped with 24 PF keys, PF keys 13 to 24 are assigned the same values as PF keys 1 to 12 as discussed here. On a display terminal without PF keys, you can enter QUIT on the command line to exit from the screen.

Pressing the PA1 key while in the NAMES menu displays the WM window, unless the CP TERMINAL BRKKEY has been assigned to PA1.

4. Updating a "userid NAMES" File

You can make changes to the file by using the menu and appropriate PF keys (see above), or by editing the file (XEDIT userid NAMES). If you issue NAMES from a line mode terminal, you are placed in edit mode, editing the file "userid NAMES." The format of a "userid NAMES" file is shown in the "Examples" section of the NAMEFIND command.

5. If you want to issue NAMES from an EXEC program, you should precede it with the EXEC command; that is, specify

```
exec names
```

**Responses:**

- `name` has been added to your userid NAMES file.
- Entry has been deleted from your userid NAMES file.
- Entry changed in your userid NAMES file.
- Warning: There is|are| nn undisplayed tag(s).

The following response is displayed on a line mode terminal:

```
You are now editing your Userid NAMES File.
```
NAMES

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSWNM006E No read/write disk accessed [RC = 36]
DMSWNM653E Error executing GLOBALV [RC = nn]

Messages when in the NAMES panel:

DMSWNM645W The user tag name name is too long to display in the panel
DMSWNM656E Error searching your NAMES file; rc = nn from NAMEFIND command [RC = 100]
DMSWNM657E Undefined PFkey/PAkey
DMSWNM658W The value for the tag tag is too long to display on the panel
DMSWNM660E The nickname field must be filled in
DMSWNM660W Warning: this entry duplicates an existing nickname
DMSWNM662E You are not on an entry; press PF 5, 7 or 8 to move to an entry
DMSWNM663W There is/are nn undisplayed tag(s)
DMSWNM664E Entry not found
DMSWNM664E Next entry not found
DMSWNM664E Previous entry not found

Examples:

The following is an entry in the file “SNOWHITE NAMES.”
Fill in the fields and press a PFkey to display and/or change your NAMES file.

Nickname: SNOW
Userid: SNOWHITE
Node: FOREST
Notebook:

Name: Snow White
Phone: ZZZ-ZZZZ
Address: Forest Primeval

List of Names:

You can enter optional information below. Describe it by giving it a "tag."

Tag: Value:

1 = Help  2 = Add  3 = Quit  4 = Clear  5 = Find  6 = Change
7 = Previous  8 = Next  9 = Delete  11 =
12 = Cursor

Macro-read 1 File

Figure 20. Sample NAMES Screen

The following menu shows an entry for a list of names. Each name in the list is the nickname of an entry in the names file.

Fill in the fields and press a PFkey to display and/or change your NAMES file.

Nickname: DWARFS
Userid: Node: Notebook:

Name:
Phone:
Address:

List of Names: SNOOZY DUMMY BOSS SMILEY GROUCHY SNIFFLES WISTFUL

You can enter optional information below. Describe it by giving it a "tag."

Tag: Value:

1 = Help  2 = Add  3 = Quit  4 = Clear  5 = Find  6 = Change
7 = Previous  8 = Next  9 = Delete  11 =
12 = Cursor

Macro-read 1 File

Figure 21. Sample Entry for a List of names.
NOTE

Use the NOTE command to prepare a “note” for one or more computer users on your computer or on other computers connected to yours via the Remote Spooling Communications Subsystem (RSCS) network. A “note” is a short communication, the kind usually done by letter. Some of the features of the NOTE command are:

- The System Product editor (XEDIT) controls the environment in which a note is prepared. Therefore, the full power of the editor is available to help you prepare notes.

- NOTE is one of several commands that references a “userid NAMES” file. By setting up a names file, you can identify recipients just by using nicknames, which are automatically converted into node and userid. For information on creating a names file, see the NAMES command.

- Notes can be sent not only to individual users but also to everyone on a list.

- Headings identifying the sender and the recipients are automatically generated on each note. The information in the headings is collected from the “userid NAMES” file. Notes can be prepared with either short or long headings. An example of each is shown in the “Examples” section, below.

- PF keys are assigned to frequently used functions like sending the note, tabbing, calling for HELP, etc.

The format of the NOTE command is:

```
NOTE [ name... ] [ CC: name... ] [ (options... [ ]) ]
```

Options:

- [ACK]
- [NOACK]
- [ADD]
- [CANCEL]
- [NOTEBOOK fn]
- [NOTEBACK *]
- [NOTONEBACK]
- [LOG]
- [NOLOG]
- [LONG]
- [SHORT]
- [REPLACE]
- [PROFILE fn]
where:

name

is one or more “names” of the computer users to whom the note is to be sent. If the same recipient is specified more than once, he receives only one copy of the note. The “name” may take any of the following forms, and the different forms can be freely intermixed:

- a “nickname” that can be found in the file “userid NAMES,” where “userid” is your userid. This nickname may represent a single person (on your computer or on another computer), or a list of several people.

- a userid of a user on your computer. If a name cannot be found in the “userid NAMES” file, it is assumed to be a userid of a user on your computer.

- “userid AT node,” which identifies a user (“userid”) on your computer or another computer (“node”).

A userid cannot be “AT” or “CC:.”

CC:

indicates the following name(s) are “complimentary copy” recipients of the note. A name can take any of the forms described above. Complimentary copy recipients are designated as such in the note header.

Issued without parameters, NOTE is used to continue a note that was started previously. For more information on saving and continuing notes, see the usage note, “Continuing Notes.”

Options:

ACK

requests an acknowledgment be sent to you when the addressee receives your note. For more information on acknowledgments, see the RECEIVE command description.

NOACK

requests that no acknowledgment be sent.

ADD

causes the addressees to be added to the current invocation of NOTE. No other options may be specified when the ADD option is used. This option is intended to be used from within the NOTE command environment. For more information on this option, see the usage note, “Adding and Deleting Names of Recipients.”
Cancel
causes the note you are currently editing to be erased. You are returned to the file you were previously editing or to CMS, and no note is sent. You enter NOTE with the CANCEL option from the XEDIT command line. All other options are ignored if CANCEL is specified.

NOTebook \textit{fn}
causes the text of the outgoing note to be saved in a file named “\textit{fn NOTEBOOK}.” You can use this option if you want a copy of the note(s) sent to a particular recipient to be kept in a separate file.

If you do not specify a notebook filename here, a filename is first searched for in the (first) recipient’s entry in your “userid NAMES” file, and then in a file set up by the DEFAULTS command. If neither contains a notebook filename, the note is saved in the default notebook file, “ALL NOTEBOOK.” A note is saved by appending it to the NOTEBOOK file, with a line of 73 equal signs (=) separating each note.

(See the NAMEFIND or NAMES command for more information on the relationship between the NAMES file and the NOTEBOOK file.)

NOTebook *
specifies that the text of the outgoing note is saved in a file named “\textit{name NOTEBOOK},” where “\textit{name}” is the value of the Notebook tag in the recipient’s entry in your “userid NAMES” file, or the recipient’s nickname, or the recipient’s userid (whichever is located first).

When there is more than one recipient, the full text of the note is saved in the NOTEBOOK file of the first addressee (selected as described above). In the notebook files of the other addressees and complimentary copy recipients (if any), only the note header and a line referencing the file in which the full text exists is saved. The search order for the notebook filename for these recipients is the same as described above.

NONotebook
specifies that a copy of the outgoing note is not to be saved.

LOG
specifies that the addressees, date, and time of this note are logged in a file called “userid NETLOG,” where “userid” is your userid. This log is updated when acknowledgments are received (if they were requested).

NOLog
specifies that this note is not to be logged.
LONG
causes the long form of the note header to be used. An example of the long form is shown in the “Examples” section, below.

Short
causes the short form of the note header to be used. An example of the short form is shown in the “Examples” section, below.

Replace
causes the work file from a previously interrupted note to be erased before NOTE is entered. If there is no work file, this option has no effect.

PROFile fn
specifies the name of an XEDIT macro to be executed when XEDIT is invoked by the NOTE command. By default the macro PROFNOTE XEDIT is invoked. For more information on the PROFNOTE macro, see the usage note, “Default PF Key Settings.”

Usage Notes:

1. Tailoring the NOTE Command Options

You can use the DEFAULTS command to set up options and/or override command defaults for NOTE. However, the options you specify in the command line when entering the NOTE command override those specified in the DEFAULTS command. This allows you to customize the defaults of the NOTE command, yet override them when you desire. Refer to the DEFAULTS command description for more information.

The current options for an invocation of the NOTE command are displayed as the second line of the file while the note is being prepared. You can alter some of these options (such as LOG or ACK, but not LONG or SHORT) by typing over this line. The options line is not sent with the note.

2. Composing the Note

When you enter the NOTE command, the note screen appears (with the headings). An example of a note screen is shown in the “Examples” section, below. You type in the text of your note in the XEDIT environment. The full power of XEDIT is available while you compose your note. Initially, you are placed in edit mode (although no prefix area or scale is displayed). Text may be entered on the first available line, but it may only begin in the columns prior to the first character of the recipients’ names. For example, if the recipients’ names begin in column 7, the text must begin before column 7. Otherwise, you can leave the first line blank and begin your text anywhere on the next available lines. You can also enter input or power typing mode by entering the appropriate XEDIT subcommand.
The PROFNOTE macro is executed when you issue the NOTE command. It assigns values to PF keys and creates two synonyms that make the NOTE command easier to use. The synonyms are SEND and CANCEL, for "SENDFILE (NOTE)" and "NOTE (CANCEL)," respectively. SEND is also assigned to a PF key. (You can specify the name of a different macro in the PROFILE option if you do not want the PROFNOTE macro to be executed.)

3. Sending the Note

To send the note, you can do one of the following:

- Press the PF5 key.
- Enter SENDFILE (NOTE or SENDFILE (NOTE OLD. The OLD option should be used when the recipient does not have the RECEIVE command available to read the note. For more information on the OLD option, see the SENDFILE command.
- Enter SEND (a synonym for "SENDFILE (NOTE").

The note is sent to the addressees and is logged or saved as specified. Control is returned either to CMS or to the file that was being edited when NOTE was issued.

4. Continuing Notes

If you want to save a note and finish it later, issue the XEDIT subcommand FILE from the command line. No note is sent, but the note is kept on your disk as "userid NOTE AO." To continue the note later, issue the NOTE command with no parameters.

5. Adding and Deleting Names of Recipients

You can add recipients to a note while composing it, that is, after you have already entered a NOTE command. To do this, issue a NOTE command with the ADD option (from the XEDIT command line), specifying the names of the additional recipient(s). For example,

```plaintext
==> NOTE name1 name2 (ADD
```

Any nicknames are resolved, and the additional recipients are automatically added to the note header.

You can also alter the address list and complimentary copy list by typing over the header lines. However, with this method, no nicknames are resolved, and no userids are checked for validity. Therefore, issuing the NOTE command with the ADD option is the preferred way to add recipients.

You can delete the names of recipients directly from the note screen. Just blank out the names you wish deleted from the header lines.
6. Naming Conventions for Userid and Node

You cannot send a note to a userid (or nickname) or node named AT or CC.; nor can your userid be AT or CC.; Also, your userid must contain only those characters that are valid for CMS filenames.

7. Conflicting Options

If conflicting options are entered (such as ACK and NOACK) the last one entered (the rightmost) overrides the others.

8. Default PF Key Settings

The PROFNOTE XEDIT macro is executed when the NOTE command is invoked. It sets the PF keys to the following functions:

<table>
<thead>
<tr>
<th>PF</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Help</td>
</tr>
<tr>
<td>2</td>
<td>Add line</td>
</tr>
<tr>
<td>3</td>
<td>Quit</td>
</tr>
<tr>
<td>4</td>
<td>Tab</td>
</tr>
<tr>
<td>5</td>
<td>Send</td>
</tr>
<tr>
<td>6</td>
<td>?</td>
</tr>
<tr>
<td>7</td>
<td>Backward</td>
</tr>
<tr>
<td>8</td>
<td>Forward</td>
</tr>
<tr>
<td>9</td>
<td>=</td>
</tr>
<tr>
<td>10</td>
<td>Rgtleft</td>
</tr>
<tr>
<td>11</td>
<td>Splitjoin</td>
</tr>
<tr>
<td>12</td>
<td>Powerinput</td>
</tr>
</tbody>
</table>

Note: On a terminal equipped with 24 PF keys, PF keys 13 to 24 are assigned the same values as PF keys 1 to 12 as discussed here. If you enter the "PROFILE fn" option, the macro specified (fn XEDIT) is invoked instead of PROFNOTE XEDIT. In "fn XEDIT," you can easily change the PF key settings.

Some XEDIT subcommands are stacked by the NOTE command (for example, SET TRUNC, SET LRECL, and SET VERIFY). In order to override these settings in a profile, these SET subcommands must be stacked FIFO.

9. The format of the file created by NOTE and sent by the SENDFILE command is described in the SENDFILE command description, in the section "Format of the File Sent by SENDFILE."

10. You cannot start a new note while in NOTE.

11. Format of the Note Header Records
Header records are generated automatically in the note file. The information in the headers is collected from the defaults and options you supplied in the NOTE command.

You can change the information displayed on these lines simply by typing over them. Changing the recipients (the users listed in the “To:” line) is discussed in the usage note, “Adding and Deleting Names of Recipients.”

You can also type over the NOTE command options (the “OPTIONS:” line). Because the information listed in these lines is positional, you must type over the options in the correct order.

The format of the options header record is as follows:

OPTIONS: opt1 opt2 opt3 opt4 opt5

where:

opt1 is either ACK or NOACK.

opt2 is LOG or NOLOG.

opt3 is LONG or SHORT. (This option cannot be altered.)

opt4 is NOTEBOOK or NONOTEBOOK.

opt5 is the NOTEBOOK filename: ALL, *, or the filename specified in the NOTE command.

The other header records are:

Date: is the date and time the note is prepared.

From: is information about the sender. The format of this line depends on whether LONG or SHORT is specified.

To: is information about the recipient(s). The format of this line depends on whether LONG or SHORT is specified.

cc: is information about the complimentary copy recipient(s). The format of this line depends on whether LONG or SHORT is specified.

12. The full power of XEDIT is available to you when using the NOTE command. For example, you may want to use XEDIT subcommands to scroll through the note or file, to locate a particular word, etc.

However, some XEDIT subcommands are inappropriate in this environment. Subcommands that alter the format or heading information of “userid NOTE” (such as SET TRUNC, SET FTYPE, SET LINEND, or SET LRECL) may cause unpredictable results.
If AUTOSAVE is on and you make enough changes in the note to create an autosave file, CMS does not erase the AUTOSAVE file when you send the note. Use the ERASE command to erase the AUTOSAVE file.

13. If you want to issue NOTE from an EXEC program, you should precede it with the EXEC command; that is, specify

   exec note

Responses:

Note cancelled.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSWNT006E  No read/write disk accessed [RC = 36]
DMSWNT399E  Too many tags or tag too long for nickname in userid NAMES file [RC = 88]
DMSWNT637E  Missing {value|nodeid} for the {option option |operand operand} [RC = 24]
DMSWNT647E  Userid not specified for nickname in userid NAMES file [RC = 24]
DMSWNT648E  Userid name not found; check the userid NAMES file [RC = 32]
DMSWNT651E  ADD must be issued from NOTE [RC = 40]
DMSWNT651E  CANCEL must be issued from NOTE [RC = 40]
DMSWNT665E  File userid NOTE * not found; to begin a new note, enter NOTE name [RC = 28]
DMSWNT666E  NOTE already exists; enter NOTE to continue or specify REPLACE option [RC = 28]
DMSWNT668E  ADD option must be specified alone [RC = 40]
DMSWNT669E  List of addressees cannot begin with CC: [RC = 24]
DMSWNT670E  No names to be added were specified [RC = 24]
DMSWNT676E  Invalid character * for Network ID [RC = 20]
DMSWNT677E  Invalid option: option in option line [RC = 32]

Messages when in the NOTE environment (in XEDIT):

DMSWNT667E  Note header does not contain the {keyword From:|keyword To:|OPTIONS line|DATE line} [RC = 32]
Examples:

When a NOTE command is issued, the type of heading generated depends on whether the SHORT option (the default) or LONG is specified. The short form lists only the userids and nodes (if different from the sender's) of the addressees. The long form also lists the name and phone number of each addressee.

An example of each type of heading is shown below. The information in the headings was collected from the names file shown in the “Examples” section of the NAMEFIND command.

The command “NOTE DWARFS CC: GORGEOUS,” where DWARFS and GORGEOUS are nicknames in the names file referenced above, produced the following screen:

![Image of Sample Note with Short Headings]

Figure 22. Sample Note with Short Headings

If the command “NOTE DWARFS CC: GORGEOUS (LONG)” is issued, the headings look like this:
Figure 23. Example of Long Headings
NUCXDROP

NUCXDROP

Use the NUCXDROP command to cancel nucleus extensions and release the storage occupied by the corresponding program. The NUCXDROP command uses the NUCEXT function which is described in detail in the VM/SP CMS Macros and Functions Reference.

The command format is:

```
NUCXDROP [ name1 [ name2... ] ]
```

*where:*

*name*

Is the nucleus extension to be cancelled.

*Means all currently loaded nucleus extensions.

Usage Notes:

1. If a nucleus extension has the 'SERVICE' attribute, it is called by NUCXDROP with the following parameter list:

```
DS OF
DC CL8 'NUCLEUS EXTENSION NAME'
DC CL8 'RESET'
DC 8X'FF'
```

The high order byte in register 1 is set to X'FF'.

2. It is the responsibility of the unloaded program to cancel any secondary nucleus extension entry points by reacting to the RESET service call issued by NUCXDROP before the main entry point is cancelled and the program unloaded. The RESET call allows programs with several entry points to cancel these at the same time, or to free static storage areas obtained from free storage.
Responses:

None.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSNXd050E Parameter missing after NUCXDROP [RC = 24]
DMSNXd070E Invalid argument argument [RC = 24]
DMSNXd616W name does not exist [RC = 28]
DMSNXd617E Error code nn from DMSFRET while unloading module module [RC = 3]
DMSNXd624W No nucleus extensions are loaded [RC = 28]
NUCXLOAD

Use NUCXLOAD to install nucleus extensions. The command loads the following types of modules into free storage and installs them as nucleus extensions:

- an ADCON-free module
- a relocatable member of an OS or CMS load library
- a CMS module file that has relocation information saved.

These modules must be serially reusable modules. The nucleus extension is invoked by issuing the name of the nucleus extension. The NUCXLOAD command uses the NUCEXT function. See the NUCEXT function description in the VM/SP CMS Macros and Functions Reference.

The format of the command is:

\[
\text{NUCXLOAD } \{ \text{name fn} \} \{ \text{name member ddname} \} \{ ([\text{S}ystem] [\text{S}ervice]) [\text{ENdcmd}] [\text{IMcmd}] [\text{Push}] [\text{]}) \}
\]

where:

\text{name fn}

\text{name} is the name associated with this nucleus extension. The \text{fn} is the optional filename of a module file to be loaded and associated with \text{name}. The module being loaded must be an ADCON-free CMS module, a relocatable member of an OS load library, or a CMS module file that has relocation information saved. The term ADCON-free implies that the program needs no relocation, i.e., it runs correctly when loaded at an address different from that at which it was generated (via GENMOD). It allows the object module to be read directly into storage obtained from the free storage manager, after determining the size needed from the module header (or the file format, for the one-record fixed format CMS system transient routines). The term serially reusable implies that the same copy of a routine may be used by another task after the current use has been concluded. If the second argument (i.e., \text{fn}) is not specified, the command name is also used as the filename of the module. The relocation information in a module file is saved by using the RLDSAVE option on the LOAD or INCLUDE commands.
name

is the name to be associated with this nucleus extension.

member

must be a member of a CMS or OS load library. To create a CMS load library, see the CMS command LKED.

ddname

is the ddname from the FILEDEF command that must be issued prior to calling the NUCXLOAD that identifies the load library.

SYstem

indicates that system free storage should be used, and the program is to receive control disabled, key 0. The SYSTEM option is assumed by default for transient routines generated with the SYSTEM option of the GENMOD command.

SErvice

indicates that service calls are accepted (for instance a PURGE from an abend).

ENdcmd

indicates that the nucleus extension receives control at normal end-of-command processing.

IMmcmd

indicates that this nucleus extension can be invoked as an Immediate command.

Push

causes no check to be made to see if there is already a nucleus extension of the same name. Otherwise, an existing nucleus extension is not overridden.

Usage Notes:

1. Nucleus extensions remain in effect until cancelled, either explicitly or implicitly. Implicit cancellation normally occurs only for nucleus extensions of the ‘user’ type (during an abnormal end cleanup time when all storage of ‘user’ type is reclaimed). Explicit cancellation does not release the storage (if any) occupied by the nucleus extension: that is the responsibility of the program that issues the cancellation (usually the program NUCXDROP).

2. Overlay modules may not be loaded by NUCXLOAD.

3. GETMAIN storage management should generally not be used by modules loaded as nucleus extensions, unless all such storage is released before returning from the nucleus extension and no command is issued from the nucleus extension which may perform a STRINIT function.
4. If a module generated from a higher level language is loaded using NUCXLOAD, caution should be taken when passing parameters to the module. See the register contents in the NUCEXT Function in the VM/SP CMS Macros and Functions Reference.

5. Not all CMS macro instructions generate ADCON-free code when expanded. The macro expansion should be inspected to determine if ADCONS are present. If ADCONS are present, the program should either be loaded from a CMS load library via the NUCXLOAD command or from a CMS module file that has relocation information saved. See the LOAD or INCLUDE commands for more information about saving relocation information.

6. You cannot save the relocation information from any CMS modules that were originally loaded in the transient area and generated with the SYSTEM option.

7. The ENDCMD nucleus extensions only receive control after a command is entered from the virtual console. They do not receive control if a command is issued from an EXEC, a user program, or CMS SUBSET mode. All ENDCMD nucleus extensions receive control before any end-of-command cleanup is done by the CMS console command handler.

8. NUCXLOAD with the IMMCMD option allows you to give control to a nucleus extension routine whenever a specified Immediate command is invoked. These exit routines receive control as an extension of CMS I/O interrupt handling. Therefore, they receive control with a PSW key of 0 and are disabled for interrupts. The routine must not perform any I/O operations or issue any SVCs that result in I/O operations.

   In addition, the exit routine must not enable itself for interrupts. DIAGNOSE instructions can be used within the exit, but the exit routine must not enable itself for interruptions that may be caused by the DIAGNOSE (for example, DIAGNOSE X'58'). See the NUCEXT function discussion in the VM/SP CMS Macros and Functions Reference and in the VM/SP CMS User’s Guide.

9. Modules whose size is greater than or equal to 64K bytes must be loaded from a CMS or an OS load library. Not all high level programming languages can generate serially reusable object code. For example, OS/VS COBOL cannot be loaded as a nucleus extension because it does not generate serially reusable code.

Responses:

None.
NUCXLOAD

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSNXL001E  No filename specified [RC = 24]
DMSNXL070E  Invalid parameter parameter [RC = 24]
DMSNXL589E  Missing FILEDEF for DDNAME SYSIN [RC = 32]
DMSNXL618E  NUCEXT failed [RC = nnn]
DMSNXL619E  Module module not found [RC = 28]
DMSNXL622E  Insufficient free storage [RC = rc]
DMSNXL639E  Error in DMSRLD routine; return code was nnn [RC = 24]

Return Codes:

1  Nucleus Extension already exists.

4  Module is marked “not executable.” The module is not loaded; no nucleus extension is defined. To determine why the “not executable” flag was set, examine the information provided by the linkage editor at the time the module was created.

10 Module is an overlay structure. The module is not loaded; no nucleus extension is defined. Overlay structures may not be used as nucleus extensions, because CMS does not support more than one such program at a time. Only an overlay structure in the user area is supported. If this program is to be used as a nucleus extension, it must be restructured so that it does not require overlays.

12 Module is marked “only loadable.” The module is not loaded; no nucleus extension is defined. Modules are marked “only loadable” because of an explicit command to do so at the time they are link-edited. This is typically done when a module contains data, but not executable instructions. Such a nature makes a module unsuitable for use as a nucleus extension.

13 The nucleus extension could not be installed.

24 A filename was not specified, an invalid operand was specified, or too many or extraneous operands were specified.

28 This is the return code generated by NUCXLOAD when the specified module cannot be found. It is also used in the case of an error when opening a LOADLIB file, in which case message DMSSOP036E is produced by the open routine.

32 For NUCXLOAD, a FILEDEF command identifying the load library must be issued prior to calling NUCXLOAD.

41 There was not enough free storage to build the table of SCBLOCK addresses.
100 An unrecoverable error occurred while reading the module from disk.
Use the NUCXMAP command to get information about the currently defined nucleus extensions. NUCXMAP displays on the console or stacks a list of the nucleus extensions. The NUCXMAP command uses the NUCEXT function which is described in detail in the VM/SP CMS Macros and Functions Reference.

The command format is:

```
NUCXMAP [(STACK) [LIFO [FIFO]]]
```

**where:**

If no options are specified, one line describing each nucleus extension is printed on the user's virtual console.

**STACK [LIFO [FIFO]]**

Specifies that the information should be placed in the program stack (for use by an EXEC or other program) instead of being displayed at the terminal. The information is stacked either FIFO (first in first out) or LIFO (last in first out). The default order is FIFO.

**FIFO**

Specifies that the information should be placed in the program stack rather than displayed at the terminal. The information is stacked FIFO. The options STACK, STACK FIFO, and FIFO are all equivalent.

**LIFO**

Specifies that the information should be placed in the program stack rather than displayed at the terminal. The information is stacked LIFO. This option is equivalent to STACK LIFO.

**Responses:**

<table>
<thead>
<tr>
<th>NAME</th>
<th>ENTRY</th>
<th>USERWORD</th>
<th>ORIGIN</th>
<th>BYTES (ALL NUMBERS ARE HEXADECIMAL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLOBALV</td>
<td>0E9888</td>
<td>00000000</td>
<td>0E9888</td>
<td>001258 SYSTEM SERVICE</td>
</tr>
<tr>
<td>EXECIO</td>
<td>1F4CC0</td>
<td>000AD670</td>
<td>1F4CC0</td>
<td>000000 SYSTEM SERVICE</td>
</tr>
<tr>
<td>NAMEFIND</td>
<td>1F7020</td>
<td>000AAB70</td>
<td>1F7020</td>
<td>000000 SYSTEM SERVICE</td>
</tr>
<tr>
<td>IDENTIFY</td>
<td>0E5E48</td>
<td>00000000</td>
<td>0E5E48</td>
<td>0001C8 SYSTEM SERVICE IMMCMD</td>
</tr>
<tr>
<td>CANCEL</td>
<td>01ACC0</td>
<td>00000000</td>
<td>01ACC0</td>
<td>000050 SYSTEM SERVICE ENDCMD</td>
</tr>
<tr>
<td>CLEANUP</td>
<td>01EEF0</td>
<td>00000000</td>
<td>01EEF0</td>
<td>000050</td>
</tr>
</tbody>
</table>
Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSNXM070E  Invalid parameter parameter [RC = 24]
DMSNXM622E  Insufficient free storage (nn entries) [RC = rc]
DMSNXM624I  No nucleus extensions are loaded
Use the OPTION command to change any or all of the options in effect for the DOS/VS COBOL compiler and the RPG II compiler in CMS/DOS.

The format of the OPTION command is:

```
OPTION [options...]

Options:
[DUMP] [DECK] [LIST] [LISTX] [SYM] [XREF] [ERRS] [48C] [TERM]
[NODUMP] [NODECK] [NOLIST] [NolistX] [NOSYM] [NOXREF] [NOERRS] [60C] [NOTERM]
```

Options:

If an invalid option is specified on the command line, an error message is issued for that option; all other valid options are accepted. Only those options specified are altered, and all other options remain unchanged.

**DUMP**

dumps the registers and the virtual partition on the virtual SYSLST device in the case of abnormal program end.

**NODUMP**

suppresses the DUMP option.

**DECK**

punches the resulting object module on the virtual SYSPCH device. If you do not issue an ASSGN command for the logical unit SYSPCH before invoking the compiler, the text deck is written to your CMS A-disk.

**NODECK**

suppresses the DECK option.

**LIST**

writes the output listing of the source module on the SYSLST device.
**OPTION**

**NOLIST**
suppresses the LIST option. This option overrides the XREF option as
it does in DOS/VS.

**LISTX**
produces a procedure division map on the SYSLST device.

**NOLISTX**
suppresses the LISTX option.

**SYM**
prints a Data Division map on SYSLST.

**NOSYM**
suppresses the SYM option.

**XREF**
writes the output symbolic cross-reference list on SYSLST.

**NOXREF**
suppresses the XREF option.

**ERRS**
writes an output listing of all errors in the source program on SYSLST.

**NOERRS**
suppresses the ERRS option.

**48C**
Uses the 48-character set.

**60C**
Uses the 60-character set.

**TERM**
Writes all compiler messages to the user's terminal.

**NOTERM**
Suppresses the TERM option.

**Usage Notes:**

1. If you enter the OPTION command with no options, all options are reset
to their default values, that is, the default settings that are in effect
when you enter the CMS/DOS environment. CMS/DOS defaults are not
necessarily the same as the defaults generated on the VSE system being
used and do not include additional options that are available with some
DOS compilers.
2. The OPTION command has no effect on the DOS PL/I compiler nor on any of the OS language compilers in CMS.

Responses:

None. To display a list of options currently in effect, use the QUERY command with the OPTION operand.

Messages and Return Codes:

- **DMSOPT070E**  Invalid parameter *parameter* [RC = 24]
- **DMSOPT099E**  CMS/DOS environment not active [RC = 40]
OSRUN

Use the OSRUN command to execute a load module from a CMS LOADLIB or an OS module library. The library containing the module must have been previously identified by a GLOBAL command. For an OS module library, the library must also have been defined in a FILEDEF command. If no library has been identified by a GLOBAL command, the OSRUN command searches the $SYSLIB LOADLIB library for the specified module.

The format of the OSRUN command is:

```
OSRUN member [PARM=parameters]
```

where:

*member* is the member of a CMS LOADLIB or an OS module library to be executed.

*PARM* = are the OS parameters that the user wants to pass to the module. If the parameters contain blanks or special characters, they must be enclosed in quotes. To include quotes in the parameters, use double quotes. The parameters are passed in OS format: register 1 points to a fullword containing the address of a character string headed by a halfword field containing the length of the character string. The parameters are restricted to a maximum length of 100 characters.

*Note:* You may not pass parameters (PARM = ) to the module if you issue the OSRUN command from a CMS EXEC file. The OSRUN command can be issued from a System Product interpreter or an EXEC 2 file with no restrictions.

**Messages and Return Codes:**

- **DMSLOS002I** File(s) fn LOADLIB not found
- **DMSLOS013E** Member *membername* not found in library *libname* [RC = 32]
- **DMSLOS073E** Unable to open file *ddname* [RC = 28]
- **DMSOSR001E** No filename specified [RC = 24]
- **DMSOSR052E** More than 100 characters of options specified [RC = 24]
- **DMSOSR070E** Invalid parameter *parameter* [RC = 24]
PARSECMD

Use PARSECMD to call the parsing facility from within an exec. The CMS parsing facility parses and translates command arguments.

The format of the PARSECMD command is:

```
PARSECMD uniqueid [ (options...[]) ]
```

Options:

```
[ TYPE ] [ APPLID applid ] [ STRING cmdstring ]
```

where:

- `uniqueid` is the unique identifier for a command syntax definition in a file containing Definition Language for Command Syntax (DLCS) statements. The uniqueid may be up to sixteen characters in length.

Options:

- **TYPE** displays error messages at the terminal for syntax errors that are found while parsing the exec arguments or `cmdstring`.

- **NOTYPE** returns the text of syntax error messages for exec arguments or `cmdstring` in a REXX or EXEC2 variable called “MESSAGE.n.” MESSAGE.0 contains the number of lines in the message. Variables MESSAGE.1 to MESSAGE.n (where n is the value in MESSAGE.0) contain the lines of the error message text. MESSAGE.0 is 0 if there are no errors.

- **APPLID applid** is an application identifier. The `applid` must be three alphanumeric characters, and the first character must be alphabetic. For example, “DMS” is the applid for CMS, the default application.
STRING cmdstring
    parses the cmdstring rather than the exec's arguments normally obtained from EXECCOMM. This must be the last option specified. Specify a complete command definition, including the command name, as if it were entered from the command line.

Usage Notes:

1. PARSECMD returns parsing information to the exec in a series of REXX (or EXEC 2) variables in the form

   token.<i>n</i> and code.<i>n</i>

   where <i>n</i> is a subscript that distinguishes the different values returned. The variables are in the following format:

   token.0 number of tokens returned
   code.0 number of validation codes returned
   token.1 command name from the DLCS definition
   code.1 validation code for command name
   token.2 second token in the command string
   code.2 validation code for the second token
   ...
   ...
   ...
   token.<i>n</i> nth token in the command string
   code.<i>n</i> validation code for the nth token

2. Possible code.<i>n</i> values are:

   COMMAND Command name
   KEYWORD Keyword
   OPTSTART Option start ( 
   OPTEND Option end )
   COMMENT Comment (everything following OPTEND)
   ALPHANUM Alphanumeric string
   APPLID Any three character alphanumeric string with first alphabetic
   CHAR A single character
   CUU Device address: X'001', X'002', ..., X'FFF'
   DIGITS Any unsigned number made up of digits 0-9
   FN Filename
   FT Filetype
   EFN Filename with '*' or '%' valid also
   EFT Filetype with '*' or '%' valid also
   EXECNAME Execname
   EXECTYPE Exectype
   FM Filemode
   HEX Hexadecimal number
PARSECMD

| INTEGER | Integer: ..., -2, -1, 0, +1, +2, ...
| NINTEGER | Negative integer: ..., -2, -1
| PINTEGER | Positive integer: +1, +2, ...
| MODE | Uppercase alphabetic character
| STRING | Any character string (no blanks)
| TEXT | Any string
| usercode | A user-supplied validation code between 128 and 255 that describes the corresponding token returned by the parsing facility.

3. The TYPE and NOTYPE options apply while parsing the syntax of cmdstring or EXEC arguments. TYPE and NOTYPE depend on the EMSG setting in the CP SET command.

Syntax errors produced when calling PARSECMD are displayed at the terminal regardless of the TYPE and NOTYPE option.

4. The uniqueid you specify in the PARSECMD command is matched to the uniqueid specified in the DLCS file. For a detailed description of uniqueids, see the manual, VM/SP CMS for System Programming.

5. PARSECMD is also the name of a CMS macro that calls the parsing facility from an assembler language program.

6. Keywords are uppercased according to the National Language Uppercase Table for the active application. If the table is not found, the CMS National Language Table is used.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

- DMSPMD407E  Invalid unique ID uniqueid  [RC = 24]
- DMSPMD622E  Insufficient free storage  [RC = 104]
- DMSPMD631E  cmdname can only be executed from an EXEC-2 or REXX EXEC  [RC = 40]
- DMSPMD639E  Error in routine routine; return code was retcode  [RC = 256]
**PEEK**

Use the PEEK command to display a file that is in your virtual reader without reading it onto your disk. Once you issue the PEEK command you can use XEDIT subcommands to view the file. In most cases the files in your reader were sent to you by other computer users, on your computer or on other computers that are connected to yours via the Remote Spooling Communications Subsystem (RSCS) network.

The format of the PEEK command is:

```
PEEK [spoolid] [(options...[)])
```

**Options:**

- **FRom recno**
  - is the starting record number to be read. The default is 1 (one).

- **FOR numrec**
  - is the number of records of the file to be read. Specifying an asterisk (*) causes the entire file to be used. The default is to read up to 200 records.

- **PROFile fn**
  - specifies the name of an XEDIT macro to be executed when XEDIT is invoked by the PEEK command. The default macro is PROFPEEK XEDIT. For more information on the PROFPEEK macro, see the usage note, “PF Key Settings on the PEEK Screen.”

**where:**

- **spoolid**
  - is the spoolid of the file to be displayed. The default is the “next” file in the virtual reader.

  The “next” file is the one for which the RDR command returns information. Which file this is depends on the class of the reader, the class of the files in the reader, and whether or not they are held.
Usage Notes:

1. Tailoring the PEEK Command Options

You can use the DEFAULTS command to set up options and/or override command defaults for PEEK. However, the options you specify in the command line when entering the PEEK command override those specified in the DEFAULTS command. This allows you to customize the defaults of the PEEK command, yet override them when you desire. Refer to the DEFAULTS command description for more information.

2. Editing from the PEEK Screen

When you invoke the PEEK command you are placed in the XEDIT environment, editing the file "spoolid PEEK A0." The full power of XEDIT is available to you while you “peek” at the file. You can make changes to this file and then issue the XEDIT subcommand FILE or SAVE from the XEDIT command line on the PEEK screen. In this case, the reader spool file is not changed. The changes are made only to the file that is saved or filed.

You can purge the file by entering the DISCARD command from the XEDIT command line. The DISCARD command is described in the section “Special Command,” below.

3. PF Key Settings on the PEEK Screen

The PROFPEEK macro is executed when the PEEK command is invoked, unless you specified a different macro as an option in the PEEK command. It assigns the following values to the PF keys:

PF 1 Help Display PEEK command description.

PF 2 Add line Add a blank line after the current line.

PF 3 Quit Exit from the PEEK display.

PF 4 Tab Tab the cursor.

PF 5 Clocate Locate the string specified in an XEDIT subcommand CLOCATE or CHANGE that is typed in the command line. This PF key is set to the XEDIT macro SCHANGE 6. For more information on its use, see the publication VM/SP System Product Editor Command and Macro Reference.

PF 6 ?/Change Display the last command, or change the string specified in a CHANGE subcommand. (The Change function is the XEDIT SCHANGE macro and must be used in conjunction with PF5.)

PF 7 Backward Scroll backward one screen.
PF 8  **Forward**  Scroll forward one screen.

PF 9  **Receive**  Write this file on the A-disk, using the same filename and filetype.

PF 10  **Rgleft**  Shift the view to the right; press again to shift back to original display.

PF 11  **Splitjoin**  Split a line or join two lines, at the cursor.

PF 12  **Cursor**  If cursor is in the file area, move it to the command line; if cursor is on the command line, move it back to its previous location in the file (or to the current line).

*Note:*  On a terminal equipped with 24 PF keys, PF keys 13 to 24 are assigned the same values as PF keys 1 to 12 as discussed here.

If you enter the “PROFILE fn” option, the file “fn XEDIT” is invoked instead of the file PROFP'EEK XEDIT. In “fn XEDIT,” you can easily change the PF key settings. Some XEDIT subcommands are stacked by the FILELIST command (for example, SET TRUNC, SET LRECL, and SET VERIFY). In order to override these settings in a profile, these SET subcommands must be stacked FIFO.

4. Files in DISK DUMP or NETDATA Format

Files in DISK DUMP or NETDATA format are reformatted so that they are readable. However, the entire file must be “peeked” at and have a logical record length of less than 256 in order to be reformatted. For more information on NETDATA format, see the SENDFILE command.

Note that if multiple files are sent with continuous spooling (using CP SPOOL PUNCH CONT) and a series of DISK DUMP commands, RECEIVE recognize only the first file identifier (filename and filetype). Any files having the same file identifier as existing files on your A-disk will overlay those files on your A-disk.

As a sender, you can avoid imposing this problem on file recipients by doing any of the following:

a.  Always use SENDFILE, which resets any continuous spooling options in effect.

b.  Do not spool the punch continuous.

c.  If you must send files with continuous spooling, warn the recipient(s) that files are being sent in this manner and list the file identifiers of the files you are sending.

Similarly, if the punch is spooled continuous and PUNCH is used to send multiple files, the file is read in as one file with “:READ” cards
Imbedded. In this case, although no files are overlaid, the recipient
must divide the file into individual files. This problem can also be
avoided by using SENDFILE or by not spooling the punch continuous.

5. Using the PEEK Command

This command is useful not only when issued in the CMS environment
but also in the RDRLIST command environment. In the RDRLIST
display, the PF11 key is set to the PEEK command.

6. Special NETDATA Files from MVS with TSO Extensions (PP)

The MVS with TSO Extensions program product (program number
5662-285) can send an empty file. It can also send two files in
NETDATA format in a single transmission. Peeking at an empty (null)
file results in a warning message that the file is empty. Peeking at two
files sent in one transmission results in two messages, identifying each
of the files. A line of equal signs (=) separates the two files.

7. The PEEK command does not handle MONITOR files or files with a
   SPECIAL status of YES. (The SPECIAL status indicates whether or
   not the file contains records with X'5A' carriage control characters.
   See the CP QUERY command to determine SPECIAL status of a file.)

8. If you want to issue PEEK from an EXEC program, you should precede
   it with the EXEC command; that is, specify

   exec peek

Responses:

File fn ft from userid at node Format is transmission format

Note from userid at node Format is transmission format

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly
are listed on page 24.

DMSWPK132S  File is too large [RC = 88]
DMSWPK156E  FROM nnn not found--the file fn ft fm has only nnn
            records [RC = 32]
DMSWPK630S  Error accessing spool file [RC = 36]
DMSWPK643E  No class class files in your reader [RC = 28]
DMSWPK644E  All reader files are in HOLD status or not class class
            [RC = 28]
DMSWPK653E  Error executing EXECIO [RC = nn]
DMSWPK655E  Spoolid nnnn does not exist [RC = 28]
DMSWPK672E  Virtual reader invalid or not defined [RC = 36]
DMSWPK674E  Reader is not ready [RC = 36]
DMSWPK683E  The entire file must be peeked at to be formatted [RC = 32]
PEEK

DMSWPK683W  The file has an LRECL greater than 255 and cannot be reformatted [RC = 32]
DMSWPK684E  File contains invalid records and cannot be reformatted [RC = 32]
DMSWPK687E  This is a {SYSTEM\{HELD\}DUMP} file with a special format and cannot be peeked [RC = 1]

Special Command Used in the PEEK Environment

Use the DISCARD command to purge the file displayed on the PEEK screen. DISCARD can be used only in the PEEK, FILELIST, and RDRLIST environments. When DISCARD is used to purge a file, an acknowledgment is sent to the sender (if requested). For more information on acknowledgments, see the RECEIVE command, the usage note, “Acknowledgments.” You enter DISCARD in the XEDIT command line at the bottom of the PEEK screen. The spoolid of this reader file is automatically appended to the DISCARD command and it is displayed on the top line of the PEEK screen, as the filename of the PEEK file.

The format of the DISCARD command as used in the PEEK environment is:

| DISCARD |

Messages and Return Codes for DISCARD:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSWDC651E  DISCARD must be issued from FILELIST, RDRLIST, MACLIST or PEEK [RC = 40]
DMSWDC653E  Error executing command; rc = nn from fn ft fm [RC = nn]
Examples:

A sample PEEK screen follows:

```
3001 PEEK AO V 255 Trunc=255 Size=20 Line=0 Col=1 Alt=0
File NEW IDEA from OHARA at BLUESKY. Format is DISK-DUMP.
** * Top of File ** *
Small business Opportunity

Greetings...

I am planning to open a store, in which I will sell
computer microforms and integrated circuits. I plan to call
it Bob's Fiche and Chips.

Bob

1= Help  2= Add line  3= Quit  4= Tab  5= Clocate  6= ?/Change
7= Backward 8= Forward 9= Receive 10= Rgtleft 11= Splitjoin 12= Cursor

====>

X E D I T 1 File
```

Figure 24. Sample PEEK Screen
POP WINDOW

Use the POP WINDOW command to move a window up in the order of displayed windows.

The format of the POP WINDOW command is:

\[
\text{POP WINDOW} \left\{ \begin{array}{l}
\text{name} \\
\text{WM}
\end{array} \right\} \left[ \begin{array}{l}
n \\
\ast
\end{array} \right]
\]

where:

\text{name}

is the name of the window to be moved up.

\text{WM}

displays the WM window, allowing you to enter commands from the command line or with WMPF keys. The \( n \) and \( \ast \) have no effect when specified with WM. See Usage Note 5 for a list of commands that you can enter in the WM environment.

\( n \)

is the number of positions the window is to be moved up. An \( \ast \) indicates that the window will be positioned on top of the other displayed windows. If this operand is not specified, \( \ast \) is assumed.

Usage Notes:

1. If the window is hidden (see HIDE WINDOW) then POP WINDOW has no effect.

2. A variable size window is only displayed when there is at least one scrollable line to show. If a variable size window is showing a virtual screen that does not contain any scrollable lines, the POP WINDOW command for that window moves the window up in the display order, but it is not displayed.

3. When a variable size window has been cleared (using the CLEAR WINDOW command or by scrolling forward), the POP WINDOW command moves the window up in the order of displayed windows and scrolls it to the bottom of the virtual screen that it is showing.

4. When you are using full-screen CMS and you enter the POP WINDOW command from the command line, the command is executed and then
the screen is refreshed. As part of the refresh processing, any pop-type window that has output waiting is moved to the top of the order of displayed windows. Therefore, the window that you specified may not be displayed at the top of the display order because another has been popped afterwards.

5. You can display the WM window whether or not you are using full-screen CMS, that is, whether SET FULLSCREEN is ON, OFF, or SUSPEND. From the WM window you can issue the following commands:

<table>
<thead>
<tr>
<th>Command</th>
<th>Command</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEAR WINDOW</td>
<td>PUT SCREEN</td>
<td>SCROLL</td>
</tr>
<tr>
<td>CP</td>
<td>QUERY BORDER</td>
<td>SET BORDER</td>
</tr>
<tr>
<td>DROP WINDOW</td>
<td>QUERY HIDE</td>
<td>SET LOCATION</td>
</tr>
<tr>
<td>HELP</td>
<td>QUERY LOCATION</td>
<td>SET RESERVED</td>
</tr>
<tr>
<td>HIDE WINDOW</td>
<td>QUERY RESERVED</td>
<td>SET WINDOW</td>
</tr>
<tr>
<td>MAXIMIZE WINDOW</td>
<td>QUERY SHOW</td>
<td>SHOW WINDOW</td>
</tr>
<tr>
<td>MINIMIZE WINDOW</td>
<td>QUERY WMPF</td>
<td>SIZE WINDOW</td>
</tr>
<tr>
<td>POP WINDOW</td>
<td>REACTOR WMPF</td>
<td></td>
</tr>
<tr>
<td>POSITION WINDOW</td>
<td>RESTORE WINDOW</td>
<td></td>
</tr>
</tbody>
</table>

In the WM environment, you can enter HELP (WMPF 1) to see the list of commands that are available. The WM environment creates a WMHELP window and WMHELP virtual screen to display the list. To exit the WM window, use the DROP WINDOW command.

6. The POP WINDOW WM command automatically defines the WM window and WM virtual screen if they do not already exist.

7. You may encounter a situation where the entire screen is protected and you are unable to enter commands. For example, you may maximize a window so that it fills the entire screen and covers all other windows. You may not be able to enter commands in the window because it is protected. In such cases, the WM window is automatically displayed, and the WMPF keys and command line are available to manipulate the window. For example, use the POP WINDOW command to change the order of displayed windows or the RESTORE WINDOW command to restore a maximized window. The DROP WINDOW WM command (or the default WMPF 3 key) exits the WM environment and displays all the windows that are showing the virtual screen that is waiting for a response. If that virtual screen is the CMS virtual screen, the STATUS window is also displayed. Also, you can use the PA2 and CLEAR keys to scroll the topmost window forward. When there is no more data to scroll, you automatically exit the WM window.
Responses:

None.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSPO386E  Missing operand(s)  [RC = 24]
- DMSPO388E  Invalid keyword: keyword  [RC = 24]
- DMSPO389E  Invalid operand: operand  [RC = 24]
- DMSPO391E  Unexpected operand(s): operand  [RC = 24]
- DMSPO921E  Window uname is not defined  [RC = 28]
- DMSPO929E  Window uname is not connected to a virtual screen  [RC = 36]
POSITION WINDOW

Use the POSITION WINDOW command to change the location of a window on the physical screen.

The format of the POSITION WINDOW command is:

```
POSition WINdow { wname } psline pscol
```

where:

`wname` is the name of the window to be moved. An "=" indicates that the topmost window is moved.

`psline` is the line on the physical screen where the upper or lower edge of the window is positioned. A positive number positions the upper edge of the window on the specified line relative to the top of the screen. A negative number positions the lower edge of the window on the specified line relative to the bottom of the screen.

`pscol` is the column on the physical screen where the left edge of the window is positioned.

Usage Notes:

1. The window’s size and location must be such that, excluding borders, the entire window fits on the physical screen.

2. ‘Psline’ may be specified as either a positive or a negative number. When positive, the window’s upper left corner is placed on the physical screen at the line and column number specified. When negative, the window’s lower left corner is placed relative to the bottom of the physical screen. Thus, a psline value of +1 positions a window by its upper left corner to the top line of the physical screen. A psline value of -1 positions it by its lower left corner to the bottom line of the physical screen.
POSITION WINDOW

Responses:

None.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSPST386E  Missing operand(s)  [RC = 24]
- DMSPST388E  Invalid keyword: keyword  [RC = 24]
- DMSPST389E  Invalid operand: operand  [RC = 24]
- DMSPST391E  Unexpected operand(s): operand  [RC = 24]
- DMSPST921E  Window uname is not defined  [RC = 28]
- DMSPST922E  Window does not fit entirely on the screen  [RC = 32]
PRINT

Use the PRINT command to print a CMS file with a preceding header page on the spooled virtual printer.

The format of the PRINT command is:

\[
\text{PRint} \quad fn \ ft \ [fm] \ [(options\ldots[\ ]]\]
\]

Options:

\[
[\text{OVersize}] \ \ [\text{CC\ [HEA}d\text{er\ ]\ [UPCASE\ [TRC\ [\NOTRC\ ]\ ]\ ]\ ]\ ]\ ]\ ]\]
\[
[\text{LINECOUN}\ nnn] \ \ [\text{MEMBER\ {*\ [membername\ ]\ [HEX\ ]\ ]\ ]\ ]\ ]\]
\]

where:

- \( fn \) is the filename of the file to be printed.
- \( ft \) is the filetype of the file to be printed.
- \( fm \) is the filemode of the file to be printed. If this field is specified as an asterisk (*), the standard order of search is followed and the first file found with the given filename and filetype is printed. If \( fm \) is not specified, the A-disk and its extensions are searched.

Options:

- **OVersize** allows you to print:
  - Files that have records larger than the carriage size of the virtual printer, and
  - Files that have a SPECIAL status of YES.
The records that are larger than the virtual printer's carriage size are printed, but they are truncated to the carriage size. Records with a SPECIAL status of YES are printed if the record length is not greater than 32767 bytes. (The SPECIAL status indicates whether or not the file contains records with X'5A' carriage control characters. See the CP QUERY command to determine SPECIAL status of a file.)

The OVERSIZE (and CC) option is assumed if the filetype is LISTCPDS, LIST3820, or LIST38PP. If OVERSIZE is not specified and the file you want to print is larger than the virtual printer's carriage size, the message "Records exceeds allowable maximum" is displayed.

CC [HEADER]
interprets the first character of each record as a carriage control character. If the filetype is LISTING, LIST3800, or LISTCPDS, the CC option is assumed. If CC is in effect, the PRINT command neither performs page ejects nor counts the number of lines per page; these functions are controlled by the carriage control characters in the file. The LINECOUN option has no effect if CC is in effect.

HEADER creates a shortened header page with only the filename, filetype, and filemode at the top of the page that follows the standard header page. The records in the file being printed begin on a new page following both header pages. The HEADER option can only be used in conjunction with the CC option. If the CC option is not specified HEADER has no effect.

NOCC
does not interpret the first character of each record as a carriage control character. In this case, the PRINT command ejects a new page and prints a heading after the number of lines specified by LINECOUN are printed. If NOCC is specified, it is in effect even if the filetype is LISTING, LIST3800, LISTCPDS, LIST3820, or LIST38PP. The maximum page value in the header is 99999.

UPcase
 translates the lowercase letters in the file to uppercase for printing.

TRC
interprets the first data byte in each record as a TRC (Table Reference Character) byte. The value of the TRC byte determines which translate table the 3800 printer selects to print a record. The value of the TRC byte corresponds to the order in which you have loaded WCGMs (via the CHAR$ keyword of the SETPRT command). Valid values for TRC are 0, 1, 2, and 3. If an invalid value is found, a TRC byte of 0 is assumed. If the filetype is LIST3800, TRC is assumed.

NOTRC
does not interpret the first data byte in each record as a TRC byte. NOTRC is the default.
PRINT

**LI-necoun [nnn]**

allows you to set the number of lines to be printed on each page. nnn can be any decimal number from 0 through 144. If a number is not specified, the default value is 55. If nnn is set to zero, the effect is that of an infinite line count and page ejection does not occur. This option has no effect if the CC option is also specified.

When calculating nnn, remember that the total number of lines printed on a page equals nnn plus 3. The 3 extra lines are for the heading (1 heading line and 2 blank lines).

**MEMber { * { membername } }**

prints the members of macro or text libraries. This option may be specified if the file is a simulated partitioned data set (filetype MACLIB, TXTLIB, or LOADLIB). If an asterisk (*) is entered, all individual members of that library are printed. If a membername is specified, only that member is printed.

**HEX**

prints the file in graphic hexadecimal format. If HEX is specified, the options CC and UPCASE are ignored, even if specified, and even if the filetype is LISTING, LIST3800, LISTCPDS, LIST3820, or LIST38PP.

**Usage Notes:**

1. The file may contain carriage control characters and may have either fixed- or variable-length records, but no record may exceed 132 characters for a 1403, 3203, or 3289 Model 4 printer, 150 characters for a 3211 printer, or 168 characters for a 4248 printer. There are exceptions:

   - If the CC option is in effect, the record length can be one character longer (133, 151, or 169) to allow for the carriage control character.

   - If the virtual printer is a 3800, you can specify a carriage control byte, a TRC byte, or both, for a total line length of up to 206 bytes.

   - If the HEX option is in effect, a record of any length can be printed, up to the CMS file system maximum of 65,535 bytes.

2. If you want the first character of each line to be interpreted as a carriage control character, you must use the CC option. When you use the CC option for files that do not contain carriage control characters, the first character of each line is stripped off. An attempt is made to interpret the first character for carriage control purposes. If the character is not valid, the results are unpredictable because CMS does not check for valid carriage control characters.

Files with a filetype of UPDLOG (produced by the UPDATE command) must be printed with the CC option.
3. If the virtual printer is not a 3800 and you have specified TRC, PRINT strips off the first data byte before each line is printed.

4. One spool printer file is produced for each PRINT command; for example:
   
   print mylib maclib (member get
   
   prints the member GET from the file MYLIB MACLIB. If you want to print a number of files as a single file (so that you do not get output separator pages, for example), use the CP command SPOOL to spool your virtual printer with the CONT option.

5. If the MEMBER option is specified more than once, only the last member specified will be printed. However, if one MEMBER option is coded with an asterisk (*), and another MEMBER option is specified with a membername, only the specified member will be printed, regardless of their order on the command line.

   For example, if you code:
   
   print one maclib (member example1 member example2
   
   only EXAMPLE2 will be printed. If you code:
   
   print one maclib (member example1 member *
   
   only EXAMPLE1 will be printed.

6. If the printer has an extended FCB with the duplication option specified, the PRINT command is not valid because the header line is too long to be duplicated.

Responses:

None.

The CMS ready message indicates the command completed without error (that is, the file is written to the spooled printer). The file is now under the control of CP spooling functions. If a CP SPOOL command option such as HOLD or COPY is in effect, you may receive a message from CP.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSPRTR002E File [fn [ft [fm]]] not found [RC = 28]
DMSPRTR008E Device vdev {invalid or nonexistent|is an unsupported device type} [RC = 36]
DMSPRTR013E Member membername not found in library libname [RC = 32]
DMSPRT029E  Invalid parameter parameter [in the [option] option field]  
[RC = 24]
DMSPRT033E  File fn ft fm is not a library [RC = 32]
DMSPRT039E  No entries in library fn ft fm [RC = 32]
DMSPRT044E  Record exceeds allowable maximum [RC = 32]
DMSPRT069E  Disk mode not accessed [RC = 36]
DMSPRT104S  Error nn reading file fn ft fm from disk [RC = 100]
DMSPRT109S  Insufficient free storage available [RC = 104]
DMSPRT123S  Error nn printing file fn ft fm [RC = 100]
Use the PSERV command in CMS/DOS to copy, display, print, or punch a procedure from the VSE procedure library.

The format of the PSERV command is:

```
PSERV procedure [ft (options... [])]
```

where:

- **procedure** specifies the name of the procedure in the VSE procedure library that you want to copy, print, punch, or display.

- **ft** specifies the filetype of the file to be created on your A-disk. ft defaults to PROC if a filetype is not specified; the filename is always the same as the procedure name.

**Options:**

You may enter as many options as you wish, depending on the functions you want to perform.

**DISK**

copies the procedure to a CMS file. If no options are specified, DISK is the default.

**PRINT**

spools a copy of the procedure to the virtual printer.

**PUNCH**

spools a copy of the procedure to the virtual punch.

**TERM**

displays the procedure on your terminal.
Usage Notes:

1. You cannot execute VSE procedures in CMS/DOS. You can use the PSERV command to copy an existing VSE procedure onto a CMS disk, use the CMS Editor to change or add VSE job control statements to it, and then spool it to the reader of a VSE virtual machine for execution.

2. The PSERV command ignores current assignments of logical units, and directs output according to the option list.

3. The PSERV command does not support a private procedure library.

Responses:

When you issue the TERM option, the procedure is displayed at your terminal.

Messages and Return Codes:

- DMSPRV003E Invalid option: option [RC = 24]
- DMSPRV004E Procedure procedure not found [RC = 28]
- DMSPRV006E No read/write A disk accessed [RC = 36]
- DMSPRV070E Invalid parameter parameter [RC = 24]
- DMSPRV097E No SYSRES volume active [RC = 36]
- DMSPRV098E No procedure name specified [RC = 24]
- DMSPRV099E CMS/DOS environment not active [RC = 40]
- DMSPRV105S Error nn writing file fn ft fm on disk [RC = 100]
- DMSPRV113S Disk(vdev) not attached [RC = 100]
- DMSPRV411S Input error code nn on SYSRES [RC = rc]
PUNCH

Use the PUNCH command to punch a CMS disk file to your virtual punch.

The format of the PUNCH command is:

```
Punch  fn  ft  [fm]  [(options... [ ])]
```

<table>
<thead>
<tr>
<th>Options:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
</tr>
<tr>
<td>NOHeader</td>
</tr>
<tr>
<td>MEMber [ *  membername ]</td>
</tr>
</tbody>
</table>

**where:**

**fn**

is the filename of the file to be punched. This field must be specified.

**ft**

is the filetype of the file to be punched. This field must be specified.

**fm**

is the filemode of the file to be punched. If you specify it as an asterisk (*), the standard order of search is followed and the first file found with the specified filename and filetype is punched. If fm is not specified, your A-disk and its extensions are searched.

**Options:**

**Header**

inserts a control card in front of the punched output. This control card indicates the filename and filetype for a subsequent READCARD command to restore the file to a disk. The control card format is shown in25.

**NOHeader**

does not punch a header control card.

**MEMber**

```
  { *  membername }
```

punches members of MACLIBs or TXTLIBs. If an asterisk (*) is entered, all individual members of that macro or text library are punched. If membername is specified, only that member is punched. If the filetype is MACLIB and the MEMBER membername option is specified, the header contains MEMBER as the filetype. If the filetype
is TXTLIB and the MEMBER membername option is specified, the header card contains TEXT as the filetype.

<table>
<thead>
<tr>
<th>Column</th>
<th>Number of Characters</th>
<th>Contents</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>Identifies card as a control card.</td>
</tr>
<tr>
<td>2-5</td>
<td>4</td>
<td>READ</td>
<td>Identifies card as a READ control card.</td>
</tr>
<tr>
<td>6-7</td>
<td>2</td>
<td>blank</td>
<td></td>
</tr>
<tr>
<td>8-15</td>
<td>8</td>
<td>fname</td>
<td>Filename of the file punched.</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>blank</td>
<td></td>
</tr>
<tr>
<td>17-24</td>
<td>8</td>
<td>ftype</td>
<td>Filetype of the file punched.</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>blank</td>
<td></td>
</tr>
<tr>
<td>26-27</td>
<td>2</td>
<td>fmode</td>
<td>Filemode of the file punched.</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>blank</td>
<td></td>
</tr>
<tr>
<td>29-34</td>
<td>6</td>
<td>valid</td>
<td>Label of the disk from which the file was read.</td>
</tr>
<tr>
<td>35</td>
<td>1</td>
<td>blank</td>
<td></td>
</tr>
<tr>
<td>36-43</td>
<td>8</td>
<td>mm/dd/yy</td>
<td>The date that the file was last written.</td>
</tr>
<tr>
<td>44-45</td>
<td>2</td>
<td>blank</td>
<td></td>
</tr>
<tr>
<td>46-50</td>
<td>5</td>
<td>hh:mm</td>
<td>The time of day that the file was written to disk.</td>
</tr>
<tr>
<td>51-80</td>
<td>30</td>
<td>blank</td>
<td></td>
</tr>
</tbody>
</table>

Figure 25. Header Card Format

Usage Notes:

1. You can punch fixed- or variable-length records with the PUNCH command, as long as no record exceeds 80 characters. Records with less than 80 characters are right-padded with blanks. Records longer than 80 characters are rejected. PUNCH changes variable-length record files to fixed-length 80-byte record files.

2. If you punch a MACLIB or TXTLIB file specifying the MEMBER * option, a READ control card is placed in front of each library member. If you punch a library without specifying the MEMBER * option, only one READ control card is placed at the front of the deck.

3. One spool punch file is produced for each PUNCH command; for example:

   punch compute assemble (noh)

   punches the file COMPUTE ASSEMBLE, without inserting a header card. To transmit multiple CMS files as a single punch file, use the CP SPOOL command to spool the punch with the CONT option.

Note that if multiple files are sent with continuous spooling (using CP SPOOL PUNCH CONT) and a series of DISK DUMP commands, RECEIVE recognizes only the first file identifier (filename and filetype). Any files having the same file identifier as existing files on your A-disk will overlay those files on your A-disk.

As a sender, you can avoid imposing this problem on file recipients by doing any of the following:
Responses:

None. The CMS ready message indicates that the command completed without error (the file was successfully spooled); the file is now under control of CP spooling functions. You may receive a message from CP indicating that the file is being spooled to a particular user's virtual reader.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

- DMSPUN002E  File [fn [ft [fm]]] not found [RC = 28]
- DMSPUN008E  Device vdev {invalid or nonexistent|is an unsupported device type} [RC = 36]
- DMSPUN013E  Member membername not found [RC = 32]
- DMSPUN033E  File fn ft fm is not a library [RC = 32]
DMSPUN039E  No entries in library \textit{fn ft fm} [RC = 32]
DMSPUN044E  Record exceeds allowable maximum [RC = 32]
DMSPUN069E  Disk \textit{mode} not accessed [RC = 36]
DMSPUN026E  Invalid parameter \textit{parameter} in the option field \textit{fieldname} [RC = 24]
DMSPUN104S  Error \textit{nn} reading file \textit{fn ft fm} from disk [RC = 100]
DMSPUN123S  Error \textit{nn} punching file \textit{fn ft fm} [RC = 100]
PUT SCREEN

Use the PUT SCREEN command to copy the image of a physical screen to a CMS file.

The format of the PUT SCREEN command is:

```
PUT SCREEN fn ft [fm]
```

where:

- `fn` is the filename of the file into which the image is copied.
- `ft` is the filetype of the file into which the image is copied.
- `fm` is the filemode of the file. The default is *, which is the first read/write disk in the search order containing the specified file. See the Usage Notes for more information.

Usage Notes:

1. The PUT SCREEN command copies the last physical screen image that was displayed in a window.

2. If the file does not exist, it is created on the A-disk and the lines are inserted. If the file already exists, the data is appended to the end of the file.

3. For a file in fixed format, each line of virtual screen that is longer than the logical record length of the file is truncated. Lines that are shorter than the logical record length are padded with blanks.
Responses:

None.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

- **DMSERD107S**  Disk mode(vdev) is full [RC = 100]
- **DMSPUT069E**  Disk mode not accessed [RC = 36]
- **DMSPUT386E**  Missing operand(s) [RC = 24]
- **DMSPUT388E**  Invalid keyword: keyword [RC = 24]
- **DMSPUT391E**  Unexpected operand(s): operand [RC = 24]
- **DMSPUT917E**  No windows are displayed [RC = 4]
- **DMSSTT048E**  Invalid mode mode [RC = 24]
- **DMSSTT062E**  Invalid character char in fileid [fn ft ffm] [RC = 20]
- **DMSWVL037E**  Disk mode is accessed as read/only [RC = 12]
- **DMSWVL105S**  Error nn writing file fn ft fm on disk [RC = 100]
- **DMSWVL531E**  Disk is full; set new filemode or clear some disk space [RC = 13]
- **DMSWVL924E**  Data was truncated [RC = 3]
PUT VSCREEN

Use the PUT VSCREEN command to write the data from the scrollable data area of a virtual screen to a CMS file.

The format of the PUT VSCREEN command is:

\[
\text{PUT VSCREEN} \quad \text{uname} \quad \text{fn} \quad \text{ft} \quad \{ \text{fromlin} \} \quad \{ \text{numlin} \}
\]

where:

- **uname**
  - is the name of the virtual screen from which the data is written.

- **fn**
  - is the filename of the file into which the data is written.

- **ft**
  - is the filetype of the file into which the data is written.

- **fm**
  - is the filemode of the file. The default is *, which is the first read/write disk in the search order containing the specified file. See the Usage Notes for more information.

- **fromlin**
  - is the starting line in the virtual screen to be copied into the file. PUT VSCREEN starts with the first line in the virtual screen by default. The number specified for fromlin must be within the range of the current top and bottom of the virtual screen.

- **numlin**
  - is the number of lines to be written into the specified file. If numlin is not specified, the default is *, meaning all the lines in the virtual screen starting with the specified line (fromlin) are copied into the file.
PUT VSCREEN

Usage Notes:

If the specified file does not exist, the file is created on the A-disk and the lines are inserted. If the file already exists, the data is appended to the end of the file.

Responses:

None.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSERD107S Disk mode (vdev) is full [RC = 100]
- DMSPUT069E Disk mode not accessed [RC = 36]
- DMSPUT386E Missing operand(s) [RC = 24]
- DMSPUT388E Invalid keyword: keyword [RC = 24]
- DMSPUT389E Invalid operand: operand [RC = 24]
- DMSPUT391E Unexpected operand(s): operand [RC = 24]
- DMSPUT921E Virtual screen vname is not defined [RC = 28]
- DMSSTT048E Invalid mode mode [RC = 24]
- DMSSTT062E Invalid character char in fileid [fn ft fm] [RC = 20]
- DMSWVL037E Disk mode is read only [RC = 12]
- DMSWVL105S Error nn writing file fn ft fm on disk [RC = 100]
- DMSWVL109S Virtual storage capacity exceeded [RC = 104]
- DMSWVL531E Disk is full; set new filemode or clear some disk space [RC = 13]
- DMSWVL923E Specified location is outside the virtual screen [RC = 32]
- DMSWVL924E Data was truncated [RC = 3]
- DMSWVL928E Command is not valid for virtual screen vname [RC = 12]
- DMSWVL936W Virtual screen vname is empty [RC = 0]
Use the QUERY command to gather information about your CMS virtual machine. You can determine:

- The state of virtual machine characteristics that are controlled by the CMS SET command
- File definitions (set with the FILEDEF and DLBL commands) that are in effect
- The status of accessed disks
- The status of CMS/DOS functions
- Physical screen characteristics
- CMSPF and WMPF key settings
- Window characteristics and the order in which the windows are being displayed
- Virtual screen characteristics
- Cursor position.
Query

```
ABBREV
APL
AUTOREAD
BLIP
BORDER  \textit{wname}  [\textit{ALL}]
CHARMODE
CMSLEVEL

CMSPF  \begin{array}{c}
\texttt{nn} \\
\texttt{*}
\end{array}

CMSTYPE
CURSOR  \texttt{vname}

DISK  \begin{array}{c}
\texttt{mode} \\
\texttt{R/W} \\
\texttt{MAX}
\end{array}

DISPLAY
DLBL
DOS
DOSLIB
DOSLNCT
DOSPART
EXECTRAC
FILEDEF
FULLREAD
HIDE  \begin{array}{c}
\texttt{wname} \\
\texttt{*}
\end{array}

IMESCAPE
IMPCP
IMPEX
INPUT
INSTSEG
KEY
```

[(\textit{options} \ldots [\textit{)})]

Chapter 2. CMS Commands 423
QUERY

<table>
<thead>
<tr>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>LABELDEF</td>
</tr>
<tr>
<td>LANGLIST</td>
</tr>
<tr>
<td>LANGUAGE</td>
</tr>
<tr>
<td>LDRTBLS</td>
</tr>
<tr>
<td>LIBRARY</td>
</tr>
<tr>
<td>LINEND</td>
</tr>
<tr>
<td>LOADLIB</td>
</tr>
<tr>
<td>LOCATION</td>
</tr>
<tr>
<td>LOGFILE</td>
</tr>
<tr>
<td>MACLIB</td>
</tr>
<tr>
<td>NONDISP</td>
</tr>
<tr>
<td>OPTION</td>
</tr>
<tr>
<td>OUTPUT</td>
</tr>
<tr>
<td>PROTECT</td>
</tr>
<tr>
<td>RDYMSG</td>
</tr>
<tr>
<td>REDTYPE</td>
</tr>
<tr>
<td>RELPAGE</td>
</tr>
<tr>
<td>REMOTE</td>
</tr>
<tr>
<td>RESERVED</td>
</tr>
<tr>
<td>ROUTE</td>
</tr>
<tr>
<td>SEARCH</td>
</tr>
<tr>
<td>SHOW</td>
</tr>
<tr>
<td>SYNONYM</td>
</tr>
<tr>
<td>SYSTNames</td>
</tr>
<tr>
<td>TEXT</td>
</tr>
<tr>
<td>TRANslate</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>TXTLIB</td>
</tr>
<tr>
<td>UPSI</td>
</tr>
<tr>
<td>Query</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Options: [STACK [FIFO | LIFO], FIFO, LIFO]
ABBREV

displays the status of the minimum truncation indicator.

Response:

\[
\text{ABBREV} = \begin{cases} \text{ON} \\ \text{OFF} \end{cases}
\]

where:

ON
indicates that truncations are accepted for CMS commands and all translations.

OFF
indicates that truncations are not accepted.

APL

displays the status of APL character code conversion.

Response:

\[
\text{APL} = \begin{cases} \text{ON} \\ \text{OFF} \end{cases}
\]

where:

ON
converts APL characters for windows.

OFF
does not convert APL characters.

AUTOREAD

displays the status of the console read.

Response:

\[
\text{AUTOREAD} = \begin{cases} \text{ON} \\ \text{OFF} \end{cases}
\]

where:

ON indicates that a console read is issued immediately after command execution.

OFF indicates that no console read is issued until the enter key (or its equivalent) is pressed.

BLIP

displays the BLIP character(s).

Response:
BLIP = \{xxxxxxx\}
  \{OFF\}

**BORDER wname**
displays whether or not the window borders are ON, and for each edge that is defined, it displays the edge name and character.

**Response:**

```
BORDER wname  ON ([TOP char] [BOTTOM char] [LEFT char] [RIGHT char]
OFF
```

**where:**

- **wname**
  - is the name of the window.
- **char**
  - is the character assigned to the border edge.

**BORDER wname ALL**
displays whether or not the window borders are ON and, for each edge that is defined, displays the edge name and character. In addition, the border attributes are also displayed.

**Response:**

```
BORDER wname  ON|OFF ( [TOP char] [BOTTOM char] [LEFT char] [RIGHT char]
attr  color  exthi psset
```

**where:**

- **attr**
  - is the attribute of the border. It may be a HIGH or NOHIGH.
- **color**
  - is the color of the border.
- **exthi**
  - is the extended highlighting of the border.
- **psset**
  - is the Programmed Symbol set of the border.

**Note:** When using the ALL operand, the response may be too long for one line. If so, it wraps to the next one.

**CHARMODE**
displays whether character attributes or field attributes are used when displaying virtual screen data on the physical screen.

**Response:**

```
CHARMODE  \{ ON \}
```

---

Chapter 2. CMS Commands  427
where:

ON
indicates that character attributes are used when displaying virtual screen data on the physical screen.

OFF
indicates that field attributes are used when displaying virtual screen data on the physical screen.

CMSLEVEL
returns the feature or program product, release, and the service level of CMS.

Response: Displays the VM/SP Release Level and the Service Level.
such as:

VM/SP RELEASE 5, SERVICE LEVEL 103

CMSPF nn
displays the definition of a specific CMS PF key represented by nn.

Responses:

CMSPF nn  [pseudonym keyword string]

where:

nn
is the number of the PF key.

pseudonym
is a 9-character representation that is displayed in the PF Key definition area at the bottom of the CMS window.

keyword
indicates when the command associated with the PF key is executed in relation to other commands entered at the terminal. It may be DELAYED, ECHO, or NOECHO. A CMSPF key set to RETRIEVE does not have a keyword associated with it.

string
is the command string associated with the PF Key.

A CMSPF key that is undefined is displayed as:

CMSPF nn

CMSPF *
displays the definitions for all full-screen CMS PF keys. This is the default.

Response:
The response is the same as for QUERY CMSPF nn; one line is displayed for each PF key.

**CMSTYPE**
indicates the status of the CMS terminal display. This option is valid only from an EXEC environment. The STACK and/or LIFO or FIFO options must be specified.

*Response:* CMSTYPE = \{HT\}RT

*where:*

**HT**
indicates that the CMS terminal display within an EXEC is suppressed. All CMS terminal display from an EXEC, except for CMS error messages with a suffix letter of ‘S’ or ‘T’, is suppressed until the end of the EXEC file or until the SET CMSTYPE \{RT\} command is executed.

**RT**
indicates that the CMS terminal display is not suppressed.

**CURSOR**
displays the location of the cursor on the physical screen when the screen was last read. If there is associated virtual screen information, the name of the virtual screen and the location of the cursor in the virtual screen when the screen was last read is also displayed.

*Responses:*

CURSOR pline pcol [IN vname vline vcol RESERVED DATA]

*where:*

**pline**
is the line number of the physical screen.

**pcol**
is the column number of the physical screen.

**vname**
is the name of the virtual screen that last set the cursor.

**vline**
is the line number of the virtual screen. A value of zero (0) indicates that the cursor is below the current bottom.

**vcol**
is the column number of the virtual screen.

**RESERVED DATA**
is the area where the cursor is set.
CURSOR vname

displays the name of the virtual screen and the current location of the
cursor in the virtual screen. If the cursor has not yet been set, vline
and vcol are -1.

Response:

VSCREEN vname vline vcol [ (RESERVED)DATA]

DISK mode

displays the status of the single disk represented by “mode.”

Response:

<table>
<thead>
<tr>
<th>LABEL</th>
<th>CUU</th>
<th>M</th>
<th>STAT</th>
<th>CYL</th>
<th>TYPE</th>
<th>BLKSIZE</th>
<th>FILES</th>
<th>BLKS USED-(%)</th>
<th>BLKS LEFT</th>
<th>BLK TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>label cuu m</td>
<td>{R/O}</td>
<td>cyl type</td>
<td>blksize</td>
<td>nnnn</td>
<td>nnnn-nn</td>
<td>nnnn</td>
<td>nnnnn</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If the disk is an OS or DOS disk, the response is:

<table>
<thead>
<tr>
<th>LABEL</th>
<th>CUU</th>
<th>M</th>
<th>STAT</th>
<th>CYL</th>
<th>TYPE</th>
<th>BLKSIZE</th>
<th>FILES</th>
<th>BLKS USED-(%)</th>
<th>BLKS LEFT</th>
<th>BLK TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>label cuu m</td>
<td>{R/O}</td>
<td>cyl type</td>
<td>FB</td>
<td>{OS}</td>
<td>DOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where:

label

is the label assigned to the disk when it was formatted; or,
if it is an OS or DOS disk, the volume label.

cuu

is the virtual device address.

m

is the access mode letter.

{R/O}  
{R/W}

indicates whether disk status is read/write or read-only.

cyl

is the number of cylinders available on the disk. For an
FB-512 device, this field contains the notation ‘FB’ rather
than the number of cylinders. For an OS-formatted disk the
number of cylinders available may appear to be larger that
the number actually available, because it includes the space
used by the VTOC.

type

is the device type of the disk.

blkszize

is the CMS disk block size when the minidisk was
formatted.
nnnn FILES

is the number of CMS files on the disk.

BLKS USED

indicates the number of CMS disk blocks in use. The CMS disk blocks include both data blocks and control blocks.

nn %

indicates the percentage of blocks in use.

nnnn BLKS LEFT

indicates the number of disk blocks left. This is a high approximation because control blocks are included.

nnnnn BLK TOTAL

indicates the total number of disk blocks.

\{OS \ DOS\}

indicates an OS or DOS disk.

If the disk with the specified mode is not accessed, the response is:

Disk mode not accessed

DISK *

displays the status of all CMS disks that are currently accessed. This is the default.

Response: Is the same as for QUERY DISK mode; one line is displayed for each accessed disk.

DISK R/W

displays the status of all CMS disks that have been accessed in the Read/Write mode.

Response: Is of the same format as QUERY DISK mode; one header is displayed followed by one line for each accessed CMS Read/Write disk.

If there are no disks accessed in Read/Write mode, the response is:

No Read/Write disk accessed

DISK MAX

displays the status of the CMS disk accessed in Read/Write mode having the most available space.

Response: Is of the same format as QUERY DISK mode; a header and one line are displayed for the CMS Read/Write disk with the most available space.

If there are no disks accessed in Read/Write mode, the response is:

No Read/Write disk accessed
If there is no space available on any of the disks accessed in Read/Write mode, the response is:

No Read/Write disk with space available

DISPLAY
indicates the characteristics of the physical screen.

Responses:

```
DISPLAY lines cols devtype addrtype dbcs color exthi pss pssets
```

where:

- **lines** is the number of lines on the physical screen.
- **cols** is the number of columns on the physical screen.
- **devtype** is the type of display terminal:
  - ANR - such as 3270 to 3277 type displays.
  - NDS - such as 3278, 3279, 3290 type displays.
- **addrtype** is either 12BIT type address or 14BIT type address.
- **dbcs** is DBCS if Double-Byte Character Set (DBCS) strings are supported, or NODBCS if DBCS strings are not supported.
- **color** is COLOR, if color is available, or NOCOLOR.
- **exthi** is EXTHI, if extended highlighting is available, or NOEXTHI.
- **pss** is PSS, if Programmed Symbol sets are available, or NOPSS.
- **pssets** is the list of Program Symbol sets (PSSET) that are currently loaded. Each character represents a PSSET. The PSSET value of 0 is always displayed.

DLBL
in order to display the contents of the current data set definitions, it is necessary only to enter:

DLBL or QUERY DLBL

Entering the command yields the following information:
DDNAME
the VSE filename or OS ddname.

MODE
the CMS disk mode identifying the disk on which the data set resides.

LOGUNIT
the VSE logical unit specification (SYSxxx). This operand will be blank for a data set defined while in CMS/OS environment; that is, the SET DOS ON command had not been issued at DLBL definition time.

TYPE
indicates the type of data set defined. This field may only have the values SEQ (sequential) and VSAM.

CATALOG
indicates the ddname of the VSAM catalog to be searched for the specified data set. This field will be blank for sequential (SEQ) dataset definitions.

EXT
specifies the number of extents defined for the data set. The actual extents may be displayed by entering either the DLBL (EXTENT) or the QUERY DLBL EXTENT command. This field will be blank if no extents are active for a VSAM data set or if the data set is sequential (SEQ). If no DLBL MULT definitions are active, the response is:

No user defined MULTs in effect.

VOL
specifies the number (if greater than one) of volumes on which the VSAM data set resides. The actual volumes may be displayed by entering either the DLBL (MULT) or the QUERY DLBL MULT commands. This field will be blank if the VSAM data set resides only on one volume or if the data set is sequential (SEQ). If no DLBL EXTENT definitions are active, the response is:

No user defined EXTENTs in effect.

BUFSP
indicates the size of the VSAM buffer space if entered at DLBL definition time. This field will be blank if the dataset is sequential (SEQ).

PERM
indicates whether the DLBL definition was made with the PERM option. The field will contain YES or NO.
DISK
indicates whether the data set resided on a CMS or DOS/OS disk at DLBL definition time. The values for this field are DOS and CMS.

DATASET.NAME
for a data set residing on a CMS disk, the CMS filename and filetype are given; for a data set residing on a DOS/OS disk, the data set name (maximum 44 characters) is given. This field will be blank if no DOS/OS data set name is entered at DLBL definition time. If no DLBL definitions are active, the following message is issued:

No user defined DLBLs in effect.

DOS
displays whether the CMS/DOS environment is active or not.

Response:
DOS = \{ON \}
\{OFF\}

Options:

DOSLIB
displays the names of all files with a filetype of DOSLIB that are to be searched for executable phases (that is, all DOSLIBs specified on the last GLOBAL DOSLIB command, if any).

Response:
DOSLIB = libnamel . . . libname8

Up to eight names are displayed per line, for as many lines as are necessary. If no DOSLIBs are to be searched, the response is:

DOSLIB = NONE

DOSLNCNT
displays the number of SYSLST lines per page.

Response:  DOSLNCNT = nn

where:

nn is an integer from 30 to 99.

DOSPART
displays the current setting of the virtual partition size.

Response:
where:

\{ \text{n
n
n
K} \}

\text{n
n
n
K}

indicates the size of the virtual partition to be used at program execution time.

\text{NONE}

indicates that CMS determines the virtual partition size at program execution time.

\text{EXECTRAC}

displays the setting of the tracing bit (in EXECFLAG in NUCON).
Setting is either ON or OFF.

\text{Response:}

\text{EXECTRAC} = \{ \text{ON} \}

\text{OFF} \}

\text{FILEDEF}

displays all file definitions in effect.

\text{Response:}

\text{ddname device [fn [ft]]}

\text{FULLREAD}

indicates whether or not 3270 null characters are recognized in the middle of the physical screen.

\text{Response:}

\text{FULLREAD} = \{ \text{ON} \}

\text{OFF} \}

\text{where:}

\text{ON}

indicates that null characters are recognized in the middle of lines, making it easier for you to enter tabular or pictorial data.

\text{OFF}

inhibits transmission of nulls from the terminal.
**FULLSCREEN**
indicates the status of full-screen CMS.

*Response:*

```
FULLSCREEN { ON
OFF
SUSPEND }
```

*where:*

ON
indicates that full-screen CMS is active.

OFF
indicates that full-screen CMS is inactive.

SUSPEND
indicates that full-screen CMS is suspended.

**HIDE wname**
displays the name of the window, the name of the virtual screen to which the window is connected, and the location (line and column) of the window on the virtual screen.

*Responses:*

```
WINDOW wname ON vname line column
```

*where:*

wname
is the name of the window.

vname
is the name of the virtual screen to which the window is connected.

line
is the line number in the virtual screen where the window is connected.

column
is the column number in the virtual screen where the window is connected.

**HIDE**
displays the name of the window, the name of the virtual screen to which the window is connected, and the location (line and column) of the window on the virtual screen for all hidden windows. This is the default.

*Responses:*
The response is the same as for QUERY HIDE *wname*; one line is displayed for each hidden window.

**IMESCAPE**
displays the immediate command escape character.

*Response:*

\[
\text{IMESCAPE} = \begin{cases} \text{char} \\ \text{OFF} \end{cases}
\]

*where:*

*char*

is the escape character in effect. The default character is a semi-colon (;).

*OFF*

no immediate command escape character is in effect.

**IMPCP**
displays the status of implied CP command indicator.

*Response:*

\[
\text{IMPCP} = \begin{cases} \text{ON} \\ \text{OFF} \end{cases}
\]

*where:*

*ON*

indicates that CP commands can be entered from the CMS environment.

*OFF*

indicates that you must use the CP command or the CP commands from the CMS environment.

**IMPEX**
displays status of implied EXEC indicator.

*Response:*

\[
\text{IMPEX} = \begin{cases} \text{ON} \\ \text{OFF} \end{cases}
\]

*where:*

*ON*

indicates that EXEC files can be executed by entering the filename of the file.

*OFF*

indicates that the EXEC command must be explicitly entered to execute EXEC files.
INPUT
displays the contents of any input translate table in effect.

Response:

```
INPUT  al  xxl
   :
   :
   an  xxn
```

If you do not have an input translate table in effect, the response is:

-NO USER DEFINED INPUT TRANSLATE TABLE IN USE

INSTSEG
displays the status of the Installation Discontiguous Shared Segment (DCSS).

Response:

```
INSTSEG= {ON mode|LAST}
```

where:

- **ON mode|LAST** indicates that the Installation DCSS is being used. The *mode* specifies the location in the command search order where the DCSS is being accessed. LAST indicates that the Installation DCSS is searched after the last accessed disk in the search order.
- **OFF** indicates that the Installation DCSS is not being used.

KEY
displays the last key pressed that caused an attention interrupt.

Responses:

```
KEY  =  keypressed
```

where:

- **keypressed** is the attention key that was pressed, such as ENTER, PAKEY n, PFKEY n, CLEAR, or UNKNOWN.

LABELDEF
displays all label definitions in effect.

Response:
Only fields you have explicitly specified are displayed. Defaulted fields are not displayed. If no label definitions are in effect, the following message is displayed at the terminal:

No user defined LABELDEFs in effect

For an OS simulation user, if SCRATCH was entered at command time and the file has not been opened, then all the VOLIDs and SCRATCH will be displayed. If SCRATCH was entered at command time, and the file has been opened, then all the VOLIDs and the VOLIDs of all the scratch tapes will be displayed. When multiple VOLIDs have been specified, the response is:

LANGLIST
displays all the language identifiers (langids) that can be set for CMS in your virtual machine. You can use this command to determine whether or not a certain language is valid for your virtual machine.

Response:
langid1
langid2
langid3

where:
langid1
is the language identifier of the default language for CMS.

langid2
is the language identifier of the language that is currently active for CMS in your virtual machine. If the current, active language and the default language are the same, langid2 is not displayed.

langid3 . .
are other valid language identifiers.

Note: The response may change if you alter the size of your virtual machine.
Example:

The response to QUERY LANGLIST may be:

```
AMENG
GER
FRANC
```

where:

- AMENG is American English, the default language for CMS.
- GER is German, the language that is currently active for CMS in your virtual machine.
- FRANC is French, another valid language for your virtual machine.

**LANGUAGE**

displays the language identifier of the language that is currently active for CMS in your virtual machine.

**Response:** `langid`

where:

- `langid` is the language identifier of the language that is currently active for CMS in your virtual machine.

**LANGUAGE ALL**
displays the language identifier and all active application identifiers. For applications, ALL displays whether the applications have system-provided language files, user additions (message repositories, CMS command syntax file), or both.

**Response:**

```
langid
applid1
USER|SYSTEM|ALL
applid2
USER|SYSTEM|ALL
.
.
.
```

where:

- `langid` is the language identifier of the language that is currently active for CMS in your virtual machine.
- `applid1` `applid2` . . .

is an application identifier.
USER
indicates that user repositories, command syntax tables, and/or command synonym tables are loaded into storage.

SYSTEM
indicates that the system-provided language files for the application named are active. No user language files are active.

ALL
specifies that the system-provided language files and user additions are active.

Example:
The response to QUERY LANGUAGE ALL may be:

```
AMENG
DMS
ALL
AWG
SYSTEM
```

where:

- AMENG is the langid for American English.
- DMS is the application id for CMS.
- ALL indicates that you have made additions to the CMS message repository and/or command syntax files.
- AWG is an application id for an application program.
- SYSTEM indicates that you are using only the system language information for the AWG application.

LDRTBLS
displays the number of loader tables.

Response:

```
LDRTBLS = nn
```

LIBRARY
displays the names of all library files with filetypes of MACLIB, TXTLIB, DOSLIB, and LOADLIB that are to be searched.

Response:

```
libtype = libname1 . . . libname8
          . . . .
```

Up to eight names are displayed per line, for as many lines as necessary. If no libraries are to be searched, the response is:
MCLIB = NONE
TXTLIB = NONE
DOSLIB = NONE
LOADLIB = NONE

LINEND
indicates whether or not the logical line end character is activated and the character defined as the logical line end.

Response:
LINEND \{ ON \}
OFF char

where:
ON
indicates that the logical line end character is activated.
OFF
indicates that the logical line end character is not activated.
char
is the logical line end character.

LOADLIB
displays the names of all files, that have a filetype of LOADLIB, that are to be searched for load modules (that is, all LOADLIBs specified on the last GLOBAL LOADLIB command, if any).

Response:
LOADLIB = libname1 . . . libname8
: : :
Up to eight names are displayed per line, for as many lines as necessary. If no LOADLIBs are to be searched, the following message is displayed at the terminal:
LOADLIB = NONE

LOCATION \texttt{wname}
indicates whether or not the location indicator is displayed in the specified window when the data in the virtual screen exceeds the size of the window.

Response:
LOCATION \texttt{wname} ON
OFF

where:
wname
    is the name of the specified window.

ON
    the location indicator is displayed when there is data to be viewed outside the window.

OFF
    the location indicator is not displayed.

LOGFILE vname
    displays whether or not a log file is updated with data written to the virtual screen.

Response:
    LOGFILE vname  {ON} fn ft fm

where:

ON
    indicates that a log file is updated with data written to the virtual screen.

OFF
    indicates that a log file is not updated for the virtual screen.

vname
    is the name of the virtual screen.

fn
    is the filename of the log file.

ft
    is the filetype of the log file.

fm
    is the filemode of the log file.

MACLIB
    displays the names of all files, with a filetype of MACLIB, that are to be searched for macro definitions (that is, all MACLIBs specified on the last GLOBAL MACLIB command, if any).

Response:
    MACLIB = libname1 ... libname8

Up to eight names are displayed per line, for as many lines as necessary. If no macro libraries are to be searched for macro definitions, the response is:
MACLIB = NONE

NONDISP
specifies the character that is set to be displayed in place of nondisplayable characters.

Response:
NONDISP char

where:
char
is the character that is displayed in place of nondisplayable characters.

OPTION
displays the compiler options that are currently in effect.

Response: OPTION = options...

OUTPUT
displays the contents of any output translate table in effect.

Response:
OUTPUT xx1 al
      :
      :
      xxn an

If you do not have an output translate table defined, the response is:
NO USER DEFINED OUTPUT TRANSLATE TABLE IN USE

PROTECT
displays the status of CMS nucleus protection.

Response:
PROTECT = \{ ON OFF \}

where:
ON
means CMS nucleus protection is in effect.
OFF
means CMS nucleus protection is not in effect.

RDYMSG
displays the format of the CMS ready message.
Response:

\[ \text{RDYMSG} = \{ \text{LMSG} \} \]

where:

\text{LMSG}

is the standard CMS ready message:

\text{Ready; T = 0.12/0.33 17:06:20}

\text{SMSG}

is the shortened CMS ready message:

\text{Ready;}

\text{REDTYPE}

displays the status of the REDTYPE indicator.

Response:

\[ \text{REDTYPE} = \{ \text{ON} \} \]

where:

\text{ON}

types CMS error messages in red, for certain terminals equipped with the appropriate terminal feature and a two-color ribbon. Supported terminals are described in the \textit{VM/SP Terminal Reference}.

\text{OFF}

does not type CMS error messages in red.

\text{RELPAGE}

indicates whether pages of storage are to be released or retained after certain commands complete execution.

Response:

\[ \text{RELPAGE} = \{ \text{ON} \} \]

where:

\text{ON}

releases pages.

\text{OFF}

retains pages.

\text{REMOTE}

displays the manner in which data transmission is handled.

Response:
REMOTE \{ ON OFF \}

where:

ON
specifies that data is compressed by removing nulls and combining data when five or more of the same characters occur consecutively in a data stream.

OFF
specifies that the data stream is not compressed. Data is transmitted with no minimization.

RESERVED wname
displays the maximum number of reserved lines on the top and bottom of the specified window.

Response:
RESERVED wname rtop rbot

where:

wname
is the name of the window.

rtop
is the maximum number of top reserved lines that may be displayed.

rbot
is the maximum number of bottom reserved lines that may be displayed.

ROUTE msgclass
displays the name of the virtual screen, alarm status, and notification information for the specified message class when SET FULLSCREEN is ON or SUSPEND. The following classes of output may be specified:

CMS
CP
MESSAGE
WARNING
SCIF
NETWORK

Response:
msgclass TO vname (alarm notify

where:
**QUERY**

msgclass
is the message class which is directed to the virtual screen.

vname
is the virtual screen receiving the output.

alarm
is ALARM if the alarm is sounded when a message is received
and NOALARM if the alarm does not sound when a message is
received.

notify
is NOTIFY if the message class name is displayed in the status
area when you receive a message and is NONOTIFY if the
message class name is not displayed in the status area when you
receive a message.

**ROUTE** *

displays the routing of all message classes. This is the default.

Response:

The response is the same as for QUERY ROUTE msgclass; one line is
displayed for each message class.

**SEARCH**
displays the search order of all disks currently accessed.

Response:

<table>
<thead>
<tr>
<th>label</th>
<th>cuu</th>
<th>mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>{R/O}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[R/W]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[-OS]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[-DOS]</td>
</tr>
</tbody>
</table>

where:

label is the label assigned to the disk when it was formatted; or, if
it is an OS or DOS disk, the volume label.

cuu is the virtual device address.

mode is the filemode letter assigned to the disk when it was
accessed.

{R/O}
{R/W}

indicates whether read/write or read-only is the status of the
disk.
QUERY

OS
DOS

indicates an OS or DOS disk.

SHOW wname
displays the name of the window, the name of the virtual screen that
the window is showing, and the location (line and column) of the
window on the virtual screen.

Note: If a variable size window is showing a virtual screen that does
not contain any scrollable data lines or if a variable size window is
clear, the window is not displayed on the physical screen. It does
appear in the response.

Responses:
WINDOW wname ON vname line column

where:

wname
is the name of the window.

vname
is the name of the virtual screen to which the window is
connected.

line
is the line number in the virtual screen where the window is
connected.

column
is the column number in the virtual screen where the window is
connected.

SHOW *
displays the name of the window, the name of the virtual screen to
which the window is connected, and the location (line and column) of
the window on the virtual screen for all windows in the order in
which they are being displayed on the physical screen. This is the
default.

Responses:
The response is the same as for QUERY SHOW wname; one line is
displayed for each visible window.

SYNONYM SYSTEM
displays the CMS system synonyms in effect that are enabled by the
SYNONYM command.

Response:
If no system synonyms are in effect, the following message is displayed at the terminal:

No system synonyms in effect

SYNONYM USER

displays user synonyms in effect that are enabled by the SYNONYM command.

Response:

<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>USER</th>
<th>SHORTEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMAND</td>
<td>SYNONYM</td>
<td>FORM (IF ANY)</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------------</td>
</tr>
<tr>
<td>command synonym</td>
<td>minimum truncation</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td></td>
</tr>
</tbody>
</table>

If no user synonyms are in effect, the following message is displayed at the terminal:

No user synonyms in effect

SYNONYM ALL

displays all synonyms in effect that are enabled by the SYNONYM command.

Response: The response to the command QUERY SYNONYM SYSTEM is followed by the response to QUERY SYNONYM USER.

SYSNAMES

displays the names of the standard saved systems.

Response:

SYSNAMES: CMSVSAM CMSAMS CMSDOS CMSBAM
ENTRIES: entry... entry... entry... entry...

where:

SYSNAMES are the standard names that identify the discontiguous saved systems.

ENTRIES are the standard system default names or the system names established via the SET SYSNAME command.
TEXT displays the status of TEXT character code conversion.

Response:

TEXT \{ ON \ OFF \}

where:

ON converts TEXT characters for windows.

OFF does not convert TEXT characters.

TRANslate displays translations and translation synonyms that are in effect.

SYStem displays only the System National Language Translation Table.

USER displays only the User National Language Translation Table.

ALL displays System and User National Language Translation Tables.

TRANslate displays only the national language translations.

SYNonym displays only the national language translation synonyms.

BOTH displays both the national language translations and translation synonyms.

APPlid applid is an application identifier that defines information about a particular application. It must be three alphanumeric characters, and the first character must be alphabetic. The default, *, displays tables for all applications.

Response:

The QUERY TRANSLATE command responds with:

• One or two messages stating whether or not the translations and translation synonyms are active, and
• A table displaying the command name, translations/synonyms, and minimum abbreviation, or a message stating that there are no entries in the table.

In the first part of the response, you receive one or more of the following messages, depending on the options that you specify:

System translations, application id: applid
A heading for the currently active system translations

System translation synonyms, application id: applid
A heading for the currently active system translation synonyms

User translations, application id: applid
A heading for the currently active user translations

User translation synonyms, application id: applid
A heading for the currently active user translation synonyms

User translations off, application id: applid
A response when user translations are currently not active

User translation synonyms off, application id: applid
A response when user translation synonyms are currently not active

System translation off, application id: applid
A response when system translations are currently not active

System translation synonyms off, application id: applid
A response when system translation synonyms are currently not active

No entries in table
A response when there are no entries in table being displayed

When the tables contain entries for translations and translation synonyms, the tables are displayed in the following order:

User Translations
System Translations
User Translation Synonyms

System Translation Synonyms

Each table is displayed in the following format:

<table>
<thead>
<tr>
<th>command name</th>
<th>translation/synonym</th>
<th>minimum abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

Example:

To display only the system translations that are in effect, enter

```
query translate system translate
```

**TXTLIB**

displays the names of all files, with a filetype of TXTLIB, that are to be searched for unresolved references (that is, all TXTLIBs specified on the last GLOBAL TXT LIB command, if any).

**Response:**

```
TXTLIB = libname1 . . . libname8
```

Up to eight names are displayed per line, for as many lines as necessary. If no TXTLIBs are to be searched for unresolved references, the following message is displayed at the terminal:

```
TXTLIB = NONE
```

**UPSI**

displays the current setting of the UPSI byte. The eight individual bits are displayed as zeros or ones depending upon whether the corresponding bit is on or off.

**Response:**

```
UPSI = nnnnnnnn
```

**VSCREEN vname**

displays the name of the virtual screen, the number of lines and columns, and the number of top and bottom reserved lines for the specified virtual screen.

**Responses:**

```
VSCREEN vname lines cols rtop rbot
```

**where:**
vname
  is the name of the virtual screen.

lines
  is the number of lines in the virtual screen.

cols
  is the number of columns in the virtual screen.

rtop
  is the number of lines in the top reserved area.

rbot
  is the number of lines in the bottom reserved area.

VSCREEN *
  displays the name of the virtual screen, the number of lines and number of columns, and the number of top and bottom reserved lines for all virtual screens. This is the default.

Responses:

The response is the same as for QUERY VSCREEN vname; one line is displayed for each virtual screen.

VSCREEN vname ALL
  *
  displays the attributes and extended attributes for a virtual screen or all virtual screens in addition to the same information as QUERY VSCREEN.

Response:

VSCREEN vname lines cols rtop rbot (high protect color exthi psset type system

where:

vname
  is the name of the virtual screen.

high
  is the intensity attribute of the data in the virtual screen. It may be HIGH or NOHIGH.

protect
  indicates whether or not the data in the virtual screen is protected. It may be PROTECT or NOPROT (NOPROTECT).

color
  is the color of the virtual screen.

exthi
  is the extended highlighting of the virtual screen.
psset
    is the Programmed Symbol set of the virtual screen.

type
    indicates that data is moved to the virtual screen when the
    virtual screen queue is processed (TYPE) or that the virtual
    screen is not updated (NOTYPE).

system
    indicates whether the virtual screen is retained (SYSTEM) or
    deleted (USER) when a task abnormally ends (abend) or when
    the HX (halt execution) command is issued.

Note: When using the ALL operand, the response may be too long for
one line. If so, it wraps to the next one.

WINDOW wname
    displays the name, the number of lines and columns, and the location
    on the physical screen for a specific window.

    Responses:

    WINDOW wname lines cols psline pscol

    where:

    wname
        is the name of the window.

    lines
        is the number of lines in the window.

    cols
        is the number of columns in the window.

    psline
        is the line on the physical screen where the window is placed.

    pscol
        is the column on the physical screen where the window is placed.

WINDOW =
    displays the name, the number of lines and columns, and the location
    on the physical screen for the topmost window.

    Responses:

    The response is the same as for QUERY WINDOW wname; one line is
    displayed for the topmost window.

WINDOW *
    displays the name, the number of lines and columns, and the location
    on the physical screen for all windows. This is the default.
Responses:

The response is the same as for QUERY WINDOW wname; one line is displayed for each window.

WINDOW wname ALL
  *
  displays the options for a window or all windows in addition to the same information as QUERY WINDOW.

Response:

WINDOW wname lines cols psline pscol (type border pop top system

where:

type
  is VARIABLE if the window is a variable size or FIXED if the window is a fixed size.

border
  is BORDER if the border is set ON, or NOBORDER if the borders are OFF.

pop
  indicates whether the window is displayed on top of all other windows (POP) or that there is no effect on the window's position in the ordered list of windows (NOPOP) when the virtual screen that the window is showing is updated.

top
  indicates whether the window may qualify as the topmost window (TOP) or cannot qualify as the topmost window (NOTOP)

system
  indicates whether the window is retained (SYSTEM) or deleted (USER) when a task abnormally ends (abend) or when the HX (halt execution) command is issued.

Note: When using the ALL operand, the response may be too long for one line. If so, it wraps to the next one.

WMPF nn
  displays the definition of a WMPF key specified as nn.

Responses:

WMPF nn pseudonym keyword string

where:

pseudonym
  is a 9-character representation that is displayed in the PF Key definition area at the bottom of the WM window.
QUERY

keyword
indicates when the command associated with the PF key is executed in relation to other commands entered at the terminal. It may be DELAYED, ECHO, or NOECHO. A WMPF key set to RETRIEVE does not have a keyword associated with it.

string
is the command string associated with the PF Key.

An undefined WMPF key is displayed as

WMPF nn

WMPF *
displays the definitions for all WMPF keys. This is the default.

Responses: The response is the same as for QUERY WMPF nn; one line is displayed for each PF key.

Options:

STACK
causes the results of the QUERY command to be placed in the program stack instead of being displayed at the terminal. The information is stacked either FIFO (first in first out) or LIFO (last in first out). The default order is FIFO.

If CMS passes the command to CP, then the response from CP is also put in the program stack. If CP precedes the QUERY command, CMS does not stack the results. The STACK option is valid only when issued from CMS.

FIFO
(first-in first-out) is the default option for STACK. FIFO causes the results of the QUERY command to be placed in the program stack instead of being displayed at the terminal. The information is stacked FIFO. The options STACK, STACK FIFO, and FIFO are all equivalent.

LIFO
(last-in first-out) causes the results of the QUERY command to be placed in the program stack rather than being displayed at the terminal. The information is stacked LIFO. This option is equivalent to STACK LIFO.
Usage Notes:

1. You may specify only one QUERY parameter at a time.

2. If the implied CP (IMPCP) function is in effect and you enter an invalid QUERY parameter, you may receive the message DMKCQG045E - userid NOT LOGGED ON.

3. If an invalid QUERY parameter is specified from an EXEC and the implied CP (IMPCP) function is in effect, then the return code is -0003.

4. The DOSPART, OPTION, and UPSI functions are valid only if the CMS/DOS environment is active.

5. When the STACK option is specified, the header is included in the program stack. If the information that you query requires a response from CP and the response is longer than 8192 characters, nothing is stacked and you receive a return code of 88.

6. The language for QUERY command responses depends on the language used to enter the command and whether or not the command was translated. If the QUERY command is entered in American English (or AMENG, which is always available), CMS responds in American English. If the command is entered in the current national language, or if the translation of the command is the same as American English, the response is displayed (or stacked) in the current national language. The language of the response is especially important for language dependent EXECs. For more information, see the section "Using Translations" in the VM/SP CMS User's Guide.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

- DMSQRF109S Virtual storage capacity exceeded [RC = 109]
- DMSQRF525E Invalid PFkey number [RC = 24]
- DMSQRF921E Virtual screen vname is not defined [RC = 28]
- DMSQRF926E Command is only valid on a display terminal [RC = 88]
- DMSQRF926E Command is only valid in CMS FULLSCREEN mode [RC = 88]
- DMSQRG109S Virtual storage capacity exceeded [RC = 109]
- DMSQRG918E No windows are defined [RC = 4]
- DMSQRG918E No virtual screens are defined [RC = 4]
- DMSQRG921E Window wname is not defined [RC = 28]
- DMSQRG921E Virtual screen vname is not defined [RC = 28]
- DMSQRG1082E No window qualifies as the window on top [RC = 4]
- DMSQRH109S Virtual storage capacity exceeded [RC = 109]
- DMSQRH916E Window wname is not displayed [RC = 28]
- DMSQRH916E Window wname is not hidden [RC = 28]
- DMSQRH917E No windows are displayed [RC = 4]
- DMSQRH917E No windows are hidden [RC = 4]
DMSQRH921E Virtual screen vname is not defined [RC = 28]
DMSQRH921E Window wname is not defined [RC = 28]
DMSQRH926E Command is only valid in CMS FULLSCREEN mode
[RC = 88]
DMSQRS006E No read/write disk accessed [RC = 36]
DMSQRS104S Error nn reading SYSTEM LANGUAGE S from disk
[RC = 1nn]
DMSQRS109S Virtual storage capacity exceeded [RC = 109]
DMSQRS280E Application applid not active [RC = 28]
DMSQRS639E Error in QUERY routine; return code was nnnn [RC = 24]
DMSQRT109S Virtual storage capacity exceeded [RC = 109]
DMSQRT639E Error in QUERY routine; return code was nnnn [RC = 24]
DMSQRU109S Virtual storage capacity exceeded [RC = 109]
DMSQRU639E Error in QUERY routine; return code was nnnn [RC = 24]
DMSQRV109S Virtual storage capacity exceeded [RC = 109]
DMSQRV639E Error in QUERY routine; return code was nnnn [RC = 24]
DMSQRW109T Virtual storage capacity exceeded
DMSQRW639E Error in QUERY routine; return code was nnnn [RC = 24]
DMSQRX070E Invalid parameter parameter [RC = 24]
DMSQRX099E CMS/DOS environment [not] active [RC = 40]
DMSQRX109S Virtual storage capacity exceeded [RC = 109]
DMSQRX639E Error in QUERY routine; return code was nnnn
DMSQRY003E Invalid option: option [RC = 24]
DMSQRY014E Invalid function function [RC = 24]
DMSQRY065E option option specified twice [RC = 24]
DMSQRY066E option1 and option2 are conflicting options [RC = 24]
DMSQRY109S Virtual storage capacity exceeded [RC = 109]
DMSQRY618E NUCEXT failed [RC = nn]
DMSQRY621E Bad plist: message [RC = 24]
RDR

Use RDR to determine the characteristics of the next file in your virtual reader. RDR generates a return code and either displays or stacks a message for each type of file recognized. Which file is "next" depends upon the class of the reader, the class of the files in the reader, and whether or not they are held.

The format of the RDR command is:

\[
\text{RDR} \quad [\text{spool-class}] \quad [(\text{options}...[)])
\]

Options:

- **NOTYPE**
- **STACK**
- **FIFO**
- **LIFO**

**where:**

- **spool-class**
  - is the class of the spool file for which information is to be returned. The virtual reader remains spooled to the class specified in the RDR command.

  =

  - indicates that information is to be returned for a file having the same spool file class as that of the virtual reader. This is the default.

**Options:**

**NOTYPE**

specifies that no message is to be displayed or stacked. However, a return code is generated, which is accessible from within an EXEC 2 (or EXEC) procedure, by examining the variable &RC (or RETCODE).

**STACK**

specifies that the message is placed in the program stack rather than displayed at the terminal. The information is stacked either FIFO (first in first out) or LIFO (last in first out). The default order is FIFO.
FIFO

specifies that the information is placed in the program stack rather than displayed at the terminal. The information is stacked FIFO. The options STACK, STACK FIFO, and FIFO are all equivalent.

LIFO

specifies that the information is placed in the program stack rather than displayed at the terminal. The information is stacked LIFO. This option is equivalent to STACK LIFO.

Usage Notes:

1. Issued with no options, RDR displays the return code and message.

2. Issued with the NOTYPE option from an EXEC (written in the REXX language) RDR places a return code in the variable RC. Appropriate action can be taken by examining this variable. For example:

   rdr '(' notype
   If rc=22 Then disk load
   Else If RC=7 Then readcard

3. If the spool-class specified is different from the current spool class of the virtual reader, the virtual reader’s spool class is changed to the one specified. The current spool class of the virtual reader can be determined by issuing the CP command QUERY VIRTUAL OOC or QUERY VIRTUAL UR.

4. The RDR command changes the order of the files in your virtual reader. Files that are not held are re-ordered according to class.

5. The RDR command does not handle MONITOR files.

Responses:

The return codes and messages are:

0  Reader empty
1  System dump file
2  PRINTER FILE (ITEM LENGTH lrecl)
3  DISK LOAD fn ft fm
4  :READ fn ft fm originid mm/dd/yy hh:mm:ss
5  Cards for IPL
6  Unnamed card deck
7  :READ fn ft fm
9  Reader not operational
13 Reader not ready
18 Console spool file
22 DISK LOAD fn ft fm
23 Netdata file
26 Message

Explanations of the messages follow. (The return code is not part of the message.)
Reader empty RC = 0
    The reader is empty, the reader file is held, or there are no files
    in the reader of the current reader spool class. You can check to
    make sure the reader corresponds to the current spool class, or
    check for held files.

System dump file RC = 1
    The reader contains a system dump file, which can be handled
    using the appropriate system utility.

PRINTER FILE (ITEM LENGTH lrecl) RC = 2
    The reader contains an unnamed printer file with a logical
    record length of lrecl.

DISK LOAD fn ft fm RC = 3
    The reader contains a file sent via DISK DUMP from a VM/370
    Release 6 (or earlier version) CMS file system. The CMS file
    system in VM/370 Release 6 (or earlier) only supports minidisks
    formatted in 800-byte physical blocks.

:READ fn ft fm originid mm/dd/yy hh:mm:ss RC = 4
    The reader contains a non-console or a non-printer file that has a
    READ control card as the first real record. If the PUNCH
    command produced this file, then the result will be in the above
    format; otherwise, the READ control card will be displayed
    through column 72.

Cards for IPL RC = 5
    The reader contains a file that has IPL cards as the first cards in
    the file.

Unnamed card deck RC = 6
    The reader contains a PUNCH file that can not be identified.

:READ fn ft fm RC = 7
    The reader contains a file that is a printer file.

Reader not operational RC = 9
    The reader is not operational: device 00C does not exist in the
    virtual machine configuration or device 00C is not a reader.
    Possible causes: the reader is not defined in the directory; the
    reader was detached; some other device was at 00C. (CMS
    assumes the reader to be at address 00C.)

Reader not ready RC = 13
    The reader is not ready. To reverse the not ready status, issue
    the CP command READY 00C.

Console spool file RC = 18
    The reader contains a file that is from a console.
DISK LOAD fn ft fm  RC = 22
The reader contains a file sent via DISK DUMP from a post-VM/370 Release 6 CMS file system. In addition to the 800-byte physical blocksize used by the VM/370 file system, the enhanced file system supports minidisks formatted in 512-, 1024-, 2048-, or 4096-byte logical blocks.

Netdata file  RC = 23
The reader contains a file that was sent using the SENDFILE command with the NEW option.

Message  RC = 26
The reader contains non-console or non-printer file that has a MSG control card as the first real record.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSRDR070E  Invalid parameter parameter [RC = 24]
DMSRDR630S  Error accessing spool file [RC = 36]
RDRLIST

Use the RDRLIST command to display information about the files in your virtual reader. The RDRLIST environment is controlled by the System Product editor. Therefore, you can use XEDIT subcommands to manipulate the files. In addition, you can look at a given reader file, discard it, copy it to a CMS mini-disk, or transfer it to someone else (local or remote).

In most cases these files were sent to you by other computer users, on your computer or on other computers that are connected to yours via the Remote Spooling Communications Subsystem (RSCS) network.

The format of the RDRLIST command is:

```
RDRLIST

[Options...[]]

Options: [PROFile fn] [Append]
```

where:

**PROFile fn**

specifies the name of an XEDIT macro to be executed when XEDIT is invoked by the RDRLIST command. If not specified, the default macro PROFRLST XEDIT is invoked. For more information on the PROFRLST macro, see the usage note, “Default PF Key Settings.”

**Append**

specifies that the list of files in your reader should be appended to the existing list. This option has meaning only when issued from within RDRLIST, and is ignored otherwise.
1. Tailoring the RDRLIST Command Options

You can use the DEFAULTS command to set up options and/or override command defaults for RDRLIST. However, the options you specify in the command line when entering the RDRLIST command override those specified in the DEFAULTS command. This allows you to customize the defaults of the RDRLIST command, yet override them when you desire. Refer to the DEFAULTS command description for more information.

2. Format of the List

When you invoke the RDRLIST command you are placed in the XEDIT environment, editing a file “userid RDRLIST A1.” The existing copy of this file is erased if it exists.

The file you are editing is a list of files with information collected from the CP QUERY RDR ALL command. Each line contains:

- a command area
- filename and filetype
- class and type
- number of records
- whether or not the file is held
- creation date and time
- originating userid and node

The full power of XEDIT is available to you while you issue commands against the list of files. For example, you may want to use XEDIT subcommands to scroll through the list of files, locate a particular file, etc.

However, some XEDIT subcommands are inappropriate in this environment. Subcommands that alter the format or the contents of “userid RDRLIST” (for example, SET TRUNC, SET FTYPE, or SET LINEND) may cause unpredictable results.
3. Issuing Commands from the List

On a full screen display, you can issue commands directly from the line on which a reader file is displayed. These commands must be CP or CMS commands that operate on reader files (for example, CHANGE RDR, PURGE RDR, TRANSFER RDR, PEEK, DISCARD). For the above commands that operate on the reader files, the spoolid number is automatically appended to the end of the command. Use the slash (/) symbols described below to specify the spoolid elsewhere in the command. For example:

```
CHANGE RDR / CLASS A
RECEIVE / fn ft ( REPLACE
```

To enter a command, just move the cursor to the line that describes the file to be used by the command, and type the command in the space provided to the left of the filename. If a command is longer than the command space provided on the screen, just continue typing over the rest of the line. You press the ENTER key to execute the command. (The ENTER key is set to EXECUTE, which is described in the section “Special Commands,” below.)

For example, to transfer a reader file to a user on your computer (local), you would move the cursor to that line on the screen and type:

```
TRANSFER RDR / USERA
```

where USERA is the userid of the recipient. To transfer a reader file to a user on another computer that is connected to yours (remote), you must know the userid of the user, the userid (rscsid) of the virtual machine at your location that is running the Remote Spooling Communications Subsystem (RSCS), and the location identification (locid) of the computer at the remote location. Move the cursor to that line on the screen and type:

```
TAG FILE / REMOTE1 USERB
```

where REMOTE1 is the locid of the remote computer and USERB is the userid of the recipient. After you have entered the TAG command, on the same line type:

```
TRANSFER RDR / NET
```

where NET is the userid (rscsid) of the virtual machine at your location that is running RSCS.

To purge a file, you would move the cursor up to that line on the screen, and type “discard” in the space provided to the left of the filename. DISCARD is another special command described in the section “Special Commands.” When the ENTER key is pressed, all the commands typed on one screen are executed. The screen is restored to its previous state; however, the list is updated to reflect the current status of the files (see “Responses”).
You may want to enter commands from the RDRLIST command line before executing commands that are typed on the list. To do this, move the cursor to the command line by using the PF12 key (instead of the ENTER key). After typing a command on the command line and pressing ENTER, you can use PF12 to move the cursor back to its previous position on the list.

Another way to issue commands that make use of the reader files displayed is to issue EXECUTE from the RDRLIST command line. A complete description of EXECUTE follows, in the section "Special Commands."

4. Default Key Settings and Synonyms

The PROFRLST XEDIT macro is executed when the RDRLIST command is invoked, unless you specified a different macro in the RDRLIST command. It sets the keys to the following values:

<table>
<thead>
<tr>
<th>Key</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTER</td>
<td>Execute commands typed in the file line(s) or on the command line. (The ENTER key is set by the XEDIT subcommand, SET ENTER IGNORE MACRO EXECUTE).</td>
</tr>
<tr>
<td>PF 1</td>
<td>Help</td>
</tr>
<tr>
<td>PF 2</td>
<td>Refresh</td>
</tr>
<tr>
<td>PF 3</td>
<td>Quit</td>
</tr>
<tr>
<td>PF 4</td>
<td>Sort by filetype, filename.</td>
</tr>
<tr>
<td>PF 5</td>
<td>Sort by date and time, in a calendar year, oldest to newest.</td>
</tr>
<tr>
<td>PF 6</td>
<td>Sort by userid, in alphabetical order.</td>
</tr>
<tr>
<td>PF 7</td>
<td>Backward</td>
</tr>
<tr>
<td>PF 8</td>
<td>Forward</td>
</tr>
<tr>
<td>PF 9</td>
<td>Receive</td>
</tr>
<tr>
<td>PF 10</td>
<td>Not assigned</td>
</tr>
<tr>
<td>PF 11</td>
<td>Peek</td>
</tr>
<tr>
<td>PF 12</td>
<td>Cursor</td>
</tr>
</tbody>
</table>

Note: On a terminal equipped with 24 PF keys, PF keys 13 to 24 are assigned the same values as PF keys 1 to 12 as discussed here.

Some XEDIT subcommands are stacked by the FILELIST command (for example, SET TRUNC, SET LRECL, and SET VERIFY). In order to override these settings in a profile, these SET subcommands must be stacked FIFO.

In addition to setting the above PF keys, the PROFRLST XEDIT macro sets the synonyms that sort your RDRLIST files. Enter the synonyms
on the RDRLIST command line. The synonyms and their descriptions are:

**SNAME** Sorts the list alphabetically by spool filename and spool filetype.

**STYPE** Sorts the list alphabetically by spool filetype and spool filename.

**SCLAS** Sorts the list by device type, class, and hold status.

**SHOLD** Sorts the list by hold status, device type, and class.

**SUSER** Sorts the list by origin userid, node, and date.

**SSIZE** Sorts the list by the number of records (greatest to least).

**SDATE** Sorts the list month, day, and time (oldest to most recent).

5. Displaying a File

To display a file on the screen without reading it onto disk, position the cursor at the file you want to see and press the PF11 key, which is set to the PEEK command. Refer to the PEEK command for more information on the PEEK screen.

6. If you want to issue RDRLIST from an EXEC program, you should precede it with the EXEC command; that is, specify

```plaintext
eexec rdrlist
```

7. You may receive a response on the RDRLIST screen on the line where the reader file is listed. To issue commands on that line, type over the response, press the ERASE EOF key (or space over the rest of the line) and press ENTER. Alternatively, press the PF2 key (REFRESH), move your cursor to the appropriate line, and enter the command.

8. When you press PF9 (RECEIVE), you may receive prompting messages that require a response. See the RECEIVE command for an explanation of the prompts and responses.

**Responses:**

After a command is executed, one of the following symbols is displayed in the “Cmd” space to the left of the file for which it was executed.

* Means the command was executed successfully (RC = 0).

*n Is the return code from the command executed (RC = n).

*? Means the command was an unknown CP/CMS command (RC = -3).
**RDRLIST**

*! Means the command was not valid in CMS subset. For a list of commands valid in CMS subset mode, see the VM/SP CMS User's Guide.

The following responses can also appear on the RDRLIST screen after you have issued a command for a file from your virtual reader:

* spoolfn spoolft ** Discarded or Received **
* spoolfn spoolft has been discarded.
File spoolfn spoolft has been discarded.
* spoolfn spoolft has been left in your reader.

The following response can also appear if RDRLIST is invoked in the CMS environment and you have no reader files:

No files in your reader.

**Messages and Return Codes:**

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSWRL205E No files in your reader [RC = 28]
DMSWRL651E APPEND must be issued from RDRLIST or FILELIST [RC = 40]

**Special Commands Used in the RDRLIST Environment**

Two commands, EXECUTE and DISCARD, make use of the list of files displayed by the RDRLIST command. EXECUTE can be used only in the RDRLIST and FILELIST command environments, and DISCARD can be used only in the RDRLIST, FILELIST, and PEEK command environments.

**EXECUTE**

Use EXECUTE (an XEDIT macro) to issue CP/CMS commands (or EXECs) that make use of the reader spool files displayed by RDRLIST.

EXECUTE may be used in two ways. First, on a display terminal, the command(s) to be executed can be typed directly on the RDRLIST screen and "EXECUTE" entered either on the command line or by pressing the ENTER key. Second, the command to be executed can be typed in the command line, following "EXECUTE," as one of its operands. The command is then executed against one or more reader files in the list.

The format of the EXECUTE macro is:

```
EXECUTE [Cursor]lines [command]
```

where:
Cursor

means that a command is to be executed against the line that contains the cursor. The command can either be typed on the line that describes the file, or it can be typed as an operand of EXECUTE. The CURSOR operand is valid only on display terminals and is particularly useful when assigned to a PF key. For example, if EXECUTE CURSOR PEEK is assigned to a PF key, you can place the cursor on the line describing the file you want to peek at and then press the PF key.

lines

is the number of lines in the file the command is to be executed for. If a command is specified, the default is one (1). You can specify an asterisk (*), which means “execute this command on all lines from the current line to the end of the file.”

command

is a CMS or CP command (or any program or EXEC) that makes use of reader spool files. You can either type out the command operands, or you can use the symbols described below to represent the filename, filetype, spoolid, and device type. (See the usage note, “Using Symbols as Part of a Command.”)

Usage Notes:

1. Entering Commands on a Full Screen Display

You can type commands that operate on reader spool files directly on the lines of the RDRLIST display. When you enter EXECUTE (either from the command line or by pressing the ENTER key), all commands entered on the lines in the file that are currently displayed are executed. The spoolid number of the reader file is appended automatically to the command, unless you typed one of the symbols described below (in usage note 3).

2. Entering Commands on the Command Line

Another way to issue commands that make use of the files displayed is to move the current line to the first (or only) file you want the command to use, and then to issue the EXECUTE subcommand (in the form “EXECUTE lines command”) from the XEDIT command line. This method may be used on both display and typewriter terminals. For example:

First move the current line (by using XEDIT subcommands like UP or DOWN) to the first file you want to use in the command. On a full screen display, the current line is the first file on the screen. Then (in the XEDIT command line) you type:

execute n peek
where “n” is the number of lines to be peeked, starting with the current line. (You can use any command, not just PEEK.)

Note: You can use synonyms or macros to make issuing common commands easier. For example, you might want to set up a command “SEE” to be a synonym for “EXECUTE 1 PEEK.”

3. Using Symbols as Part of a Command

Symbols can be used to represent operands in the command to be executed. They can be used in the commands typed on the list, or as part of the command in EXECUTE (on the command line). Symbols are needed if the command to be executed has operands or options that follow the command name. Examples of using symbols are in the “Examples” section, below. The following symbols can be used:

/ means the spoolid of the file displayed on the line.
/n means the filename displayed on the line.
/t means the filetype displayed on the line.
/o means execute the line as is, without appending anything.
/m means the device type (from which the file was sent).

Any combinations of symbols can be used. For example:

/n /t means: filename followed by filetype.
/nt means: filename followed by filetype.

Note: If the symbol ‘/’ appears in a command or in its operands, it must be issued from the command line, and not as part of an EXECUTE command.

4. Special Symbols Used Alone

The following special symbols can be typed alone on the lines of the RDRLIST display. They have the following meanings:

= means execute the previous command for this file. Commands are executed starting at the top of the screen. For example, suppose you enter DISCARD on a line. You can then type an equal sign on any other line(s) below the DISCARD command. Those files preceded by equal signs are discarded when the EXECUTE command is entered (from the command line or by pressing the ENTER key).

? means display the last command executed. The command is displayed on the line in which the ? is entered.
Messages and Return Codes for EXECUTE:

DMSWEX526E  Option CURSOR valid in display mode only [RC = 3]
DMSWEX543E  Invalid number: number [RC = 5]
DMSWEX561E  Cursor is not on a valid data field [RC = 1 or 3]
DMSWEX651E  EXECUTE must be issued from RDRLIST, FILELIST or MACLIST [RC = 40]
DMSWEX654E  Invalid symbol symbol; /{0 must be specified alone|invalid character char following / symbol} [RC = 24]

On a typewriter terminal only:

Executing: command
+++E(nn)+++ 

DISCARD

Use the DISCARD command to purge a file displayed in the reader list. DISCARD is equivalent to a RECEIVE command issued with the PURGE option. Unlike the CP PURGE command, DISCARD allows an acknowledgment to be sent to the sender (if he requested one). The acknowledgment indicates that the file was discarded. DISCARD also makes an entry in your "userid NETLOG" file, which indicates that this file was discarded. A log entry is made only if the LOG option (the default) is in effect in the RECEIVE command. (For more information on acknowledgments and the "userid NETLOG" file, see the RECEIVE command.)

DISCARD can either be typed in the command area of the line that describes the reader file you want purged, or it can be entered from the command line (at the bottom of the screen).

The format of the DISCARD command as used in the RDRLIST environment is:

```
DISCARD [spoolid]
```

where:

<table>
<thead>
<tr>
<th>spoolid</th>
</tr>
</thead>
<tbody>
<tr>
<td>is the spoolid of the reader file to be purged. If DISCARD is typed on the line that describes the file to be purged, the spoolid is appended automatically.</td>
</tr>
</tbody>
</table>
**RDRLIST**

**Usage Note:**

If you want to enter DISCARD from the XEDIT command line, use EXECUTE, which automatically appends the spoolid. For example,

EXECUTE 1 DISCARD

purges the first file displayed in the reader list (the current line). EXECUTE automatically appends the spoolid to the command (DISCARD) specified. This method is particularly useful on a typewriter terminal.

**Messages and Return Codes for DISCARD:**

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSWDC651E DISCARD must be issued from FILELIST, RDRLIST, MACLIST or PEEK [RC=40]

DMSWDC653E Error executing command; rc = nn [RC=nn]

**Examples:**

In the RDRLIST environment, information about the user’s virtual reader is displayed in a format similar to what the FILELIST command provides about a CMS mini-disk.

The following is a sample RDRLIST screen.

<table>
<thead>
<tr>
<th>Filetype</th>
<th>Class</th>
<th>User</th>
<th>At Node</th>
<th>Hold</th>
<th>Records</th>
<th>Date</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOPPINGS</td>
<td>PUN</td>
<td>A</td>
<td>KEN</td>
<td>NODE04</td>
<td>NONE</td>
<td>10/06</td>
<td>10:39:38</td>
</tr>
<tr>
<td>ASSEMBLE</td>
<td>PUN</td>
<td>A</td>
<td>KEN</td>
<td>NODE04</td>
<td>NONE</td>
<td>10/06</td>
<td>10:25:11</td>
</tr>
<tr>
<td>SCRIPT</td>
<td>PRT</td>
<td>A</td>
<td>KEN</td>
<td>NODE04</td>
<td>7</td>
<td>10/06</td>
<td>10:15:50</td>
</tr>
<tr>
<td>TIPS</td>
<td>PUN</td>
<td>A</td>
<td>KEN</td>
<td>NODE04</td>
<td>11</td>
<td>10/06</td>
<td>09:40:28</td>
</tr>
<tr>
<td>NOTE</td>
<td>PUN</td>
<td>A</td>
<td>KEN</td>
<td>NODE04</td>
<td>2</td>
<td>10/05</td>
<td>11:44:34</td>
</tr>
</tbody>
</table>

Figure 26. Sample RDRLIST Screen

472 VM/SP CMS Command Reference
Examples of Using Symbols:

The following examples show how symbols can be used to represent operands in a command. The values substituted for the symbols and the resulting command are shown. In each case, the command can be entered in either of the following ways:

- typed in the “Cmd” area of the screen. The command is executed either by pressing the ENTER key or by entering EXECUTE on the XEDIT command line and then pressing ENTER.

- entered from the XEDIT command line, as an operand of EXECUTE (in the form “EXECUTE lines command”).

If a symbol is not specified, the spoolid number of the reader file is appended automatically to the command.

<table>
<thead>
<tr>
<th>SPOOL FILEID</th>
<th>COMMAND</th>
<th>RESULTING COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>pizza toppings</td>
<td>DISCARD</td>
<td>DISCARD spoolid</td>
</tr>
<tr>
<td>cookie assemble</td>
<td>RECEIVE / CAKE</td>
<td>RECEIVE spoolid CAKE</td>
</tr>
<tr>
<td></td>
<td>/t (REPLACE</td>
<td>ASSEMBLE (REPLACE</td>
</tr>
<tr>
<td>ken note</td>
<td>PEEK</td>
<td>PEEK spoolid</td>
</tr>
<tr>
<td>send exec</td>
<td>FILELIST /n **</td>
<td>FILELIST SEND **</td>
</tr>
<tr>
<td>$jelly script</td>
<td>TRANSFER RDR /</td>
<td>TRANSFER RDR spoolid</td>
</tr>
<tr>
<td></td>
<td>TO * PRT</td>
<td>TO * PRT (prints the file)</td>
</tr>
</tbody>
</table>
READCARD

Use the READCARD command to read data records from your virtual reader and to create CMS disk files containing the data records.

The format of the READCARD command is:

```
READcard { fn ft [fm] } [options...[]] [options...[]] [options...[]]
```

where:

- **fn** is the filename you want to assign to the file being read.

- **ft** is the filetype you want to assign to the file being read.

- **[*]** indicates that file identifiers are to be assigned according to READ control cards in the input deck.

- **fm** is the filemode letter of the disk onto which the file is to be read and the filemode number of the file. A filemode of A1 is assumed if this field is omitted or specified as an asterisk (*) on the command. However, if a filemode number is on the control card, that number is used with A. When a filemode letter is specified on the command, the default filemode number is 1, unless the filemode number is on the control card. Specifying a filemode letter and number on the command ignores the filemode letter and number on the control card and assigns that filemode to the file.
Options:

**Fullprompt**

specifies that a prompt is issued for each file.

**Minprompt**

specifies that a prompt is issued when the name of the first (or only) file differs from the name of the spool file; the prompt for the first file is suppressed when it has the same name as the spool file. A prompt is always issued for the second and subsequent files. This will only occur if READCARD * is specified.

**NOPrompt**

specifies that a prompt is not issued to you as a file is received. NOPROMPT is the default.

**Replace**

specifies that if a file of the same filename and filetype exists on the disk onto which the incoming file is to be loaded, it is to be replaced with this one. REPLACE is the default.

**NOREplace**

specifies that a file is not received that would overlay an existing file on the receiving disk.

Usage Notes:

1. Data records read by the READCARD command must be fixed-length records, and may be a minimum of 80 and a maximum of 204 characters.

2. If you specify the FULLPROMPT or MINPROMPT option the valid responses include:

   - One of the digits specified in the prompt
   - One of the parenthetical words that follows a digit or any initial truncation of the word.

The meanings of these responses are:

<table>
<thead>
<tr>
<th>Response</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 or No</td>
<td>If this file is one of a set of files that constitutes a single spool file, the file is not received and prompting continues for the next file, if there is one. If this is the last file of a set of files or if this is the only file in the spool file, the command is ended.</td>
</tr>
<tr>
<td>1 or Yes</td>
<td>Receives the file under the name fn1 ft1 fm1 (or fn3 ft3 fm3).</td>
</tr>
</tbody>
</table>
2 or Quit  Ends the command.

3 or Rename  Requests prompt message DMSRDC1080R so that the incoming file can be received using a different name.

3. If you receive prompt message DMSRDC1081R the valid responses include:

- One of the digits specified in the prompt
- One of the parenthetical words that follows a digit or any initial truncation of the word.

The meanings of these responses are:

<table>
<thead>
<tr>
<th>Response</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 or No</td>
<td>Does not receive the file under the name fn ft fm and repeats the original prompt message DMSRDC1080R. This allows you to specify a different name for the incoming file.</td>
</tr>
<tr>
<td>1 or Yes</td>
<td>Receives the file under the name fn ft fm.</td>
</tr>
<tr>
<td>2 or Quit</td>
<td>Ends the command.</td>
</tr>
</tbody>
</table>

4. CMS disk file identifiers are assigned according to READ control cards in the input deck (the PUNCH command header card is a valid READ control card). When you enter the command:

    readcard *

CMS reads the first spool reader file in the queue and if there are READ control cards in the input stream, it names the files as indicated on the control cards.

The first card in the deck may not be a READ control card. If not, CMS writes a file named READCARD CMSUT1 A1 to contain the data, until a READ control card is encountered or until the end-of-file is reached.

5. If you specify a filename and filetype on the READCARD command, for example:

    readcard junk file

CMS does not check the input stream for READ control cards, but reads the entire spool file onto the disk and assigns it the specified filename and filetype.

If there were any READ control cards in the deck, they are not removed. Delete them using the editor if you do not want them in your file. If the file is too large, you can either increase the size of your
virtual storage (using the CP DEFINE command), or use the COPYFILE command to copy all records except the READ control cards (using the FROM and FOR options).

6. READCARD loads a file from the reader into a temporary work file called "READCARD CMSUT2." The existing file with the same name as the one being loaded from the reader is then erased. The name of the temporary work file just created is changed to the name of the file just received. However, if the file you are loading has the name "READCARD CMSUT2," it will be changed to "READCARD CMSUT3." "READCARD CMSUT2" is a reserved work filename of the READCARD command.

7. To read a file onto a disk other than your A-disk, specify the filemode letter when you specify the filename and filetype; for example:

    readcard junk file c

Or, if you want the READ control card to determine the filenames and filetypes, you can enter:

    readcard * * c

8. If you are preparing real or virtual card decks to send to your own or another user's virtual card reader, you may insert READ control cards to designate filenames, filetypes, and optionally, filemode numbers, to be assigned to the disk file(s).

A READ control card must begin in column 1 and has the format:

    :READ filename filetype filemode

Each field must be separated by at least one blank; the second character of the filemode field, if specified, must be a valid filemode number (0 through 6). The filemode letter is ignored when this file is read, since the mode letter is determined by specifications on the READCARD command line.

9. To send a real card deck to your own or another user's virtual card reader, punch a CP ID card to precede the deck. The ID card has the keyword ID or USERID in column 1, followed by the userid you want to receive the file and optionally, spool file class and name designations; for example:

    ID MARY CLASS A NAME LITTLE LAMB

Each field must be separated from the others by at least one blank.

10. If the reader file being processed contains carriage control characters, the READCARD command returns the records with the carriage control characters stripped off.
READCARD

Responses:

READCARD issues the following responses, depending on the situation.

1. If the spooled card reader contains no records after the control card:

   DMSRDC701I Null file

2. If READCARD * was issued and prompting is not in effect, this response indicates that a record beginning with :READ has been found in the spool file and the following fileid is invalid:

   DMSRDC702E Missing, invalid, or incomplete fileid in following READ control card:
   :READ... Command terminated

3. If READCARD * was issued and a control card was encountered in the input card stream, this response indicates the names assigned to each file:

   DMSRDC702I :READ...

4. If READCARD * was issued and the first record in the spool file is not a READ control card, this response is issued when a READ control card in the spool file has been identified and validated, and it is listed at the terminal:

   DMSRDC702I READ control card missing. Following assumed:
   :READ READCARD CMSUT1 A1

5. If READCARD * was issued and prompting is in effect, this response indicates that a record beginning with :READ has been found in the spool file and the following fileid is invalid:

   DMSRDC702W Missing, invalid, or incomplete fileid in following READ control card:
   :READ... Fileid changed to READCARD CMSUT1

6. If the records being read are not 80 bytes long, this message gives the length:

   DMSRDC738I Record length is nnn bytes

7. If you specify the FULLPROMPT or MINPROMPT option one of these prompts will be displayed:
READCARD

DMSRDC1079R Receive fn1 ft1 fm1?
   Reply 0 (NO), 1 (YES), 2 (QUIT), or 3 (RENAME)

Receive fn1 ft1 fm1 and replace the existing file
   of the same name?
   Reply 0 (NO), 1 (YES), 2 (QUIT), or 3 (RENAME)

Receive fn1 ft1 fm1 and replace fn2 ft2 fm2?
   Reply 0 (NO), 1 (YES), 2 (QUIT), or 3 (RENAME)

Receive fn1 ft1 fm1 as fn3 ft3 fm3?
   Reply 0 (NO), 1 (YES), 2 (QUIT), or 3 (RENAME)

Receive fn1 ft1 fm1 as fn3 ft3 fm3 and
   replace the existing file of the same name?
   Reply 0 (NO), 1 (YES), 2 (QUIT), or 3 (RENAME)

Receive fn1 ft1 fm1 as fn3 ft3 fm3 and replace
   fn2 ft2 fm2?
   Reply 0 (NO), 1 (YES), 2 (QUIT), or 3 (RENAME)

- The fileid fn1 ft1 fm1 is the name from the card stream of the spool file.
- The phrase "and replace the existing file of the same name?" appears when the operation replaces an existing file and the filemode of that file is the same as fm1.
- The phrase "and replace fn2 ft2 fm2." appears when the operation replaces an existing file and the filemode of that file is not fm1.
- The fileid fn3 ft3 fm3 is the name from the card stream of the spool file that you may specify when the name differs from the name of the incoming file.

8. If you respond with a 3 (or RENAME) to prompt message DMSRDC1079R, the following message appears and you must enter a fileid in the form fn [ft [fm]].

DMSRDC1080R Enter the new name for fn ft fm

9. If you respond to prompt message DMSRDC1080R with a fileid that names an existing file, you receive this prompt:

DMSRDC1081R Replace fn ft fm?
   Reply 0 (NO), 1 (YES), or 2 (QUIT)

10. When READCARD * is entered it will list each READ control card in the spool file and, after it loads an incoming file, it issues one of the following responses. Also, if READCARD fn ft is entered it issues one of the following responses.

- If the incoming file (fn1 ft1 fm1) does not already exist and is received without being renamed, you receive

  fn1 ft1 fm1 created

Chapter 2. CMS Commands  479
READCARD

- If the incoming file \((fn1 ft1 fm1)\) is renamed to a filename \((fn2 ft2 fm2)\) that does not already exist, you receive
  \[\text{fn2 ft2 fm2 created from fn1 ft1 fm1}\]

- If the incoming file \((fn1 ft1 fm1)\) is copied to an existing data set that has the same name as the incoming file, you receive
  \[\text{fn1 ft1 fm1 replaced}\]

- If the incoming file \((fn1 ft1 fm1)\) is copied to an existing file \((fn2 ft2 fm2)\) with a name different from that of the incoming file, you receive
  \[\text{fn2 ft2 fm2 replaced by fn1 ft1 fm1}\]

- If the incoming file \((fn1 ft1 fm1)\) replaces an existing file \((fn2 ft2 fm2)\), but is given a mode \((fm1)\) that differs from the mode of the existing file \((fm2)\), you receive
  \[\text{fn1 ft1 fm1 replaced fn2 ft2 fm2}\]

- If the incoming file \((fn1 ft1 fm1)\) replaces an existing file \((fn2 ft2 fm2)\), but is given a mode \((fm3)\) that differs from the mode of the existing file \((fm2)\), you receive
  \[\text{fn3 ft3 fm3 replaced fn2 ft2 fm2 sent as fn1 ft1 fm1}\]

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

\begin{itemize}
  \item DMSRDC008E Device \textit{vdev} [invalid or nonexistent] is an unsupported device type [RC = 36]
  \item DMSRDC024E File \textit{fn ft fm} already exists; specify REPLACE option [RC = 28]
  \item DMSRDC037E Disk \textit{mode} ([\textit{vdev}]) is accessed as read/only [RC = 36]
  \item DMSRDC042E No fileid specified [RC = 24]
  \item DMSRDC054E Incomplete fileid specified [RC = 24]
  \item DMSRDC062E Invalid * in fileid [RC = 20]
  \item DMSRDC069E Disk \textit{mode} not accessed [RC = 36]
  \item DMSRDC105S Error \textit{nn} writing file \textit{fn ft fm} on disk [RC = 100]
  \item DMSRDC124S Error reading card file [RC = 100]
  \item DMSRDC205W Reader empty or not ready [RC = 8]
  \item DMSRDC257T Internal system error at address \textit{address} (offset \textit{offset})
  \item DMSRDC639E Error in \textit{routine} routine; return code was \textit{nnnn}
  \item DMSRDC671E Error loading [file] \textit{fn ft fm}; \textit{rc = nn} from RENAME [RC = 100]
  \item DMSRDC1123E Unknown response \textit{text} ignored
  \item DMSRDC1124W Spool file \textit{spoolid} has been left in your reader because one or more files were not received [RC = 1]\
\end{itemize}
Use the RECEIVE command to read onto disk one of the files or notes that is in your virtual reader. In most cases these files were sent to you by other computer users, on your computer or on other computers that are connected to yours via the Remote Spooling Communications Subsystem (RSCS) network.

The format of the RECEIVE command is:

```
RECEIVE [spoolid [fn [ft [fm]]]] [options...[]]
```

Options:

- `NOTebook fn`
- `NOTebook *`
- `Log NOlog`
- `Purge Fullprompt`
- `Replace NOReplace`
- `Olddate`
- `NEwdate` `STack`

**where:**

- **spoolid**
  
  specifies which file in the virtual reader is to be received. The default is ‘=’ or ‘next’ which means the ‘next’ file in the reader is received.

  The ‘next’ file is the one for which the RDR command returns information. Which file this is depends on the class of the reader, the class of the files in the reader, and whether or not they are held.

  To give the file a new file identifier, you must specify the spoolid, ‘=’, or ‘next’.

- **fn**
  
  is the filename the file is to be given. The default is =, which means the file’s present name is used.

- **ft**
  
  is the filetype the file is to be given. The default is =, which means the file’s present type is used.
receive

$fm$
is the filemode the file is to be given. If not specified, the default is
"A."

If the file being received is a note (prepared by the NOTE command), or if
the PURGE option is specified, the operands fn, ft, and fm are ignored. If
the file being received is an acknowledgment, all parameters and all options
(except the spoolid and the PURGE option) are ignored. (See the usage
note, "Acknowledgments," for more information on acknowledgments.)

Options:

NOTEbook fn
causes the file to be saved as a note in a file named “fn NOTEBOOK.”
You can use this option if you want the note(s) from this person to be
kept in a separate file. If you do not specify a notebook filename here,
a filename is first searched for in the sender’s entry in your “userid
NAMES” file and then in a file set up by the DEFAULTS command.
If neither contains a notebook filename, the note is saved in the
default notebook file, “ALL NOTEBOOK.” A note is saved by
appending it to the NOTEBOOK file, with a line of 73 equal signs (=)
separating each note.

If the file is not a note (prepared by the NOTE command), this option
is ignored.

See the NAMEFIND or NAMES command description for more
information on the relationship between a “userid NAMES” file and
the NOTEBOOK file.

NOTEbook *
specifies that note is saved in a file named “name NOTEBOOK,”
where “name” is the value of the Notebook tag in the sender’s entry in
your “userid NAMES” file, or the sender’s nickname, or the sender’s
userid (whichever is located first).

If the file is not a note (prepared by the NOTE command), this option
is ignored.

Log
specifies that the recipients, date, and time of this file transmission
are logged in a file called “userid NETLOG.” This log is updated when
acknowledgments of sent files are received (if they were requested).
Do not use this option if you have no read/write disk accessed.

NOLog
specifies that this file transmission is not to be logged.

Purge
specifies that this file is to be purged and not read onto disk.
Fullprompt
specifies that a prompt is issued for each file without regard to the format of the spool file being processed.

If the file being received is a NOTE file, FULLPROMPT is ignored.

Minprompt
specifies that a prompt is issued when the name of the first (or only) file differs from the name of the spool file; the prompt for the first file is suppressed when it has the same name as the spool file. A prompt is always issued for the second and subsequent files. This prompt is issued only for files in DISK DUMP and NETDATA format. MINPROMPT is the default.

If the file being received is a NOTE file, MINPROMPT is ignored.

NOPrompt
specifies that a prompt is not issued to you as a file is received.

Replace
specifies that if a file of the same filename and filetype exists on the disk onto which the incoming file is to be loaded, it is to be replaced with this one.

NOREPLACE
specifies that a file is not received that would overlay an existing file on the receiving disk. NOREPLACE is the default.

Olddate
means that when a file that was sent in DISK DUMP format is received, it is written to disk with its original date and time (that is, the date and time it was created or last updated by the sender), not the date and time you received it. When a file that was sent in NETDATA format is received, it is written to disk with its original date and time unless it was sent from a different time zone, in which case the date and time are changed to reflect GMT (Greenwich Mean Time). See the usage note, “Receiving NETDATA Files Using the Olddate Option.” For more information on NETDATA format, see the SENDFILE command, the usage note “Format of the File Sent by SENDFILE.”

NEwdate
means to re-date the file to the current date and time it is received.

STack
specifies that the message returned when RECEIVE completes successfully should be stacked (LIFO). If this option is not specified, the messages from RECEIVE are displayed at the terminal.
Usage Notes:

1. Tailoring the RECEIVE Command Options

You can use the DEFAULTS command to set up options and/or override command defaults for RECEIVE. However, the options you specify in the command line when entering the RECEIVE command override those specified in the DEFAULTS command. This allows you to customize the defaults of the RECEIVE command, yet override them when you desire. Refer to the DEFAULTS command description for more information.

2. Why Should I Use Receive?

You should use RECEIVE instead of READCARD or DISK for general purpose use, because RECEIVE calls either READCARD or DISK, whichever is appropriate. It also handles notes, acknowledgments, etc. In fact, RECEIVE handles most of the various formats of files that can appear in your virtual reader. RECEIVE is the only way to read a file that was sent using the SENDFILE command issued with the NEW option.

RECEIVE is particularly useful within the RDRLIST command environment, where it is assigned to the PF9 key.

You may send multiple files by continuous spooling (using CP SPOOL PUNCH CONT) or by a series of DISK DUMP commands but those methods are discouraged. As a sender, you are encouraged to do the following:

- Always use SENDFILE, which resets any continuous spooling options in effect.
- Do not spool the punch continuous.

Similarly, if the punch is spooled continuous and PUNCH is used to send multiple files, the file is read in as one file with "READ" cards imbedded. In this case, although no files are overlaid, the recipient must divide the file into individual files. This problem can also be avoided by using SENDFILE or by not spooling the punch continuous.

3. Acknowledgments

Acknowledgments can be sent to users on different computers connected by the RSCS network so that they can be sure that a file they sent was received.

The sender can specify on the SENDFILE or NOTE command that an acknowledgment be returned to him when a file is RECEIVED. The SENDFILE command must be issued with the NEW option (the default) in order to request an acknowledgment; otherwise, the request is ignored. Even if a recipient discards a file (using the DISCARD
RECEIVE

command), an acknowledgment is returned to the sender. This is possible because DISCARD is equivalent to a RECEIVE issued with the PURGE option. (For more information on DISCARD, see the RDRLIST command.) The acknowledgment indicates whether the file was received (written to disk) or discarded (purged).

When you RECEIVE an acknowledgment that appears in your reader, all parameters and all options (except the spoolid and the PURGE option) are ignored. The acknowledgment is used to make an entry in your "userid NETLOG" file. This entry confirms that the file you sent was received (or discarded). The format of entries in the "userid NETLOG" file is shown in the "Examples" section, below.

4. Responding to Prompting Messages

If you specify the FULLPROMPT or MINPROMPT option the valid responses include:

- One of the digits specified in the prompt
- One of the parenthetical words that follows a digit or any initial truncation of the word.

The meanings of these responses are:

<table>
<thead>
<tr>
<th>Response</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 or No</td>
<td>If this file is one of a set of files that constitutes a single spool file, the file is not received and prompting continues for the next file, if there is one. If this is the last file of a set of files or if this is the only file in the spool file, the command is ended.</td>
</tr>
<tr>
<td>1 or Yes</td>
<td>Receives the file under the name fn1 ft1 fm1 (or fn3 ft3 fm3).</td>
</tr>
<tr>
<td>2 or Quit</td>
<td>Ends the command.</td>
</tr>
<tr>
<td>3 or Rename</td>
<td>Requests prompt message DMSWRC1080R so the incoming file can be received using a different name.</td>
</tr>
</tbody>
</table>

5. If you receive prompt message DMSWRC1081R the valid responses include:

- One of the digits specified in the prompt
- One of the parenthetical words that follows a digit or any initial truncation of the word.

The meanings of these responses are:
**RECEIVE**

<table>
<thead>
<tr>
<th>Response</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 or No</td>
<td>Does not receive the file under the name fn ft fm and repeats the original prompt message DMSWRC1080R. This allows you to specify a different name for the incoming file.</td>
</tr>
<tr>
<td>1 or Yes</td>
<td>Receives the file under the name fn ft fm.</td>
</tr>
<tr>
<td>2 or Quit</td>
<td>Ends the command.</td>
</tr>
</tbody>
</table>

6. Which prompt is most useful for you?

- If you do not issue either a QUERY RDR command or a RDRLIST command specify FULLPROMPT.
- If you do issue a QUERY RDR command before issuing the RECEIVE command or if you issue RECEIVE from a RDRLIST screen specify MINPROMPT.
- If you issue the RECEIVE command in a controlled environment where the identity of all incoming files are known, specify NOPROMPT.

7. Special NETDATA Files from MVS with TSO Extensions (PP)

The MVS with TSO Extensions Program Product can send an empty file, in which case RECEIVE will give you an error message indicating that no file was created on disk. It can also send, as a unique case of multiple files in one transmission, one note and a data file together. The note will be the first file in the transmission and the data file will be second. RECEIVE will add the note to the appropriate notebook, receive the data file, and give informative messages for each action. This is the only form of multiple NETDATA files supported by the RECEIVE command.

*Note:* RECEIVE will not handle partitioned data sets or data sets that have been encrypted by Access Method Services. These files will not be received and remain in the reader. Multiple NETDATA transmissions that do not have a note as the first file, result in the first file being received and the other file(s) ignored. The entire spool file is left in the user’s reader.

8. How does RECEIVE determine the fileid?

If you do not specify a fn, ft, or fm, then RECEIVE determines the fileid according to the method that was used to send the file.
Responses:

1. If you specify the FULLPROMPT or MINPROMPT option one of these prompts is displayed:
**RECEIVE**

DMSWRCl079R Receive fnl ft1 fml?
Reply 0 (NO), 1 (YES), 2 (QUIT), or 3 (RENAME)

Receive fnl ft1 fml and replace the existing file of the same name?
Reply 0 (NO), 1 (YES), 2 (QUIT), or 3 (RENAME)

Receive fnl ft1 fml and replace fn2 ft2 fm2?
Reply 0 (NO), 1 (YES), 2 (QUIT), or 3 (RENAME)

Receive fnl ft1 fml as fn3 ft3 fm3?
Reply 0 (NO), 1 (YES), 2 (QUIT), or 3 (RENAME)

Receive fnl ft1 fml as fn3 ft3 fm3 and replace the existing file of the same name?
Reply 0 (NO), 1 (YES), 2 (QUIT), or 3 (RENAME)

Receive fnl ft1 fml as fn3 ft3 fm3 and replace fn2 ft2 fm2?
Reply 0 (NO), 1 (YES), 2 (QUIT), or 3 (RENAME)

- The fileid fnl ft1 fml is the name from the card stream of the spool file.
- The phrase “and replace the existing file of the same name?” appears when the operation replaces an existing file and the filemode of that file is the same as fml.
- The phrase “and replace fn2 ft2 fm2.” appears when the operation replaces an existing file and the filemode of that file is not fml.
- The fileid fn3 ft3 fm3 is the name from the card stream of the spool file that you may specify when the name differs from the name of the incoming file.

2. If you respond with a 3 (or RENAME) to prompt message DMSWRCl079R, this message appears and you must enter a fileid of the form fn [ft [fm]].

DMSWRCl080R Enter the new name for fn ft fm

3. If you respond to prompt message DMSWRCl080R with a fileid that names an existing file, you receive this prompt:

DMSWRCl081R Replace fn ft fm?
Reply 0 (NO), 1 (YES), or 2 (QUIT)

4. Spool files in the DISK DUMP or PUNCH format will issue the following response.

File fnl ft1 fml received from userid at node sent as fn2 ft2 fm2

5. Spool files in the NETDATA format issue any one of the following responses.

- If the incoming file (fnl ft1 fml) does not already exist:
File fn2 ft2 fm2 created from fn1 ft1 fml received from userid at node

- If the incoming file (fn1 ft1 fml) replaces an existing file (fn2 ft2 fm2):

  File fn2 ft2 fm2 replaced by fn1 ft1 fml received from userid at node

- If the incoming file (fn1 ft1 fml) replaces an existing file (fn2 ft2 fm2), but is given a mode (fm3) that differs from the mode of the existing file (fm2):

  File fn3 ft3 fm3 replaced fn2 ft2 fm2 with fn1 ft1 fml received from userid at node

6. RECEIVE issues the following responses for spool files in the DISK DUMP format if you specify the operands spoolid fn ft fm and if the incoming file is given a different name or placed on a disk other than your A-disk.

- If the first (or only) file in the spool file has the same filename and filetype as an existing file on your A-disk, you will get these responses:

  fnl ftl fml created
  fnl ftl fml saved in a temporary file

  - If an error does not occur you receive this response so that you are aware that your original file has not been destroyed:

    fnl ftl fml copied to fn2 ft2 fm2 and original fnl ftl fml restored

    - If an error does occur you receive this response and RECEIVE resumes processing.

      original fnl ftl fml restored

- If the first (or only) file in the spool file has a filename and filetype that is not the name of an existing file on your A-disk, and the incoming file is to be placed on a disk other than your A-disk, you will get these responses.

  fnl ftl fml created
  fnl ftl fml copied to fn2 ft2 fm2 and fnl ftl fml then erased

7. Other responses include:
RECEIVE

File spfn spft has been discarded
Note spfn spft has been discarded
Note spfn spft added to fn NOTEBOOK fm
Ackn date time added to userid NETLOG
Ackn date time has been discarded

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSWRC006E  No read/write disk accessed [RC = 36]
DMSWRC024E  File fn [ft fm] already exists[; specify REPLACE option] [RC = 28]
DMSWRC037E  Disk mode is accessed as read/only[; A must be R/W for DISK LOAD] [RC = 36]
DMSWRC062E  Invalid {character [char]*} in [output] fileid [fn ft [fm]] [RC = 20]
DMSWRC069E  Disk mode not accessed [RC = 36]
DSMWRC630S  Error accessing spool file [RC = 36]
DMSWRC643E  No class class files in your reader [RC = 28]
DMSWRC644E  All reader files are in HOLD status or not class class [RC = 28]
DMSWRC655E  Spoolid nnnn does not exist [RC = 28]
DMSWRC671E  Error receiving file fn ft fm; rc = nn from command [RC = 100]
DMSWRC672E  Virtual reader invalid or not defined [RC = 36]
DMSWRC674E  Reader is not ready [RC = 36]
DMSWRC681E  This is an unnamed file; specify filename and filetype [RC = 88]
DMSWRC682E  Error copying file fn ft A to {fn ft fm|mode disk.}; rc = nn from COPYFILE [RC = 100]
DMSWRC687E  This is a [SYSTEM|HELD|DUMP]file|file with a special format] and cannot be received [RC = 1]
DMSWRC1123E Unknown response text ignored
DMSWRC1124W Spool file spoolid has been left in your reader because one or more files were not received [RC = 1]

Examples:

Format of the “userid NETLOG” File:

The format of entries in the “userid NETLOG” file maintained by SENDFILE and RECEIVE is shown below. If both the “ACK” and “LOG” options of SENDFILE or NOTE are specified, a “sent to” record is placed in the NETLOG file. When an acknowledgment is received, it is also placed in this file.
In this example, the user sent himself a file (SMALL DATA) using SENDFILE with the LOG and ACK options specified. The first line in the NETLOG file was placed in the file by the SENDFILE command.

He then used RECEIVE (with the LOG option) to read the file onto disk. The second line was added when the file was received. (In this case the sender was the receiver.) The “recv” in this line means “received.” If a file is discarded (using DISCARD), the line contains “disc” instead of “recv.” (The file can be RECEIVEd with a different fileid than it was sent as.)

Last, he received an acknowledgment. It indicates whether the recipient received (“recv”) or discarded (“disc”) the file.
REFRESH

Use the REFRESH command to update virtual screens and their associated windows and refresh the screen.

The format of the REFRESH command is:

| REFresh |

Usage Notes:

None.

Responses:

None.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSREF391E  Unexpected operand(s): operand  [RC = 24]
- DMSREF622E  Insufficient free storage  [RC = 104]
- DMSREF917E  No windows are displayed  [RC = 4]
- DMSREF925E  I/O error on screen  [RC = 100]
- DMSREF926E  Command is only valid on a display terminal  [RC = 88]
RELEASE

Use the RELEASE command to free an accessed disk and make the files on it unavailable.

The format of the RELEASE command is:

\[
\text{RELEASE} \{vdev\} \[(\text{DET}[)]\]
\]

where:

- **vdev**
  - is the virtual device address of the disk that is to be released.
  - Valid addresses are 001 through 5FF for a virtual machine in basic control mode and 001 through FFF for a virtual machine in extended control mode.

- **mode**
  - is the mode letter at which the disk is currently accessed.

Option:

- **DET**
  - specifies that the disk is to be detached from your virtual machine configuration; CMS calls the CP command DETACH.

Usage Notes:

1. If a disk is accessed at more than one mode letter, the RELEASE vdev command releases all modes. If you access a disk specifying the mode letter of an active disk, the first disk is released.

2. You cannot release the system disk (S-disk).

3. When a disk is released, the user file directory is freed from storage and that storage becomes available for other CMS commands and programs. When you release a read/write CMS disk, either with the RELEASE command or implicitly with the FORMAT command or ACCESS commands, the user file directory is sorted and rewritten on disk; user(s) who may subsequently access the same disk may have a resultant favorable decrease in file search time.
RELEASE

4. When a disk is released, any read-only extensions it may have are not released. The extensions may be referred to by their own mode letters. If a disk is then accessed with the same mode as the original parent disk, the original read-only extensions remain extensions to the new disk at that mode.

5. In CMS/DOS, when you release a disk, any system or programmer logical unit assignments made for the disk are unassigned.

6. The RELEASE command rewrites the file directory on any CMS disk accessed in R/W mode whether or not the disk was altered.

Example:

If you want to release and detach the 498 disk that is accessed as your B-disk, then issue:

release 498 (det
or
release b (det

Responses:

DASD vdev DETACHED
   This is a CP message that is issued when you use the DET option. It indicates that the disk has been detached.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSARE017E Invalid device address vdev [RC = 24]
- DMSARE028E No device specified [RC = 24]
- DMSARE048E Invalid mode mode [RC = 24]
- DMSARE069E Disk mode not accessed [RC = 36]
- DMSARE070E Invalid parameter parameter [RC = 24]
RENAME

Use the RENAME command to change the fileid of one or more CMS files on a read/write CMS disk.

The format of the RENAME command is:

```
Rename fileid1 fileid2 [ (options...[])]
```

Options:

<table>
<thead>
<tr>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Displays, at the terminal, the new identifiers of all the files that are renamed. The file identifiers are displayed only when an asterisk (*) is specified for one or more of the file identifiers (fn, ft, or fm) in fileid1.</td>
</tr>
<tr>
<td>NOType</td>
<td>Suppresses at the terminal, displaying of the new file identifiers of all files renamed.</td>
</tr>
<tr>
<td>UPdirt</td>
<td>Updates the master file directory upon completion of this command.</td>
</tr>
</tbody>
</table>

where:

`fileid1` is the file identifier of the original file whose name is to be changed. All components of the fileid (filename, filetype, and filemode) must be coded, with either a name or an asterisk. If an asterisk is coded in any field, any file that satisfies the other qualifications is renamed.

`fileid2` is the new file identifier of the file. All components of the file (filename, filetype, and filemode) must be coded, with either a name or an equal sign; if an equal sign (=) is coded, the corresponding file identifier is unchanged. The output filemode can also be specified as an asterisk (*), indicating that the filemode is not changed.
# RENAME

**NOUPdirt**

suppresses the updating of the master file directory upon completion of this command. (See Usage Note 3.)

**Usage Notes:**

1. When you code an asterisk (*) in any portion of the input fileid, any or all of the files that satisfy the other qualifiers may be renamed, depending upon how you specify the output fileid. For example:

   ```rename * assemble a test file a```

   results in the first ASSEMBLE file found on the A-disk being renamed to TEST FILE. If more than one ASSEMBLE file exists, error messages are issued to indicate that they cannot be renamed.

   If you code an equal sign (=) in an output fileid in a position corresponding to an asterisk in an input fileid, all files that satisfy the condition are renamed. For example:

   ```rename * assemble a = oldasm =```

   renames all files with a filetype of ASSEMBLE to files with a filetype of OLDASM. Current filenames are retained.

2. You cannot use the RENAME command to move a file from one disk to another. You must use the COPYFILE command if you want to change filemode letters.

   You can use the RENAME command to modify filemode numbers, for example,

   ```rename * module a1 = = a2```

   changes the filemode number on all MODULE files that have a mode number of 1 to a mode number of 2.

   **Note:** You can invoke the RENAME command from the terminal, from an EXEC file, or as a function from a program. If RENAME is invoked as a function or from an EXEC file that has the &CONTROL NOMSG option in effect, the message DMSRNM002E (“File fn ft fm not found”) is not issued.

3. Normally, the file directory for a CMS disk is updated whenever you issue a command that affects files on the disk. When you use the NOUPDIRT option of the RENAME command, the file directory is not updated until you issue a command that writes, updates, or deletes any file on the disk, or until you explicitly release the disk (with the RELEASE command).
Responses:

```
newfn newft newfm
```

The new filename, filetype, and filemode of each file altered is displayed when the TYPE option is specified and an asterisk was specified for at least one of the file identifiers (fn, ft or fm) of the input fileid.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

- **DMSRNM002E** File [fn [ft [fm/]]] not found [RC = 28]
- **DMSRNM019E** Identical fileids [RC = 24]
- **DMSRNM024E** File fn ft fm already exists [RC = 28]
- **DMSRNM030E** File fn ft fm already active [RC = 28]
- **DMSRNM037E** Disk mode[(vdev)] is accessed as read/only [RC = 36]
- **DMSRNM048E** Invalid filemode mode [RC = 24]
- **DMSRNM051E** Invalid mode change [RC = 24]
- **DMSRNM054E** Incomplete fileid specified [RC = 24]
- **DMSRNM062E** Invalid * in output in fileid fn ft [RC = 20]
- **DMSRNM069E** Disk mode not accessed [RC = 36]
RESERVE

Use the RESERVE command to allocate all available blocks of a 512-, 1K-, 2K-, or 4K-byte block formatted minidisk to a unique CMS file.

The format of the RESERVE command is:

| RESERVE | { fn ft fm } |

where:

fn
is the filename of the file to which you are allocating the available blocks.

ft
is the filetype of the file to which you are allocating the available blocks.

fm
is the filemode of the file to which you are allocating the available blocks. If you do not specify a filemode number (0-6), then the filemode number defaults to 6.

Format of the File

When the RESERVE command completes, the defined file has the following format:

- filename, filetype, and filemode letter as defined by the user.
- filemode number 6 (indicating “update in place”) if not specified on the RESERVE command.
- logical record length (lrecl) equal to the CMS disk block size.
- fixed (F) record format.
- the number of records is the total number of blocks available on the disk minus the number of blocks used by CMS. You can use the DISKID function to get this number. This CMS overhead varies with the size of the minidisk. The data blocks physically follow the blocks used by CMS.
The file that is created can be read or written via the DASD Block I/O System Service or the CMS file system. Because a CMS file structure has been created on the disk, the file may be accessed using the CMS file system.

**Organization of the Mini-disk**

When the RESERVE command completes, the physical organization of the minidisk on a CKD device is:

<table>
<thead>
<tr>
<th>Start Block</th>
<th>No. of Blocks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>IPL record</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Volume label</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>CMS file directory</td>
</tr>
<tr>
<td>6</td>
<td>*</td>
<td>Pointer blocks for allocation map</td>
</tr>
<tr>
<td>next</td>
<td>*</td>
<td>CMS allocation map</td>
</tr>
<tr>
<td>next</td>
<td>*</td>
<td>Duplicate for CMS allocation map backup</td>
</tr>
<tr>
<td>next</td>
<td>*</td>
<td>File pointer blocks</td>
</tr>
<tr>
<td>next</td>
<td>*</td>
<td>File data blocks</td>
</tr>
</tbody>
</table>

**Example:**

Suppose you have a 3330 device with one cylinder formatted with 1024-byte block size. There will be 209 blocks available. After you issue the RESERVE command, the file created will have the following format:

```
1 2 3 | 4 5 6 | 7 8 | 9 | 10 | 11 | ... | 207 | 208 | 209

Blocks used by CMS file system

{ } { } { data blocks }
```

**where:**

**Physical block Number**

**Description**

1 and 2 Contain the IPL records
3 Contains the volume label
4 or 5 Contain the CMS directory file
6 Contains the allocation map
7 Contains the alternate allocation map
8 Is the CMS level 1 pointer block
9 through 209 Are data blocks 1 through 201

The data blocks of the file created are now organized sequentially (blocks 9-209) after the blocks used by the CMS file system (blocks 1-8). Data block 1 is actually physical block 9, data block 2 is actually physical block 10, ..., and the last block (data block 201) is actually physical block 209.
RESERVE

Usage Notes:

1. The RESERVE command is valid only for an accessed minidisk.

2. The RESERVE command does not rewrite the data blocks of the file being created. The data blocks contain whatever was left in the slot they occupy on disk. To clear these blocks with binary zeros, use the FORMAT command before the RESERVE command is issued.

3. If the disk specified is formatted with the RECOMP option, the RESERVE command ignores this option and assigns all cylinders or blocks to the file.

4. The block in the section above refers to a CMS physical block.

Responses:

DMSRSV603R RESERVE will erase all files on disk mode(vdev). Do you wish to continue? Enter 1 (YES) or 0 (NO).

To reply yes, enter 1 or ‘YES’. To reply no, enter 0 or ‘NO’. If you respond ‘YES’, you must only enter the character string ‘YES’. You have indicated that a disk area is to be initialized; all existing files are erased. If the character string contains leading or trailing blanks, such as ‘YES’ or ‘YES ’, the response is processed as a ‘NO’ response. Responding ‘NO’, pressing the ENTER key, or entering a character string other than ‘YES’ cancels execution of the FORMAT command.

DMSRSV705I Disk remains unchanged

The response to continuing the execution of the RESERVE command was anything but YES.

DMSRSV733I Reserving disk mode

The disk represented by the mode letter mode is reserved. The response to continue the execution of the RESERVE command was YES or 1.
Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSRSV037E  Disk mode(vdev)] is accessed as read/only [RC = 36]
DMSRSV042E  No fileid(s) specified [RC = 24]
DMSRSV054E  Incomplete fileid specified [RC = 24]
DMSRSV069E  Disk mode not accessed [RC = 36]
DMSRSV069E  Output disk mode not accessed [RC = 36]
DMSRSV070E  Invalid parameter parameter [RC = 24]
DMSRSV109T  Virtual storage capacity exceeded
DMSRSV113S  Disk(vdev) not attached [RC = 100]
DMSRSV223E  No filemode specified [RC = 24]
DMSRSV260E  Disk not properly formatted for RESERVE [RC = 16]
DMSRSV908E  File system error detected at virtual address vdev; reason code nn [RC = 100]
DMSRSV909E  Permanent I/O error on vdev; csw = csu, sense = sense [RC = 100]
RESTORE WINDOW

RESTORE WINDOW

Use the RESTORE WINDOW command to return a maximized or minimized window to its size and location prior to the maximize or minimize.

The format of the RESTORE WINDOW command is:

```
RESTore WINdow [wname]
```

**where:**

- **wname** is the name of the window to be restored. An "=" indicates that the topmost window will be restored. "=" is the default.

**Usage Notes:**

None.

**Responses:**

None.

**Messages and Return Codes:**

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSRES386E  Missing operand(s) [RC = 24]
- DMSRES388E  Invalid keyword: keyword [RC = 24]
- DMSRES391E  Unexpected operand(s): operand [RC = 24]
- DMSRES921E  Window wname is not defined [RC = 28]
Use the ROUTE command to direct data of a particular message class to a virtual screen.

The format of the ROUTE command is:

```
ROUTE msgclass TO vname [(options . . .)]
```

Options: 

- ALARM
- NOTify
- NOALARM
- NONotify

where:

msgclass identifies the message class which is being directed to the virtual screen. The following classes of output may be specified:

- CMS directs the responses generated by the virtual machine. The responses include any CMS error messages and line mode I/O performed by the virtual machine.
- CP directs messages and responses generated by CP.
- MESSAGE directs messages sent by the MSG and MSGNOH (message-noheader) commands from other users.
- WARNING directs warning messages sent by the CP WARNING command from the system operator.
- SCIF directs any messages from a secondary userid to a virtual screen.
- NETWORK directs RSCS file routing messages from the RSCS virtual machine. All other RSCS messages are handled as regular messages. See Usage Note 4 for a list of the messages classified as NETWORK.
- * directs all classes of information to a specified virtual screen.

vname specifies which virtual screen receives the output.
ROUTE

Options:

The options apply to the MESSAGE, WARNING, SCIF, and NETWORK message classes. Although you can specify the options for CP and CMS, they are ignored.

ALArm
sounds the alarm when a message is received.

NOAlarm
does not sound the alarm. NOALARM is the default.

NOTify
displays the message class name in the status area when you receive a message with a message class of MESSAGE, WARNING, SCIF, or NETWORK.

NONotify
will not display the virtual screen name in the status area when you receive a message with a message class of MESSAGE, WARNING, SCIF, or NETWORK. NONOTIFY is the default.

Usage Notes:

1. When SET FULLSCREEN is ON, the various message classes are routed to virtual screens as follows:

<table>
<thead>
<tr>
<th>Message Class</th>
<th>Virtual Screen</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMS</td>
<td>CMS</td>
<td>NOALARM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NONOTIFY</td>
</tr>
<tr>
<td>CP</td>
<td>CMS</td>
<td>NOALARM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NONOTIFY</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>MESSAGE</td>
<td>ALARM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOTIFY</td>
</tr>
<tr>
<td>WARNING</td>
<td>WARNING</td>
<td>ALARM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOTIFY</td>
</tr>
<tr>
<td>SCIF</td>
<td>MESSAGE</td>
<td>NOALARM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NONOTIFY</td>
</tr>
<tr>
<td>NETWORK</td>
<td>NETWORK</td>
<td>NOALARM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOTIFY</td>
</tr>
</tbody>
</table>

Figure 27. Default Settings for Message Routing

Commands entered in the CMS virtual screen are always echoed in the CMS virtual screen regardless of the routing of the CMS message class.

2. The CP and CMS message classes always have options of NOALARM and NONOTIFY.

3. When you specify the message class as *, the options are recognized for MESSAGE, WARNING, SCIF, and NETWORK. CP and CMS always have the NOALARM and NONOTIFY options.
4. The following messages from the Remote Spooling Communications Subsystem (RSCS) are classified as NETWORK messages:

- **101I** FILE spoolid ENQUEUED ON LINK linkid
- **104I** FILE (orgid) {SPOOLED|TRANSFERRED} TO userid1 ORG nodeid(userid2) mm/dd/yy hh:mm:ss: zzz
- **147I** SENT FILE spoolid(orgid) ON LINK linkid TO nodeid(userid2)
- **170I** FROM nodeid: message-text

All other RSCS messages are handled as regular messages.

The first three characters of the identifier of an RSCS message are DMT. The next three characters denote the module origin of the message. For more information on messages from RSCS, see *Remote Spooling Communications Subsystem Program Reference and Operations Manual* or *Remote Spooling Communications Subsystem Networking Version 2 Operation and Use.*

**Responses:**

None.

**Messages and Return Codes:**

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSROU386E Missing operand(s) [RC = 24]
- DMSROU388E Invalid keyword: keyword [RC = 24]
- DMSROU391E Unexpected operand(s): operand [RC = 24]
- DMSROU394E Invalid option: option [RC = 24]
- DMSROU921E Virtual screen *vname* is not defined [RC = 28]
- DMSROU926E Command is only valid in CMS FULLSCREEN mode [RC = 88]
- DMSROU928E Command is not valid for virtual screen *vname* [RC = 12]
RSERV

Use the RSERV command in CMS/DOS to copy, display, print, or punch a VSE relocatable module from a private or system library.

The format of the RSERV command is:

```
RSERV modname [ ft ] [(options...[ ])]
```

Options: [DISK] [PRINT] [PUNCH] [TERM]

**where:**

**modname**

specifies the name of the module on the VSE private or system relocatable library. The private library, if any, is searched before the system library.

**ft**

specifies the filetype of the file to be created on your A-disk. “ft” defaults to TEXT if a filetype is not specified. The filename is always the same as the module name.

**Options:**

You may specify as many options as you wish on the RSERV command, depending on which functions you want to perform.

**DISK**

copies the relocatable module onto your A-disk. If no other options are specified, DISK is the default.

**PRINT**

prints the relocatable module on the virtual printer.

**PUNCH**

punches the relocatable module on the virtual punch.

**TERM**

displays the relocatable module at your terminal.
Usage Notes:

1. If you want to copy modules from a private relocatable library, you must issue an ASSGN command for the logical unit SYSRLB and identify the library on a DLBL command line using the ddname IJSYSRL.

   To copy modules from the system relocatable library, you must have entered the CMS/DOS environment specifying a mode letter on the SET DOS ON command line.

2. The RSERV command ignores the assignment of logical units, and directs output to the devices specified on the option list.

Responses:

If you use the TERM option, the relocatable module is displayed at the terminal.

Messages and Return Codes:

DMSRRV003E Invalid option: option [RC = 24]
DMSRRV004E Module module not found [RC = 28]
DMSRRV006E No read/write A disk accessed [RC = 36]
DMSRRV070E Invalid parameter parameter [RC = 24]
DMSRRV089E Open error code nn on SYSRLB [RC = 36]
DMSRRV097E No SYSRES volume active [RC = 36]
DMSRRV098E No module name specified [RC = 24]
DMSRRV099E CMS/DOS environment not active [RC = 40]
DMSRRV105S Error nn writing file fn ft fm on disk [RC = 100]
DMSRRV113S Disk(vdev) not attached [RC = 100]
DMSRRV411S Input error code nn on SYSa aa [RC = rc]
Use the RUN command to initiate a series of functions on a file depending on the filetype. The RUN command can select or combine the procedures required to compile, load, or start execution of the specified file.

The format of the RUN command is:

```
RUN fn [ft [fm]] [(args...][]]
```

where:

- **fn** is the filename of the file to be manipulated.
- **ft** is the filetype of the file to be manipulated. If filetype is not specified, a search is made for a file with the specified filename and the filetype of EXEC, MODULE, or TEXT (the search is performed in that order). If the filetype of an input file for a language processor is specified, the language processor is invoked to compile the source statements and produce a TEXT file. If no compilation errors are found, LOAD and START may then be called to initiate program execution. The valid filetypes and resulting action for this command are:

<table>
<thead>
<tr>
<th>Filetype</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXEC</td>
<td>The EXEC processor is called to process the file.</td>
</tr>
<tr>
<td>MODULE</td>
<td>The LOADMOD command is issued to load the program into storage and the START * command begins execution of the program at the default entry point.</td>
</tr>
<tr>
<td>TEXT</td>
<td>The LOAD command brings the file into storage in an executable format and the START * command executes the program beginning at the default entry point.</td>
</tr>
<tr>
<td>FORTRAN</td>
<td>The FORTRAN processor module that is called is FORTVS, FORTRAN, FORTGI, GOFORT, or FORTHX, whichever is found first. Object text successfully compiled by the FORTGI or FORTHX processors will be loaded and executed.</td>
</tr>
<tr>
<td>TESTFORT</td>
<td>The TESTFORT module is called to initiate FORTRAN Interactive Debug and will process a TEXT file that has been compiled with the TEST option.</td>
</tr>
</tbody>
</table>
FREEFORT  The GOFORT module is called to process the file.

COBOL    The COBOL processor module that is called is COBOL or TESTCOB, whichever is found first. After successful compilation, the program text will be loaded and executed.

PLI PLOPT The PLIOPT processor module is called to process the file. After successful compilation, the program text will be loaded and executed.

\( fm \)
is the filemode of the file to be loaded by the LOADMOD command. If \( fm \) is specified, a filetype must also be specified. The \( fm \) field is only beneficial when attempting to execute a module. All other functions use the default search order to locate a file on your minidisk.

\( args \)
are arguments you want to pass to your program. You can specify up to 13 arguments in the RUN command, provided they fit on a single input line. Each argument is left-justified, and any argument more than eight characters long is truncated from the right.

**Usage Notes:**

1. If you are executing a CMS EXEC or EXEC 2 file, the arguments you enter on the RUN command line are assigned to the variable symbols \&1, \&2, and so on. If you are executing a System Product interpreter program, the arguments that you enter on the RUN command line are available via the arg instruction.

   The RUN command passes only the filename (fn) of an EXEC to the EXEC processor. Therefore, you cannot use “\( fm \)” to select a particular EXEC.

2. Before using the RUN command, you should issue the GLOBAL command to identify the required libraries.

3. If you are executing a TEXT or MODULE file, or compiling and executing a program, the arguments are placed in a parameter list and passed to your program when it executes. The arguments are placed in a series of doublewords in storage, terminated by X'FF'. If you enter:

   \[ \text{run myprog (charlie dog)} \]

   the arguments *, CHARLIE, and DOG are placed in doublewords in a parameter list, and the address of the list is in register 1 when your program receives control.

   **Note:** You cannot use the argument list to override default options for the compilers or for the LOAD or START commands.
RUN

4. The RUN command is not designed for use with CMS/DOS.

5. The RUN command cannot be used for COBOL and PL/I programs that require facilities not supported under CMS. For specific language support limitations, see VM/SP Planning Guide and Reference.

6. If you want to issue RUN from an EXEC program, you should precede it with the EXEC command; that is, specify

    exec run

Responses:

Any responses are from the programs or procedures that executed within the RUN command.

Messages and Return Codes:

- DMSRUN001E  No filename specified [RC = 24]
- DMSRUN002E  File fn ft fm not found [RC = 28]
- DMSRUN048E  Invalid mode fm [RC = 24]
- DMSRUN070E  Invalid parameter parameter [RC = 24]
- DMSRUN999E  No ft module found [RC = 28]
Use the SCROLL command to move a window to a new location on the virtual screen to which it is connected.

The format of the SCROLL command is:

<table>
<thead>
<tr>
<th>SCROLL</th>
<th><strong>BAckward</strong></th>
<th><strong>Bottom</strong></th>
<th><strong>Down</strong></th>
<th><strong>Forward</strong></th>
<th><strong>Left</strong></th>
<th><strong>Next</strong></th>
<th><strong>Right</strong></th>
<th><strong>Top</strong></th>
<th><strong>Up</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[ <strong>wname</strong></td>
<td>= [ [n]</td>
<td>= [ ]</td>
<td>= [n]</td>
<td>= [n]</td>
<td>= [1]</td>
<td>= [n]</td>
<td>= [ ]</td>
<td>= [n]</td>
</tr>
</tbody>
</table>

* **wname**

* SCROLL

**where:**

**wname**

is the name of the window being scrolled. An "=" indicates that the topmost window is scrolled. If this operand is not specified, "=" is assumed.
SCROLL

BAckward
moves the window toward the top of the virtual screen so that the first
data line displayed in the window becomes the last data line displayed.
If you specify a number for \( n \), the window is scrolled backward \( "n" \)
displays. The * means scroll to the top. The default is 1 display.

Bottom
moves the window to the last line (bottom) of the virtual screen. The
first data line of the window displays the last line of the virtual
screen.

Down
moves the window toward the bottom of the virtual screen the number
of data lines specified. The default is 1 line. DOWN is the same as
NEXT. The * means scroll to the bottom.

Forward
moves the window toward the bottom of the virtual screen so that the
last data line displayed in the window becomes the first data line
displayed. If you specify a number for \( n \), the window is scrolled
forward \( "n" \) displays. The default is 1. The * means scroll to the
bottom.

Left
moves the window toward the left edge of the virtual screen the
number of columns specified. The default for \( n \) is 1. If \( n \) is specified
as 0, the window is moved to column 1.

Next
moves the window toward the bottom of the virtual screen the number
of lines specified. The default is 1 line. NEXT is the same as DOWN.
The * means scroll to the bottom.

Right
moves the window toward the right edge of the virtual screen the
number of columns specified. The default for \( n \) is 1. If \( n \) is specified
as 0, the window is moved to column 1.

Top
moves the window to the first line (top) of the virtual screen. The first
data line of the window displays the first line of the virtual screen.

Up
moves the window toward the top of the virtual screen the number of
data lines specified. The default is 1 line. The * means scroll to the
top.
Usage Notes:

1. The SCROLL command is valid only when the window is connected to a virtual screen (see the SHOW WINDOW and HIDE WINDOW commands).

2. If the window has been cleared (see the CLEAR WINDOW command), the following rules apply:
   - If you SCROLL FORWARD or TOP, you scroll to the top of the virtual screen.
   - If you SCROLL BACKWARD or BOTTOM, you scroll to the bottom of the virtual screen.
   - SCROLLing UP or DOWN has no effect.

3. If the window is positioned in the middle of the virtual screen and you scroll backward specifying an "n," which would result in scrolling past the virtual screen top, the window is repositioned at the virtual screen top and stops. If the window is positioned at the virtual screen top and you scroll backward, the window is repositioned at the virtual screen bottom and stops.

4. If the window is positioned in the middle of the virtual screen and you scroll forward specifying an "n," which results in scrolling past the bottom, the window is repositioned at the virtual screen bottom, and stops. If the window is positioned at the virtual screen bottom and you scroll forward, the window is cleared. A subsequent scroll forward positions the window at the virtual screen top. If the window being scrolled is variable size and it is positioned at the bottom of the virtual screen, the window is not displayed at the next refresh when it is scrolled forward (see the CLEAR WINDOW command).

5. When you receive the status area message “Scroll forward for more information in vscreen uname” (in full-screen CMS) and there are multiple windows showing the specified virtual screen, it is recommended that you scroll forward the window closest to the top of the ordered list of windows. This enables data that is in the virtual screen queue to be written to the virtual screen and displayed.

Responses:

None.
SCROLL

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSSCL386E  Missing operand(s) [RC = 24]
DMSSCL388E  Invalid keyword: keyword [RC = 24]
DMSSCL389E  Invalid operand: operand [RC = 24]
DMSSCL391E  Unexpected operand(s): operand [RC = 24]
DMSSCL921E  Window wname is not defined [RC = 28]
DMSSCL929E  Window wname is not connected to a virtual screen [RC = 36]

Return Codes:

0  Normal
1  Vscreen top or bottom reached, or window is cleared
3  Scroll amount truncated
SENDFILE

Use the SENDFILE command to send files or notes to one or more computer users on your computer or on other computers that are connected to yours via the Remote Spooling Communications Subsystem (RSCS) network.

SENDFILE is one of several commands that references a "userid NAMES" file. By setting up a names file, you can identify recipients just by using nicknames, which are automatically converted into node and userid. For information on creating a names file, see the NAMES command.

The format of the SENDFILE command is:

```
SENDFile SFile [fn ft fm] [[ TO] name...] [ (options... ]
```

where:

- **fn**
  - is the filename of the file to be sent.

- **ft**
  - is the filetype of the file to be sent.

- **fm**
  - is the filemode of the file to be sent. If "*" is specified (the default), all accessed disks are searched, and the first file found is sent. This operand can be omitted if the first "name" would not be misinterpreted as a filemode, or if the keyword "TO" is used.

- **TO**
  - is a keyword operand. It can be omitted if the first "name" is not "TO."
SENDFILE

_name_ is one or more “names” of the computer users to whom the file is to be sent. If the same recipient is specified more than once, he receives only one copy of the file. The “name” may take any of the following forms, and the different forms can be freely intermixed:

• a “nickname” that can be found in the file “userid NAMES,” where “userid” is your userid. This nickname may represent a single person (on your computer or on another computer), or a list of several people. See the NAMES command for more information on nicknames.

• a userid of a computer user on your computer. If a name cannot be found in the “userid NAMES” file, it is assumed to be a userid of someone on your computer.

• “userid AT node,” which identifies a user (“userid”) on your computer or another computer (“node”).

You cannot send files to a userid named “AT” or “CC:.”

If no operands are specified, a menu is displayed. This menu is described in the Usage Note below, “Using the SENDFILE Menu.”

Options:

**Ack** requests an acknowledgment be returned to you and logged when the recipient receives your file (using the RECEIVE command). Acknowledgments are added to your “userid NETLOG” file. An acknowledgment is sent only if the NEW option is also in effect.

**NOAck** requests that no acknowledgment be returned when the recipient RECEIVES a file.

**Filelist** specifies that the file (fn ft fm) is a list of files in the format of a CMS EXEC file produced by the LISTFILE command issued with the EXEC option, or a file saved from a FILELIST command. This option is used to send multiple files with only one invocation of SENDFILE. Both the file containing the list of files and each file in the list are sent.

Lines beginning with an asterisk (*) and blank lines are ignored. All EXEC tokens (for example, &1, &2) or any token beginning with an ampersand (&) is ignored.

For information on creating a list of files that can be saved and used to send multiple files, see the FILELIST command, the usage note “Saving a List of Files.”
NOFilelist
specifies that the file is not a list of files.

Log
specifies that the recipients, date, and time of this file transmission are logged in a file called “userid NETLOG.” This log is updated when acknowledgments of sent files are received (if they were requested). Do not use this option if you have no read/write disk accessed.

NOLog
specifies that this file transmission is not to be logged.

NEW
specifies that header records are added and the file is sent as described below, in “Format of the File Sent by SENDFILE.” If this option is specified, the recipient must use RECEIVE to read the file.

Old
specifies that the file is sent using DISK DUMP. This option should be specified when the recipient of the file does not have the RECEIVE command available to read the file. When OLD is specified, no acknowledgment (the ACK option) can be requested.

NOTE
specifies that the file is to be sent as a note (that was prepared using the NOTE command). The “TO” operand and the list of names cannot be specified if this option is given. If no file is specified, the file “userid NOTE *” is sent as a note. (On a display terminal, the PF5 key is set to this option in the NOTE command environment.)

Type
specifies that the files sent and the userids and nodes to which the files were sent are displayed at the terminal.

NOType
specifies that no information is to be displayed.

Usage Notes:

1. Tailoring the SENDFILE Command Options
You can use the DEFAULTS command to set up options and/or override command defaults for SENDFILE. However, the options you specify in the command line when entering the SENDFILE command override those specified in the DEFAULTS command. This allows you to customize the defaults of the SENDFILE command, yet override them when you desire. Refer to the DEFAULTS command description for more information.

2. Using the SENDFILE Menu (Display Terminals Only)
Enter the SENDFILE command without operands to display a menu, on which you “fill in the blanks” with the necessary information. A sample SENDFILE menu is shown in the “Examples,” below.

**The File Identifier:**

You type the filename, filetype, and filemode of a file that you want to send directly on the menu in the spaces provided. If you do not enter a filemode, the default is “A.”

If you want to select the files from a list, you can type an asterisk (*) for filename, filetype, and/or filemode. An asterisk means that you want the list to contain all filenames (or filetypes, or filemodes).

You can also use two special characters in the filename and/or filetype to request that the list contain a specific subset of files. The special characters are * (asterisk) and % (percent), where:

- * represents any number of character(s). As many asterisks as required can appear anywhere in a filename or filetype.
- % means any single character, but any character will do. As many percent symbols as necessary may appear anywhere in a filename or filetype.

To display the list, first finish filling out the menu, and then press PF5. A special FILELIST screen is displayed instead of the SENDFILE menu. You select the files by typing a letter “s” in front of the filename of each file to be sent. Then press the ENTER key to send the files.

Another way to select files to be sent from the FILELIST screen is to position the cursor on the line describing a file you want to send, and then press PF5.

**The Recipient(s):**

You type the name(s) of the recipient(s) in the space provided. A name can take any of the forms listed above, in the “name” operand description.

**The Options:**

A list of options also appears on the menu. The default for each option appears to its left. You type 1 for YES or 0 for NO over any options for which you do not want the default. The options are as follows:

- 0 Request acknowledgment when the file has been received?

  Type 1 for YES only if you want to get an acknowledgment when the person receives your file. The acknowledgment shows the date and time the file was received, and the recipient’s userid and node.
When you get an acknowledgment, it appears in your reader. If you choose to receive it, an entry is made in a “userid NETLOG” file, which is explained below.

1 Make a log entry when the file has been sent?

Each time you send a file, an entry is automatically made in the file “userid NETLOG.” A typical entry might look like this:

File MY DATA A1 sent to JONES at NODE1 on 10/10/81 11:30:25 EDT

If you specified 1 on the first option (acknowledgment), an entry is also made when you receive the acknowledgment.

Type 0 if you don’t want an entry made in the log file.

1 Display the file name when the file has been sent?

The names of the file(s) and the userid(s) and node(s) of the recipients are displayed on a cleared screen. Type 0 if you do not want this information displayed.

0 This file is actually a list of files to be sent?

See the FILELIST command, the usage note “Saving a List of Files,” for information on saving a file list. By saving a list of files created by either the FILELIST command or the LISTFILE command issued with the EXEC option, you can send all the files (and the list of files) at once. Type 1 if your file is a list of files.

Sending a File:

If you specified only one fileid, press either PF5 or the ENTER key after filling out the SENDFILE menu. PF5 sends the file and exits from the menu. Pressing the ENTER key sends the file but keeps the menu.

If you are selecting files from a FILELIST screen type a letter “s” in front of each filename you want to send. Then press the ENTER key to send the file(s).

Keys on the SENDFILE Menu:

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTER</td>
<td>Execute the command typed on the command line, or if none, send the file. (The ENTER key is set by the XEDIT subcommand, SET ENTER IGNORE MACRO EXECUTE.)</td>
</tr>
<tr>
<td>PF 1</td>
<td>Help</td>
</tr>
<tr>
<td>PF 2</td>
<td>Not assigned.</td>
</tr>
<tr>
<td>PF 3</td>
<td>Quit</td>
</tr>
<tr>
<td>PF 4</td>
<td>Not assigned</td>
</tr>
<tr>
<td>PF 5</td>
<td>Send</td>
</tr>
<tr>
<td>PF 6</td>
<td>Not assigned</td>
</tr>
</tbody>
</table>
SENDFILE

PF 7   Not assigned
PF 8   Not assigned
PF 9   Not assigned
PF 10  Not assigned
PF 11  Not assigned
PF 12  Cursor If cursor is on the menu, move it to the command line; if cursor is on the command line, move it back to its previous location on the menu.

Note: On a terminal equipped with 24 PF keys, PF keys 13 to 24 are assigned the same values as PF keys 1 to 12 as discussed here.

On a display terminal without PF keys, you can enter QUIT from the command line to exit from the screen.

Pressing the PA1 key while in the SENDFILE menu displays the WM window, unless the CP TERMINAL BRKKEY has been assigned to PA1.

Keys on the FILELIST Screen:

ENTER Execute the command(s) typed on file line(s), or on the command line. (The ENTER key is set by the XEDIT subcommand, SET ENTER IGNORE MACRO EXECUTE.)
PF 1 Help Display information about the FILELIST command.
PF 2 Refresh Update the list to indicate new files, erased files, etc., using the same parameters as those specified on the SENDFILE panel.
PF 3 Quit Exit from the list.
PF 4 Sort files by filetype, filename.
PF 5 Sendfile at cursor. Append the fn ft fm on this line and send the file.
PF 6 Sort files by size, largest first.
PF 7 Forward Scroll forward one screen.
PF 8 Backward Scroll backward one screen.
PF 9 FL/n Issue the command FILELIST /n * * at the cursor, so that a list is displayed, containing all files that have the filename that is displayed on the line with the cursor.
PF 10 Not assigned.
PF 11 XEDIT Edit the file pointed to by the cursor.
PF 12 Cursor If cursor is in the file area, move it to the command line; if cursor is on the command line, move it back to its previous location in the file.

In addition to setting the above PF keys, the PROFSEND XEDIT macro sets synonyms that you can use to sort your FILELIST files. Enter the synonyms on the SENDFILE command line. The synonyms are:

520 VM/SP CMS Command Reference
SENDFILE

SNAME Sorts the list alphabetically by filename, filetype, and filemode.

STYPE Sorts the list alphabetically by filetype, filename, and filemode.

SMODE Sorts the list by filemode, filename, and filetype.

SRECF Sorts the list by record format, filename, filetype, and filemode.

SLREC Sorts the list by logical record length and then by size (greatest to least).

SSIZE Sorts the list by number of blocks and number of records (greatest to least).

SDATE Sorts the list by year, month, day, and time (most recent to oldest).

An example of a SENDFILE menu and a FILELIST screen are shown in the “Examples” section, below.

3. Format of the File Sent by SENDFILE

The format of the file that is sent depends on whether the OLD or NEW (the default) option is specified.

Important note: Unless the OLD option is specified, the RECEIVE command is the only way you can read a file sent by SENDFILE.

The OLD Option:

If the OLD and NOTE options are specified and the width (LRECL) of the note (prepared using the NOTE command) is 80 or less, SENDFILE uses the PUNCH command (with the HEADER option) to send the file. Otherwise, DISK DUMP is used to send the file. The OLD option should be used if the recipient does not have the RECEIVE command available to read the file.

The NEW Option:

If the NEW option is specified, control records are added and the file is sent in a format called “NETDATA.”

The transmitted file is composed of several control records, followed by the data records, and ending with a trailer record. If the file is an acknowledgment, it consists only of control records. An acknowledgment can be requested only with the NEW option.

The NEW option should be used when the recipient can read the file with the RECEIVE command on his CMS system, or when the file is
being sent to the MVS operating system with TSO Extensions program product.

4. Priority

When SENDFILE is issued with the NEW option to send a file across the network (to a node different from yours), the file is assigned a priority. The order and speed of transmission are based on both this priority and the size of the file.

The priorities are assigned as follows:

NOTE files at least ten blocks in size: Priority = 00 (high)
Other files: Priority = 50 (medium)
Acknowledgments: Priority = 90 (low)

5. SENDFILE always sends files as CLASS A NOCONT NOHOLD regardless of the class to which you spool your PUNCH. The CP message that is generated, containing the spoolid, etc., is suppressed.

6. If you want to issue SENDFILE from an EXEC program, you should precede it with the EXEC command; that is, specify

exec sendfile

Responses:

Body of the note kept in 'fn NOTEBOOK fm'
Header only added to other NOTEBOOK files.
File 'fn ft fm' sent to 'userid' at 'node' on 'date time timezone'
nnn files have been sent.
File fn ft fm not found.
Note added to fn NOTEBOOK fm

The following message appears on the FILELIST screen invoked from a SENDFILE menu:
Type 'S' in front of each file to be sent, and press ENTER.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSWSF002E  File fn ft fm not found [RC = 28]
DMSWSF006E  No read/write disk accessed [RC = 36]
DMSWSF048E  Invalid mode mode [RC = 24]
DMSWSF054E  Incomplete fileid specified [RC = 24]
Messages from the SENDFILE Panel:

DMSWSF002E  File * fn ft fm not found [RC = 28]
DMSWSF048E  Invalid mode mode [RC = 24]
DMSWSF054E  Incomplete fileid specified [RC = 24]
DMSWSF069E  Disk mode not accessed [RC = 36]
DMSWSF081E  Invalid reply; answer 1 for YES and 0 for NO
DMSWSF637E  Missing nodeid for the AT operand [RC = 24]
DMSWSF647E  Userid not specified for nickname in userid NAMES file [RC = 32]
DMSWSF648E  Userid name not found; no files have been sent [RC = 32]
DMSWSF667E  NOTE header does not contain the {keyword From:|keyword To:|OPTIONS line|DATE line} [RC = 32]
DMSWSF671E  Error sending file fn ft fm; rc = nn from command [RC = 100]
DMSWSF672E  Virtual punch invalid or not defined [RC = 36]
DMSWSF673E  Addressees are in the note header cards; do not specify names with NOTE option [RC = 24]
DMSWSF674E  Punch is not ready [RC = 36]
DMSWSF675E  No names specified [RC = 24]
DMSWSF676E  Invalid character * for Network ID [RC = 20]
DMSWSF677E  Invalid option: option in option line [RC = 32]
DMSWSF678E  Invalid note header format; note cannot be sent [RC = 32]
DMSWSF679E  Disk mode is not accessed; note cannot be sent [RC = 36]
DMSWSF680E  Invalid fileid specified with FILELIST option [RC = 20]
Examples:

The following is a sample SENDFILE menu:

```
----------- SENDFILE -----------
File(s) to be sent   (use * for Filename, Filetype and/or Filemode
to select from a list of files)
Enter filename : *
   filetype : data
   filemode : a
Send files to : sleepy
Type over 1 for YES or 0 for NO to change the options:
  0  Request acknowledgement when the file has been received?
  1  Make a log entry when the file has been sent?
  1  Display the file name when the file has been sent?
  0  This file is actually a list of files to be sent?
1= Help    3= Quit    5= Send   12= Cursor
====>
Macro-read 1 File
```

Figure 28. Sample SENDFILE Menu

In Figure 28, the sender typed an asterisk for filename, “data” for filetype, and “a” for filemode. The name of the recipient (sleepy) is also typed on the screen. When PF5 is pressed, a special FILELIST screen is displayed. The files to be sent can be selected from this screen (shown in Figure 29).
Figure 29. Sample FILELIST Screen Invoked from SENDFILE

To send one or more of these files, you can type a letter "s" in front of the filename of each file you want sent (see above) and then press the ENTER key. You can also position the cursor on the line describing the file you want to send, and then press the PF5 key.
SENTRIES

Use the SENTRYES command to determine the number of lines currently in the program stack. When you issue a SENTRYES command, CMS returns the number of lines in the program stack (but not the terminal input buffer) as a return code.

The format of the SENTRYES command is:

SENTRIES

Usage Note:

If you issue a SENTRYES command in an EXEC that has set up a procedure to be done when an error occurs, a nonzero SENTRYES return code causes that procedure to execute.

Responses:

None

Messages and Return Codes:

None
SET

Use the SET command to establish, turn off, or reset a particular function in your CMS virtual machine. Only one function may be specified per SET command. SET cannot be issued without an option.

The options available with SET are summarized below. A complete description of each option follows.

<table>
<thead>
<tr>
<th>ABBREV</th>
<th>DOSPART</th>
<th>LDRTBLS</th>
<th>RELPAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>APL</td>
<td>EXECTRAC</td>
<td>LINEND</td>
<td>REMOTE</td>
</tr>
<tr>
<td>AUTOREAD</td>
<td>FULLREAD</td>
<td>LOCATION</td>
<td>RESERVED</td>
</tr>
<tr>
<td>BLIP</td>
<td>FULLSCREEN</td>
<td>LOGFILE</td>
<td>SYSNAME</td>
</tr>
<tr>
<td>BORDER</td>
<td>IMESCAPE</td>
<td>NONDISP</td>
<td>TEXT</td>
</tr>
<tr>
<td>CHARMODE</td>
<td>IMPCP</td>
<td>NONSHARE</td>
<td>TRANSLATE</td>
</tr>
<tr>
<td>CMSPF</td>
<td>IMPF</td>
<td>OUTPUT</td>
<td>UPSI</td>
</tr>
<tr>
<td>CMSTYPE</td>
<td>INPUT</td>
<td>PROTECT</td>
<td>VSCREEN</td>
</tr>
<tr>
<td>DOS</td>
<td>INSTSEG</td>
<td>RDYMSG</td>
<td>WINDOW</td>
</tr>
<tr>
<td>DOSLNCNT</td>
<td>LANGUAGE</td>
<td>REDTYPE</td>
<td>WMPF</td>
</tr>
</tbody>
</table>

Usage Notes:

1. If you issue the SET command specifying an invalid function and the implied CP function is in effect, you may receive message DMKCFC003E Invalid option - option

2. If an invalid SET command function is specified from an EXEC and the implied CP function is in effect, then the return code is -0003.

3. To determine or verify the setting of most functions, use the QUERY command.

Responses:

None.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.
SET ABBREV

Use the ABBREV option to control whether the system accepts system and user abbreviations for command names and their translations, or only full command names or the full synonym or translation.

The format of the SET ABBREV command is:

\[
\begin{array}{|c|c|}
\hline
\text{SET} & \text{ABBREV} \{ \text{ON} \} \{ \text{OFF} \} \\
\hline
\end{array}
\]

*where:*

**ON**
accepts system and user abbreviations for command names and their translations. The SYNONYM command makes synonym abbreviations available. The SET TRANSLAT command makes translation abbreviations available.

For example, if GETDISK is a synonym for ACCESS and the minimum abbreviation is three, then you can enter GET, GETD, GETDL, GETDIS, or GETDISK to issue the ACCESS command. The same is true if GETDISK is a translation of ACCESS.

**OFF**
accepts only the full command name or the full synonym or translation (if one is available) for the command name.

For a discussion of the relationship of the SET ABBREV and SYNONYM commands, refer to the SYNONYM command description.

*Initial Setting:*
ABBREV ON

*Responses:*
None.
Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSET109S   Virtual storage capacity exceeded [RC = 104]
SET APL

Use the APL option to activate character code conversion for APL characters for the System Product editor and CMS.

The format of the SET APL command is:

<table>
<thead>
<tr>
<th>SET</th>
<th>APL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[ON]</td>
</tr>
<tr>
<td></td>
<td>[OFF]</td>
</tr>
</tbody>
</table>

*where:*

**ON**
activates character code conversion for APL characters. Before using APL keys, issue SET APL ON to ensure proper character code conversion.

**OFF**
specifies that no character code conversion is performed for APL characters and keys.

*Initial Setting:*

APL OFF

*Usage Notes:*

1. The APL setting is valid only when performing full-screen I/O (for example, in XEDIT or in CMS with SET FULLSCREEN ON). If you are in CP or using a line-mode terminal, SET APL has no effect.

   If you are in CP, you can issue the TERMINAL APL ON command to have CP convert APL character codes.

2. Because the APL character code conversion is costly, it is recommended that you issue SET APL OFF when you stop using the special APL keys.

3. When SET APL ON is specified, TEXT is set OFF.

4. Changing the APL setting for CMS also changes the APL setting for the System Product editor, and vice versa.
Responses:

None.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

- DMSSEF109S Virtual storage capacity exceeded [RC = 104]
- DMSSEF524W NONDISP character reset to "."
- DMSSEF926E Command is only valid on a display terminal [RC = 88]
- DMSSET109S Virtual storage capacity exceeded [RC = 104]
- DMSWIR329W Warning: APL/TEXT option not in effect
SET AUTOREAD

Use the AUTOREAD option to specify whether or not a console read is to be issued after command execution.

The format of the SET AUTOREAD command is:

```
SET AUTOREAD [ON \nOFF]
```

**where:**

ON

specifies that a console read is to be issued immediately after command execution. ON is the initial setting for non-display, non-buffered terminals.

OFF

specifies that you do not want a console read to be issued until you press the Enter key or its equivalent. OFF is the initial setting for display terminals because the display terminal does not lock, even when there is no READ active for it.

*Note:* If you disconnect from one type of terminal and reconnect on another type, the AUTOREAD status remains unchanged.

**Initial Setting:**

AUTOREAD ON for non-display, non-buffered terminals.

AUTOREAD OFF for display terminals.

**Responses:**

None.
Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSET109S Virtual storage capacity exceeded [RC = 104]
SET BLIP

Use the BLIP option to define and control the BLIP character. BLIP characters are displayed at the terminal to indicate every two seconds of virtual interval timer time.

The format of the SET BLIP command is:

<table>
<thead>
<tr>
<th>SET</th>
<th>BLIP {string [(count)]}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ON</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
</tr>
</tbody>
</table>

where:

string[(count)]

defines the characters that are displayed at the terminal to indicate every two seconds of virtual interval timer time. This time is made up of virtual processor time plus, if the REALTIMER option is in effect, self-imposed wait time. Blips may also be caused by the execution of the STIMER macro.

You can define up to eight characters as a blip string; if you want trailing blanks, you must specify count. ON and OFF must not be used as BLIP characters.

ON

sets the BLIP character string to its default, which is a string of nonprintable characters. ON is the initial setting for typewriter devices. The default BLIP character provides no visual or audio-visual signal on a 3767 terminal. You must define a BLIP character for a 3767 if you want the BLIP function.

OFF

turns off BLIP. OFF is the initial setting for graphics devices.

Note: The BLIP operand will be ignored when issued from the CMS batch machine.

Initial Setting:

BLIP ON for typewriter devices.

BLIP OFF for graphics devices.
Responses:

None.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSET109S Virtual storage capacity exceeded [RC = 104]
SET BORDER

Use the BORDER option to define borders around windows. Borders visually separate information displayed in different windows. Also, border corners can be used to enter Border Commands (see the Border Commands section in this book for more information).

The format of the SET BORDER command is:

```
SET BORDER wname [ON | OFF] [([optionA] [optionB] [)])]
```

where:

- **wname**
  - is the name of the window.

- **ON**
  - indicates that the borders of the window will be displayed (if they fit on the physical screen). If no options are specified, the currently defined border characters and attributes are used.

- **OFF**
  - indicates that the borders will not be displayed.

**OptionA:**

allows you to specify new characters for window borders. One or more of the following options may be specified:

- **TOP char**
  - where *char* is the character that is displayed in the top border.

- **BOTTOM char**
  - where *char* is the character that is displayed in the bottom border.

- **LEFT char**
  - where *char* is the character that is displayed in the left-hand border.
**SET BORDER**

**Option B:**

indicates how the borders should be displayed. The following can be specified:

- **High**
  - borders are high intensity.

- **NOHigh**
  - borders are normal intensity.

- **color**
  - is the border color. It may be Default, Blue, Red, Pink, Green, Turquoise, Yellow, or White.

- **exthi**
  - is the border extended highlighting. It may be None, REVvideo, BLInk, or Underline.

- **psset**
  - is the Programmed Symbol Set (PSset) used to display the borders (PS0, PS1, PSA, PSB, PSC, PSD, PSE, or PSF). The PSset must be loaded in the display to be used. If not, the default PSset is used.

**Usage Notes:**

1. To override the default border characteristics (see DEFINE WINDOW for the defaults) or to display a particular edge of the border you must use option A. Only those edges specified in option A are set “ON.” All other edges are set “OFF.”

   For example, if you issue the command:

   ```
   set border message on (top *
   ```

   only the top border is displayed and it is all asterisks (*). The bottom, left, and right borders are not displayed.

   The settings specified remain in effect for the duration of the session or until you change them.
2. Window borders are built outside the area defined for the window. Therefore, due to the size and position of the window, it is possible that any or all of the borders may not fit on the physical screen.

   Note: Left and right border characters take up two columns on the physical screen. (One column is for a start field and another column is for the border character.) Top and bottom borders take only one line. Thus, on a 24 x 80 physical screen, a window must not start before column 3 and must not extend past column 78 for the left and right borders to be displayed. To display top and bottom borders, the window must not start before line 2 or extend past line 23.

3. The corner characters of the border (identified by plus (+) signs) are available for entering single character windowing or scrolling commands (see the Border Commands section of this book). All other border characters are protected.

4. Location information for the number of lines or columns, or both, is displayed using the color, highlight, and program symbol set defined for the window border.

Responses:

None.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSEF921E Window wname is not defined [RC=28]
DMSSET109S Virtual storage capacity exceeded [RC=104]
SET CHARMODE

Use the CHARMODE option to specify whether character attributes or field attributes should be used when displaying virtual screen data on the physical screen.

The format of the SET CHARMODE command is:

| SET | CHARMODE | { ON } | { OFF } |

*where:*

**ON**
- displays each character with its own attribute. Each displayed character can be given individual color, extended highlighting, and PSset.

**OFF**
- displays each character in a field with the attributes of the field. In this case, individual character attributes are ignored.

*Initial Setting:*

CHARMODE OFF

*Usage Notes:*

1. The structure of virtual screens allows you to give different attributes to each character. For example, adjacent characters can be displayed with different colors. See the WRITE VSCREEN command for a description of how to specify character attributes when writing data.

2. To use character attributes, the display device must support character attributes. If the device does not support them (for example, the terminal is a 3277), then field attributes are used regardless of the CHARMODE setting. For more information, refer to the *IBM 3270 Information Display System Data Stream Programmer's Reference*, GA23-0059.

3. SET CHARMODE must be ON to update COLOR, EXTHI, and PSS in the WRITE VSCREEN command. If SET CHARMODE is OFF they are ignored. Switching from SET CHARMODE ON to OFF may produce some undesirable results, such as a field having attributes that you intended only for a character.
4. For color and extended highlighting in a DBCS string, the first byte of a double-byte character determines the attributes for both bytes. You cannot specify character attributes for PSS in the WRITE VSCREEN command within a DBCS string.

Responses:

None.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

- DMSSEF109S  Virtual storage capacity exceeded [RC = 104]
- DMSSEF926E  Command is only valid on a display terminal  [RC = 88]
- DMSSET109S  Virtual storage capacity exceeded [RC = 104]
**SET CMSPF**

Use the CMSPF option to set a CMSPF key to execute a specified command when the PF key is pressed when SET FULLSCREEN is ON.

The format of the SET CMSPF command is:

```
SET CMSPF nn [pseudonym [NOWRITE] [keyword. DELAYED] string]
```

*where:*

*nn*

is a number from 1 to 24 indicating which PF key is being set.

*pseudonym*

is a 9-character representation of the PF key definition. The pseudonym is displayed in the PF key definition area at the bottom of the CMS window. The pseudonym may be up to nine characters in length.

**NOWRITE**

suppresses overwriting of the PF key pseudonym when you set a CMSPF key.

**keyword**

indicates when the command associated with the PF key is executed in relation to other commands entered at the terminal. The *keyword* may be any of the following:

**DELAYED**

delays the execution of the command string. When the key is pressed, the command is displayed in the input area and is not executed until you press the ENTER key. If anything is currently in the input area, it is overlaid and no commands entered on the physical screen are processed. DELAYED is the default setting if no keyword is specified on the SET CMSPF command.

**ECHO**

executes the command immediately when the program function key is pressed. The key definition is echoed on the CMS virtual screen.
SET CMSPF

NOECHO
executes the command immediately when the program function key is pressed. The key definition is not echoed on the CMS virtual screen.

Note: When a CMSPF key is set to RETRIEVE the keyword is ignored.

string
is the command(s) to be executed when the key is pressed.

Initial Settings:

<table>
<thead>
<tr>
<th>CMSPF</th>
<th>Command</th>
<th>ECHO</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Help</td>
<td>ECHO</td>
<td>HELP</td>
</tr>
<tr>
<td>02</td>
<td>Pop_Msg</td>
<td>NOECHO</td>
<td>POP WINDOW MESSAGE *</td>
</tr>
<tr>
<td>03</td>
<td>Quit</td>
<td>NOECHO</td>
<td>SET FULLSCREEN SUSPEND</td>
</tr>
<tr>
<td>04</td>
<td>Clear_Top</td>
<td>NOECHO</td>
<td>#WM CLEAR WINDOW =</td>
</tr>
<tr>
<td>05</td>
<td>Filelist</td>
<td>ECHO</td>
<td>EXEC FILELIST</td>
</tr>
<tr>
<td>06</td>
<td>Retrieve</td>
<td>NOECHO</td>
<td>RETRIEVE</td>
</tr>
<tr>
<td>07</td>
<td>Backward</td>
<td>NOECHO</td>
<td>#WM SCROLL BACKWARD CMS 1</td>
</tr>
<tr>
<td>08</td>
<td>Forward</td>
<td>NOECHO</td>
<td>#WM SCROLL FORWARD CMS 1</td>
</tr>
<tr>
<td>09</td>
<td>Rdrlist</td>
<td>ECHO</td>
<td>EXEC RDRLIST</td>
</tr>
<tr>
<td>10</td>
<td>Left</td>
<td>NOECHO</td>
<td>#WM SCROLL LEFT CMS 10</td>
</tr>
<tr>
<td>11</td>
<td>Right</td>
<td>NOECHO</td>
<td>#WM SCROLL RIGHT CMS 10</td>
</tr>
<tr>
<td>12</td>
<td>Cmdline</td>
<td>NOECHO</td>
<td>CURSOR VSCREEN CMS -2 8 (RESERVED)</td>
</tr>
</tbody>
</table>

Note: On a terminal equipped with 24 PF keys, PF keys 13 through 24 have the same values as PF keys 1 through 12, respectively.

Usage Notes:

1. You can assign a sequence of commands to a single PF key by:
   a. Setting off the LINEND character
   b. Setting the PF key to the commands separated by the LINEND character
   c. Setting the LINEND character ON before using the PF key

   You cannot assign a sequence of #WM commands to a CMSPF key. The SET CMSPF command will accept the sequence, but the commands will not execute.

2. To cancel a PF key definition, enter:
   
   set cmspf nn
   
   substituting the number of the PF key for nn.

3. When you press a PA key or a CMSPF key in the CMS window, any input on the screen that has not been processed is discarded except input that is typed on the command line. If the key that was pressed
does not update the command line, then input on the command line is rewritten. The next time you press ENTER it is executed.

4. The RETRIEVE function saves previously entered commands in a buffer that is 256 characters long. When you enter full-screen CMS, the buffer contains an asterisk (comment), and commands are added to the buffer as they are entered until it is full. As you continue to enter commands, the oldest commands are deleted and the most current commands are added.

Pressing the PF key assigned to RETRIEVE displays the next command in the buffer on the command line. Each time you press the key, the previously entered command is displayed until the oldest one is reached. Then, RETRIEVE returns the most current command. Once the command is on the command line, press ENTER to execute it. You may also modify the command, then press ENTER to execute the new command.

5. The NOWRITE option is particularly useful when you have changed the bottom reserved area in the CMS virtual screen and you do not want the area overwritten when you set a CMSPF key.

Responses:

None.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

- DMSSEF525E Invalid PFkey number [RC = 24]
- DMSSEF926E Command is only valid in CMS FULLSCREEN mode [RC = 88]
- DMSSET109S Virtual storage capacity exceeded [RC = 104]


SET CMSTYPE

Use the CMSTYPE option to suppress or resume CMS terminal display within an EXEC.

The format of the SET CMSTYPE command is:

<table>
<thead>
<tr>
<th>SET</th>
<th>CMSTYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{ HT    }</td>
</tr>
<tr>
<td></td>
<td>{ RT    }</td>
</tr>
</tbody>
</table>

*where:*

HT
suppresses CMS terminal display within an EXEC. All CMS terminal display from an EXEC, except for CMS error messages with a suffix letter of 'S' or 'T', is suppressed until the end of the EXEC file or until a SET CMSTYPE RT command is executed. Some CMS commands may reset CMSTYPE to RT. In general, those commands that interact with the user through the console (i.e. HELP, XEDIT, or any command or module that issues a READ to the console or the &READ EXEC control word) may reset CMSTYPE.

RT
resumes CMS terminal display which has been suppressed as a result of a previous SET CMSTYPE HT command.

*Initial Setting:*

CMSTYPE RT

*Usage Notes:*

1. &STACK HT and SET CMSTYPE HT have the same effect when interpreted by the CMS EXEC processor. Similarly, &STACK RT and SET CMSTYPE RT are equivalent for the CMS EXEC processor. However, when using EXEC 2, the commands &STACK HT and &STACK RT cause the characters “HT” and “RT” to be placed in the program stack and do not affect the console output. These characters must be used by a program or cleared from the stack. Otherwise, you will receive an “UNKNOWN CP/CMS COMMAND” error message when they are read from the program stack.
2. In fullscreen-CMS, SET CMSTYPE HT purges nonpriority output that is in the queue for the virtual screen to which message class CMS is routed.

Responses:

None.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.
SET DOS

Use the DOS option to place your virtual machine in the CMS/DOS environment or return it to the normal CMS environment. In addition, you can specify:

- The mode letter at which the VSE system residence is accessed.
- That you are going to use the AMSERV command or you are going to execute programs to access VSAM data sets.

The format of the SET DOS command is:

```
SET DOS {ON [mode [(VSAM [ ) ])]} OFF
```

where:

**ON**

places your CMS virtual machine in the CMS/DOS environment. The logical unit SYSLOG is assigned to your terminal.

**mode**

specifies the mode letter at which the VSE system residence is accessed; the logical assignment of SYSRES is made for the indicated mode letter.

**VSAM**

specifies that you are going to use the AMSERV command or you are going to execute programs to access VSAM data sets.

**OFF**

returns your virtual machine to the normal CMS environment. All previously assigned system and programmer logical units are unassigned.

**Initial Setting:**

DOS OFF
Responses:

None.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

- DMSSET048E Invalid mode mode [RC = 24]
- DMSSET109S Virtual storage capacity exceeded [RC = 104]
- DMSSET400S System sysname does not exist [RC = 44]
- DMSSET401S VM size (size) cannot exceed sysname start address (vstor) [RC = 104]
- DMSSET402W DMSLBR no in CMSBAM segment; ESERY support not available.
- DMSSET410S Control program error indication xxx [RC = rc]
- DMSSET444E Volume valid is not a DOS SYSRES [RC = 32]

Note: In RC = rc, the rc represents the actual error code generated by CP.
SET DOSLNCNT

Use the DOSLNCNT option to specify the number of SYSLST lines per page for the CMS/DOS environment.

The format of the SET DOSLNCNT command is:

| SET | DOSLNCNT nn |

where:

DOSLNCNT nn specifies the number of SYSLST lines per page. nn is an integer from 30 to 99.

Initial Setting:

056

Responses:

None.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSET099E CMS/DOS environment not active [RC = 40]
DMSSET109S Virtual storage capacity exceeded [RC = 104]
SET DOSPART

Use the DOSPART option to specify the size of the partition in which you want a program to execute in the CMS/DOS environment.

The format of the SET DOSPART command is:

```
SET DOSPART { nnnnnK }
```

where:

`nnnnnK`

specifies the size of the virtual partition in which you want a program to execute. The value, `nnnnnK`, may not exceed the amount of user free storage available in your virtual machine.

You should use this function only when you can control the performance of a particular program by reducing the amount of available virtual storage.

*Note:* In rare circumstances, it may happen that when a program is executed, the amount of storage available is less than the current DOSPART. Then, only the amount of storage available is obtained; no message is issued.

`OFF`

specifies that you no longer want to control your virtual machine partition size. When the DOSPART setting is OFF, CMS computes the partition size whenever a program is executed.

*Initial Setting:*

DOSPART OFF

*Responses:*

None.
SET DOSPART

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSET099E  CMS/DOS environment not active [RC = 40]
DMSSET109S  Virtual storage capacity exceeded [RC = 104]
DMSSET333E  nnnnnK partition too large for this virtual machine [RC = 24]
SET EXECTRAC

Use the EXECTRAC option set tracing on or off for your System Product Interpreter or EXEC 2 program.

The format of the SET EXECTRAC command is:

<table>
<thead>
<tr>
<th>SET</th>
<th>EXECTRAC { ON }</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{ OFF }</td>
</tr>
</tbody>
</table>

*where:*

**ON**

specifies that you want tracing turned on for your System Product Interpreter or EXEC 2 program. The tracing bit in the EXECFLAG in NUCON is turned on. The tracing bit is cleared upon return to CMS or XEDIT, turning the tracing off.

**OFF**

specifies that you want tracing turned off for your System Product Interpreter or EXEC 2 program.

*Initial Setting:*

EXECTRAC OFF

*Responses:*

None.

*Messages and Return Codes:*

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSET109S Virtual storage capacity exceeded [RC = 104]
SET FULLREAD

Use the FULLREAD option to allow 3270 null characters to be recognized in the middle of the physical screen by CMS and the System Product editor.

The format of the SET FULLREAD command is:

<table>
<thead>
<tr>
<th>SET</th>
<th>FULLREAD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[ON]</td>
</tr>
<tr>
<td></td>
<td>[OFF]</td>
</tr>
</tbody>
</table>

**where:**

**ON**

enables nulls to be recognized in the middle of lines, making it easier for you to enter tabular or pictorial data.

**OFF**

inhibits transmission of nulls from the terminal.

**Initial Setting:**

FULLREAD OFF

**Usage Notes:**

1. When FULLREAD ON is issued, nulls at the end of screen lines that are part of a logical line that occupies more than one physical screen line are dropped. This allows you to delete characters in a screen line and still have the line reconstructed flush together even though multi-line 327X lines do not “wrap” when the character delete key (or the insert mode key) is used.

2. FULLREAD ON increases communication to the CPU, which generally results in increased response time.

3. Setting FULLREAD ON will prevent you from losing any screen changes when you press a PA key and a message is displayed on a cleared screen.

4. A certain terminal configuration, which imposes several restrictions on your session, occurs when going through a VM/Passthru Facility (5749-RC1) (PVM) 327X Emulator link to another VM system. These PVM links can be identified by an S to the immediate left of the nodeid in the PVM selection screen. The following is a list of these restrictions:
a. The SET FULLREAD ON command may not be used.

b. All PA keys (except for the CP defined TERMINAL BRKKEY) are non-functional.

5. Changing the FULLREAD setting for CMS also changes the FULLREAD setting for the System Product editor.

Responses:

None.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSEF109S  Virtual storage capacity exceeded [RC = 104]
DMSSEF926E  Command is only valid on a display terminal [RC = 88]
DMSSET109S  Virtual storage capacity exceeded [RC = 104]
SET FULLSCREEN

Use the FULLSCREEN option to use full-screen CMS.

The format of the SET FULLSCREEN command is:

```
<table>
<thead>
<tr>
<th>SET</th>
<th>FULLSCREEN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ON</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
</tr>
<tr>
<td></td>
<td>SUSPEND</td>
</tr>
<tr>
<td></td>
<td>RESUME</td>
</tr>
</tbody>
</table>
```

where:

ON initializes full-screen CMS. SET FULLSCREEN ON defines the default virtual screens and windows for CMS. Output that is normally displayed by CP is trapped by CMS (by the IUCV Message All System Service) so that messages and VM output are displayed in windows.

OFF returns CMS to line-mode operation.

SUSPEND specifies that CMS should temporarily return to line-mode operation. CMS discontinues trapping I/O (by severing the *MSGALL connection) so that CP displays messages and VM output. This option could be used by applications that perform their own full-screen management, such as those that use DIAGNOSE Code X'58'.

RESUME returns a CMS session to the state that preceded a SET FULLSCREEN SUSPEND command.

Initial Setting:

FULLSCREEN OFF
Things You Should Know About Full-Screen CMS

1. When you issue SET FULLSCREEN ON, CMS is placed in full-screen mode with the default and/or user-defined windows and features (for example: CMSPF Keys, Command Line, Status Area, etc.) being displayed.

2. When entering full-screen CMS:
   - All default virtual screens that you have not defined are defined, such as a virtual screen for CMS and CP output, and virtual screens for messages, network messages, warnings from the operator, and status information
   - All reserved areas are written for the default virtual screens
   - All default windows that you have not defined are defined
   - Default windows are connected to appropriate virtual screens
   - CMSPF key definitions are established
   - A connection to the IUCV Message All System Service is established and various classes of output are routed to virtual screens
   - Logging is started for the MESSAGE and WARNING virtual screens
   - The cursor is set in the CMS virtual screen
   - The CP TERMINAL BRKKEY NONE command is issued.

3. When returning to full-screen CMS after it has been suspended:
   - The CMS window is shown
   - The STATUS window is displayed
   - A connection is reestablished to the IUCV Message All System Service
   - Window and virtual screen definitions, logging, message routing, and the CP TERMINAL BRKKEY setting are not affected.

4. When lines are written to a virtual screen sequentially, such as in the CMS virtual screen, lines are added to the virtual screen starting at the virtual screen top. Once the virtual screen is full and you scroll forward, the oldest lines that have been scrolled are deleted, new lines are appended at the bottom, and the lines are renumbered. (Lines that have not been scrolled are not deleted.) Because the lines are renumbered, the scroll location information may appear to remain the same as you scroll forward.
When the virtual screen is full and new information is waiting to be added, scrolling the virtual screen forward or entering one of the following commands: CLEAR WINDOW, CLEAR VSCREEN, SHOW WINDOW, or HIDE WINDOW, allows the virtual screen to be updated. That is, the oldest information that has been scrolled is deleted off the top so that the newest information can be added at the bottom. This updating process occurs even if the window connected to the virtual screen is hidden or overlaid by other windows.

5. When you receive the status area message “Scroll forward for more information in vscreen vname” and there are multiple windows showing the specified virtual screen, it is recommended that you scroll forward the window closest to the top of the ordered list of windows. This enables data that is in the virtual screen queue to be written to the virtual screen and displayed.

6. When you SET FULLSCREEN OFF, all information that has not been updated to a virtual screen prior to execution of the command will be typed out in line mode. Any default windows and virtual screens defined by full-screen CMS will be deleted.

In addition, the CP TERMINAL BRKKEY remains as NONE. To reset it to PA1 (the default setting), use the CP TERMINAL command.

7. Commands can only be entered in the CMS virtual screen and the WM window. Commands entered in the CMS virtual screen are always echoed in the CMS virtual screen regardless of the routing of the CMS message class.

8. You must always have a window showing the CMS virtual screen when using full-screen CMS. If you hide all the windows showing the CMS virtual screen, the CMS window is automatically shown at the top of the CMS virtual screen. The CMS window is on top of all other windows, including the STATUS window. You should then issue the POP WINDOW STATUS command.

9. The WM window is automatically displayed in full-screen CMS when no windows are showing the active virtual screen. For example, you may maximize a window so that it fills the entire screen and covers all other windows. You may not be able to enter commands in the window because it is protected. In such cases, the WM window is automatically displayed, and the WMPF keys and command line are available to manipulate the window.

In addition, the WM window is automatically displayed in full-screen CMS when you run an application that uses the CONSOLE macro to perform I/O and

- The CMS virtual screen is updated, or
- Any virtual screen (other than CMS) is updated and a pop-type window is showing it.
The WMPF keys and a command line are available. Issuing the DROP WINDOW WM command (default WMPF 3) returns you to the application.

**System Defaults for Full-Screen CMS**

1. The default windows for full-screen CMS are:

<table>
<thead>
<tr>
<th>Wname</th>
<th>Lines</th>
<th>Cols</th>
<th>Pline</th>
<th>Pscol</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATUS</td>
<td>1</td>
<td>Pscr</td>
<td>-1</td>
<td>1</td>
<td>Fixed Noborder Nopop Notop</td>
</tr>
<tr>
<td>CMS</td>
<td>Pscr</td>
<td>Pscr</td>
<td>1</td>
<td>1</td>
<td>Fixed Border Nopop Top</td>
</tr>
<tr>
<td>NETWORK</td>
<td>8</td>
<td>71</td>
<td>-12</td>
<td>7</td>
<td>Variable Border Nopop Top</td>
</tr>
<tr>
<td>WARNING</td>
<td>6</td>
<td>71</td>
<td>3</td>
<td>3</td>
<td>Variable Border Pop Top</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>8</td>
<td>71</td>
<td>11</td>
<td>3</td>
<td>Variable Border Pop Top</td>
</tr>
<tr>
<td>WM</td>
<td>5</td>
<td>Pscr</td>
<td>-1</td>
<td>1</td>
<td>Fixed Border Nopop Notop</td>
</tr>
<tr>
<td>CMSOUT</td>
<td>8</td>
<td>75</td>
<td>9</td>
<td>3</td>
<td>Variable Border Pop Top</td>
</tr>
</tbody>
</table>

Figure 30. Default Windows

**Pscr**

Size of the physical screen

**Psline**

The line on the physical screen where the upper (when psline is positive) or lower (when psline is negative) edge of the window is placed.

**Pscol**

The column on the physical screen where the upper left corner of the window is placed.
Variable
indicates that the number of lines in the window may vary
depending on the amount of scrollable data displayed.

Fixed
indicates that the number of lines in the window is always
constant.

Border
indicates that the borders are displayed when possible. For the
CMS window, the borders are on but you cannot see them because
the window is the size of the physical screen.

Noborder
indicates that borders are not displayed.

Pop
specifies that the window is displayed on top of all other windows
when the virtual screen that the window is showing is updated.

Nopop
specifies that there is no effect on the window’s position in the
ordered list of windows when the virtual screen that the window is
showing is updated.

Top
specifies that the window may qualify as the topmost window.

Notop
specifies that the window cannot qualify as the topmost window.

Note: Although the WM window is a default window, it is not defined
when you enter full-screen CMS. The WM window is defined when you
issue the command POP WINDOW WM, press the PA1 key, or when the
WM window is automatically displayed.

All default windows are SYSTEM windows, which means the window is
retained when the system abnormally terminates (abend) or when the
HX (halt execution) command is issued.
2. The default virtual screens for full-screen CMS are:

<table>
<thead>
<tr>
<th>Vname</th>
<th>Lines</th>
<th>Cols</th>
<th>Rtop</th>
<th>Rbot</th>
<th>Dcolor</th>
<th>Protected?*</th>
</tr>
</thead>
<tbody>
<tr>
<td>WM</td>
<td>1</td>
<td>Pscr</td>
<td>0</td>
<td>5</td>
<td>White</td>
<td>No</td>
</tr>
<tr>
<td>STATUS</td>
<td>1</td>
<td>Pscr</td>
<td>0</td>
<td>0</td>
<td>White</td>
<td>Yes</td>
</tr>
<tr>
<td>NETWORK</td>
<td>16</td>
<td>70</td>
<td>2</td>
<td>0</td>
<td>Blue</td>
<td>Yes</td>
</tr>
<tr>
<td>WARNING</td>
<td>4</td>
<td>70</td>
<td>2</td>
<td>0</td>
<td>Red</td>
<td>Yes</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>20</td>
<td>70</td>
<td>2</td>
<td>0</td>
<td>White</td>
<td>Yes</td>
</tr>
<tr>
<td>CMS</td>
<td>120</td>
<td>Pscr</td>
<td>2</td>
<td>5</td>
<td>Green</td>
<td>No</td>
</tr>
</tbody>
</table>

Figure 31. Default Virtual Screens

**Pscr**

Physical screen size. For terminals with a screen size of 80 columns or less, the CMS virtual screen contains 81 columns. For terminals with a screen size greater than 80 columns, the CMS virtual screen contains the same number of columns as the physical screen. The STATUS and WM virtual screens always contain the same number of columns as the physical screen.

**Rtop**

Top reserved lines

**Rbot**

Bottom reserved lines

**Dcolor**

Data color

*If protected, you cannot type into the window(s) connected to the virtual screen.

**Note:** Although the WM virtual screen is a default virtual screen, it is not defined when you enter full-screen CMS. The WM virtual screen is defined when you issue the command POP WINDOW WM, press the PA1 key, or when the WM window is automatically displayed.

All default virtual screens are TYPE and SYSTEM virtual screens. TYPE means data is moved to the virtual screen when the virtual screen is updated. SYSTEM means the virtual screen is retained when the system abnormally terminates (abend) or when the HX (halt execution) command is issued.
3. Default windows are connected to default virtual screens in the following manner:

<table>
<thead>
<tr>
<th>Window</th>
<th>Virtual Screen</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMS</td>
<td>CMS</td>
<td>Displays CMS and CP output</td>
</tr>
<tr>
<td>CMSOUT</td>
<td>CMS</td>
<td>Displays CMS and CP output while in XEDIT</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>MESSAGE</td>
<td>Displays user messages and SCIF messages</td>
</tr>
<tr>
<td>NETWORK</td>
<td>NETWORK</td>
<td>Displays network messages</td>
</tr>
<tr>
<td>STATUS</td>
<td>STATUS</td>
<td>Displays status messages</td>
</tr>
<tr>
<td>WARNING</td>
<td>WARNING</td>
<td>Displays warnings</td>
</tr>
<tr>
<td>WM</td>
<td>WM</td>
<td>Provides the capability to enter windowing commands</td>
</tr>
</tbody>
</table>

Figure 32. Default Windows and Virtual Screens

4. When SET FULLSCREEN is ON, the various message classes are routed to virtual screens as follows:

<table>
<thead>
<tr>
<th>Message Class</th>
<th>Virtual Screen</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMS</td>
<td>CMS</td>
<td>NOALARM NONOTIFY</td>
</tr>
<tr>
<td>CP</td>
<td>CMS</td>
<td>NOALARM NONOTIFY</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>MESSAGE</td>
<td>ALARM NOTIFY</td>
</tr>
<tr>
<td>WARNING</td>
<td>WARNING</td>
<td>ALARM NOTIFY</td>
</tr>
<tr>
<td>SCIF</td>
<td>MESSAGE</td>
<td>NOALARM NONOTIFY</td>
</tr>
<tr>
<td>NETWORK</td>
<td>NETWORK</td>
<td>NOALARM NOTIFY</td>
</tr>
</tbody>
</table>

Figure 33. Default Settings for Message Routing

ALARM
sounds the alarm when a message is received.

NOALARM
does not sound the alarm.

NOTIFY
displays the message class name in the status area when you receive a message.
**SET FULLSCREEN**

**NONOTIFY**

will not display the virtual screen name in the status area when you receive a message.

See the ROUTE command for information on changing the default message routing.

Commands entered in the CMS virtual screen are always echoed in the CMS virtual screen regardless of the routing of the CMS message class.

5. Any messages or warnings received during your full-screen CMS session are displayed in windows and logged into files. Messages are logged into a file called MESSAGE LOGFILE, and warnings are logged into a file called WARNING LOGFILE.

6. Pressing the PA1 key while in full-screen CMS displays the WM window. The PA2 key and CLEAR key scroll the topmost window forward. See the SET CMSPF command for the default settings for the CMSPF keys.

In the WM window, the PA2 key and CLEAR key also scroll the topmost window forward. When there is no more data in the window to scroll, you automatically exit the WM environment.

**Considerations for Applications in Full-Screen CMS**

1. If an application performs full-screen management while in full-screen CMS and it does not use the CONSOLE macro, the output written to full-screen CMS is not displayed until the application completes. Prior to running the application, issue the SET FULLSCREEN SUSPEND command; when the application completes, issue the SET FULLSCREEN RESUME command.

2. When returning to full-screen CMS from an application that performs its own full-screen management (such as DIAGNOSE Code X'58'), your screen may contain mixed data. Press the CLEAR key to scroll forward and refresh the screen.

Alternatively, issue the SET FULLSCREEN SUSPEND command, run the application, and then issue SET FULLSCREEN RESUME when the application completes. For more information, see *VM System Facilities for Programming*.

3. SET FULLSCREEN SUSPEND and SET FULLSCREEN RESUME can be “nested.” For example, suppose full-screen CMS is ON and an application called MYPROG issues SET FULLSCREEN SUSPEND and calls another application, TESTPROG. TESTPROG also issues SET FULLSCREEN SUSPEND. When TESTPROG completes, it issues SET FULLSCREEN RESUME and control returns to MYPROG. SET FULLSCREEN is still in the SUSPEND state and MYPROG continues to execute. Upon completion MYPROG issues SET FULLSCREEN RESUME, which returns FULLSCREEN to the ON state.
SET FULLSCREEN

To preserve the nesting, do not issue ON or OFF between SUSPEND and RESUME. If an application issues SET FULLSCREEN ON or OFF, the nesting is cleared and FULLSCREEN status changes to ON or OFF. Use the QUERY FULLSCREEN command to determine the FULLSCREEN status.

4. If full-screen CMS has never been set ON, and either SET FULLSCREEN OFF, SET FULLSCREEN SUSPEND or SET FULLSCREEN RESUME is issued, no action is taken.

5. The following messages are not trapped by the IUCV Message All System Service and are sent directly to the terminal:
   - Asynchronous CPCONIO, including PER/TRACE events
   - EMSGs not generated as part of a DIAGNOSE code X'08' operation instruction.
   - Accounting messages

6. The IUCV Message All System Service can stack up to 255 messages at any one time. If this limit is exceeded, any additional incoming messages are sent directly to the terminal.

7. When SET FULLSCREEN is ON, most CMS console output is not passed to CP. In addition, applications that use the IUCV Message System Service (*MSG) and SET VMCONIO to IUCV will not trap all CMS output when using full-screen CMS. Prior to using such applications, it is recommended to issue SET FULLSCREEN SUSPEND.

8. When developing an application to be used with full-screen CMS, you may want to reset the CP TERMINAL BRKKEY to PA1 (it is set to NONE in full-screen CMS). Then, you can enter CP mode to debug the application.

Responses:

None.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

- **DMSSEF109S** Virtual storage capacity exceeded [RC = 104]
- **DMSSEF926E** Command is only valid on a display terminal [RC = 88]
- **DMSSEF927E** The physical screen must contain at least 20 lines and 80 columns [RC = 24]
- **DMSSET109S** Virtual storage capacity exceeded [RC = 104]
Special Command Used in the Full-Screen Environment

The #WM command is a special command that can only be used in the CMS virtual screen in full-screen CMS.

#WM

Use the #WM command to execute a command immediately from the CMS virtual screen.

The format of the #WM command is:

```
#WM  wmcommand
```

where:

**wmcommand**

specifies the command that you want to execute. See Usage Note 2 for a list of commands you can specify with #WM.

Usage Notes:

1. The pound sign (#) represents the default logical line end symbol for full-screen CMS. If you have redefined the line end symbol to another character (using the SET LINEND command), #WM is an invalid command; you must substitute your line end symbol for the pound sign to use the command.

   The #WM command is independent of the CP logical line end symbol.

2. You can enter any of the following commands with #WM:

<table>
<thead>
<tr>
<th>CLEAR WINDOW</th>
<th>QUERY BORDER</th>
<th>SCROLL</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP</td>
<td>QUERY HIDE</td>
<td>SET BORDER</td>
</tr>
<tr>
<td>DROP WINDOW</td>
<td>QUERY LOCATION</td>
<td>SET LOCATION</td>
</tr>
<tr>
<td>HIDE WINDOW</td>
<td>QUERY RESERVED</td>
<td>SET RESERVED</td>
</tr>
<tr>
<td>MAXIMIZE WINDOW</td>
<td>QUERY SHOW</td>
<td>SET WINDOW</td>
</tr>
<tr>
<td>MINIMIZE WINDOW</td>
<td>QUERY WINDOW</td>
<td>SET WMPF</td>
</tr>
<tr>
<td>POP WINDOW</td>
<td>QUERY WMPF</td>
<td>SHOW WINDOW</td>
</tr>
<tr>
<td>POSITION WINDOW</td>
<td>RESTORE WINDOW</td>
<td>SIZE WINDOW</td>
</tr>
<tr>
<td>PUT SCREEN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that HELP is not a valid command with #WM; however, it is a valid command in the WM environment.
SET FULLSCREEN

Responses:

None.

Messages and Return Codes:

DMSWEN931E Invalid WM command: command
DMSWEN1125E command is not allowed as an immediate command
SET IMESCAPE

Use the IMESCAPE to indicate whether or not an escape character is required to execute immediate commands.

The format of the SET IMESCAPE command is:

\[
\begin{array}{|c|c|}
\hline
\text{SET} & \text{IMESCAPE} \{ \text{ON} \} \\
& \{ \text{OFF} \} \\
& \{ \text{char} \} \\
\hline
\end{array}
\]

\textit{where:}

\textbf{ON}

indicates that an escape character is required to execute immediate commands. The default escape character is a semi-colon (;).

\textbf{OFF}

indicates that an escape character is not required to execute immediate commands. This is the default setting.

\textbf{char}

indicates that an escape character is required to execute immediate commands. The escape character is a single character and it cannot be A-Z or 0-9.

\textit{Initial Setting:}

IMESCAPE OFF

\textbf{Responses:}

None.
Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSET109S Virtual storage capacity exceeded [RC = 104]
**SET IMPCP**

Use the IMPCP option to control handling of command names that CMS does not recognize. Unknown commands may be considered CP commands or an error.

The format of the SET IMPCP command is:

```
SET IMPCP [ON] [OFF]
```

**where:**

**ON**

passes command names that CMS does not recognize to CP; that is, unknown commands are considered to be CP commands.

**OFF**

generates an error message at the terminal if a command is not recognized by CMS.

**Initial Setting:**

IMPCP ON

**Responses:**

None.

**Messages and Return Codes:**

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSET109S Virtual storage capacity exceeded [RC=104]
SET IMPEX

Use the IMPEX option to control whether or not EXEC files are treated as commands.

The format of the SET IMPEX command is:

\[
\begin{array}{|c|c|}
\hline
\text{SET} & \text{IMPEX} \quad \{\text{ON}\} \\
\hline
\end{array}
\]

where:

ON

treats EXEC files as commands; an EXEC file is invoked when the filename of the EXEC file is entered.

OFF

does not consider EXEC files as commands. You must issue the EXEC command to execute an EXEC file.

If you issue SET IMPEX OFF, you may not be able to use PF keys predefined to perform EXEC functions in productivity aids such as NOTE, RDRLIST, FILELIST, etc.

Initial Setting:

IMPEX ON

Responses:

None.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSET109S Virtual storage capacity exceeded [RC = 104]
SET INPUT

Use the INPUT option to translate characters entered from your terminal to hexadecimal code. You can also reset the hexadecimal code to a specified hexadecimal code in your translate table.

The format of the SET INPUT command is:

| SET | INPUT [ a xx ] [ xx yy ] |

where:

INPUT \( a \ xx \)
- translates the specified character \( a \) to the specified hexadecimal code \( xx \) for characters entered from the terminal.

INPUT \( xx \ yy \)
- allows you to reset the hexadecimal code \( xx \) to the specified hexadecimal code \( yy \) in your translate table.

Note: If you issue SET INPUT and SET OUTPUT commands for the same characters, issue the SET OUTPUT command first.

INPUT
- returns all characters to their default translation.

Initial Setting:

None

Responses:

None.
**SET INPUT**

**Messages and Return Codes:**

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSET109S  Virtual storage capacity exceeded [RC = 104]
SET INSTSEG

Use the INSTSEG option to specify whether or not the system should search the Installation Discontiguous Shared Segment (DCSS) to locate an EXEC or System Product Editor macro.

The format of the SET INSTSEG command is:

| SET | INSTSEG {ON | OFF [mode | LAST ]} |

*where:*

**ON**
indicates that you want the system to search the Installation Discontiguous Shared Segment (DCSS) when locating an EXEC. The initial setting is ON.

**[mode][LAST]**
specifies the location of the Installation DCSS in the command search order. It is searched immediately before the disk having the mode letter that you specify. If you haven’t accessed a disk as that mode, the Installation DCSS will be searched in that position. The default mode is S. LAST indicates that the DCSS will be searched after all accessed disks have been searched.

**OFF**
indicates that you do not want the system to search the Installation Discontiguous Shared Segment (DCSS) when locating an EXEC.

*Initial Setting:*

INSTSEG ON

*Responses:*

None.
SET INSTSEG

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSET048E  Invalid mode mode [RC = 24]
DMSSET109S  Virtual storage capacity exceeded [RC = 104]
Use the SET LANGUAGE command to change the current language of your CMS session and any application running on CMS that uses national language support. When you SET another language, you will be able to receive messages, enter commands, and see CMS panels (like the FILELIST screen) in that language.

The SET LANGUAGE command also informs CP to change the language used to issue CP messages to your virtual machine.

The format of the SET LANGUAGE command is:

```
SET LANGUAGE \[langid\] [ (options ... [)]]
```

Options:

```
ADD applid
DELETE applid
USER SYSTEM
ALL
TYPE NOTYPE
```

where:

`langid` is the language identifier of the language to which the virtual machine will be set. If omitted, it defaults to the langid of the current language. A langid may be 1 to 5 characters in length and must be made up of only CMS file system characters.

Options:

ADD activates the system and/or user language files, for the application named (applid). (You can make additions to system message repositories and the CMS command syntax file; see the `VM/SP CMS for System Programming` for details.)

If you specify ADD and a langid that is different from the current language setting, then

- the current language is changed for all active applications,
- the application named is activated, and
- a request is made for CP to change the language used to issue CP messages to your virtual machine.
SET LANGUAGE

DELETE
deactivates the system and/or user language files, for the application named.

applid
specifies the application whose system and/or user language files that should be added or deleted. The applid must be three characters in length.

USER
specifies that user repositories, and/or command syntax tables, and/or command synonym tables are loaded into storage or removed from storage. See Usage Note 1.

SYSTEM
specifies that the DCSS with system-provided language files for the application named is activated or deactivated. No user language files are affected if you specify SYSTEM. This is the default.

ALL
specifies that the DCSS with system-provided language files and user additions are activated or deactivated.

TYPE
specifies that all messages from the SET LANGUAGE command are to be typed to the console. This is the default.

NOTYPE
specifies that no messages from the SET LANGUAGE command are to be typed to the console. Messages resulting from syntax errors will be displayed even if NOTYPE is specified.

Usage Notes:

1. For the USER option, CMS searches for the following fileids:

<table>
<thead>
<tr>
<th>Fileid</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>applidUME TXTlangid</td>
<td>User message repositories.</td>
</tr>
<tr>
<td>applidUPA TXTlangid</td>
<td>User command syntax definitions.</td>
</tr>
<tr>
<td>applidUSY TXTlangid</td>
<td>User national language translation and synonyms. This file is generated automatically from user additions to the Definition Language for Command Syntax (DLCS) file. For more information, see VM/SP CMS for System Programming.</td>
</tr>
</tbody>
</table>
2. The QUERY LANGLIST command displays all the valid langids that you can set in CMS. (Note that to set a language in CMS, the CP language files must also be available.) To display the language that is active for CMS in your virtual machine, issue the QUERY LANGUAGE command. Contact your system administrator if you have any questions about the languages available on your VM/SP system.

3. To delete the application DMS, use the USER option—you cannot delete DMS with the SYSTEM or ALL options.

4. If you ADD an application that already has active language files, your new language files for that application will replace the current language files.

5. Be aware of language-related terminal restrictions when specifying a language. For example, you may have a terminal set up to display the German character set; if you then SET your language to French while using the same terminal, you may get unexpected characters on your screen. If you are not sure what characters your terminal can display, check with your system administrator.

6. When you issue the SET LANGUAGE command in full-screen CMS, the reserved lines are not updated to reflect the new language. To update the reserved lines, issue SET FULLSCREEN OFF, enter the SET LANGUAGE command, and then issue SET FULLSCREEN ON.

   Similarly, if you are in a CMS environment such as FILELIST, MACLIST, RDRLIST, and SENDFILE, and you issue the SET LANGUAGE command, the reserved lines are not updated to reflect the new language. To update the reserved lines, exit the environment, enter the SET LANGUAGE command, and then issue the command to enter the environment.

Example:

Suppose your virtual machine is set to American English (langid = AMENG) and you are working in CMS (applid = DMS).

- To work in French (langid = FRANC) and run an application called APPLICATION1 (applid = AP1), enter:
  
  set language franc (add apl system

  which changes the language to French for CMS and APPLICATION1, and informs CP to change its language to French.

- To return to American English, enter:
  
  set language ameng

  which changes CMS and APPLICATION1 to American English, and tells CP to change its language back to American English (if APPLICATION1 is available in American English).
SET LANGUAGE

- To deactivate APPLICATION1, yet preserve American English as the current language, you enter:

  set language ( delete apl all

  which returns your virtual machine to where you started: running CMS in American English with no other active applications.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSSET109S  Virtual storage capacity exceeded [RC = 104]
DMSSLG005E  No application id specified  [RC = 24]
DMSSLG109S  Virtual storage capacity exceeded  [RC = 104]
DMSSLG276E  Invalid language id langid  [RC = 24]
DMSSLG277E  The DCSS is located partially or entirely inside the virtual machine  [RC = 88]
DMSSLG278E  Unable to set requested language: langid. langid forced [by CP]  [RC = 104]
DMSSLG278E  The requested language langid is not available; langid forced [by CP]  [RC = 104]
DMSSLG279E  Application applid not found in the language DCSS [RC = 28]
DMSSLG279I  Application applid not found in the language DCSS
DMSSLG280E  Application applid not active  [RC = 28]
DMSSLG281E  Application DMS cannot be deleted  [RC = 24]
DMSSLG332E  No user additions were loaded  [RC = 28]
DMSSLG332I  No user additions were loaded
DMSSLG334E  No system information or user additions were found for application applid  [RC = 28]
DMSSLG346E  Error nn loading fn ft fm from disk  [RC = 32]
DMSSLG770E  Invalid application id applid  [RC = 24]
**SET LDRTBLS**

Use the LDRTBLS option to define the number of pages of storage to be used for loader tables.

The format of the SET LDRTBLS command is:

| SET | LDRTBLS [nn] |

where:

**LDRTBLS nn**

defines the number (nn) of pages of storage to be used for loader tables. To successfully set the size of the loader tables, the SET LDRTBLS command should be issued immediately after IPL. By default, a virtual machine having up to 384K of addressable real storage has two pages of loader tables; a larger virtual machine has three pages. Each loader table page has a capacity of 204 external names. During LOAD and INCLUDE command processing, each unique external name encountered in a TEXT deck is entered in the loader table. The LOAD command clears the table before reading TEXT files; INCLUDE does not. This number can be changed with the SET LDRTBLS nn command provided that: (1) nn is a decimal number between 0 and 128, and (2) the virtual machine has enough storage available to allow nn pages to be used for loader tables. If these two conditions are met, nn pages are set aside for loader tables. If you plan to change the number of pages allocated for loader tables, you should deallocate storage at the high end of storage so that the storage for the loader tables may be obtained from that area. Usually, you can de-allocate storage by releasing one or more of the disks that were accessed.

**Initial Setting:**

Dependent on size of virtual machine.

**Responses:**

None.
SET LDRTBLS

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

- DMSSET031E  Loader tables cannot be modified [RC = 40]
- DMSSET109S  Virtual storage capacity exceeded [RC = 104]
SET LINEND

Use the LINEND option to activate and/or define the logical line end for full-screen CMS.

The format of the SET LINEND command is:

```
SET LINEND { ON } [char ]
```

where:

**ON**

allows you to enter several commands on the same line, separated by the line end character.

**OFF**

specifies that the logical line end character is not recognized.

*char*

is the character to be used as a line end character. The default line end character is a pound sign (#).

Initial Setting:

LINEND ON #

Usage Notes:

If you redefine the line end character to a symbol other than a pound sign (#), the #WM command is invalid. (Therefore, the default CMSPF keys that issue #WM commands do not function.) To use the #WM command you must substitute your line end symbol for the pound sign.

Responses:

None.
Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSEF926E Command is only valid in CMS FULLSCREEN mode [RC = 88]
DMSSET109S Virtual storage capacity exceeded [RC = 104]
SET LOCATION

Use the LOCATION option to display the location indicator in the window when the data in the virtual screen exceeds the size of the window.

The format of the SET LOCATION command is:

| SET  | LOCATION  | wname  | [ON]  | [OFF] |

where:

wname

is the name of the window.

ON
displays the location indicator when there is data to be viewed outside of the window.

OFF
does not display the location indicator.

Initial Setting:

LOCATION ON

Usage Notes:

1. Displaying the location indicator overlays data in the window. If reserved lines are defined, the information overlays those lines. If not, the information overlays the scrollable data area. The data is not changed in the virtual screen and you are prohibited from typing over the location information. To view the data being overlaid, issue SET LOCATION wname OFF.

2. If the window is not wide enough to display the entire location indicator, the location information is truncated.
Responses:

Location information for the number of lines or columns, or both, is displayed in the upper right corner of the window, using the color, highlight, and program symbol set defined for the window border. For example:

Lines 25 - 44 of 44
Columns 1 - 20 of 80

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSEF921E  Window *name* is not defined  [RC=28]
DMSSET109S  Virtual storage capacity exceeded [RC=104]
SET LOGFILE

Use the LOGFILE option to control whether or not a log file is updated.

The format of the SET LOGFILE command is:

```
SET LOGFILE vname {ON {OFF}} [fn] LOGFILE [fm [* A1]]
```

**where:**

- **vname** is the name of the virtual screen.
- **ON** begins logging for the specified virtual screen.
- **OFF** discontinues logging for the specified virtual screen.
- **fn** is the filename of the file to which data is logged. The default filename is the name of the virtual screen.
- **LOGFILE** is the filetype of the file to which data is logged.
- **fm** is the filemode of the file. The default is *, which is the first read/write disk in the search order containing the specified file. See the Usage Notes for more information.

**Initial Setting:**

LOGFILE OFF

**Usage Notes:**

1. When the NOTYPE option is in effect for a specified virtual screen, data in the queue is not written to the virtual screen. However, the data is logged to a CMS file if logging was requested.
SET LOGFILE

2. If the CMS file already exists, the data written to the virtual screen is appended to the existing CMS file. If the specified file does not exist, the file is created on the A-disk and the lines are inserted.

3. If you issue:

   SET LOGFILE vname ON

   and do not specify a filename, filetype and/or a filemode, then the current values from previous settings for the log fileid are used.

4. To specify the filemode, you must also specify fn and LOGFILE.

5. For each full-screen session, the following line is added to the file when the first message or warning is logged:

   **** Logging started for virtual screen vname on mm/dd/yr at hh:mm:sec

Responses:

None.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSEF921E  Virtual screen vname is not defined  [RC = 28]
DMSSET109S  Virtual storage capacity exceeded  [RC = 104]
DMSWVL928E  Command is not valid for virtual screen vname  [RC = 12]
SET NONDISP

Use the NONDISP option to define a character for CMS and the System Product editor that is displayed in place of nondisplayable characters.

The format of the SET NONDISP command is:

```
SET NONDISP [char]
```

where:

char defines a character that is displayed in place of nondisplayable characters. If not specified, a blank is used.

Initial Setting:

NONDISP "

Usage Notes:

1. The translation of the nondisplayable character depends upon the type of terminal, whether SET APL ON or SET TEXT ON is in effect, and the current language being used (see SET LANGUAGE).

2. Changing the NONDISP setting for CMS also changes the NONDISP setting for the System Product editor, and vice versa.

3. Changing the NONDISP character does not change characters already displayed on the screen unless that line is altered.

Responses:

None.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

- DMSSEF109S Virtual storage capacity exceeded [RC = 104]
- DMSSEF926E Command is only valid on display terminal [RC = 88]
- DMSSET109S Virtual storage capacity exceeded [RC = 104]
The format of the SET NONSHARE command is:

<table>
<thead>
<tr>
<th>SET</th>
<th>NONSHARE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMSDOS</td>
<td>CMSVSAM</td>
</tr>
<tr>
<td>CMSAMS</td>
<td>CMSBAM</td>
</tr>
</tbody>
</table>

where:

- CMSDOS
- CMSVSAM
- CMSAMS
- CMSBAM

specifies that you want your own non-shared copy of a normally shared named system.

Initial Setting:

None.

Responses:

None.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

- DMSSET109S Virtual storage capacity exceeded [RC = 104]
- DMSSET400S System sysname does not exist [RC = 44]
- DMSSET401S VM size (size) cannot exceed sysname start address (vstor) [RC = 104]
- DMSSET410S Control program error indication xxx [RC = rc]

Note: In RC = rc, the rc represents the actual error code generated by CP.
SET OUTPUT

Use the OUTPUT option to translate a hexadecimal representation displayed at the terminal to a specified character.

The format of the SET OUTPUT command is:

| SET | OUTPUT [xx a] |

where:

OUTPUT xx a
translates the specified hexadecimal representation xx to the specified character “a” for all xx characters displayed at the terminal.

OUTPUT
returns all characters to their default translation.

Initial Setting:
None

Usage Notes:

1. Output translation does not occur for SCRIPT files when the SCRIPT command output is directed to the terminal, nor when you use the CMS editor on a display terminal in display mode.

2. Changing the OUTPUT setting does not translate characters already displayed on the screen unless that line is altered.

3. The OUTPUT setting does not affect trailing nulls or blanks that are at the end of a line.

Responses:
None.
Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSET061E  No translation character specified [RC = 24]
DMSSET109S  Virtual storage capacity exceeded [RC = 104]
SET PROTECT

Use the PROTECT option to specify whether the CMS nucleus is protected against writing in its storage area.

The format of the SET PROTECT command is:

| SET | PROTECT [ON ] [OFF] |

*where:*

ON  
protects the CMS nucleus against writing in its storage area.

OFF  
does not protect the storage area containing the CMS nucleus.

*Initial Setting:*

PROTECT ON

*Responses:*

None.

*Messages and Return Codes:*

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSET109S   Virtual storage capacity exceeded [RC = 104]

Chapter 2. CMS Commands 589
SET RDYMSG

Use the RDYMSG option to specify whether CMS issues the standard CMS ready message or a shortened form.

The format of the SET RDYMSG command is:

<table>
<thead>
<tr>
<th>SET</th>
<th>RDYMSG</th>
<th>LMSG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SMSG</td>
</tr>
</tbody>
</table>

where:

LMSG
indicates that the standard CMS ready message, including current and elapsed time, is used. The format of the standard Ready message is:

Ready; T=s.ss/s.ss hh:mm:ss

where the virtual processor time (in seconds), real processor time (in seconds), and clock time are listed.

SMSG
Indicates that a shortened form of the CMS ready message (Ready;) which does not include the time is used.

Initial Setting:
RDYMSG LMSG

Responses:
None.

Messages and Return Codes:
The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSET109S Virtual storage capacity exceeded [RC=104]
**SET REDTYPE**

Use the REDTYPE option to have CMS error messages typed in red for certain terminals equipped with the appropriate terminal feature and a two-color ribbon.

The format of the SET REDTYPE command is:

```
SET REDTYPE {ON | OFF}
```

*where:*

**ON**

types CMS error messages in red for certain terminals equipped with the appropriate terminal feature and a two-color ribbon. Supported terminals are described in the VM/SP Terminal Reference.

**OFF**

suppresses red typing of error messages.

*Initial Setting:*

REDTYPE OFF

*Responses:*

None.

*Messages and Return Codes:*

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSET109S  Virtual storage capacity exceeded [RC = 104]
SET RELPAGE

Use the RELPAGE option to release page frames of storage and set them to binary zeros, or to hold the pages of storage.

The format of the SET RELPAGE command is:

| SET | RELPAGE [ON | OFF] |

where:

ON
releases (returns to CP) unused page frames of storage and sets them to binary zeros. A page frame is considered unused whenever all of the storage allocated from it has been returned to CMS.

OFF
prevents the release of unused page frames of storage as described in SET RELPAGE ON. Use the SET RELPAGE OFF function when debugging or analyzing a problem so that the storage used is not released and can be examined. Be aware that indiscriminate use of SET RELPAGE OFF can have a negative impact on overall system performance.

Initial Setting:
RELPAGE ON

Responses:
None.
Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSET109S  Virtual storage capacity exceeded [RC = 104]
SET REMOTE

Use the REMOTE option to control the display of data transmissions for CMS and the System Product editor.

The format of the SET REMOTE command is:

| SET | REMOTE | ON | OFF |

where:

ON
specifies that data is to be compressed by removing nulls and combining data when five or more of the same characters occur consecutively in a data stream. This minimizes the amount of data transmitted and shortens the buffer, thus speeding transmission.

OFF
specifies that the data stream is not to be compressed. Data is transmitted with no minimization.

Initial Setting:
REMOTE ON for remote displays
REMOTE OFF for local displays.

Usage Notes:

Changing the REMOTE setting for CMS also changes the REMOTE setting for the System Product editor, and vice versa.

Responses:

None.
### Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

<table>
<thead>
<tr>
<th>Message Code</th>
<th>Message Description</th>
<th>Return Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSSEF109S</td>
<td>Virtual storage capacity exceeded [RC = 104]</td>
<td></td>
</tr>
<tr>
<td>DMSSEF926E</td>
<td>Command is only valid on a display terminal [RC = 88]</td>
<td></td>
</tr>
<tr>
<td>DMSSET109S</td>
<td>Virtual storage capacity exceeded [RC = 104]</td>
<td></td>
</tr>
</tbody>
</table>
SET RESERVED

Use the RESERVED option to specify the maximum number of lines in a window that are used to display virtual screen reserved lines.

The format of the SET RESERVED command is:

```
SET RESERVED wname {rtop} {rbot}
```

where:

- `wname` is the name of the window.
- `rtop` is the maximum number of reserved lines that are displayed in the top of the window. The number displayed depends on the number of reserved lines defined in the virtual screen to which the window is connected, and on the number of lines in the window (see Usage Notes 2 and 3).
- `rbot` is the maximum number of reserved lines that are displayed in the bottom of the window. The number displayed depends on the number of reserved lines defined in the virtual screen to which the window is connected, and on the number of lines in the window (see Usage Notes 2 and 3).

Initial Setting:

`RESERVED wname **`

Usage Notes:

1. Reserved lines are maintained in the virtual screen buffers and are used to display such things as title lines and PF key descriptions. Reserved line data is not scrollable and takes up space above and below the scrollable data area in the middle of the window. See the description of the DEFINE VSCREEN and WRITE VSCREEN commands for more information regarding virtual screen reserved lines.

2. The number of reserved lines displayed in the window is the MINIMUM of the number specified with the SET RESERVED command or the number defined in the virtual screen to which the window is connected.
When * is specified, the number of reserved lines displayed is the number defined in the virtual screen to which the window is connected.

For example, suppose you have a window called MESSAGE that is 10 lines long. The virtual screen to which the window is connected contains 3 top reserved lines and 5 bottom reserved lines. If you issue the command:

```
set reserved message 5 2
```

when the window is displayed on the physical screen, it contains 3 top reserved lines, because the virtual screen has 3 top reserved lines, and 2 bottom reserved lines, because that is the maximum number you requested.

If you then change the size of MESSAGE so that it is now only 3 lines long, the window is displayed with 1 top reserved line and 2 bottom reserved lines, according to the rules of this note and Usage Note 3.

3. Setting reserved lines for a window is independent of the window size. The number of reserved lines to be displayed in the window is determined when the physical screen is refreshed. Lines are handled according to the following priority:

   a. Bottom reserved lines
   b. Top reserved lines
   c. Data lines (top down)

**Responses:**

None.

**Messages and Return Codes:**

The possible error messages for specifying a command format incorrectly are listed on page 24.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Return Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSSEF921E</td>
<td>Window <code>uname</code> is not defined [RC = 28]</td>
<td></td>
</tr>
<tr>
<td>DMSSET109S</td>
<td>Virtual storage capacity exceeded [RC = 104]</td>
<td></td>
</tr>
</tbody>
</table>
Use the SYSNAME option to replace a saved system name entry in the SYSNAMES table with the name of an alternate, or backup system.

The format of the SET SYSNAME command is:

```
SET SYSNAME {CMSDOS CMSVSAM CMSAMS CMSBAM} entryname
```

*where:*

\[
\begin{align*}
\{ & \text{CMSDOS} \\
\{ & \text{CMSVSAM} \\
\{ & \text{CMSAMS} \\
\{ & \text{CMSBAM} \\
\end{align*}
\]

allows you to replace a saved system name entry in the SYSNAMES table with the name of an alternative, or backup system. A separate SET SYSNAME command must be issued for each name entry to be changed. CMSVSAM, CMSAMS, CMSDOS, and CMSBAM are the default names assigned to the systems when the CMS system is generated.

*Initial Setting:*

None.

*Responses:*

None.

*Messages and Return Codes:*

The possible error messages for specifying a command format incorrectly are listed on page 24.

- DMSSET109S Virtual storage capacity exceeded [RC = 104]
- DMSSET142S Saved system name *sysname* invalid [RC = 24]
**SET TEXT**

Use TEXT option to activate character code conversion for TEXT characters for the System Product editor and CMS.

The format of the SET TEXT command is:

```
<table>
<thead>
<tr>
<th>SET</th>
<th>TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[ON</td>
</tr>
<tr>
<td></td>
<td>{OFF}</td>
</tr>
</tbody>
</table>
```

**where:**

**ON**
activates character code conversion for TEXT characters. Before using TEXT keys, issue SET TEXT ON to ensure proper character code conversion.

**OFF**
specifies that no character code conversion is performed for TEXT characters and keys.

**Initial Setting:**

TEXT OFF

**Usage Notes:**

1. The TEXT setting is valid only when performing full-screen I/O (for example, in XEDIT or in CMS with SET FULLSCREEN ON). If you are in CP or using a line-mode terminal, SET TEXT has no effect.

   If you are in CP, you can issue the TERMINAL TEXT ON command to have CP convert TEXT character codes.

2. Because the text character code conversion is costly, it is recommended that you issue SET TEXT OFF when you stop using the special text keys.

3. When SET TEXT ON is specified, APL is set OFF.

4. Changing the TEXT setting for CMS also changes the TEXT setting for the System Product editor.
SET TEXT

Responses:

None.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

- DMSSEF109S  Virtual storage capacity exceeded \[RC = 104\]
- DMSSEF524W  NONDISP character reset to ".".
- DMSSEF926E  Command is only valid on a display terminal \[RC = 88\]
- DMSSET109S  Virtual storage capacity exceeded \[RC = 104\]
- DMSWIR329W  Warning: APL/TEXT option not in effect
SET TRANSLATE

Use the SET TRANSLATE command to suppress translations and translation synonyms of command names for a language. The SET TRANSLATE command specifies the way in which the User Language Translation Tables and the System National Language Translation Tables are used.

The format of the SET TRANSLATE command is:

```
SET TRANSLate [ON OFF] [SYStem USER ALL] [TRANSlate SYNonym BOTH] [APPLID applid]
```

where:

ON
allows you to specify the table to be used for command name translation.

OFF
allows you to specify that a table will not be used for command name translation.

SYStem
translates command names using only the System National Language Translation Table.

USER
translates command names using only the User National Language Translation Table.

ALL
translates command names using both the User and System National Language Translation Tables.

TRANSlate
indicates that only the national language translations are set ON or OFF.

SYNonym
indicates that only the national language translation synonyms are set ON or OFF.
**SET TRANSLATE**

**BOTH**
indicates that both national language translations and translation synonyms are set ON or OFF.

**APPlid applid**
specifies the application for which a table is to be set ON or OFF. It must be three alphanumeric characters, and the first character must be alphabetic. The default, *, sets the tables for all applications.

**Initial Setting:**
None

**Usage Notes:**

1. If you issue the SET command specifying an invalid function and the implied CP function is in effect, you may receive message DlMKCF003E INVALID OPTIONS - option.

2. If an invalid SET command function is specified from an EXEC and the implied CP function is in effect, then the return code is -0003.

3. To determine or verify the setting of most functions, use the QUERY command.

4. Translation synonyms cannot be set ON unless translations are also set ON. Likewise, translations cannot be set OFF unless translation synonyms are also set OFF.

5. The settings for the TRANSLATE operand are enabled in the following order: System Translations, System Translation Synonyms, User Translations, and User Translation Synonyms.

**Example:**
To set the translation synonym table OFF for all applications, enter

```
set translate off user syn
```

**Responses:**
None.
SET TRANSLATE

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSET109S Virtual storage capacity exceeded [RC = 104]
DMSSET258E {user|system} translation synonyms can not be set ON unless {user|system} translations are also set ON, application id: applid [RC = 28]
DMSSET258E {user|system} translations can not be set OFF unless {user|system} translation synonyms are also set OFF, application id: applid [RC = 28]
DMSSET280E Application applid not active [RC = 28]
Use the UPSI option in the CMS/DOS environment to set the User Program Switch Indicator (UPSI) byte to the specified bit string of 0's and 1's, or to reset the UPSI byte to binary zeros.

The format of the SET UPSI command is:

| SET | UPSI [nnnnnnnn] OFF |

where:

nnnnnnnn
sets the UPSI (User Program Switch Indicator) byte to the specified bit string of 0’s and 1’s. If you enter fewer than eight digits, the UPSI byte is filled in from the left and zero-padded to the right. If you enter an “x” for any digit, the corresponding bit in the UPSI byte is left unchanged.

OFF
resets the UPSI byte to binary zeros.

Initial Setting:

UPSI OFF

Responses:

None.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSET099E CMS/DOS environment not active [RC = 40]
DMSSET109S Virtual storage capacity exceeded [RC = 104]
Use the VSCREEN option to indicate what action should take place when the virtual screen is updated with data.

The format of the SET VSCREEN command is:

```
SET VSCREEN vname
```

**where:**

- **vname**
  is the name of the virtual screen.

- **TYPE**
  specifies that data is moved to the virtual screen when the virtual screen queue is processed.

- **NOType**
  specifies that the virtual screen is not updated.

- **PRotect**
  the data is protected.

- **NOPRotect**
  the data is not protected.

- **High**
  data is displayed in high intensity.

- **NOHigh**
  data is displayed in a normal intensity.

- **color**
  the color may be Default, Blue, Red, Pink, Green, Turquoise, Yellow, or White.

- **exthi**
  the extended highlighting may be None (default), REVvideo, BLInk, or Underline.
SET VSCREEN

`psset`

the Programmed Symbol Set (PSset) may be specified as PS0 (the default), PS1, PSA, PSB, PSC, PSD, PSE, or PSF.

*Initial Setting:*

See DEFINE VSCREEN.

**Usage Notes:**

1. When NOTYPE is specified, the data is not written to the virtual screen when the queue is processed. However, the data is logged to a CMS file if logging was requested (see the SET LOGFILE command).

2. In full-screen CMS, SET VSCREEN CMS NOTYPE suppresses all updates to the CMS virtual screen. However, to suppress only the error messages from within an EXEC, use the SET CMSTYPE HT command.

**Responses:**

None.

**Messages and Return Codes:**

The possible error messages for specifying a command format incorrectly are listed on page 24.

- DMSSEF921E Virtual screen `uname` is not defined [RC = 28]
- DMSSEF928E Command is not valid for virtual screen `uname` [RC = 12]
- DMSSET108S Virtual storage capacity exceeded [RC = 104]
SET WINDOW

Use the WINDOW option to specify

- whether a window is variable or fixed size
- if the window is affected when the virtual screen that the window is showing is updated.
- whether or not the window qualifies as the topmost window.

The format of the SET WINDOW command is:

```
SET WINDOW wname [VARiable] [FIXed] [POP] [NOPop] [TOP] [NOTop]
```

where:

- `wname` is the name of the window.
- `VARiable` indicates that the current number of lines in the window may vary from 0 to the number of lines defined for the window, depending on how much data is displayed,
- `FIXed` indicates that the number of lines in the window is always constant.
- `POP` specifies that the window is displayed on top of all other windows when the virtual screen that the window is showing is updated.
- `NOPop` specifies that there is no effect on the window's position in the ordered list of windows when the virtual screen that the window is showing is updated.
- `TOP` specifies that the window may qualify as the topmost window. Most windowing commands process the topmost window by default or when `=` is specified as the window name.
SET WINDOW

NOTOp
specifies that the window cannot qualify as the topmost window.
Windows defined as NOTOP are not processed by default or when a
command is specified with = for the window name.

Initial Setting:
See DEFINE WINDOW.

Responses:
None.

Messages and Return Codes:
The possible error messages for specifying a command format incorrectly
are listed on page 24.

DMSSEF921E  Window wname is not defined [RC = 28]
DMSSET109S  Virtual storage capacity exceeded [RC = 104]
Use the WMPF option to set a WMPF key to execute a windowing command.

The format of the SET WMPF command is:

```
SET WMPF nn [pseudonym] [keyword] string
```

where:

**nn**

is a number from 1 to 24 indicating which PF key is being set.

**pseudonym**

is a 9-character representation of the PF key definition. The pseudonym is displayed in the PF key definition area at the bottom of the CMS window. The pseudonym may be up to nine characters in length.

**NOWRITE**

suppresses overwriting of the PF key pseudonym when you set a WMPF key.

**keyword**

indicates when the command associated with the PF key is executed in relation to commands entered at the terminal. The *keyword* may be one of the following:

**DELAYED**

delays the execution of the command string. When the key is pressed, the command string is displayed in the input area and is not executed until you press the ENTER key. If anything is currently in the input area, it is overlaid and no commands entered on the physical screen are processed. This is the default setting if no keyword is specified on the SET WMPF command.

**ECHO**

executes the command immediately when the program function key is pressed. The key definition is echoed above the command line in the WM window.
SET WMPF

NOECHO

executes the command immediately when the program function key is pressed. The key definition is not echoed on the physical screen.

Note: When a WMPF key is set to RETRIEVE the keyword is ignored.

string

is the command(s) to be executed when the key is pressed.

Initial Settings:

| WMPF 01 Help   | NOECHO HELP         |
| WMPF 02 Top    | NOECHO SCROLL TOP = |
| WMPF 03 Quit   | NOECHO DROP WINDOW WM |
| WMPF 04 Clear  | NOECHO CLEAR WINDOW = |
| WMPF 05 Copy   | NOECHO PUT SCREEN COPY SCREEN |
| WMPF 06 Retrieve | NOECHO RETRIEVE |
| WMPF 07 Backward | NOECHO SCROLL BACKWARD = 1 |
| WMPF 08 Forward | NOECHO SCROLL FORWARD = 1 |
| WMPF 09 Maximize | NOECHO MAXIMIZE WINDOW = |
| WMPF 10 Left   | NOECHO SCROLL LEFT = 10 |
| WMPF 11 Right  | NOECHO SCROLL RIGHT = 10 |
| WMPF 12 Restore | NOECHO RESTORE WINDOW = |

Note: These are initial settings. On a terminal equipped with 24 PF keys, PF 13 through 24 have the same values as PF keys 1 through 12, respectively.

Usage Notes:

1. You can set a WMPF key to execute any of the following commands:

| CLEAR WINDOW | PUT SCREEN | SCROLL |
| CP           | QUERY BORDER | SET BORDER |
| DROP WINDOW  | QUERY HIDE | SET LOCATION |
| HELP         | QUERY LOCATION | SET RESERVED |
| HIDE WINDOW  | QUERY RESERVED | SET WINDOW |
| MAXIMIZE WINDOW | QUERY SHOW | SET WMPF |
| MINIMIZE WINDOW | QUERY WINDOW | SHOW WINDOW |
| POP WINDOW   | QUERY WMPF | SIZE WINDOW |
| POSITION WINDOW | RESTORE WINDOW |

In the WM environment, you can enter HELP (WMPF 1) to see the list of commands that are available. The WM environment creates a WMHELP window and WMHELP virtual screen to display the list. To exit the WM window, use the DROP WINDOW command (WMPF 3).

2. To cancel a PF key definition, enter:

    SET WMPF nn
3. When you press a PA key or a WMPF key in the WM window and the key that was pressed does not update the command line, then input on the command line is rewritten. The next time you press enter it is executed.

4. The RETRIEVE function saves previously entered commands in a buffer that is 256 characters long. When you enter the WM environment, the buffer contains an asterisk (comment), and commands are added to the buffer as they are entered until it is full. As you continue to enter commands, the oldest commands are deleted and the most current commands are added.

Pressing the PF key assigned to RETRIEVE displays the next command in the buffer on the command line. Each time you press the key, the previously entered command is displayed until the oldest one is reached. Then, RETRIEVE returns the most current command. Once the command is on the command line, press enter to execute it. You may also modify the command, then press enter to execute the new command.

5. The NOWRITE option is particularly useful when you have changed the bottom reserved area in the WM virtual screen and you do not want the area overwritten when you set a WMPF key.

Responses:

None.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSEF109S Virtual storage capacity exceeded [RC = 104]
DMSSEF525E Invalid PFkey number [RC = 24]
DMSSEF926E Command is only valid on a display terminal [RC = 88]
DMSSET109S Virtual storage capacity exceeded [RC = 104]
SETPRT

Use the SETPRT command to load virtual 3800 printers. The SETPRT command is valid only for a virtual 3800 Model 1 or 3800 Model 3 printer. The format of the SETPRT command is:

```
SETPRT
  [Chars [()]ccc...[]]
  [Copies [()]nnn[]]
  [COPYnr [()]nnn[]]
  [Fcb [()]fff[]]
  [FLASH [()]id nnn[]]
  [Init]
  [Modify [()]mmm[n][]]
```

where:

Chars cccc...
  specifies the names of from one to four character arrangement tables (CATs) to be loaded into the virtual 3800 printer. CAT names may be from one to four alphanumerical characters. The CATs must exist as 'XTB1ccc TEXT' files on an accessed CMS disk.

Copies nnn
  specifies the total number of copies of each page to be printed. The value of nnn must be a number from 1 to 255. The default value is 1.

COPYnr nnn
  specifies the copy number of the first copy in a copy group. The value of nnn must be a number from 1 to 255. If COPYNR is not specified, a starting copy number of 1 is assumed.

Fcb ffff
  specifies the FCB to be loaded into the virtual 3800 printer. The FCB must exist as a 'FCB3fff TEXT' file on an accessed CMS disk unless ffff is specified as 6, 8, 12, or for the 3800 Model 3 printer only, 10. In that case, the FCB is not loaded from a CMS file. CP determines the appropriate FCB to load and prints the entire file at 6, 8, 12, or for the 3800 Model 3 printer, 10 lines per inch.

FLASH id nnn
  specifies the one- to four-character overlay name (id) and the number of copies of each page (nnn) to be printed with the overlay indicated by 'id'. nnn may be a number from 0 to 255. If n is not specified, 1 is
the default. If the FLASH keyword is omitted, no copies are printed with an overlay.

**Init**

specifies that an “Initialize Printer” CCW will be issued before any other functions specified in this command are performed.

**Modify mmmm [n]**

specifies copy modification data to be loaded. The copy modification must exist as a ‘MOD1mmmm TEXT’ file on an accessed CMS disk. Further, n specifies the CAT to use for the copy modification load. If n is omitted, 0 is the default.

*Note:* Keyword values must be enclosed in parentheses only if they could be interpreted as a SETPRT keyword or keyword abbreviation. Otherwise the parentheses may be omitted.

**Usage Notes:**

1. CATs must be specified so that they correspond to the appropriate TRC bytes. The first CAT specified corresponds to TRC byte 0, the second CAT corresponds to TRC byte 1, and so on.

2. CATs can reference the Library Character Set (LCS) modules and Graphic Character Modification Modules (GRAPHMODS) for both the 3800 Model 1 and Model 3 printers. SETPRT uses naming conventions to select the correct modules for the defined 3800 model printer.

3. Customized 3800 Model 1 character sets must be converted from the 180 x 144 to the 240 x 240 pel density format before they may be used in a 3800 Model 3 printer.

4. If the number of copies specified with the FLASH keyword is greater than the number of copies specified in COPIES nnn, the actual number of copies printed will equal the number specified with the FLASH keyword. Thus, if you want all copies to be printed with an overlay, you can specify the number with the FLASH keyword and omit the COPIES keyword.

5. The use of ‘INIT’ and ‘FCB 6|8|10|12’ together causes the printer to always be reset to 6 lines per inch. Both the INIT CCW and the ‘CP SPOOL 00E FCB 6|8|10|12’ generated by the ‘FCB 6|8|10|12’ are passed to CP. The LOADFCB CCW is sent to the printer before the INIT CCW. This resets the FCB to the Init IMPL Default of 6 lines per inch. ‘INIT’ and ‘FCB ffff’ do not have this problem, since ‘FCB ffff’ is handled directly by CMS.

6. SETPRT FCB 6|8|10|12 sets the FCB of the virtual printer until it is specifically changed. SETPRT FLASH sets the FLASH until another SETPRT is issued. All other SETPRT options only affect the next file to be printed.
SETPRT

7. The length of the FCB to be loaded must agree with the forms length specified in the SIZE parameter of the CP DEFINE 3800 command.

Example:

For example, to indicate that you want to use character set GF10 printed at 6 lines per inch, enter:

```
setprt chars gf10 fcb 6
```

Responses:

```
DMSSPR196I  Printer vdev setup complete
```

The virtual 3800 printer was successfully loaded.

Messages and Return Codes:

- **DMSSPR002E**: File [fn ft [fim]] not found [RC = 28]
- **DMSSPR014E**: Invalid keyword function [RC = 24]
- **DMSSPR026E**: Invalid value value for keyword keyword [RC = 24]
- **DMSSPR113S**: Printer 00E not attached [RC = 100]
- **DMSSPR145S**: Intervention required on printer [RC = 100]
- **DMSSPR197S**: Undiagnosed error from printer 00E [RC = 100]
- **DMSSPR198E**: SETPRT load check; sense = sense
- **DMSSPR198S**: Printer 00E not a virtual 3800 Model 1 or Model 3
- **DMSSPR204E**: Too many WCGMs needed for CHARS
- **DMSSPR352E**: Invalid SETPRT data in file fn ft
SHOW WINDOW

Use the SHOW WINDOW command to place a window at the top of the window display order and to connect the window to a virtual screen.

The format of the SHOW WINDOW command is:

```
SHOW WINdow    wname [ON vname [line col]]
```

where:

- `wname` is the name of the window.
- `vname` is the name of the virtual screen to which the window is connected.
- `line` is the line number of the virtual screen where the upper left corner of the window is placed.
- `col` is the column number of the virtual screen where the upper left corner of the window is placed.

Usage Notes:

1. Multiple windows may be connected to a single virtual screen.

2. If the window is already connected to a virtual screen when you issue the SHOW WINDOW command, you do not have to specify the virtual screen information.

3. When you specify a virtual screen name, line, and column, the line and column values must be less than or equal to the corresponding virtual screen dimensions. The minimum line and column value is 1. If line and column are not specified, the default is 1 for both. If the line specified is past the current virtual screen bottom, the window is connected to the virtual screen bottom.

4. A variable size window is only displayed when there is at least one scrollable line to show. If a variable size window is showing a virtual screen that does not contain any scrollable lines, the SHOW WINDOW command places the window at the top of the window display order and connects it to the specified virtual screen, but it is not displayed.
SHOW WINDOW

5. When you are using full-screen CMS and you enter the SHOW WINDOW command from the command line, the command is executed and then the screen is refreshed. As part of the refresh processing, any pop-type window that has output waiting is moved to the top of the order of displayed windows. Therefore, the window that you specified may not be displayed at the top of the display order because another has been popped afterwards.

Responses:

None.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSSH0386E  Missing operand(s)  [RC = 24]
DMSSH0388E  Invalid keyword: keyword  [RC = 24]
DMSSH0389E  Invalid operand: operand  [RC = 24]
DMSSH0391E  Unexpected operand(s): operand  [RC = 24]
DMSSWMM921E Virtual screen vname is not defined  [RC = 28]
DMSSWMM921E Window wname is not defined  [RC = 28]
DMSSWMM923E Specified location is outside the virtual screen  [RC = 32]
DMSSWMM929E Window wname is not connected to a virtual screen  [RC = 36]
SIZE WINDOW

Use the SIZE WINDOW command to change the number of lines and columns for a specified window.

The format of the SIZE WINDOW command is:

```
SIZE WINDOW {wname = } lines [cols]
```

**where:**

- **wname**
  - is the name of the window. An "=" indicates that the size of the topmost window is changed.

- **lines**
  - is the number of lines in the window.

- **cols**
  - is the number of columns in the window. The default is the current number of columns.

**Usage Note:**

The window’s size and location must be such that, excluding borders, the entire window fits on the physical screen.

**Responses:**

None.
Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSSIZ386E  Missing operand(s) [RC = 24]
DMSSIZ388E  Invalid keyword: keyword [RC = 24]
DMSSIZ389E  Invalid operand: operand [RC = 24]
DMSSIZ921E  Unexpected operand(s): operand [RC = 24]
DMSSIZ921E  Window uname is not defined RC = 28
DMSSIZ922E  Window does not fit entirely on the screen RC = 32
SORT

Use the SORT command to read fixed-length records from a CMS input file, arrange them in ascending EBCDIC order according to specified sort fields, and create a new file containing the sorted records.

The format of the SORT command is:

```
SORT fileid1 fileid2
```

where:

- `fileid1` is the file identifier (filename, filetype, filemode) of the file containing the records to be sorted.

- `fileid2` is the file identifier (filename, filetype, filemode) of the new output file to contain the sorted records.

Usage Note:

The input and output files must not have the same file identifiers, since SORT cannot write the sorted output back into the space occupied by the input file. If `fileid2` is the same as `fileid1`, message DMSSRT019E Identical fileids is issued and the SORT operation does not take place. If `fileid1` and `fileid2` are different and a file with the same name as `fileid2` already exists, the existing file is replaced when the SORT operation takes place.

Entering Sort Control Fields: After the SORT command is entered, CMS responds with the following message on the terminal:

```
DMSSRT604R Enter sort fields:
```

Respond by entering one or more pairs of numbers of the form “xx yy”; separate each pair by one or more blanks. Each “xx” is the starting character position of a sort field within each input record and “yy” is the ending character position. The leftmost pair of numbers denotes the major sort field. The number of sort fields is limited to the number of fields you can enter on one line. The records can be sorted on up to a total of 253 positions.

Virtual Storage Requirements for Sorting: The sorting operation takes place with two passes of the input file. The first pass creates an ordered pointer table in virtual storage. The second pass uses the pointer table to read the input file in a random manner and write the output file.
Therefore, the size of storage and the size and number of sort fields are the limiting factors in determining the number of records that can be sorted at any one time. An estimate of the maximum number of records that can be sorted is:

\[
\text{FREELOWE} - 132K \\
\text{NR} = \frac{\text{FREELOWE}}{14 + \text{NC}} \\
\]

**where:**

- **NR** is the estimated maximum number of input records.
- **NC** is the total number of characters in the defined sort fields.

**FREELOWE** is the upper limit of GETMAIN storage available to the virtual machine.

**132K** is the size of the resident CMS nucleus plus the SORT module.

**FREELOWE** can be obtained from x'514' in the CMS nucleus.

*Note:* **FREELOWE** is subject to change depending on GETMAIN storage previously acquired and not FREEMAINed. Refer to the manual *VM/SP CMS for System Programming* in the “Structure of CMS Storage” section.

For example, enter the command and respond to the prompting message:

```
sort name address al sortedna address bl 
```

```
DMSSRT604R Enter sort fields: 
   1 10 25 28 
```

The records in the NAME ADDRESS file are sorted on positions 1-10 and 25-28. The sorted output is written into the newly created file SORTEDNA ADDRESS. If **FREELOWE** is 320K, you can sort a maximum of 6875 records.

\[
\text{FREELOWE-132K} = 320K - 132K = 188K = 192,512 \\
\text{NR} = \frac{\text{FREELOWE}}{14 + \text{NC}} = \frac{188K}{28} = \frac{192,512}{28} = 6875 \\
\]

**Responses:**

```
DMSSRT604R Enter sort fields: 
```

You are requested to enter SORT control fields. You should enter them in the form described previously in “Entering Sort Control Fields.”
**Messages and Return Codes:**

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSSRT002E  File \([fn \ [ft \ [fm]]]\) not found \([RC = 28]\)
- DMSSRT009E  Column \(col\) exceeds record length \([RC = 24]\)
- DMSSRT019E  Identical fileids \([RC = 24]\)
- DMSSRT034E  File \(fn \ ft \ fm\) is not fixed length \([RC = 32]\)
- DMSSRT037E  Disk \(model(vdev)\) is accessed as read/only \([RC = 36]\)
- DMSSRT053E  Invalid sort field pair defined \([RC = 24]\)
- DMSSRT054E  Incomplete fileid specified \([RC = 24]\)
- DMSSRT062E  Invalid * in fileid \([fn \ ft \ [fm]]\) \([RC = 20]\)
- DMSSRT063E  No list entered \([RC = 40]\)
- DMSSRT069E  Disk \(mode\) not accessed \([RC = 36]\)
- DMSSRT070E  Invalid parameter \(parameter\) \([RC = 24]\)
- DMSSRT104S  Error \(nn\) reading file \(fn \ ft \ fm\) from disk \([RC = 100]\)
- DMSSRT105S  Error \(nn\) writing file \(fn \ ft \ fm\) on disk \([RC = 100]\)
- DMSSRT212E  Maximum number of records exceeded \([RC = 40]\)
SSERV

SSERV

Use the SSERV command in CMS/DOS to copy, display, print, or punch a book from a VSE source statement library.

The format of the SSERV command is:

```
SSERV sublib bookname [ft.. . ..] [ (options...[)] ]
```

Options: [DISK] [PRINT] [PUNCH] [TERM]

where:

sublib
specifies the source statement sublibrary in which the book is cataloged.

bookname
specifies the name of the book in the VSE private or system source statement sublibrary. The private library, if any, is searched before the system library.

ft
specifies the filetype of the file to be created on your A-disk. “ft” defaults to COPY if a filetype is not specified. The filename is always the same as the bookname.

Options:

You may enter as many options as you wish, depending upon the functions you want to perform.

DISK
copies the book to a CMS file.

PUNCH
punches the book on the virtual punch.

PRINT
spools a copy of the book to your virtual printer.
TERM displays the book on your terminal.

Usage Notes:

1. If you want to copy books from private libraries, you must issue an ASSGN command for the logical unit SYSSLB and identify the library on a DLBL command line using a ddname of IJSYSSL.

   If you want to copy books from the system library, you must have entered the CMS/DOS environment specifying the mode letter of the system residence volume.

2. You should not use the SSERV command to copy books from macro (E) sublibraries, since they are in “edited” (that is, compressed) form. Use the ESERV command to copy and de-edit macros from a macro (E) sublibrary.

Responses:

When you use the TERM option, the specified book is displayed at the terminal.

Messages and Return Codes:

- DMSSRV003E  Invalid option: option [RC = 24]
- DMSSRV004E  Book subl.book not found [RC = 28]
- DMSSRV006E  No read/write A disk accessed [RC = 36]
- DMSSRV070E  Invalid parameter parameter [RC = 24]
- DMSSRV089E  Open error code nn on SYSSLB [RC = 36]
- DMSSRV097E  No SYSRES volume active [RC = 36]
- DMSSRV098E  No book name specified [RC = 24]
- DMSSRV099E  CMS/DOS environment not active [RC = 40]
- DMSSRV105S  Error nn writing file fn ft fm on disk [RC = 100]
- DMSSRV113S  Disk(vedev) not attached [RC = 100]
- DMSSRV194S  Book subl.book contains bad records [RC = 100]
- DMSSRV411S  Input error code nn on SYSaaa [RC = rc]
START

Use the START command to begin execution of CMS, OS, or VSE programs that were previously loaded or fetched. The format of the START command is:

```
START  
  [ entry [args...]]
  *  
  \{option \} 
```

**Option:** [ NO ]

**where:**

**entry**

passes control to the control section name or entry point name at execution time. The operand, entry, may be a filename only if the filename is identical to a control section name or an entry point name.

*  

passes control to the default entry point. See the discussion of the LOAD command for a discussion of the default entry point selection.

**args...**

are arguments to be passed to the started program. If user arguments are specified, the entry or * operands must be specified; otherwise, the first argument is taken as the entry point. Arguments are passed to the program via general register 1. The entry operand and any arguments become a string of doublewords, one argument per doubleword, and the address of the list is placed in general register 1.

**Option:**

**NO**

suppresses execution of the program. Linkage editor and loader functions are performed and the program is in storage ready to execute, but control is not given to the program. START * and START (NO) are mutually exclusive.
Usage Notes:

1. Any undefined names or references specified in the files loaded into storage are defined as zero. Thus, if there is a call or branch to a subroutine from a main program, and if the subroutine has never been loaded, the call or branch transfers control at execution time to location zero of the virtual machine.

2. Do not use the START command for programs that are generated via the GENMOD command with the NOMAP option. The START command does not execute properly for such programs.

3. When arguments are passed on the START command, the requirements of both CMS and the language of the application program must be met. For example, COBOL programs require arguments separated by commas:

   START * A,B,C

   See the appropriate language guide for details on parameter requirements.

4. Issue the START command immediately following the LOAD and INCLUDE commands. If the LOAD and INCLUDE were issued in an EXEC procedure, issue the START command from within the EXEC as well.

5. If START is issued from the virtual console or from an EXEC 2 EXEC, register 0 points to an extended parameter list block. The extended parameter list for the START command pointed to by register 0 has the following structure:

   DC   A(EPLCMD)
   DC   A(EPLARGBG)
   DC   A(EPLARGGND)
   DC   A(0)

   where:

   START          entry     any arguments
                  EPLCMD      EPLARGBG     EPLARGGND .

   or:

   START          entry
                  EPLCMD      EPLARGBG     EPLARGGND .

   or:
Responses:

DMSLIO740I    Execution begins ...

is displayed when the designated entry point is validated.

This message is suppressed if CMS/DOS is active and the COMP option is specified in the FETCH command.

Messages and Return Codes:

DMSLIO021E    Entry point name not found [RC = 40]
DMSLIO055E    No entry point defined [RC = 40]
STATE/STATEW (ESTATE/ESTATEW)

Use the STATE or ESTATE command to verify the existence of a CMS, OS, or DOS file on any accessed disk; use the STATEW or ESTATEW command to verify the existence of a CMS, OS, or DOS file on any accessed read/write disk.

It may be necessary to use ESTATE/ESTATEW when writing EXECs or assembler programs. This enables STATE to handle files larger than 65533 records. (This is the equivalent of coding the FSSTATE macro with the FORM=E option.)

The formats of the STATE, STATEW, ESTATE, and ESTATEW commands are:

<table>
<thead>
<tr>
<th>STATE</th>
<th>[fn]</th>
<th>[ft]</th>
<th>[fm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATEW</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESTATE</td>
<td>[fn]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESTATEW</td>
<td>[fn]</td>
<td>[ft]</td>
<td></td>
</tr>
</tbody>
</table>

where:

fn
if the filename of the file whose existence is to be verified. If fn is specified as *, the first file found satisfying the rest of the fileid is used.

ft
is the filetype of the file whose existence is to be verified. If ft is specified as *, the first file found satisfying the rest of the fileid is used.

fm
is the filemode of the file whose existence is to be verified. If fm is specified, the parent disk and its read-only extensions will be searched. If fm is omitted, or specified as *, all your disks (A-Z) are searched.
Usage Notes:

1. If you issue the STATEW command specifying a file that exists on a read-only disk, you receive error message DMSSTT002E.

2. When you code an asterisk in the fn or ft fields, the search for the file is ended as soon as any file satisfies any of the other conditions. For example, the command:

   state * file

executes successfully if any file on any accessed disk (including the system disk) has a filetype of FILE.

3. To verify the existence of an OS or VSE file when DOS is set OFF, you must issue the FILEDEF command to establish a CMS file identifier for the file. For example, to verify the existence of the OS file TEST.DATA on an OS C-disk you could enter:

   filedef check disk check list c dsn test data state check list

   where CHECK LIST is the CMS filename and filetype associated with the OS data set name.

4. To verify the existence of an OS or VSE file when the CMS/DOS environment is active, you must issue the DLBL command to establish a CMS file identifier for the file. For example, to verify the existence of the DOS file TEST.DATA on a DOS C-disk, you could enter:

   dlb1 check c dsn test data state file check

   where FILE CHECK is the default CMS filename and filetype (FILE ddname) associated with the VSE file-id.

5. To verify the syntax of a file identifier (filename filetype filemode) without verifying the existence of the file, use the VALIDATE command.

6. You can invoke the STATE/STATEW command from the terminal, from an EXEC file, or as a function from a program. If STATE/STATEW is invoked as a function or from an EXEC file that has the message output suppressed, the message DMSSTT002E File fn ft fm not found is not issued.

7. If the STATE/STATEW (or ESTATE/ESTATEW) command is invoked via SVC202 in an assembler program, the address of the STATEFST copy is returned at X'1C' into the STATE parameter list.

8. The STATE command disregards the filemode number specified when both the filename and filetype are explicitly specified. When the filename or filetype (or both) are specified as asterisk (*), the filemode number is used.
Responses:

The CMS ready message indicates that the specified file exists.

DMSSTT227I  Processing volume no in data set data set name

The specified data set has multiple volumes; the volume being processed is shown in the message. The STATE command treats end-of-volume as end-of-file and there is no end-of-volume switching.

DMSSTT228I  User labels bypassed on data set data set name

The specified data set has disk user labels; these labels are skipped.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSSTT002E  File [fn [ft [fm]]] not found [RC = 28]
DMSSTT048E  Invalid mode mode [RC = 24]
DMSSTT054E  Incomplete fileid specified [RC = 24]
DMSSTT062E  Invalid character char in fileid fn ft [RC = 20]
DMSSTT062E  SO and SI are invalid fileid characters [RC = 20]
DMSSTT069E  Disk mode not accessed [RC = 36]
DMSSTT070E  Invalid parameter parameter [RC = 24]
DMSSTT227I  Processing volume no in data set data set name
DMSSTT228I  User labels bypassed on data set data set name
DMSSTT229E  Unsupported OS data set, error nn [RC = 8n]
DMSSTT253E  File fn ft fm cannot be handled with supplied parameter list [RC = 88]
SVCTRACE

Use the SVCTRACE command to trace and record information about supervisor calls occurring in your virtual machine.

The format of the SVCTRACE command is:

```
SVCTRACE [ON | OFF]
```

where:

ON
starts tracing all SVC instructions issued within CMS.

OFF
stops SVC tracing.

Usage Notes:

1. The trace information recorded on the printer includes:
   - The virtual storage location of the calling SVC instruction and the name of the called program or routine
   - The normal and error return addresses
   - The contents of the general registers both before the SVC-called program is given control and after a return from that program
   - The contents of the general registers when the SVC handling routine is finished processing
   - The contents of the floating-point registers before the SVC-called program is given control and after a return from that program
   - The contents of the floating-point registers when the SVC handling routine is finished processing
   - The parameter list passed to the SVC

2. To terminate tracing previously established by the SVCTRACE command, issue the HO or SVCTRACE OFF commands. SVCTRACE OFF and HO cause all trace information recorded, up to the point they are issued, to be printed on the virtual spooled printer. On typewriter terminals SVCTRACE OFF can be issued only when the keyboard is
unlocked to accept input to the CMS command environment. To terminate tracing at any other point in system processing, HO must be issued. To suspend tracing temporarily during a session, interrupt processing and enter the Immediate command SO (Suspend Tracing). To resume tracing that was suspended with the SO command, enter the Immediate command RO (Resume Tracing).

If you issue the CMS Immediate command HX or you log off the VM/SP system before termination of tracing previously set by the SVCTRACE command, the switches are cleared automatically and all recorded trace information is printed on the virtual spooled printer.

3. If a user timer exit is activated while SVCTRACE is active, SVCTRACE is disabled for the duration of the timer exit. Any SVCs issued during the timer exit are not reflected in the SVCTRACE listing.

4. If your program must remain disabled for interrupts (in an interrupt handler, for example), it must not issue any SVC's while SVCTRACE is active. SVCTRACE enables the system for interrupts. Use the CP PER command instead.

5. When tracing on a virtual machine with only one printer, the trace data is intermixed with other data sent to the virtual printer.

Responses:

A variety of information is printed whenever the:

```
svctrace on
```

command is issued.

The first line of trace output starts with a dash or plus sign or an asterisk (- or + or *). The format of the first line of trace output is:

```
{[-+]} N/D = xxx/dd name FROM loc OLDPSW = psw1 GOPSW = psw2 [RC=rc]
```

where:

- indicates information recorded before processing the SVC.

+ indicates information recorded after processing the SVC, unless the asterisk (*) applies.

* indicates information recorded after processing a CMS SVC that had an error return.

N/D is an abbreviation for SVC number and depth (or level).
**SVCTrace**

- `xxx` is the number of the SVC call (they are numbered sequentially).
- `dd` is the nesting level of the SVC call.
- `name` is the macro or routine being called.
- `loc` is the program location from which the SVC was issued.
- `psw1` is the PSW at the time the SVC was called.
- `psw2` is the PSW with which the routine being called is invoked, if the first character of this line is a dash (-). If the first character of this line is a plus sign or asterisk (+ or *), PSW2 represents the PSW that returns control to the user.
- `rc` is the return code from the SVC handling routine in general register 15. This field is omitted if the first character of this line is a dash (-), or if this is an OS SVC call. For a CMS SVC, this field is 0 if the line begins with a plus sign (+), and nonzero for an asterisk (*). Also, this field equals the contents of R15 in the "GPRS AFTER" line.

The next two lines of output are the contents of the general registers when control is passed to the SVC handling routine. This output is identified at the left by ".GPRSB." The format of the output is:

```
.GPRSB = h h h h h h h h *ddddd*  
         = h h h h h h h h *ddddd*  
```

where `h` represents the contents of a general register in hexadecimal format and `d` represents the EBCDIC translation of the contents of a general register. The contents of general registers 0 through 7 are printed on the first line, with the contents of registers 8 through F on the second line. The hexadecimal contents of the registers are printed first, followed by the EBCDIC translation. The EBCDIC translation is preceded and followed by an asterisk (*).

The next line of output is the contents of general registers 0, 1, and 15 when control is returned to your program. The output is identified at the left by ".GPRS AFTER :." The format of the output is:

```
.GPRS AFTER : RO-R1 = h h *dd* R15 = h *d*  
```

where `h` represents the hexadecimal contents of a general register and `d` is the EBCDIC translation of the contents of a general register. The only general registers that CMS routines alter are registers 0, 1, and 15 so only those registers are printed when control returns to your program. The EBCDIC translation is preceded and followed by an asterisk (*).

The next two lines of output are the contents of the general registers when the SVC handling routine is finished processing. This output is identified at the left by ".GPRSS." The format of the output is:
where \( h \) represents the hexadecimal contents of a general register and \( d \) represents the EBCDIC translation of the contents of a general register. General registers 0 through 7 are printed on the first line with registers 8 through F on the second line. The EBCDIC translation is preceded and followed by an asterisk (*).

The next line of output is the contents of the calling routine’s floating-point registers. The output is identified at the left by “.FPRS.” The format of the output is:

\[
.FPRS = f f f f *g g g g *
\]

where \( f \) represents the hexadecimal contents of a floating-point register and \( g \) is the EBCDIC translation of a floating-point register. Each floating point register is a doubleword; each \( f \) and \( g \) represents a doubleword of data. The EBCDIC translation is preceded and followed by an asterisk (*).

The next line of output is the contents of floating-point registers when the SVC handling routine is finished processing. The output is identified by “.FPRSS” at the left. The format of the output is:

\[
.FPRSS = f f f f *g g g g *
\]

where \( f \) represents the hexadecimal contents of a floating-point register and \( g \) is the EBCDIC translation. Each floating-point register is a doubleword and each \( f \) and \( g \) represents a doubleword of data. The EBCDIC translation is preceded and followed by an asterisk (*).

The last two lines of output are printed only if the address in register 1 is a valid address for the virtual machine. If printed, the output is the parameter list passed to the SVC. The output is identified by “.PARM” at the left. The output format is:

\[
.PARM = h h h h h h h h *d d d d d d *
\]

where \( h \) represents a word of hexadecimal data and \( d \) is the EBCDIC translation. The parameter list is found at the address contained in register 1 before control is passed to the SVC handling program. The EBCDIC translation is preceded and followed by an asterisk (*).
Figure 34 summarizes the types of SVC trace output.

<table>
<thead>
<tr>
<th>Identification</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>- N/D</td>
<td>The SVC and the routine that issued the SVC.</td>
</tr>
<tr>
<td>(+) N/D</td>
<td>The SVC and the routine that issued the SVC.</td>
</tr>
<tr>
<td>(*) N/D</td>
<td>The SVC and the routine that issued the SVC.</td>
</tr>
<tr>
<td>.GPRSB</td>
<td>Contents of general registers when control is passed to the SVC handling routine.</td>
</tr>
<tr>
<td>.GPRS AFTER</td>
<td>Contents of general registers 0, 1, and 15 when control is returned to your program.</td>
</tr>
<tr>
<td>.GPRSS</td>
<td>Contents of the general registers when the SVC handling routine is finished processing.</td>
</tr>
<tr>
<td>.FPRS</td>
<td>Contents of floating-point registers before the SVC-called program is given control and after returning from that program.</td>
</tr>
<tr>
<td>.FPRSS</td>
<td>Contents of the floating-point registers when the SVC handling routine is finished processing.</td>
</tr>
<tr>
<td>.PARM</td>
<td>The parameter list, when one is passed to the SVC.</td>
</tr>
</tbody>
</table>

Figure 34. Summary of SVCTRACE Output Lines

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSOVR014E  Invalid function function [RC = 24]
DMSOVR047E  No function specified [RC = 24]
DMSOVR104S  Error nn reading file fn ft fm from disk [RC = 100]
DMSOVR109S  Virtual storage capacity exceeded [RC = 104]
SYNONYM

Use the SYNONYM command to invoke a table of synonyms to be used with, or in place of, CMS and user-written command names. You create the table yourself using the editor. The form for specifying the entries for the table is described under “The User Synonym Table.”

The names you define can be used either instead of or in conjunction with the standard CMS command truncations. However, no matter what truncations, synonyms, or truncations of the synonyms are in effect, the full real name of the command is always accepted.

The format of the SYNONYM command is:

```
SYNonym [ fn SYNONYM [ fm ] ] [ (options...[]) ]
```

Options:

- **STD**
  - specifies that standard CMS abbreviations are accepted.

- **NOSTD**
  - standard CMS abbreviations are not to be accepted. (The full CMS command and the synonyms you defined can still be used.)

where:

- **fn**
  - is the filename of the file containing your synonyms table.

- **fm**
  - is the filemode of the file containing your synonyms; if omitted, your A-disk and its extensions are searched. If you specify fm, you must enter the keyword, SYNONYM. If you specify fm as an asterisk (*), all disks are searched for the specified SYNONYM file.
SYNONYM

**CLEAR**

removes any synonym table set by a previously entered SYNONYM command.

**Usage Notes:**

1. If you enter the SYNONYM command with no operands, the system synonym table and the user synonym table (if one exists) are listed.

2. The SET ABBREV ON or OFF command, in conjunction with the SYNONYM command, determines which standard and user-defined forms of a particular CMS command are acceptable.

3. The SYNONYM command cannot define national language translations.

4. EXEC procedures having a synonym defined for them can be invoked by its synonym if implied EXEC (IMPEX) function is on. However, within an EXEC procedure, only the EXEC filename can be used. A synonym is not recognized within an EXEC since the synonym tables are not searched during EXEC processing.

5. When ABBREV is OFF, the synonyms defined for command names (EXECs and MODULEs) are valid, but abbreviations are ignored.

**The User Synonym Table**

You create the synonym table using the CMS editor. The table must be a file with the filetype SYNONYM. The file consists of 80-byte fixed-length records in free-form format with columns 73-80 ignored. The format for each record is:

```
systemcommand  usersynonym  count
```

**where:**

```
systemcommand
```

is the name of the CMS command or MODULE or EXEC file for which you are creating a synonym.

```
usersynonym
```

is the synonym you are assigning to the command name. When you create the synonym, you must follow the same syntax rules as for commands; that is, you must use the character set used to create commands, the synonym may be no longer than eight characters, and so on.

```
count
```

is the minimum number of characters that must be entered for the synonym to be accepted by CMS. If omitted, the entire synonym must be entered (see the following example).
A table of command synonyms is built from the contents of this file. You may have several synonym files but only one may be active at a time. For example, if the synonym file named MYSYN contains:

```
MOVEFILE MVIT
```

then, after you have issued the command:

```
synonym mysyn
```

the synonym MVIT can be entered as a command name to execute the MOVEFILE command. It cannot be truncated since no count is specified.

If MYSYN SYNONYM contains:

```
ACCESS GETDISK 3
```

then, the synonyms GET, GETD, GETDI, GETDIS, or GETDISK can be entered as the command name instead of ACCESS.

If you have an EXEC file named TDISK, you might have a synonym entry:

```
TDISK TDISK 2
```

so that you can invoke the EXEC procedure by specifying the truncation TD.

**The Relationship between the SET ABBREV and SYNONYM Commands:**

The default values of the SET and SYNONYM commands are such that the system synonym abbreviation table is available unless otherwise specified.

The system synonym abbreviation table for the FILEDEF command states that FI is the minimum truncation. Therefore, the acceptable abbreviations for FILEDEF are: FI, FIL, FILE, FILED, FILEDE, and FILEDEF. The system synonym abbreviation table is available whenever both SET ABBREV ON and SYNONYM (STD) are in effect.

If you have a synonym table with the file identification USERTAB SYNONYM A, that has the entry:

```
FILEDEF USENAME 3
```

then, USENAME is a synonym for FILEDEF, and acceptable truncations of USENAME are: USE, USEN, USENA, USENAM, and USENAME. The user synonym abbreviation table is available whenever both SET ABBREV ON and SYNONYM USERTAB are specified.

No matter what synonyms and truncations are defined, the full real name of the command is always in effect.

Figure 35 lists the forms of the system command and user synonyms available for the various combinations of the SET ABBREV and SYNONYM commands.
SYNONYM

Responses:

When you enter the SYNONYM command with no operands, the synonym table(s) currently in effect are displayed.

<table>
<thead>
<tr>
<th>SYSTEM COMMAND</th>
<th>USER SYNONYM</th>
<th>SHORTEST FORM (IF ANY)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This response is the same as the response to the command QUERY SYNONYM ALL.

DMSSYN711I No system synonyms in effect

This response is displayed when you issue the SYNONYM command with no operands after the command SYNONYM (NOSTD) has been issued.

DMSSYN712I No synonyms (DMSINA not in nucleus)

The system routine which handles SYNONYM command processing is not in the system.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSSYN002E File [fn ft fm] not found [RC = 28]
DMSSYN003E Invalid option: option [RC = 24]
DMSSYN007E File fn ft fm [is] not fixed, 80-character records [RC = 32]
DMSSYN032E Invalid filetype ft [RC = 24]
DMSSYN056E File fn ft [fm] contains invalid record formats [RC = 32]
DMSSYN066E option1 and option2 are conflicting options [RC = 24]
DMSSYN104S Error nn reading file fn ft fm from disk [RC = 100]
<table>
<thead>
<tr>
<th>Options</th>
<th>Acceptable Command Forms</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET ABBREV ON SYN USERTAB (STD)</td>
<td>FI, FIL, ., ., FILEDEF, USE, USEN, ., ., USENAME</td>
<td>The ABBREV ON option of the SET command and the STD option of the SYNONYM command make the system table available. The user synonym, USENAME, is available because the synonym table (USERTAB) is specified on the SYNONYM command. The truncations for USENAME are available because SET ABBREV ON was specified with the USERTAB also available.</td>
</tr>
<tr>
<td>SET ABBREV OFF SYN USERTAB (STD)</td>
<td>FILEDEF, USENAME</td>
<td>The user-defined synonym, USENAME, is permitted because the user synonym table (USERTAB) is specified on the SYNONYM command. No system or user truncations are permitted.</td>
</tr>
<tr>
<td>SET ABBREV ON SYN USERTAB (NOSTD)</td>
<td>FILEDEF, USE, ., ., USENAME</td>
<td>The system synonym table is unavailable because the NOSTD option is specified on the SYNONYM command. The user synonym, USENAME, is available because the user synonym table (USERTAB) is specified on the SYNONYM command and the truncations of USENAME are permitted because SET ABBREV ON is specified with USERTAB also available.</td>
</tr>
<tr>
<td>SET ABBREV OFF SYN USERTAB (NOSTD)</td>
<td>FILEDEF, USENAME</td>
<td>The system synonym table is made unavailable either by the SET ABBREV OFF command or by the SYN (NOSTD command. The synonym, USENAME, is permitted because the user-defined synonym table (USERTAB) is specified on the SYNONYM command. The truncations for USENAME are not permitted because the SET ABBREV OFF option is in effect.</td>
</tr>
<tr>
<td>SET ABBREV ON SYN (CLEAR STD)</td>
<td>FI, FIL, ., ., FILEDEF</td>
<td>The user-defined table is now unavailable. The system synonym table is available because both the ABBREV ON option of the SET command and the STD option of the SYNONYM command are specified.</td>
</tr>
</tbody>
</table>

Figure 35 (Part 1 of 2). System and User-Defined Truncations
### Options

<table>
<thead>
<tr>
<th>Options</th>
<th>Acceptable Command Forms</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET ABBREV OFF</td>
<td>FILEDEF</td>
<td></td>
</tr>
<tr>
<td>SYN (CLEAR STD)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SET ABBREV ON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYN (CLEAR NOSTD)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SET ABBREV OFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYN (CLEAR NOSTD)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Because CLEAR is specified on the SYNONYM command, the synonym and its truncations are no longer available. Either the SET ABBREV OFF command or the SYNONYM (NOSTD command make the system synonym table unavailable.

---

**Figure 35 (Part 2 of 2). System and User-Defined Truncations**

This table does not apply to EXECs.
**TAPE**

Use the TAPE command to dump CMS-formatted files from disk to tape, load previously dumped files from tape to disk, and perform various control operations on a specified tape drive. Files processed by the TAPE command must be in a unique CMS format. The TAPE command does not process multivolume files. Disk files to be dumped can contain either fixed- or variable-length records.

The format of the TAPE command is:

<table>
<thead>
<tr>
<th>TAPE</th>
<th>DUMP</th>
<th>LOAD</th>
<th>SCAN</th>
<th>SKIP</th>
<th>DVOL1</th>
<th>WVOL1</th>
<th>MODESET</th>
<th>tapecmd</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{fn}</td>
<td>{ft}</td>
<td>{fm}</td>
<td></td>
<td>{fn}</td>
<td></td>
<td></td>
<td>n</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>{volid}</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>owner</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

**Option A:**
- WTM
- NOWTM
- BLKSIZE 4096
- BLKSIZE 800

**Option B:**
- NOPRint
- PRINT
- Term
- DISK

**Option C:**
- EOT
- EOF n
- EOF 1

**Option D:**
- TAPn
- TAP1
- 7TRACK
- 9TRACK
- 18TRACK
- [DEN den]
- [TRTCH a]
- [vdev 181]

**Option E:**
- REWIND
- LEAVE

**Option F:**
- TRANSfer BUFFERed
- TRANSfer IMMEDIATE
where:

**DUMP** \( \{f_n\} \{f_t\} \{f_m\} \)

dumps one or more disk files to tape. If \( f_n \) and/or \( f_t \) is specified as an asterisk (*) all files that satisfy the other file identifier are dumped.

If \( f_m \) is coded as a letter, that disk and its extensions are searched for the specified file(s). If \( f_m \) is coded as a letter and number, only files with that mode number and letter (and the extensions of the disk referenced by that \( f_m \) letter) are dumped. If \( f_m \) is coded as asterisk (*), all accessed disks are searched for the specified file(s). If \( f_m \) is not specified, only the A-disk and its extensions are searched.

**LOAD** \( \{f_n\} \{f_t\} \{f_m\} \)

reads tape files onto disk. If a file identifier is specified, only that one file is loaded. If the option EOF \( n \) is specified and no file identifier is entered, \( n \) tape files are written to disk. If an asterisk (*) is specified for \( f_n \) or \( f_t \), all files within EOF \( n \) that satisfy the other file identifier are loaded.

The files are written to the disk indicated by the filemode letter. The filemode number, if entered, indicates that only files with that filemode number are to be loaded.

**SCAN** \( \{f_n\} \{f_t\} \)

positions the tape at a specified point, and lists the identifiers of the files it scans. Scanning occurs over \( n \) tape marks, as specified by the option EOF \( n \) (the default is 1 tape file). However, if a file identifier (\( f_n \) and \( f_t \)) is specified, scanning stops upon encountering that file; the tape remains positioned ahead of the file.

**SKIP** \( \{f_n\} \{f_t\} \)

positions the tape at a specified point and lists the identifiers of the files it skips. Skipping occurs over \( n \) tape marks, as specified by the option EOF \( n \) (the default is 1 tape mark). However, if a file identifier (\( f_n \) and \( f_t \)) is specified, skipping stops after encountering that file; the tape remains positioned immediately following the file.

**MODESET**

sets the values specified by the DEN, TRACK, and TRTCH options if the tape is at load point. After initial specification in a TAPE command, these values remain in effect for the virtual tape device until they are changed in a subsequent TAPE command, RDTAPE, WRTAPE, or TAPECTL macro.

If density is not specified on the next tape command or if MODESET is not specified on the next tape macro issued, the tape drive reverts to the default density. For dual density drives the default density is the
highest density of the drive. See Usage Note 7 for further considerations.

\[ \text{tapcmd } [n] \]

specifies a tape control function (tapcmd) to be executed \( n \) times (default is 1 if \( n \) is not specified). These functions also work on tapes in a non-CMS format.

**Tapcmd**  
**Action**  
BSF  Backspace \( n \) tape marks  
BSR  Backspace \( n \) tape records  
ERG  Erase a defective section of the tape  
FSF  Forward-space \( n \) tape marks  
FSR  Forward-space \( n \) tape records  
REW  Rewind tape to load point  
RUN  Rewind tape and unload  
WTM  Write \( n \) tape marks

**DVOLI**

displays an 80-character VOL1 label in EBCDIC on the user's terminal if such a label exists on the tape. If the first record on the tape is not a VOL1 label, an error message is sent to the user.

**WVOLI volid \{owner\}**

writes a VOL1 label on a tape. All fields are set to the same values they are set to when a VOL1 label is written by the IBM-supplied IEHINITT utility program (see the publication *OS/VS2 MVS Utilities* for details). The volid is set to the 1- to 6-character volid specified on the command. If the user specifies owner field, it is written in the owner name and address code field of the label. It can be up to eight characters long and left-justified in the 10-byte field in the label. If not specified, the owner field is set to blanks. The WVOLI option also writes a dummy HDR1 label and tape mark after the VOL1 label.

**Options:**

If conflicting options are specified, the last one entered is in effect.

**WTM**

writes two tape marks after a file is dumped. Subsequent files write over the second tape mark. Therefore, one tape mark will separate files and two tape marks follow the last file dumped. This should be distinguished from TAPE WTM \( n \), which writes the number of tape marks specified by \( n \).

**NOWTM**

writes two tape marks after each file is dumped, then backspaces over both of them so that subsequent files written on the tape are not separated by tape marks.
BLKSIZE 800
BLKSIZE 4096
specifies the size of the tape data block at which the files are to be dumped (not including a five-byte prefix).

NOPRint
does not spool the list of files dumped, loaded, scanned, or skipped to the printer.

PRint
spools the list of files dumped, loaded, scanned, or skipped to the printer.

Term
displays a list of files dumped, loaded, scanned, or skipped at the terminal.

DISK
creates a disk file containing the list of files dumped, loaded, scanned, or skipped. The disk file has the file identification of TAPE MAP A5.

EOT
reads the tape until an end-of-tape indication is received.

EOF n
EOF 1
reads the tape through a maximum of n tape marks. The default is EOF 1.

TAPn
vdev
specifies the symbolic tape identification (TAPn) or the virtual device address (vdev) of the tape to be read from or written to. The following symbolic names and virtual device addresses are supported:

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>Virtual Address</th>
<th>Symbolic Name</th>
<th>Virtual Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAP0</td>
<td>180</td>
<td>TAP8</td>
<td>288</td>
</tr>
<tr>
<td>TAP1</td>
<td>181</td>
<td>TAP9</td>
<td>289</td>
</tr>
<tr>
<td>TAP2</td>
<td>182</td>
<td>TAPA</td>
<td>28A</td>
</tr>
<tr>
<td>TAP3</td>
<td>183</td>
<td>TAPB</td>
<td>28B</td>
</tr>
<tr>
<td>TAP4</td>
<td>184</td>
<td>TAPC</td>
<td>28C</td>
</tr>
<tr>
<td>TAP5</td>
<td>185</td>
<td>TAPD</td>
<td>28D</td>
</tr>
<tr>
<td>TAP6</td>
<td>186</td>
<td>TAPE</td>
<td>28E</td>
</tr>
<tr>
<td>TAP7</td>
<td>187</td>
<td>TAPF</td>
<td>28F</td>
</tr>
</tbody>
</table>

The default is TAP1 or 181. The unit specified by vdev must previously have been attached to your CMS virtual machine before any tape I/O operation can be attempted.
7TRACK
specifies a 7-track tape. Odd parity, data convert on, and translate off are assumed unless TRTCH is specified.

9TRACK
specifies a 9-track tape.

18TRACK
specifies an 18-track tape.

DEN \( \text{den} \)
is the tape density where \( \text{den} \) is 200, 556, 800, 1600, 6250, or 38K. If 200 or 556 is specified, 7TRACK is assumed. If 1600 or 6250 is specified, 9TRACK is assumed; if 800 is specified, 9TRACK is assumed unless 7TRACK is specified. If 38K is specified, 18TRACK is assumed. In the case of either 800/1600 or 1600/6250 dual-density drives, 1600 is the default if the 9TRACK option is specified. If neither the 9TRACK option nor the DEN option is specified, the drive operates at whatever BPI the tape drive was last set. The following densities are allowed for the given track sizes.

\[
\begin{align*}
7\text{track} & : 200, 556, 800 \\
9\text{track} & : 800, 1600, 6250 \\
18\text{track} & : 38K
\end{align*}
\]

TRTCH \( \text{a} \)
is the tape recording technique for 7-track tape. If TRTCH is specified, 7TRACK is assumed. One of the following must be specified as “\( \text{a} \)”:

\[
\begin{align*}
\text{a} & \quad \text{Meaning} \\
O & \quad \text{Odd parity, data convert off, translate off} \\
OC & \quad \text{Odd parity, data convert on, translate off} \\
OT & \quad \text{Odd parity, data convert off, translate on} \\
E & \quad \text{Even parity, data convert off, translate off} \\
ET & \quad \text{Even parity, data convert off, translate on}
\end{align*}
\]

REWIND
LEAVE
are only valid for the DVOL1 and WVOL1 functions. They specify the positioning of a tape after the VOL1 is processed. If REWIND is specified, the tape is rewound and positioned at load point. If LEAVE (the default) is specified, the tape is positioned at the record immediately after the VOL1 label.

\[
\text{[Transfer BUFFERed] [Transfer IMMEDIATE]}
\]

Specifies the tape write mode for the 3480 Magnetic Tape Subsystem. The two write modes are:
TAPE

- Buffered Write Mode (BUFFered)
- Tape Write Immediate Mode (IMMEDiate).

BUFFered mode is the default

In BUFFered mode, data is transferred from the processor to the tape control unit, then the processor and tape control unit are disconnected from each other. The tape control unit then writes the data onto the tape and performs error checking and, if necessary, error recovery procedures.

In IMMEDiate mode, data is physically written onto tape and “read-back” checked (verified) by the microprogram in the control unit while the processor and the control unit are still connected. This mode is provided, at a severe performance degradation, for nonrestartable write operations.

Usage Notes:

1. Tape records written by the CMS TAPE DUMP command are either 805 bytes long, if the option BLKSIZE is specified as 800; or 4101 bytes long if the BLKSIZE is specified as, or defaults to, 4096. The first character is a binary 2 (X'02'), followed by the characters CMS and a file format byte. For a variable format file, the file format byte is V. For a fixed format file without null blocks, the file format byte is F; otherwise the file format byte is S. In the final record, the character N replaces the file format byte, and the data area contains CMS file directory information. A tape created at 4096-byte block size is not reloadable on a CMS system that does not have the multivalue BLKSIZE option on the TAPE command; however, the 800-byte BLKSIZE option provides backward compatibility to such a system.

2. If a tape file contains more CMS files than would fit on a disk, the tape load operation may terminate if there is not enough disk space to hold the files. To prevent this, when you dump the files, separate logical files by tape marks, then forward space to the appropriate file.

3. Because the CMS file directory is the last record of the file, the TAPE command creates a separate workfile so that backspacing and rereading can be avoided when the disk file is built. If the load criteria is not satisfied, the workfile is erased; if it is satisfied, the workfile is renamed. This workfile is named TAPE CMSUT1, which may exist if a previous TAPE command has abnormally terminated. If the work file is accidentally dumped to tape and subsequently loaded, it appears on your disk as TAPE CMSUT2.

4. The RUN option (rewind and unload) indicates completion before the physical operation is completed. Thus, a subsequent operation to the same physical device may encounter a device busy situation.
5. It is possible to run a tape off the reel in at least one situation. If you specify EOF n and n is greater than the number of tape marks on the tape, the tape will run off the reel.

6. The DVOL1 and W VOL1 are the only TAPE command functions that automatically process tape labels. TAPE DUMP does not automatically write labels on a tape when it writes the dump file, and TAPE LOAD does not recognize tape labels when loading a file.

7. To reset the mode of a variable tape drive when using an IBM standard label tape for output, rewrite the VOL1 label before processing tape using the TAPE W VOL1 command. This is a hardware restriction which allows changes to the tape drive mode only when the tape is at load point. If you are writing to a non-label tape, use the TAPE MODESET command to set the mode. The first write operation will cause the mode to be reset since the tape will be at loadpoint when the write takes place.

8. Do not use TAPE DVOL1 for a tape that you suspect to be blank. If you do, and the tape is blank, it will run off the reel.

9. The options for the 8809 tape drive must be 9TRACK and DEN 1600. Note that these are the default values, so you do not need to specify them.

10. For more information on tape file handling, see the *VM/SP CMS User's Guide*. Tapes with a density of 38K BPI and 18TRACK are used only by the 3480 Magnetic Tape Subsystem. This subsystem does not read or write current half-inch tape (such as used by the 2400 and other 3400 subsystems). Data on a 3480 cartridge must be copied onto a half-inch tape to interchange with systems that do not have a 3480 subsystem.

11. The first time that the TAPE command is issued, it is invoked from the transient area; thereafter it is invoked as a nucleus extension.
Responses:

DMSTPF701I  Null file

A final record was encountered and no prior records were read in a TAPE LOAD operation. No file is created on disk.

If the TERM option is in effect, the following is displayed at the terminal depending on the operation specified:

Loading.....
    fn ft fm
    . . .
    . . .

Skipping.....
    fn ft fm
    . . .
    . . .

Dumping.....
    fn ft fm
    . . .
    . . .

Scanning.....
    fn ft fm
    . . .
    . . .

When a tape mark is encountered, the following is displayed at the terminal if the TERM option is specified:

END-OF-FILE OR END-OF-TAPE

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSLMX264E  command is not a valid command to be established as a nucleus extension by DMSLMX [RC=24]
DMSLMX514E  Return code nnn from NUCXLOAD
DMSTPF701I  Null file
DMSTPF613E  Tape must be invoked as a nucleus extension [RC=40]
DMSTPJ002E  File[s] [fn [ft [fm]]] not found [RC=28]
DMSTPJ003E  Invalid option: option [RC=24]
DMSTPJ010E  Premature EOF on file [(fn ft [fm][number nn)] [RC=40]
DMSTPJ014E  Invalid function function [RC=24]
DMSTPJ017E  Invalid device address vdev [RC=24]
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSTPJ023E</td>
<td>No filetype specified [RC = 24]</td>
</tr>
<tr>
<td>DMSTPJ027E</td>
<td>Invalid device devtype [for SYSaaa] [RC = 24]</td>
</tr>
<tr>
<td>DMSTPJ029E</td>
<td>Invalid parameter parameter in the option option field [RC = 24]</td>
</tr>
<tr>
<td>DMSTPJ037E</td>
<td>Disk mode is read/only [RC = 36]</td>
</tr>
<tr>
<td>DMSTPJ042E</td>
<td>No fileid([s]) specified [RC = 24]</td>
</tr>
<tr>
<td>DMSTPJ043E</td>
<td>Tapn(udev) is file protected [RC = 36]</td>
</tr>
<tr>
<td>DMSTPJ047E</td>
<td>No function specified [RC = 24]</td>
</tr>
<tr>
<td>DMSTPJ048E</td>
<td>Invalid mode mode [RC = 24]</td>
</tr>
<tr>
<td>DMSTPJ057E</td>
<td>Invalid record format [RC = 32]</td>
</tr>
<tr>
<td>DMSTPJ058E</td>
<td>End-of-file or end-of-tape [RC = 40]</td>
</tr>
<tr>
<td>DMSTPJ069E</td>
<td>Disk mode not accessed [RC = 36]</td>
</tr>
<tr>
<td>DMSTPJ070E</td>
<td>Invalid parameter parameter [RC = 24]</td>
</tr>
<tr>
<td>DMSTPJ096E</td>
<td>File fn ft data block count incorrect [RC = 32]</td>
</tr>
<tr>
<td>DMSTPJ104S</td>
<td>Error nn reading file fn ft fm from disk [RC = 100]</td>
</tr>
<tr>
<td>DMSTPJ105S</td>
<td>Error nn writing file fn ft fm on disk [RC = 100]</td>
</tr>
<tr>
<td>DMSTPJ109S</td>
<td>Virtual storage capacity exceeded [RC = 104]</td>
</tr>
<tr>
<td>DMSTPJ110S</td>
<td>Error reading tapn(udev) [RC = 100]</td>
</tr>
<tr>
<td>DMSTPJ111S</td>
<td>Error writing tapn(udev) [RC = 100]</td>
</tr>
<tr>
<td>DMSTPJ113S</td>
<td>Tapn(udev) not attached [RC = 100]</td>
</tr>
<tr>
<td>DMSTPJ115S</td>
<td>Conversion feature not supported on device udev [RC = 88]</td>
</tr>
<tr>
<td>DMSTPJ115S</td>
<td>Dual density feature not supported on device udev [RC = 88]</td>
</tr>
<tr>
<td>DMSTPJ115S</td>
<td>Translation feature not supported on device udev [RC = 88]</td>
</tr>
<tr>
<td>DMSTPJ115S</td>
<td>9-track feature not supported on device udev [RC = 88]</td>
</tr>
<tr>
<td>DMSTPJ115S</td>
<td>7-track feature not supported on device udev [RC = 88]</td>
</tr>
<tr>
<td>DMSTPJ115S</td>
<td>18-track feature not supported on device udev [RC = 88]</td>
</tr>
<tr>
<td>DMSTPJ115S</td>
<td>800 BPI feature not supported on device udev [RC = 88]</td>
</tr>
<tr>
<td>DMSTPJ115S</td>
<td>1600 BPI feature not supported on device udev [RC = 88]</td>
</tr>
<tr>
<td>DMSTPJ115S</td>
<td>6250 BPI feature not supported on device udev [RC = 88]</td>
</tr>
<tr>
<td>DMSTPJ431E</td>
<td>Tapn(udev) VOL1 label missing [RC = 32]</td>
</tr>
<tr>
<td>DMSTPJ671E</td>
<td>Error loading file fn ft fm; rc = nn from RENAME [RC = 100]</td>
</tr>
</tbody>
</table>
TAPEMAC

Use the TAPEMAC command to create a CMS MACLIB from an unloaded partitioned data set (PDS) from a tape created by the IEHMOVE utility program under OS. The PDS from which the tape was created can be blocked, but the logical record length must be 80.

The format of the TAPEMAC command is:

```
TAPEMAC fn [SL [labeldefid] [NSL filename [ID = identifier]]] [options... [ ) ]]
```

Options:
- `TAPn`  
- `TAPI`  
- `ITEMCT yyyy`  
- `ITEMCT 50000`

**where:**

- **fn**
  - specifies the filename of the first, or only, CMS MACLIB to be created on the A-disk. If fn MACLIB already exists on the A-disk, the old one is erased; no warning message is issued.

- **SL**
  - means that the tape has standard labels. The default is SL without a labeldefid. With the default specification, the standard header labels are only displayed on the user's terminal. If labeldefid is specified, the standard labels are not displayed, but are checked by the tape label checking routine.

- **NSL**
  - means that the tape has nonstandard labels.

- **labeldefid**
  - identifies the LABELDEF command that supplies descriptive label information for the file to be processed. The labeldefid given here must match the 1- to 8-character identifier specified as the filename on the LABELDEF command that was previously issued.

- **filename**
  - is the CMS filename of a routine to process nonstandard labels. The filetype must be TEXT or MODULE. If both TEXT and MODULE files exist, the MODULE file is used. MODULE files that are used for NSL routines with the TAPEMAC command must be created so that they start at an address above X'21000'. This prevents the NSL modules from overlaying the command. See the section "Tape Labels".
in CMS” in the VM/SP CMS User’s Guide for details on how to write routines to process nonstandard labels.

**ID = identifier**
specifies a 1- to 8-character identifier to be passed to a user-written NSL routine. You may use the identifier in any way you want to identify the file being processed. The identifier is passed to the user routine exactly as specified in the ID operand. If an identifier is not specified, blanks are passed. See the section “Tape Labels in CMS” in the VM/SP CMS User’s Guide for details on communicating with routines that process nonstandard labels.

**Options:**

**TAPn**
specifies the symbolic name of the tape. The following names are supported and represent these virtual units:

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>Virtual Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAP0</td>
<td>180</td>
</tr>
<tr>
<td>TAP1</td>
<td>181</td>
</tr>
<tr>
<td>TAP2</td>
<td>182</td>
</tr>
<tr>
<td>TAP3</td>
<td>183</td>
</tr>
<tr>
<td>TAP4</td>
<td>184</td>
</tr>
<tr>
<td>TAP5</td>
<td>185</td>
</tr>
<tr>
<td>TAP6</td>
<td>186</td>
</tr>
<tr>
<td>TAP7</td>
<td>187</td>
</tr>
<tr>
<td>TAP8</td>
<td>288</td>
</tr>
<tr>
<td>TAP9</td>
<td>289</td>
</tr>
<tr>
<td>TAPA</td>
<td>28A</td>
</tr>
<tr>
<td>TAPB</td>
<td>28B</td>
</tr>
<tr>
<td>TAPC</td>
<td>28C</td>
</tr>
<tr>
<td>TAPD</td>
<td>28D</td>
</tr>
<tr>
<td>TAPE</td>
<td>28E</td>
</tr>
<tr>
<td>TAPF</td>
<td>28F</td>
</tr>
</tbody>
</table>

The default is TAP1.

**ITEMCT yyyy**
specifies the item count threshold of each MACLIB to be created, which is the maximum number of records to be written into each file. (commas are not allowed). If ITEMCT is not specified, the default is 50000.

**Usage Notes:**

1. Tape records are read and placed into fn MACLIB until the file size exceeds the ITEMCT (item count); loading then continues until the end of the current member is reached. Then another CMS file is created; its filename consists of the number 2 appended to the end of the filename specified (fn) if the filename is seven characters or less. The appended number overlays the last character of the filename if the name is eight characters long. Loading then continues with this new name. For example, if you enter the command:

   tapemac mylib

you may create files named MYLIB MACLIB, MYLIB2 MACLIB, MYLIB3 MACLIB, and so on.
This process continues until up to nine CMS files have been created. If more data exists on the tape than can fit in nine CMS files, processing is terminated with the error message DMSTMA139S. A maclib created by the TAPEMAC command may contain a maximum of 256 directory entries.

2. Only header labels of the first file encountered are displayed or checked if SL or SL labdefid is specified. Trailer labels are not processed or displayed; they are skipped.

3. The following examples illustrate the different ways tape labels are processed by TAPEMAC. The command

```
tapemac mac6 sl
```

displays any standard VOL1 or HDR1 labels on a tape before loading maclib MAC6. It does not stop before loading the MACLIB.

If you specify

```
labeldef taplab fid macfile crdte 77106
tapemac mac8 sl taplab
```

CMS checks the HDR1 label on the tape before loading MAC8. It uses the information you supplied in the LABELDEF command TAPLAB to check the label. If there are discrepancies between fields you specified in the LABELDEF command and in the actual tape label, the MACLIB is not loaded.

If you specify

```
tapemac mac10 ns1 ns13
```

CMS uses your own routine NSL3 to process tape labels before loading MAC10.

**Responses:**

The TAPEMAC command displays the message:

```
LOADING fn MACLIB
```

for each macro library created.

**Messages and Return Codes:**

- **DMSTMA001E** No filename specified [RC = 24]
- **DMSTMA003E** Invalid option: option [RC = 24]
- **DMSTMA027E** Invalid device devtype [for SYSaaa] [RC = 24]
- **DMSTMA057E** Invalid record format [RC = 32]
- **DMSTMA069E** Disk mode not accessed [RC = 36]
- **DMSTMA070E** Invalid parameter parameter [RC = 24]
- **DMSTMA105S** Error nn writing file fn ft fm on disk [RC = 100]
<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
<th>Return Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSTMA109S</td>
<td>Virtual storage capacity exceeded [RC = 104]</td>
<td></td>
</tr>
<tr>
<td>DMSTMA110S</td>
<td>Error reading tapn(vdev) [RC = 100]</td>
<td></td>
</tr>
<tr>
<td>DMSTMA137S</td>
<td>Error nn on STATE for fn ft fm [RC = 100]</td>
<td></td>
</tr>
<tr>
<td>DMSTMA138S</td>
<td>Error nn erasing fn ft before loading tape [RC = 100]</td>
<td></td>
</tr>
<tr>
<td>DMSTMA139S</td>
<td>Tape file exceeds 9 CMS MACLIBs [RC = 104]</td>
<td></td>
</tr>
<tr>
<td>DMSTMA420E</td>
<td>NSL exit filename missing or invalid [RC = 24]</td>
<td></td>
</tr>
</tbody>
</table>
Use the TAPPDS command to create CMS disk files from tapes that are used as input to or output from the following OS utility programs:

- **IEBPTPCH** -- tape files must be the result of an IEBPTPCH punch operation from either a sequential or partitioned data set in OS. The default attributes (IEBPTPCH DCB) must have been issued:
  \[ DCB=RECFM=FA,LRECL=81,BLKSIZE=81 \]
- **IEBUPDTE** -- tape files may be blocked or unblocked and must be in the format accepted by IEBUPDTE as "control data set" (SYSIN) input with a control statement
  
  
  . / ADD...

  preceding the records to be placed in each partitioned data set member (OS) or separate CMS file (CMS)).
- **IEHMOVE** -- unloaded partitioned data sets are read.

The tape can contain OS standard labels or be unlabeled. The format of the TAPPDS command is:

<table>
<thead>
<tr>
<th>TAPPDS</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>fn[<em>] ft[</em>] fm</td>
<td>SL [labeldefid] NSL [filename ID=identifier]</td>
</tr>
<tr>
<td>PDS</td>
<td>COL1</td>
</tr>
<tr>
<td>NOPDS</td>
<td>NOCOL1</td>
</tr>
<tr>
<td>UPDATE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**where:**

- **fn**
  
  is the filename of the disk file to be created from the sequential tape file. If the tape contains members of a partitioned data set (PDS), fn must be specified as an asterisk (*); one file is created for each member with a filename the same as the member name. If NOPDS or UPDATE is specified and you do not specify fn or specify it as an asterisk (*), the default filename is TAPPDS.
The filetype of the newly created files. The default filetypes are CMSUT1 (for PDS or NOPDS) and ASSEMBLE (for UPDATE). The defaults are used if ft is omitted or specified as *.

is the mode of the disk to contain the new files. If this field is omitted or specified as an asterisk (*), A1 is assumed.

means that the tape has standard labels. The default is SL without a labeldefid. With the default specification, the standard labels are displayed at the user's terminal. If labeldefid is specified, the standard labels are not displayed, but are checked by the tape label checking routine.

means that the tape has nonstandard labels.

identifies the LABELDEF command, which supplies descriptive label information for the file to be processed. The labeldefid given here must match the 1- to 8-character specified as the filename on the LABELDEF command that was previously issued.

is the CMS filename of a routine to process nonstandard labels. The filetype must be TEXT or MODULE. If both TEXT and MODULE files exist, the MODULE file is used. MODULE files that are used for NSL routines with the TAPPDS command must be created so that they start at an address above X'21000'. This prevents the MODULE files from overlaying the command. See the section “Tape Labels in CMS” in the VM/SP CMS User's Guide for details on writing routines to process nonstandard labels.

specifies a 1- to 8-character identifier to a user-written NSL routine. You may use the identifier in any way you want to identify the file being processed. The identifier is passed to the user routine exactly as specified in the operand. If an identifier is not specified, blanks are passed. See the section “Tape Labels in CMS” in the VM/SP CMS User's Guide for details on communication with routines that process nonstandard labels.

Note: If either SL or NSL is specified for tape label processing, the fn, ft, and fm operands must all be specified. They may be specified by asterisks (*) if you want default values; however, none of the three operands may be omitted.
Options:

If conflicting options are specified, the last one entered is the one that is used. All options, except TAPn, are ignored when unloaded (IEHMOVE) PDS tapes are read.

**PDS**
indicates that the tape contains members of an OS partitioned data set, each preceded by a MEMBER NAME = name statement. The tape must have been created by the OS IEBPTPCH service program if this option is specified.

**NOPDS**
indicates that the contents of the tape will be placed in one CMS file.

**UPDATE**
indicates that the tape file is in IEBUPDTE control file format. The filename of each file is taken from the NAME = parameter in the "./ADD" record that precedes each member. (See Usage Note 2.)

**COL1**
reads data from columns 1-80. You should specify this option when you use the UPDATE option.

**NOCOL1**
reads data from columns 2-81; column 1 contains control character information. This is the format produced by the OS IEBPTPCH service program.

**TAPn**
specifies the symbolic name of the tape device. The following names are supported and represent these virtual units:

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>Virtual Address</th>
<th>Symbolic Name</th>
<th>Virtual Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAP0</td>
<td>180</td>
<td>TAP8</td>
<td>288</td>
</tr>
<tr>
<td>TAP1</td>
<td>181</td>
<td>TAP9</td>
<td>289</td>
</tr>
<tr>
<td>TAP2</td>
<td>182</td>
<td>TAPA</td>
<td>28A</td>
</tr>
<tr>
<td>TAP3</td>
<td>183</td>
<td>TAPB</td>
<td>28B</td>
</tr>
<tr>
<td>TAP4</td>
<td>184</td>
<td>TAPC</td>
<td>28C</td>
</tr>
<tr>
<td>TAP5</td>
<td>185</td>
<td>TAPD</td>
<td>28D</td>
</tr>
<tr>
<td>TAP6</td>
<td>186</td>
<td>TAPE</td>
<td>28E</td>
</tr>
<tr>
<td>TAP7</td>
<td>187</td>
<td>TAPF</td>
<td>28F</td>
</tr>
</tbody>
</table>

If not specified, TAP1 is assumed.

**END**
considers an END statement (characters 'END' in columns 2-5) a delimiter for the current member.
**NOEND**

specifies that END statements are not to be treated as member delimiters, but are to be processed as text.

**MAXTEN**

reads up to ten members. This is valid only if the PDS option is selected.

**NOMAXTEN**

reads any number of members.

**Usage Notes:**

1. You can use the TAPE command to position a tape at a particular tape file before reading it with the TAPPDS command. If the tape has OS standard labels, TAPDDS will read and display the “VOL1” and “HDR” records at the terminal. If the file you want to process is not at the beginning of the tape, the TAPE command must be used to position the tape at a particular tape file before reading it with the TAPPDS command. Be aware that each file on an OS standard label tape is actually three physical files (HDR, DATA, TRAILER). If positioning to other than the first file, the user must skip more physical tape files (3n-3 if positioning to the header labels, 3n-2 if positioning to the data file, where n is the number of the file on the tape).

2. If you use the UPDATE option, you must also specify the COL1 option. Each tape record is scanned for a “/.ADD” record beginning in column 1. When a “/.ADD” record is found, subsequent records are read onto disk until the next “/.ADD” record is encountered or until a “/.ENDUP” record is encountered.

   A “/.ENDUP” record or a tape mark ends the TAPPDS command execution; the tape is not repositioned.

   “/.label” records are not recognized by CMS and are included in the file as data records.

   If the NAME = parameter is missing on the “/.ADD” record or if it is followed by a blank, TAPDDS uses the default filename, TAPDDS, for the CMS disk file. If this happens more than once during the execution of the command, only the last unnamed member is contained in the TAPPDS file.

3. If you are reading a macro library from a tape created by the IEHMOVE utility, you can create a CMS MACLIB file directly by using the TAPEMAC command.

4. Only header labels of the first file encountered are displayed or checked if SL or SL labeldefid is specified. Trailer labels are not processed or displayed; they are skipped. When more than one file is processed by one issuance of the TAPPDS command, only the first file has its...
standard labels processed. Standard labels are skipped on succeeding files.

5. The following examples illustrate different ways in which tape labels are processed by TAPPDS. If you specify:

```
tappds fileg cmsutl * sl
```

then, before loading the PDS into fileg, CMS displays a VOL1 and HDR1 label if it exists on the tape. It does not stop before the PDS is loaded; therefore, you cannot use the tape label to suppress loading if the wrong tape has been mounted.

If you specify:

```
labeldef label2 fid pds1 volid xyz
tappds fileh cmsutl * sl label2
```

CMS uses the label information specified to check the label on the tape before loading your PDS. If there are discrepancies, the PDS is not loaded.

If you specify

```
tappds filej * * ns1 nonstd
```

CMS uses your own routine called NONSTD to process tape labels before loading the PDS.

**Responses:**

```
DMSTPD703I  File fn ft [fm] copied
```

The named file is copied to disk.

```
DMSTPD707I  Ten files copied
```

The MAXTEN option was specified and ten members have been copied.

*Note:* If the tape being read contains standard OS labels, the labels are displayed at the terminal.

**Messages and Return Codes:**

```
DMSTPD003E  Invalid option: option [RC = 24] 
DMSTPD027E  Invalid device devtype [for SYSaaa] [RC = 24] 
DMSTPD058E  End-of-file or end-of-tape [RC = 40] 
DMSTPD105S  Error nn writing file fn ft fm on disk [RC = 100] 
DMSTPD108S  Virtual storage capacity exceeded [RC = 104] 
DMSTPD110S  Error reading tapn(vdev) [RC = 100] 
DMSTPD420E  Nsl exit filename missing or invalid [RC = 24]
```
TELL

Use the TELL command to send a message to one or more computer users on your computer or on other computers that are connected to yours via the Remote Spooling Communications Subsystem (RSCS) network. These users must be logged on to receive your message.

TELL is one of several commands that references a "userid NAMES" file. By setting up a names file, you can identify recipients just by using nicknames, which are automatically converted into node and userid. For information on creating a NAMES file, see the NAMES command.

The format of the TELL command is:

```
TELL name message
```

where:

- **name** is the “name” of the computer user to whom the message is to be sent. If the same recipient is specified more than once, he receives only one message. The “name” may take any of the following forms:
  - A “nickname” that can be found in the file “userid NAMES,” where “userid” is your userid. This nickname may represent a single person (on your computer or on another computer), or a list of people. If the nickname represents a list, the message is sent to everyone on the list.
  - A userid of a user on your computer. If a name cannot be found in the “userid NAMES” file, it is assumed to be a userid of someone on your computer.
  - “userid AT node” which identifies a user (“userid”) on your computer or another computer (“node”). The “userid NAMES” file is not examined in this case.

You cannot send messages to a userid named “AT” or “CC:.”

- **message** is the text of the message that is sent.
Usage Notes:

1. If the first word of your message is "at," you must use the third form of "name" (shown above).

2. If the person to whom you are sending the message either is not logged on or is not accepting messages (by issuing CP SET MSG OFF), he will not receive the message.

3. The TELL command uses the CP MESSAGE command to send messages to users logged on to your computer. If you would like to send messages using a different CP command (specifically, MSGNOH, SMSG, or WNG) and you are an authorized user, you change the command that TELL uses with the DEFAULTS command. See the description of the DEFAULTS command in this publication. See the description of the MSGNOH, SMSG, and WARNING commands in the VM/SP CP Command Reference.

4. The TELL command uses the CP SMSG command to send messages, via RSCS, to users logged on to other computers (nodes). A warning message may occur if the SMSG command created by TELL is too long to be handled by RSCS. In this case, shorten the message text or split it into shorter messages, and then reissue the TELL command.

5. If you want to issue TELL from an EXEC program, you should precede it with the EXEC command; that is, specify

   exec tell

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSWTL399E  Too many tags or tag too long for nickname in userid NAMES file [RC = 88]
DMSWTL499E  User not authorized to issue command command [RC = 40]
DMSWTL647E  Userid not specified for nickname in userid NAMES file [RC = 32]
DMSWTL648E  Userid name not found; no message has been sent [RC = 32]
DMSWTL676E  Invalid character * for Network ID [RC = 20]
**TXTLIB**

Use the TXTLIB command to update CMS text libraries. The format of the TXTLIB command is:

```
<table>
<thead>
<tr>
<th>Command</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEN</td>
<td><code>libname fn1 [fn2...] [ (optionA [ ] ) ]</code></td>
</tr>
<tr>
<td>ADD</td>
<td><code>libname fn1 [fn2...] [ (optionA [ ] ) ]</code></td>
</tr>
<tr>
<td>DEL</td>
<td><code>libname membername1 [ membername2...]</code></td>
</tr>
<tr>
<td>MAP</td>
<td><code>libname [ (optionB[ ] ) ]</code></td>
</tr>
</tbody>
</table>
```

<table>
<thead>
<tr>
<th>OptionA:</th>
<th>OptionB:</th>
</tr>
</thead>
<tbody>
<tr>
<td>[FILENAME]</td>
<td>TERM DISK PRINT</td>
</tr>
</tbody>
</table>

*where:*

**GEN**

creates a TXTLIB on your A-disk. If a TXTLIB with the same name already exists, it is replaced.

**ADD**

adds TEXT files to the end of an existing TXTLIB on a read/write disk. No checking is done for duplicate names, entry points, or CSECTs.

**DEL**

deletes members from a TXTLIB on a read/write disk and compresses the TXTLIB to remove unused space. If more than one member exists with the same name, only the first entry is deleted.

**MAP**

lists the names (entry points) of TXTLIB members, their locations in the library, and the number of entries.

**libname**

specifies the filename of a file with a filetype of TXTLIB, which is to be created or listed or from which members are to be deleted or added.
**TXTLIB**

\[ fn1 \{fn2\ldots} \]

specifies the name(s) of file(s) with filetype(s) of TEXT, that you want to add to a TXTLIB.

\[ membername1 \{membername2\ldots} \]

specifies the name(s) of TXTLIB member(s) that you want to delete.

**Option A:**

**FILENAME**

indicates that all the filenames specified will be used as the membernames for their respective entries in the TXTLIB file instead of the first CSECT in the file’s text deck.

**Option B:**

**TERM**

displays information about the TXTLIB on your terminal.

**DISK**

writes a CMS file, named “libname MAP A5,” that contains a list of TXTLIB members.

**PRINT**

spools a copy of the TXTLIB map to the virtual printer.

**Usage Notes:**

1. The FILENAME option overrides any name card in a text file. The name card functions as before, but the specified file name becomes the membername in the TXTLIB. The name card is the only entry within that membername of the TXTLIB. If a name card is not found in the text file and you specify the FILENAME option, the file’s name is the membername. The first CSECT in the text file is the first entry point (the remaining entry points in the text file follow) within that member.

2. The FILENAME option sets the membername to the filename and makes the first CSECT the first entry point. Therefore, any text file that has 255 entry points will exceed this limit by one if you select the FILENAME option. In other words, if you select the FILENAME option, the maximum number of entry points a file may contain is 254.

3. When a TEXT file is added to a library, its membername is taken from the first CSECT name, or, if a NAME card exists, from the NAME statement in the TEXT file. All other entry points in the TEXT file become entries within this member unless there is a NAME card. In this case, the only entry created is the membername. For example, a TEXT file with a filename of TESTPROG that contains CSECTs named CHECK and RECHECK, when added to a TXTLIB, creates a member
named CHECK and an entry point named RECHECK within this member. If it contained a NAME statement at the end of the text deck, that name would be the only entry created for that text in the TXTLIB. Deletions and LOAD and INCLUDE command references must be made on the membername.

4. Members must be deleted by their initial entry in the dictionary (that is, their “name” or the first ID name). Any attempt to delete a specific alias or entry point within a member will result in a “Not found” message.

5. If you want your TXTLIBs to be searched for missing subroutines during CMS loader processing; you must identify the TXTLIB on a GLOBAL command; for example:

   global txtlib newlib

6. You may add OS linkage editor control statements NAME, ALIAS, ENTRY, and SETSSI to a TEXT file before adding it to a TXTLIB. There must be a blank in column 1 and the ALIAS, NAME, and ENTRY statements must follow the END statement. The ALIAS statement must precede the NAME statement. The specified ENTRY point must be located within the CSECT.

7. TXTLIB members are not fully link-edited, and may return erroneous entry points during dynamic loading.

8. The total number of members in the TXTLIB file cannot exceed 2000.

   When this number is reached, an error message is displayed. The total number of entry points in a member cannot exceed 255. When this number is reached, an error message is displayed and the next text file (if there is one) is processed. The text library created includes all the text files entered up to (but not including) the one that caused the overflow.

9. TERM or PRINT options will erase the old MAP file, if one exists.

10. If you delete the last remaining member of a TXTLIB, the TXTLIB is erased.

Responses:

When the TXTLIB MAP command is issued with the TERM option, the contents of the directory of the specified text library are displayed at the terminal. The number of entries in the text library (xxx) is also displayed.

Note: Alias names follow the main member and they do not have a location field.
Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

- **DMSLBT001E** No \{file|CSECT\} name specified [RC = 24]
- **DMSLBT002E** File fn \{ft \{fm\}\} not found [RC = 28]
- **DMSLBT002W** File fn \{ft \{fm\}\} not found [RC = 4 or 28]
- **DMSLBT003E** Invalid option: option [RC = 24]
- **DMSLBT013E** Member membername not found in file fn \{ft \{fm\}\} [RC = 32]
- **DMSLBT037E** Disk mode(\{vdev\}) is accessed as read/only [RC = 36]
- **DMSLBT056E** File fn \{ft \{fm\}\} contains invalid [name|alias|entry|ESD] record formats [RC = 32]
- **DMSLBT069E** Disk mode not accessed [RC = 36]
- **DMSLBT104S** Error nn reading file fn \{ft \{fm\}\} from disk [RC = 100]
- **DMSLBT105S** Error nn writing file fn \{ft \{fm\}\} on disk [RC = 100]
- **DMSLBT106S** Number of member names exceeds maximum of 2000; file fn TEXT not added [RC = 88]
- **DMSLBT213W** Library fn TXTLIB not created, or erased if empty [RC = 4]
Use the TYPE command to display all or part of a CMS file at the terminal in either EBCDIC or the hexadecimal representation of the EBCDIC code. The format of the TYPE command is:

```
Type \[(options ... \]) \]
```

**options:**

where:

- **fn**
  - is the filename of the file to be displayed.

- **ft**
  - is the filetype of the file to be displayed.

- **fm**
  - is the filemode of the file to be displayed. If this field is omitted, the A-disk and its extensions are searched to locate the file. If fm is specified as an asterisk (*), all disks are searched, and the first file found is displayed.

- **rec1**
  - is the record number of the first record to be displayed. This field cannot contain special characters. If rec1 is greater than the number of records in the file, an error message is displayed. If this field is omitted or entered as an asterisk (*), a record number of 1 is assumed.

- **recn**
  - is the record number of the last record to be displayed. This value cannot contain embedded commas. If this field is not specified, is entered as an asterisk (*), or is greater than the number of records in the file, displaying continues until end of file is reached.
Options:

COL xxxx-yyyyy
  displays only certain columns of each record. xxxx specifies the start
column and yyyyy the end column of the field within the record that is
to be displayed. The string xxxx-yyyyy may have a maximum of eight
characters; additional characters are truncated.

If columns are not specified, the entire record is displayed unless the
filetype is LISTING, in which case the first position of each record is
not displayed, since it is assumed to be a carriage control character.

HEX
  displays the file in hexadecimal format.

MEMber \{ * \ name \}
  displays member(s) of a library. If the format of the file is MACLIB or
TXTLIB, a MEMBER entry can be specified. If an asterisk (*) is
specified, all members of the library are displayed. If a name is
specified, only that particular member is displayed. The MEMBER
option should only be used with MACLIB or TXTLIB format files.
With other format files, results may be unpredictable.

Usage Notes:

1. If the HEX option is specified, each record can be displayed in its
   entirety; if not, no more than 130 characters of each record can be
displayed.

2. The length of each output line is limited to 130 characters or the
current terminal linesize (as specified by the CP TERMINAL command),
   whichever is smaller.

3. If the MEMBER option is specified more than once, only the last
   member specified will be typed. However, if one MEMBER option is
coded with an asterisk (*), and another MEMBER option is specified
   with a membername, only the member specified by membername will be
typed, regardless of their order on the command line.

For example, if you code:

\texttt{TYPE ONE MACLIB (MEMBER EXAMPLE1 MEMBER EXAMPLE2}}

only \texttt{EXAMPLE2} will be typed. If you code:

\texttt{TYPE ONE MACLIB (MEMBER EXAMPLE1 MEMBER *}}

only \texttt{EXAMPLE1} will be typed.
Responses:

The file is displayed at the terminal according to the given specifications. When you use the HEX option, each record is preceded by a header record:

```
RECORD nnnnnnnnnn LENGTH=nnnnnnnnnn
```

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSTYP002E File [fn [ft [fm]]] not found [RC = 28]
- DMSTYP003E Invalid option: option [RC = 24]
- DMSTYP005E No option specified [RC = 24]
- DMSTYP009E Column col exceeds record length [RC = 24]
- DMSTYP013E Member membername not found in library [RC = 32]
- DMSTYP029E Invalid parameter parameter in the option option field [RC = 24]
- DMSTYP033E File fn ft fm is not a library [RC = 32]
- DMSTYP039E No entries in library fn ft fm [RC = 32]
- DMSTYP049E Invalid line number nn [RC = 24]
- DMSTYP054E Incomplete fileid specified [RC = 24]
- DMSTYP062E Invalid * in fileid [RC = 20]
- DMSTYP069E Disk mode not accessed [RC = 36]
- DMSTYP104S Error nn reading file fn ft fm from disk [RC = 100]

Chapter 2. CMS Commands 667
UPDATE

UPDATE

Use the UPDATE command to modify program source files. The UPDATE command accepts a source input file and one or more files containing UPDATE control statements and updated source records; then it creates an updated source output file, an update log file indicating what changes, if any, were made, and an update record file if more than a single update file is applied to the input file.

The format of the UPDATE command is:

```
fn1 [ft1 ASSEMBLE [fm1 [fn2 [ft2 [fm2 ]]]]] [options...[] ]
```

Options:

- **REP**
- **NOREP**
- **SEQ8**
- **NOSEQ8**
- **INC**
- **NOINC**
- **CTL**
- **NOCTL**
- **OUTMODE**

```
fn1 ft1 fm1
```

is the file identifier of the source input file. The file must consist of 80-character card image records with sequence fields in positions 73 through 80 or 76 through 80. If the filetype or filemode are omitted, ASSEMBLE and A1 are assumed, respectively.

```
fn2 ft2 fm2
```

is the file identifier of the update file. If the NOCTL option is in effect, this file must contain UPDATE control statements and updated source records. The default file identifier is fn1 UPDATE A1. If the CTL option is specified, this file must be a control file that lists the update files to be applied; the default file identifier is fn1 CNTRL A1.
Options:

REP
creates an output source file with the same filename as the input source file. If the output file is placed on the same disk as the input file, the input file is erased.

NOREP
retains the old file in its original form, and assigns a different filename to the new file, consisting of a dollar sign ($) plus the first seven characters of the input filename (fn1).

SEQ8
specifies that the entire sequence field (columns 73 through 80) contains an eight-digit sequence number on every record of source input.

NOSEQ8
specifies that columns 73-75 contain a three-character label field, and that the sequence number is a five-digit value in columns 76-80.

Note: Source files sequenced by the CMS editor are sequenced, by default, with five-digit sequence numbers.

INC
increments sequence numbers in columns 73 through 80 in each record inserted into the updated output file, according to specifications in UPDATE control statements.

NOINC
puts asterisks (********) in the sequence number field of each updated record inserted from the update file.

CTL
specifies that fn2, ft2, and fm2 describe an update control file for applying multiple update files to the source input file.

Note: The CTL option implies the INC option.

NOCTL
specifies that a single update file is to be applied to the source input file.

OUTMODE fm
specifies that the files created by the UPDATE command will be written onto the fm disk. If OUTMODE is not specified, then the files are put onto the disk as outlined later in the section ‘Disk Mode of Output Files.’ The filemode number defaults to 1 if none is specified alongside the fn. This disk must be an accessed CMS read/write disk or UPDATE will terminate.
UPDATE

**STK**
stacks information from the control file in the CMS console stack. STK is valid only if the CTL option is also specified and is useful only when the UPDATE command is executed in an EXEC procedure.

**NOSTTK**
does not stack control file information in the console stack.

**TERM**
displays warning messages at the terminal whenever a sequence or update control card error is discovered. (Such warning messages appear in the update log, whether they are displayed at the terminal or not.)

**NOTERM**
suppresses the display of warning messages at the terminal. However, error messages that terminate the entire update procedure are displayed at the terminal.

**DISK**
places the update log file on disk. This file has a file identifier “fn UPDLOG,” where “fn” is the filename of the file being updated.

**PRINT**
prints the update log file directly on the virtual printer.

**STOR**
specifies that the source input file is to be read into storage and the updates performed in storage prior to placing the updated source file on disk. This option is meaningful only when used with the CTL option since the benefit of increased processing speed is realized when processing multiple updates. STOR is the default when CTL is specified.

**NOSTOR**
specifies that no updating is to take place in storage. NOSTOR is the default when single updates are being applied (CTL is omitted from the command line).

**Update Control Statements**

The UPDATE control statements let you insert, delete, and replace source records, as well as resequence the output file.

All references to the sequence field of an input record refer to the numeric data in columns 73-80 of the source record, or columns 76-80 if NOSEQ8 is specified. Leading zeros in sequence fields are not required. If no sequence numbers exist in an input file, a preliminary UPDATE with only the ‘/ S’ control statement can be used to establish file sequencing.

Sequence numbers are checked while updates are being applied; an error condition results if any sequence errors occur in the update control
statements, and warnings are issued if an error is detected in the sequencing of the input file. Any source input records with a sequence field of eight blanks are skipped, without any indication of a sequence error. Such records may be replaced or deleted only if they occur within a range of records that are being replaced or deleted entirely and if that range has limits with valid sequence numbers. There is no means provided for specifying a sequence field of blanks on an UPDATE control statement.

**Control Statement Formats:**

All UPDATE control statements are identified by the characters ".j" in columns 1 and 2 of the 80-byte record, followed by one or more blanks and additional, blank-delimited fields. Control statement data must not extend beyond column 50.

**SEQUENCE Control Statement** -- resequences the updated source output file in columns 73-80 (if SEQ8 is specified), or in columns 76-80 with the label placed in columns 73-75 (if NOSEQ8 is specified).

The format of the SEQUENCE control statement is:

```
./ S [seqstrt [seqincr [label]]]
```

**where:**

- **seqstrt**
  is a one- to eight-digit numeric field specifying the first decimal sequence number to be used. The default value is 1000 if SEQ8 is specified and 10 if NOSEQ8 is specified.

- **seqincr**
  is a one- to eight-digit numeric field specifying the decimal increment for resequencing the output file. The default is the "seqstrt" value.

- **label**
  is a three-character field to be duplicated in columns 73-75 of each source record if NOSEQ8 is specified. The default value is the first three characters of the input filename (fn1).

If you use the SEQUENCE statement, it must be the first statement in the update file. If any valid control statement precedes it, the resequence operation is suppressed.

When the sequence control statement is the first statement processed, the sequence numbers in the source file are checked and warning message DMSUPD210W is issued for any errors. If the sequence control statement is processed after updates have been applied, no warning messages will be issued.
Each source record is resequenced in columns 73-80 as it is written onto the output file, including unchanged records from the source file and records inserted from the update file.

**INSERT Control Statement** -- inserts all records following it, up to the next control statement, into the output file.

The format of the INSERT control statement is:

```
./ I seqno [$ seqstrt [seqincr]]
```

where:

- **seqno**
  - is the sequence number of the record in the source input file following which new records are to be added.

- **$**
  - is an optional delimiter indicating that the inserted records are to be sequenced by increments.

- **seqstrt**
  - is a one- to eight-digit numeric field specifying the first decimal number to be used for sequencing the inserted records.

- **seqincr**
  - is a one- to eight-digit numeric field specifying the decimal increment for sequencing the inserted records.

All records following the "./ I" statement, up to the next control statement, are inserted in the output file following the record identified by the "seqno" field. If the NOINC option is specified, each inserted record is identified with asterisks (********) in columns 73-80. If either the INC or CTL option is specified, the records are inserted unchanged in the output file, or they are sequenced according to the "seqstrt" and "seqincr" fields, if the dollar sign ($) key is specified.

The default sequence increment, if the dollar sign is included, is determined by using one tenth of the least significant, nonzero digit in the seqno field, with a maximum of 100. The default seqstrt is computed as seqno plus the default seqincr. For example, the control statement:

```
./ I 2600 $ 2610
```

causes the inserted records to be sequenced XXX02610, XXX02620, and so forth (NOSEQ8 assumed here). For the control statement:

```
./ I 240000 $
```
the defaulted seqincr is the maximum, 100, and the starting sequence number is 240100. SEQ8 is assumed, so the inserted records are sequenced 00240100, 00240200, and so forth.

If either INC or CTL is specified but the dollar sign is not included, whatever sequence number appears on the inserted records in the update file is included in the output file.

**DELETE Control Statement** -- deletes one or more records from the source file.

The format of the DELETE control statement is:

```
./ D seqno1 [seqno2] [$]
```

*where:*

*seqno1*

is the sequence number identifying the first or only record to be deleted.

*seqno2*

is the sequence number of the last record to be deleted.

$ is an optional delimiter indicating the end of the control fields.

All records of the input file, beginning at seqno1, up to and including the seqno2 record, are deleted from the output file. If the seqno2 field is omitted, only a single record is deleted.

**REPLACE Control Statement** -- replaces one or more input records with updated records from the update file.

The format of the REPLACE control statement is:

```
./ R seqno1 [seqno2] [$ [seqstrt [seqincr]]]
```

*where:*

*seqno1*

is the sequence number of the first input record to be replaced.

*seqno2*

is the sequence number of the last record to be replaced.
UPDATE

$ is an optional delimiter key indicating that the substituted records are to be sequenced incrementally.

**seqstrt**

is a one- to eight-digit numeric field specifying the first decimal number to be used for sequencing the substituted records.

**seqincr**

is a one- to eight-digit numeric field specifying the decimal increment for sequencing the substituted records.

All records of the input file, beginning with the seqnol record, up to and including the seqno2 record, are replaced in the output file by the records following the "/R" statement in the update file, up to the next control statement. As with the "/D" (delete) function, if the seqno2 field is omitted, only a single record is replaced, but it may be replaced by more than a single inserted record. The "/R" (replace) function is performed as a delete followed by an insert: thus, the number of statements inserted need not match the number deleted. The dollar sign ($), seqstrt, and seqincr processing is identical to that for the insert function.

**COMMENT Statement** -- allows inserting supplemental information that the user may want. Note that the COMMENT statement is treated as a control statement when it appears within a sequence of records to be applied in an update.

The format of the COMMENT statement is:

```
./ * [comment]
```

**where:**

* indicating that this is a comment statement and is only copied into the update log file.

**Summary of Files Used by the UPDATE Command**

The following discussion shows input and output files used by the UPDATE command for a:

- Single-level update
- Multilevel update
- Multilevel update with an auxiliary control file

**Disk Mode of Output Files:** The following steps are taken to determine the disk upon which the output files are to be placed (the search stops as soon as one of the following steps is successful):
1. If the OUTMODE option is specified, then the output files are placed on the disk specified, if it is a read/write disk. If the specified disk is accessed as read/only extension, then the following message is displayed:

\[DMSUPD037E \text{ Disk } \text{mode}[(\text{vdev})] \text{ is accessed as read/only}\]

2. If the disk on which the original source file resides is read/write, then the output files are placed on that disk.

3. If that disk is a read-only extension of a read/write disk, then the output files are placed on that particular read/write disk.

4. If neither of the other steps is successful, the output files are placed on the primary read/write disk (the A-disk).

**Single-Level Update**

\[
\begin{align*}
\text{fn ASSEMBLE} \\
\text{fn UPDATE} \\
\text{update fn}
\end{align*}
\]

\[
\begin{align*}
\text{\$fn ASSEMBLE} \\
\text{fn UPDLOG}
\end{align*}
\]

**Note:** \textit{fn ASSEMBLE} is the source input file.

\textit{fn UPDATE} contains UPDATE control statements and updated source input records.

\textit{\$fn ASSEMBLE} is the updated source file, incorporating changes, additions, and deletions specified in the update file. The output filetype is always the same as the filetype of the input file. These default filetypes and filemodes can be overridden on the command line; for example:

\texttt{update testprog cobol b fix cobol b (rep)}

results in a source file \texttt{TESTPROG COBOL B} being updated with control statements contained in the file \texttt{FIX COBOL B}. The output file replaces the existing \texttt{TESTPROG COBOL B}. \textit{fn UPDLOG} contains a record of updates applied. If you do not want this file written on disk, specify the PRINT option.
Multilevel Update

```
fn ASSEMBLE
fn CNTRL
fn UPDTABC
fn UPDTXYZ
```

update fn (ctl)

Note: `fn ASSEMBLE` is the source input file.

`fn CNTRL` is the control file that lists updates to be applied to the source file. These default filetypes and filemodes can be overridden on the command line; for example:

```
update acct pliopt a test cntrl a (ctl)
```

results in the file TEST CNTRL being used by the UPDATE command to locate the update files for ACCT PLIOPT.

`fn UPDTABC` and `fn UPDTXYZ` are update files containing UPDATE control statements and new source records. These files must have filenames that are the same as the source input file. The first four characters of the filetype must be 'UPDT'. The UPDATE command searches all accessed disks to locate the update files.

`$fn ASSEMBLE` is the updated source file, incorporating changes, additions, and deletions specified in the update files. The filetype is always the same as the filetype of the source input file.

`fn UPDLOG` contains a record of updates applied. If you do not want this file written on disk, specify the PRINT option. `fn UPDATES` summarizes the updates applied to the source file. The CONTROL FILE (fn CNTRL) may not contain UPDATE control statements. It may only list the filetypes of the files that contain UPDATE control statements. This control file contains the records:

```
TEXT MACS CMSLIB
TWO UPDTABC
ONE UPDTXYZ
```

where `UPDTABC` and `UPDTXYZ` are the filetypes of the update files. The UPDATE command applies these updates to the source file beginning with the last record in the control file. Thus, the updates in `fn UPDTXYZ` are applied before the updates in `fn UPDTABC`.

When you create update files whose filetypes begin with 'UPDT', you may omit these characters when you list the updates in the control file; thus, the CNTRL file may be written:

```
TEXT MACS CMSLIB
TWO ABC
ONE XYZ
```
**UPDATE**

**TEXT, TWO, ONE:** The first column of the control file consists of an update level identifier, which may be from one to five characters long. These identifiers are used by VM/SP updating procedures, like the VMFASMA EXEC, to locate and identify text decks produced by multilevel updates.

*MACS:* The first record in the control file must be a MACS record that contains an update level identifier (TEXT) and, optionally, lists up to 29 macro library (MACLIB) filenames (subject to the record length of the line). The length of the MACS record may not exceed 80, the record length of the control file.

UPDATE uses the information provided in the MACS card and the update level identifier only when the STK option is specified. This information is, however, required in the CNTRL file.

**Multilevel Update with Auxiliary Control File**

```
fn ASSEMBLE
fn CNTRL
fn UPDTABC
fn UPDTXYZ
fn AUXLIST
fn FIX01
fn FIX02
```

```
$fn ASSEMBLE
fn UPDLOG
fn UPDATES
```

```
update fn (ctl)
```

`fn ASSEMBLE, fn CNTRL, fn UPDTABC, fn UPDTXYZ, $fn ASSEMBLE, fn UPDLOG, and fn UPDATES` are used as described, for “Multilevel Update,” except that the CNTRL file contains:

```
TEXT MACS CMSLIB
TWO UPDTABC
ONE UPDTXYZ
TEXT AUXLIST
```

*Aux* in the filetype AUXLIST indicates that this is the filetype of an auxiliary control file that contains an additional list of updates. The first three characters of the filetype of an auxiliary control file must be “AUX”; the remaining character(s) (to a maximum of 5) may be anything. The filename must be the same as the source input file.

An auxiliary file may also be specified as:

```
xxxxx AUX
```

in the control file. For example, the record:

```
FIX TEST AUX
```

identifies the auxiliary file fn AUXTEST.

---

Chapter 2. CMS Commands 677
UPDATE

Note that if you give an auxiliary control file the filetype AUXPTF or an update level identifier of AUX, the UPDATE command assumes that it is a simple update file and does not treat it as an auxiliary file.

Preferred AUX File

A preferred AUX file may be specified. A preferred AUX file contains the version of an update that applies to your version of the source file. (There may be more than one version of the same update if there is more than one version of the source file. For example, you need one version for the source file that has a system extension licensed program installed, and you need another version for the source file that does not have a licensed program installed.)

When you specify an auxiliary control file, you can specify more than one filetype. The first filetype indicates a file that UPDATE uses only on one condition: the files that the second and subsequent filetypes indicate do not exist. If they do exist, this AUX file entry is ignored and no updating is done. The files that the second and subsequent filetypes indicate are preferred because, if they exist, UPDATE does not use the file that the first filetype indicates. For example, assume that the file 'fn ASSEMBLE' does exist. The control file MYMODS CNTRL:

TEXT MACS MYMACS CMSLIB OSMACRO
MY2 AUXTEST
MY1 AUXMINE AUXTEST

and the command:

UPDATE fn ASSEMBLE * MYMODS CNTRL (CTL

would result in UPDATE finding the preferred auxiliary control file ‘fn AUXTEST’, and therefore not using ‘fn AUXMINE’ to update ‘fn ASSEMBLE’. UPDATE would then proceed to the MY2 AUXTEST entry and update ‘fn ASSEMBLE’ with the updates listed in ‘fn AUXTEST.’ It is assumed that AUXTEST and AUXMINE list similar but mutually exclusive updates.

The search for a “preferred” auxfile will continue until one is found or until the token is an invalid filetype; that is, less than four or more than eight characters. This token and the remainder of the line are considered a comment.

fn FIX01 and fn FIX02 are update files containing UPDATE control statements and new source records to be incorporated into the input file. When update files are listed in an auxiliary control file, they can have any filetype you choose but the filename must be the same as the source input file. The update files, as well as the AUX file, may be on any accessed disk. These are indicated in fn AUXLIST as follows:

FIX02
FIX01
The updates are applied from the bottom of the auxiliary file. Thus, fn FIX01 is applied to the source file before fn FIX02. Since the auxiliary file is listed at the bottom of the control file, these updates are applied before UPDTXYZ and UPDTABC.

Additional Control File Records

In addition to the MACS record, the filetypes of update (UPDT) files, and the filetypes of auxiliary control (AUX) files, a control file may also contain:

- Comments. These records begin with an asterisk (*) in column 1. Comments are also valid in AUX files.

- PTF records. If the characters PTF appear in the update level identifier field, the UPDATE command expects the second field to contain the filetype of an update file. The filetype may be anything; the filename must be the same as the source input file.

- Update level identifiers not associated with update files.

The following example of a control file shows all the valid types of records:

```
* Example of a control file
ABC MACS MYLIB
TEXT
004 UPDTABC
003 XYZ
002 AUXLIST1
001 LIST2 AUX
PTF TESTFIX
```

The STK Option

The STK (stack) option is valid only with the CTL option and is meaningful only when the UPDATE command is invoked within an EXEC procedure.

When the STK option is specified, UPDATE stacks the following data lines in the console stack:

```
first line:  * update level identifier
second line: * library list from MACS record
```

The update level identifier is the identifier of the most recent update file that was found and applied. For example, if a control file contains

```
TEXT MACS CMSLIB OSMACRO TESTMAC
OFA UPDTOFA
PFA UPDTOFA
```

and the UPDATE command appears in an EXEC as follows:

```
UPDATE SAMPLE (CTL STK
&READ VARS &STAR &TEXT
&READ VARS &STAR &LIB1 &LIB2 &LIB3 &LIB4
```
then the variable symbols set by the &READ VARS statements have the following values if the file SAMPLE UPDTOFA is found and applied to the file SAMPLE ASSEMBLE:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;STAR</td>
<td>*</td>
</tr>
<tr>
<td>&amp;TEXT</td>
<td>OFA</td>
</tr>
<tr>
<td>&amp;LIB1</td>
<td>CMSLIB</td>
</tr>
<tr>
<td>&amp;LIB2</td>
<td>OSMACRO</td>
</tr>
<tr>
<td>&amp;LIB3</td>
<td>TESTMAC</td>
</tr>
<tr>
<td>&amp;LIB4</td>
<td>null</td>
</tr>
</tbody>
</table>

The library list may be useful to establish macro libraries in a subsequent GLOBAL command within the EXEC procedure. If no update files are found, UPDATE stacks the update level identifier on the MACS record.

**Responses:**

FILE 'fn ft fm,' REC #n = update control statement
This message is displayed when the TERM option is specified and an error is detected in an update file. It identifies the file and record number where the error is found.

DMSUPD177I Warning messages issued (severity = nn) [; REP option ignored]
Warning messages were issued during the updating process. The severity shown in the error message in the "nn" field is the highest of the return codes associated with the warning messages that were generated during the updating process.

The warning return codes have the following meanings:

RC = 4 - Sequence errors were detected in the original source file being updated.

RC = 8 - Sequence errors, which did not previously exist in the source file being updated, were introduced in the output file during the updating process.

RC = 12 - Any other warning error detected during the updating process. Such errors include invalid update file control statements and missing update or PTF files.

The severity value is passed back as the return code from the UPDATE command. In addition, if the REP option is specified in the command line, then it is ignored, and the updated source file has the fileid "$fn1 ft1," as if the REP option was not specified.

DMSUPD178I Updating fn ft fm
Applying fn ft fm
Applying fn ft fm
...
The specified update file is being applied to the source file. This message appears only if the CTL option is specified in the command line. The updating process continues.

DMSUPD304I Update processing will be done using disk. An insufficient amount of virtual storage was available to perform the updating in virtual storage, so a CMS disk must be used. This message is displayed only if NOSTOR was specified in the UPDATE command line.

DMSUPD180W Missing PTF file fn ft fm RC=12
In the event that the user receives this message during the update process, the message MISSING PTF FILE ‘fn ft fm’ will appear in the update log associated with the program being updated.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSUPD001E No filename specified [RC = 24]
DMSUPD002E File [fn [ft [fm]]) not found [RC = 28]
DMSUPD003E Invalid option: option [RC = 24]
DMSUPD007E File fn ft fm is not fixed, 80-character records [RC = 32]
DMSUPD010W Premature EOF on file fn ft fm—sequence number seqno not found [RC = 12]
DMSUPD024E File fn ft fm already exists [RC = 28]
DMSUPD048E Invalid mode mode [RC = 24]
DMSUPD065E option option specified twice [RC = 24]
DMSUPD066E option1 and option2 are conflicting options [RC = 24]
DMSUPD069E Disk mode not accessed [RC = 36]
DMSUPD070E Invalid parameter parameter [RC = 24]
DMSUPD104S Error nn reading file fn ft fm from disk [RC = 100]
DMSUPD105S Error nn writing file fn ft fm on disk [RC = 100]
DMSUPD174W Sequence error introduced in output file: seqno1 to seqno2 seqno1 to seqno2 [RC = 8]
DMSUPD176W Sequencing overflow following sequence number seqno [RC = 8]
DMSUPD179E Missing or duplicate MACS card in control file fn ft fm [RC = 32]
DMSUPD181E No update files were found [RC = 40]
DMSUPD182W Sequence increment is zero [RC = 8]
DMSUPD183E Invalid {CONTROL|AUX} file control card [RC = 32]
DMSUPD184W /S not first card in update file—ignored [RC = 12]
DMSUPD185W Invalid character in sequence field seqno [RC = 12]
DMSUPD186W Sequence number seqno not found [RC = 12]
DMSUPD187E Option STK invalid without CTL [RC = 24]
DMSUPD207W Invalid update file control card [RC = 12]
DMSUPD210W Input file sequence error: seqno1 to seqno2 [RC = 4]
DMSUPD299E Insufficient storage to complete update [RC = 41]
DMSUPD300E Insufficient storage to begin update [RC = 41]
DMSUPD361E Disk mode is not a CMS disk [RC = 36]
VALIDATE

VALIDATE

Use the VALIDATE command to verify the syntax of a file identifier (filename filetype filemode). If you specify the filemode, VALIDATE verifies whether or not the disk is accessed.

The format of the VALIDATE command is:

```
VALIDATE fn ft fm
```

where:

- **fn**
  - is the filename whose syntax is to be verified. If fn is specified as *, it is ignored.

- **ft**
  - is the filetype whose syntax is to be verified. If ft is specified as *, it is ignored.

- **fm**
  - is the filemode whose syntax is to be verified. If fm is specified, the disk will be checked for access. If fm is omitted, or specified as *, no disks are checked for access.

**Usage Notes:**

1. The filename and filetype can each be one to eight characters. The valid characters are A-Z, a-z, 0-9, $, #, @, +, - (hyphen), : (colon), and _ (underscore).

2. When you code an asterisk in the fn or ft fields, only the specified field will be verified. For example, the command:
   ```
   validate * file e
   ```
   verifies the syntax of the filetype FILE and determines if disk E is accessed.

3. You can invoke the VALIDATE command from the terminal, from an EXEC file, or as a function from a program. If VALIDATE is invoked from an EXEC file or as a function that has the message output suppressed, the messages are not issued.
4. When writing EXECs or assembler programs, you can use VALIDATE * * fm to determine if a disk is accessed. For example,

```
validate * * e
```

tells you if disk E is accessed, regardless if any files exist on the disk. If the disk is not accessed, an error message is issued.

5. To verify the syntax of a file identifier and the existence of the file on an accessed disk, use the STATE/STATEW (or ESTATE/ESTATEW) command.

Responses:

The CMS ready message indicates that the specified file identifier is valid and the filemode is an accessed disk or was specified as *.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

- DMSSTT048E Invalid mode mode [RC = 24]
- DMSSTT054E Incomplete fileid specified [RC = 24]
- DMSSTT062E Invalid character char in fileid fn ft [RC = 20]
- DMSSTT069E Disk mode not accessed [RC = 36]
- DMSSTT070E Invalid parameter parameter [RC = 24]
WAITREAD VSCREEN

Use the WAITREAD VSCREEN command from an EXEC to update the virtual screen with data in the virtual screen queue, refresh the physical screen, and wait for the next attention interrupt.

The format of the WAITREAD VSCREEN command is:

```
WAITREAD VSCREEN vname
```

where:

vname

is the name of the virtual screen that is waiting for input.

Usage Notes:

1. All windows showing virtual screens other than the virtual screen specified in the WAITREAD command are protected. The windows showing the specified virtual screen are either protected or unprotected, depending on how the data being displayed is defined in the virtual screen.

2. WAITREAD VSCREEN executes in the following manner:

   - Each virtual screen is updated with data from the virtual screen queue
   - The screen image is rebuilt based on the ordered list of windows and displayed on the physical screen
   - Wait for the next interrupt. (The virtual screen specified in the WAITREAD VSCREEN command is now active.)
   - When the interrupt is received, the screen is read
   - Information regarding the key pressed, the cursor, and modified fields are passed back to the exec in variables.

The virtual screen can then be updated with the variables by using the WRITE VSCREEN command.

The exec variables contain:
WAITREAD.0
the number of variables returned (excluding WAITREAD.0)

WAITREAD.1
the key that caused the attention interrupt (such as PAKEY n, PFKEY n, CLEAR, or ENTER)

WAITREAD.2
the word CURSOR followed by the line number and column number of the cursor in the window, followed by the word DATA or RESERVED indicating the area that the cursor was located. If the cursor was not in a DATA or RESERVED area, the line and column numbers are -1.

WAITREAD.3

WAITREAD.n
where each variable contains the word DATA or RESERVED indicating the type of text updated, followed by the line number and column number of the field that was modified, followed by the modified text.

Note: The keyword values returned in WAITREAD.0, WAITREAD.1, and WAITREAD.2, as well as the EXEC variables WAITREAD.n, are always in American English (AMENG).

3. WAITREAD VSCREEN is an important command for EXECs that read from and write to windows. A typical sequence for such an EXEC would be:

- Define a window and virtual screen (DEFINE WINDOW and DEFINE VSCREEN commands)
- Connect the window to the virtual screen (SHOW WINDOW command)
- Write data to the virtual screen (WRITE VSCREEN command)
- Issue the WAITREAD VSCREEN command

When an interrupt is received, the EXEC can process the WAITREAD.n variables and update the virtual screen using the WRITE VSCREEN command.

4. A field definition character is placed at the start of each row if:

- A window is not connected to a virtual screen at column 1, or
WAITREAD VSCREEN

- The number of columns in the window, virtual screen, and physical screen are not the same.

As a result, data is shifted one character to the right in each row so that data in column one is not lost. Lines may be truncated on the right, and the location information indicates that the window is showing one less character from the virtual screen.

5. When windows overlap on the physical screen, a field definition character is placed at each window boundary.

6. If only part of a field from a virtual screen is displayed on the physical screen and the field is modified, the entire field is returned as a modified field in a variable WAITREAD.n.

7. The same field can be modified in different windows. These modifications are processed from the top of the screen to the bottom, moving from left to right.

8. The lines in a window that are not top reserved lines, bottom reserved, or data lines from the virtual screen are called pad lines. When a window is connected to the active virtual screen, the pad lines are unprotected. If there are multiple windows connected to the same active virtual screen, only the pad lines of the top-most window are unprotected.

9. Any part of a window that does not map to the virtual screen is protected. For example, suppose you have defined a window that is 24 rows by 80 columns and a virtual screen that is 20 rows by 60 columns. When you connect the window to the virtual screen at row 1 column 1, the 4 rows and 20 columns in the window that do not map to the virtual screen are protected.

Responses:

None.

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSWIO494W</td>
<td>FULLREAD set off.</td>
</tr>
<tr>
<td>DMSWIO614E</td>
<td>Screen modifications lost. See ‘SET FULLREAD’ to use PAkeys safely.</td>
</tr>
<tr>
<td>DMSWIO629W</td>
<td>Screen modifications may be lost. Press ENTER key to process screen changes.</td>
</tr>
<tr>
<td>DMSWIR329W</td>
<td>Warning: APL/TEXT option not in effect</td>
</tr>
<tr>
<td>DMSWIR696W</td>
<td>Invalid data received from the display</td>
</tr>
<tr>
<td>DMSWRD386E</td>
<td>Missing operand(s) [RC = 24]</td>
</tr>
<tr>
<td>DMSWRD388E</td>
<td>Invalid keyword: keyword [RC = 24]</td>
</tr>
</tbody>
</table>
Chapter 2. CMS Commands

WAITREAD VSCREEN

DMSWRD391E  Unexpected operand(s): operand  [RC = 24]
DMSWRD622E  Insufficient free storage  [RC = 104]
DMSWRD631E  WAITREAD can only be executed from an EXEC-2 or
             REXX EXEC  [RC = 4]
DMSWRD917E  No windows are showing virtual screen vname  [RC = 4]
DMSWRD921E  Virtual screen vname is not defined  [RC = 28]
DMSWRD925E  I/O error on screen  [RC = 100]
DMSWRD926E  Command is only valid on a display terminal  [RC = 88]
DMSWRD928E  Command is not valid for virtual screen vname  [RC = 12]
WAITT VSCREEN

Use the WAITT VSCREEN command to update a virtual screen with data.

The format of the WAITT VSCREEN command is:

```
WAITT VSCREEN [ vname [*] ]
```

**where:**

- **vname**
  - is the name of the virtual screen to be updated. An * indicates that all the virtual screens are updated. An * is the default.

**Usage Notes:**

1. WAITT VSCREEN updates the virtual screen with any data that has been written to it. The data is moved from the queue to the virtual screen data buffer. The following also may occur:

   - If logging is requested, the data is appended to the virtual screen log file.
   - If NOTYPE is specified (or is in effect for the virtual screen), then the data is logged (if logging was requested), and then discarded.

2. Queuing is handled so that applications need not be concerned with the virtual screen size and the location of the data in the virtual screen buffers. When the screen is refreshed, the queued writes are moved into the virtual screen with one field per line of output.

   When the queue becomes full, writing stops until a window connected to a virtual screen is cleared or scrolled. As you scroll forward, the oldest lines that have been scrolled are deleted, new lines are appended at the bottom, and the lines are renumbered. (Lines that have not been scrolled are not deleted.)

3. When the virtual screen is updated, any windows defined as POP showing the virtual screen are popped to the top of the ordered list of windows.
Responses:

None.

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

DMSWAT386E  Missing operand(s) [RC = 24]
DMSWAT388E  Invalid keyword: keyword [RC = 24]
DMSWAT391E  Unexpected operand(s): operand [RC = 24]
DMSWAT921E  Virtual screen vname is not defined [RC = 28]
DMSWAT928E  Command is not valid for virtual screen vname [RC = 12]

Messages while logging and/or updating virtual screen data to disk:

DMSWVT037E  Disk mode is read only [RC = 12]
DMSSTT062E  Invalid character char in fileid fn ft fm
DMSWVT069E  Disk mode not accessed
DMSWVT105S  Error nn writing file fn ft fm on disk
DMSWVT109S  Virtual storage capacity exceeded
DMSWVT531E  Disk is full; set new filemode or clear some disk space [RC = 13]
DMSWVT924E  Data was truncated
DMSWVT933W  Logging stopped for virtual screen vname
DMSWVT934E  Text was not written to virtual screen. No field was defined.

Return Codes:

0  Normal execution
13  Virtual screen full
WRITE VSCREEN

Use the WRITE VSCREEN command to enter information in a virtual screen. Information is queued to a virtual screen and is displayed the next time the screen is refreshed.

The WRITE VSCREEN command provides many possibilities for defining fields, writing data, and updating the plane buffers associated with the virtual screen (color, extended highlighting, and Programmed Symbol Sets).

The format of the WRITE VSCREEN command is:

```
WRITE VSCREEN vname line col length [ ( [REServed] [optionA] [optionB] [optionC] [optionD] ) ]
```

**OptionA:**
- BLANKs
- NULls

**OptionB:**
- PROtect
- NOPROtect
- High
- NOHigh
- Invisible

**OptionC:**
- color
- exthi
- psset

**OptionD:**
- FIELD
- DATA
- COLOR
- EXTHI
- PSS

```
text
```

*Note: If option D is used, a right parenthesis should not be used to mark the end of the options.*

**where:**

- **vname**
  - is the name of the virtual screen.

- **line**
  - is the line number of the virtual screen where the write is to begin.

- **col**
  - is the column number of the virtual screen where the write is to begin.

- **length**
  - is the length to be written.
**WRITE VSCREEN**

REServed
indicates that the line number refers to a reserved line. If you specify RESERVED, line must be a number less than or equal to the size of the reserved area. A negative line number indicates a write in the bottom reserved area, and a positive line number indicates a write in the top reserved area.

**Option A:**

Option A is used for padding text in the data buffer. The following may be specified:

BLAnks
pad data with blanks

NULls
pad data with nulls (default)

**Option B:**

Option B is the attribute for defining a field. The following may be specified:

PROtect
protected field

NOPRotect
not protected field

High
high intensity field

NOHigh
normal intensity field

Invisible
invisible field. The data written is not displayed on the screen.

**Option C:**

Option C is the extended attribute for updating the virtual screen buffers and/or for padding the text being written. If option C is not specified, then the default characteristics of the virtual screen are used, if necessary (see the DEFINE VSCREEN command). The following may be specified:

color
is the color of the field. It may be DEFault, Blue, Red, Pink, Green, Turquoise, Yellow, or White.
**WRITE VSCREEN**

*exthi*

is the extended highlighting of the field. It may be None, REVvideo, BLInk, or Underline.

*psset*

is the Programmed Symbol Set of the field. It may be PS0, PS1, PSA, PSB, PSC, PSD, PSE, or PSF. The psset must already be loaded in the display. If not, no psset is used.

**Option D:**

Option D is the buffer for the text. The following may be specified:

**FIELD**

creates a field. All the buffers will be initialized with either the virtual screen defaults, or the settings specified in options A and/or B and/or C. Text is placed in the field left justified, and is padded or truncated to the field length.

**DATA**

indicates that the text is to be written to the data buffer of the virtual screen. The text is written for the length specified or until the next field is encountered.

**COLOR**

indicates that the text is composed of color codes that are to be written to the color buffer. The text is written for the length specified or until the next field is encountered. Use the following characters to represent the colors:

1 Blue
2 Red
3 Pink
4 Green
5 Turquoise
6 Yellow
7 White
0 Defaults to the color of the field

**EXTHI**

indicates that the text is composed of extended highlighting codes that are to be written to the extended highlight buffer. The text is written for the length specified or until the next field is encountered. The following characters represent the extended highlighting:

0 Defaults to the extended highlight of the field
1 Blink
2 Revvideo
4 Underline
PSS
indicates that the text is composed of Programmed Symbol set codes that are to be written to the PSset buffer. The text is written for the length specified or until the next field is encountered. The following characters represent the PSS:

0 (defaults to the PSS of the field) 1 A B C D E F

Note: The letters must be specified in upper case.

text
is the text to be written.

Usage Notes:

1. When WRITE VSCREEN is issued, the text and its characteristics are held in the virtual screen queue created by DEFINE VSCREEN. To move the text from the queue to the virtual screen, you should issue either WAITREAD VSCREEN, WAITT VSCREEN, or the REFRESH command. This process is required when issuing the WRITE VSCREEN command from an EXEC.

2. Use the INVISIBLE option to prevent data such as passwords from being displayed on the screen.

3. When a field is defined, the first character contains the start field. The start field is a one-byte character identifying the attributes for the field. The start field character is protected and cannot be written to. If option D is not specified, the default is FIELD.

For more information on fields, refer to the IBM 3270 Information Display System Data Stream Programmer's Reference, GA23-0059.

4. For color and extended highlighting in a DBCS string, the first byte of a double-byte character determines the attributes for both bytes. You cannot specify character attributes for PSS within a DBCS string.

5. The column operand must be greater than or equal to zero. Specifying a column number of zero is valid only when writing to the scrollable area, in which case it is equivalent to specifying column number one. When writing a field, a start field is placed in the column specified. The text always begins in the next column. When writing COLOR, EXTHI, PSS, or DATA, the text begins in the column specified, or in the next available column after the start field(s).

6. Specifying a line number of zero is valid only when writing to the scrollable area. When specifying a line number of zero and writing a field, a field is created at the line past the current bottom of the virtual screen. This provides a means for writing sequentially to the virtual screen. When doing sequential writes, a field always fills an entire line. The length of the field is determined as follows (assume a virtual screen with 80 columns):
WRITE VSSCREEN

- If the length specified is zero, as in the example:
  
  ```
  write vscreen cms 0 0 0 (field Enter your name:
  ```

  A field is created at the line past the current bottom of the CMS virtual screen. A start field is placed in column 1 and the text begins in column 2. All the text is written and the length of the field is set to 80.

- If length specified is less than the number of columns in the virtual screen, as in the example:
  
  ```
  write vscreen cms 0 0 10
  ```

  A field is defined at one line past the current bottom of the virtual screen. Even though the length was specified as 10, the length of the field is set to 80 in order to fill the entire line.

- If length specified is greater than the number of columns in the virtual screen, as in the example:
  
  ```
  write vscreen cms 0 0 100
  ```

  A field is defined at one line past the current bottom of the virtual screen. In this case, the length of the field is set to 160 and the field fills two lines.

7. When using Option D, SET CHARMODE must be ON to update COLOR, EXTHI, and PSS. If SET CHARMODE is OFF they are ignored. Switching from SET CHARMODE ON to OFF may produce some undesirable results, such as a field having attributes that you intended only for a character.

8. When specifying a line number of zero and writing COLOR, EXTHI, PSS, or DATA, you are acting on the "current field." The current field is the most recent field written to the scrollable data area of the virtual screen. The column number specifies the position in the current field. For example:
  
  ```
  write vscreen cms 0 5 0 (COLOR 11116666777
  ```

  will update the color buffer starting at the fifth position of the current field.

9. If the length specified is less than the length of the text, then the text is truncated. If the length specified is greater than the length of the text, then the text is padded with the characteristics specified in option A and/or option B and/or option C. If the options are not specified, then text is padded with the characteristics defined for the virtual screen.

Note that when you are not writing a field, the text is written for the length specified or until the next field is encountered.

For example:
write vscreen cms 1 1 20 (blank field Enter your name:
write vscreen cms 0 0 20 (red color 11111
writes “Enter your name: .” The “Enter” is displayed blue and “your
name:” is displayed red.

The example:
write vscreen cms 1 1 10 (data Enter your name:
writes “Enter your.”

10. When defining a virtual screen, the top reserved area is defined as one
continuous field and the bottom reserved area is defined as one
continuous field. However, the scrollable area is not defined as a field.
To write to the scrollable data area, you must define a field.

11. In a virtual screen, the lines in the top reserved area are numbered
starting from the top. The top line is 1, the second line is 2, etc. In the
bottom reserved area, lines are numbered starting at the bottom and
have negative values. The bottom line is -1, the second line up is -2, the
third line up is -3, etc.

12. Once a field is defined, you cannot change the attributes (PROtect,
NOPROtect, High, NOHigh) without redefining the field. However, you
can change the extended attributes (color, exthi, psset) or data of the
field. For example, suppose you define the following field:
write vscreen cms 5 20 0 (field Enter your name:
To change the color extended attribute, you can enter:
write vscreen cms 5 20 0 (COLOR 2222244445555
As a result, beginning at line 5, column 21 (because there is a start field
character in column 20), “Enter” will be red, “your” will be green, and
“name:” will be turquoise.

13. When writing sequentially (see Usage Note 6) and the text contains a
character assigned to X’15’ (end-of-line, or EOL) via the SET INPUT
command, that character indicates a line end. The text following it is
continued on the next line. A new line is written for each EOL.

Responses:

None.
## Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
<th>Return Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSWRT622E</td>
<td>Insufficient free storage</td>
<td>RC = 104</td>
</tr>
<tr>
<td>DMSWRT921E</td>
<td>Virtual screen <code>vname</code> is not defined</td>
<td>RC = 28</td>
</tr>
<tr>
<td>DMSWRT923E</td>
<td>Specified location is outside the virtual screen</td>
<td>RC = 32</td>
</tr>
<tr>
<td>DMSWRT928E</td>
<td>Command is not valid for virtual screen <code>vname</code></td>
<td>RC = 12</td>
</tr>
</tbody>
</table>
XEDIT

Use the XEDIT command to invoke the VM/SP System Product editor to create, modify, and manipulate CMS disk files. Once the VM/SP System Product editor has been invoked, you may execute XEDIT subcommands and use the System Product Interpreter or EXEC 2 macro facility.

You can return control to the CMS environment by issuing the XEDIT subcommand FILE, FFILE, QUIT, or QQUIT.

For complete details on XEDIT subcommands and macros, see the publication VM/SP System Product Editor Command and Macro Reference.

The format of the XEDIT command is:

<table>
<thead>
<tr>
<th>Xedit</th>
<th>[fn [ft [fm]]] [options...]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Options:</td>
<td>[WINdow wname]  [Width nn]  [NOSCreen]</td>
</tr>
<tr>
<td>[PROFile macroname] [NOPROFil] [NOCLear]</td>
<td></td>
</tr>
<tr>
<td>[NOMsg] MEMber membername</td>
<td></td>
</tr>
</tbody>
</table>

Options Valid Only in Update Mode:

| [Update]  [Seq8]  [Ctl fn1] |
| NOUpdate  NOSeq8  NOCtl |
| [Merge]  [UNtil filetype]  [Incr nn] |
| SIDcode string |

where:

* fn ft are the filename and filetype of the file to be edited. If they are not specified here, they must be provided in the LOAD subcommand as part of the profile.

* fm is the filemode of the file to be edited, indicating the disk on which the file resides. The editor determines the filemode of the edited file as follows:

  * Editing existing files
When the filemode is specified, that disk and its extensions are searched. If the filemode is not specified or is specified as an asterisk (*), all accessed disks are searched for the specified file.

- Creating new files

If the filemode is not specified, the editor assumes a filemode of A1.

Options:

**WINdow wname**

is the name of the window and virtual screen that XEDIT uses to display the file being edited. By default, XEDIT uses the window and virtual screen named “XEDIT.” For more information, refer to the VM/SP System Product Editor Command and Macro Reference.

**Width nn**

defines the amount of virtual storage used to contain one line of the file. If the value specified is too small, certain file lines may be truncated.

If not specified here, WIDTH may be defined in the LOAD subcommand, as a part of the profile. If WIDTH is not specified in either the XEDIT command or the LOAD subcommand, the default is the larger of the following:

- The logical record length (LRECL) of the file
- The default logical record length associated with the filetype

**NOSCreen**

forces a 3270 display terminal into line (typewriter) mode.

**PROFile macroname**

If the specified macro exists on one of the accessed disks, the editor executes it as the first subcommand.

If the specified macro is not found on an accessed CMS disk, an error message is displayed.

If this option is not specified but a macro with a macro name of PROFILE exists, the editor executes it.

In all cases, the profile macro must have a filetype of XEDIT.

**NOPROFIL**

forces the editor not to execute the default PROFILE macro.
**NOClear**
specifies that the screen is not cleared when the editor gets control. Instead, the screen is placed in a MORE... (waiting) status. Any messages remain on the screen until the CLEAR key is pressed. This option is useful when the XEDIT command is issued from a macro that displays messages.

**NOMsg**
enters a file with a default of SET MSGMODE OFF.

**MEMber membername**
specifies the name of a member to be edited specified in ‘fn ft fm’ macro library. If MEMBER is specified, ft must be MACLIB. If the member does not exist in that library, a new file is created with a fileid of ‘membername MEMBER fm’.

The following options are meaningful only if the VM/SP System Product editor is to be used in update mode:

**Update**
The editor searches all accessed CMS disks for a file with a filename of fn and a filetype of UPDATE. If the file exists, the editor applies the update statements before displaying the file to be edited. Each new modification made by the user is added to the existing UPDATE file. The original source file is not modified.

If the file does not exist, the editor creates a new UPDATE file to contain modifications made by the user.

**NOUpdate**
specifies that the editor is to apply no update statements (even if UPDATE is specified in the LOAD subcommand in the profile).

**Seq8**
specifies that the entire sequence field (columns 73-80) contains an eight-digit sequence number in every record of the file to be edited. The SEQ8 option automatically forces the UPDATE option. SEQ8 is the default value.

**NOSeq8**
specifies that columns 73-75 contain a three-character label field, and that the sequence number is a five-digit number in columns 76-80.

The NOSEQ8 option forces the UPDATE option.

**Ctl fn1**
specifies that “fn1 CNTRL” is an update control file that controls the application of multiple update files to the file to be edited. (See the CMS UPDATE command description for more information.)

This option automatically forces the UPDATE and SEQ8 options.
NOCtl
specifies that the editor is not to use the control (CTL) file (even if it is specified in the LOAD subcommand in the profile).

Merge
specifies that all the updates made through the control file and all the changes made while editing will be written into the file whose name is defined by the latest update level (that is, the most recently applied UPDATE file in a control file). This option forces the UPDATE option.

UNTil filetype
specifies the filetype of the last update to be applied to the file. Changes are applied to the file being edited from all filetypes in the control file, up to and including the filetype specified in the UNTIL option.

Filetypes of update files listed in the control file or of update files listed in an auxiliary control file can be specified with the UNTIL option. AUX filetypes (AUXxxxxx) cannot be specified with the UNTIL option.

The UNTIL option forces the UPDATE option.

Incr nn
When inserting new lines in an update file, the editor automatically computes the serialization; the INCR option forces a minimum increment between two adjacent lines. If not specified, the minimum increment is one (1). This option forces the UPDATE option.

SIDcode string
specifies a string that the editor inserts in every line of an update file whether the update file is an existing file or if it is being created. The editor inserts the specified string in columns 64-71 and pads on the right with blanks, if necessary. Any data in columns 64-71 is overlaid. This option forces the UPDATE option.

Usage Notes:

1. For the CTL, INCR, MEMBER, PROFILE, SIDCODE, UNTIL, and WIDTH options, the operand must be specified; otherwise, the next option will be interpreted as the operand. For example, in the "PROFILE macroname" option, "macroname" must be specified; if it is not, the next option will be interpreted as the operand macro name.

2. Once the XEDIT command has been executed, the XEDIT subcommand can be used to edit and display multiple files simultaneously. (See the XEDIT subcommand description in the publication VM/SP System Product Editor Command and Macro Reference).
3. You can also call the editor recursively (using “CMS XEDIT...,” for example). This ability is particularly useful when applications are developed using the editor and its macro facilities to interface with the user, for example, HELP.

4. The editor is kept in virtual storage as part of the CMS nucleus; the CMS user area is unused. As a result, assuming a large enough virtual machine, any CMS or CP command may be issued directly from the editor environment itself (if a SET IMPCMSCP subcommand is in effect).

5. When the PROFILE macro is invoked by an XEDIT command, everything following the command name XEDIT is tokenized and then assigned to the argument string that is passed to the PROFILE macro. The editor does not examine any parameters that follow a closing right parenthesis on the XEDIT command.

6. When you issue an XEDIT command for a variable-format file, trailing blanks are removed when the file is filed (or saved).

7. Comment control records are deleted from an update file whenever an update file is applied to the original source file during an editing session, and a FILE or SAVE subcommand is issued.

8. XEDIT invokes the CMS storage initialization routine, STRINIT, when invoked as a command and the extended PLIST is available. This routine frees any storage from a previous program, starting at the beginning of the user area.

XEDIT is invoked as a command under the following situations:

- issued at CMS command level
- issued from an EXEC 2 or a System Product Interpreter EXEC when prefixed by the command, CMDCALL.
- issued from a System Product Interpreter EXEC as ADDRESS CMS. For System Product Interpreter EXECs, ADDRESS CMS is the default.

If you have issued CMS SET RELPAGE ON, STRINIT causes CMS storage initialization to release the remaining pages of storage.

Note: If a program running in the user area calls XEDIT as a command (high-order byte of register 1 = x'OB'), the user area storage pointers will be reset. This reset condition could cause errors upon return to the original program.

9. Many languages have more characters than can be displayed using one-byte codes (KANJI, for example). A Double-Byte Character Set (DBCS) is used to represent those characters. The double-byte characters may appear in a sentence with characters from other languages that are displayed in one-byte codes. Files containing
double-byte characters are handled differently than files that only contain one-byte characters. Special considerations for editing files that contain double-byte characters are described in the VM/SP System Product Editor Command and Macro Reference.

10. The MEMBER option and the NOUPDATE option have no effect when preceded by an option that automatically forces update processing. Likewise, options that normally force update processing are ignored when preceded by the MEMBER option or the NOUPDATE option.

Responses:

The following messages are displayed only if you are using the VM/SP System Product editor in update mode:

DMSXUP178I Updating fn ft fm
   Applying fn ft fm
   Applying fn ft fm
   ...
DMSXUP180W Missing PTF file fn ft fm

Messages and Return Codes:

Additional error messages for specifying a command format incorrectly are listed on page 24.

- DMSXDS590E Dataset too large [RC=88]
- DMSXIN002E File fn ft fm not found [RC=28]
- DMSXIN003E Invalid option: option [RC=24]
- DMSXIN024E File fn ft fm already exists [RC=28]
- DMSXIN029E Invalid parameter parameter in the option option field [RC=24]
- DMSXIN054E Incomplete fileid specified [RC=24]
- DMSXIN065E option option specified twice [RC=24]
- DMSXIN066E option1 and option2 are conflicting options [RC=24]
- DMSXIN070E Invalid parameter parameter [RC=24]
- DMSXIN104S Error nn reading file fn ft fm from disk [RC=100]
- DMSXIN132S File fn ft fm too large [RC=88]
- DMSXIN500E Unable to unpack file fn ft fm [RC=88]
- DMSXIN928E Command is not valid for virtual screen vname [RC=12]
- DMSXSU048E Invalid mode mode [RC=24]
- DMSXSU062E Invalid character in fileid fn ft fm [RC=20]
- DMSXSU069E Disk mode not accessed [RC=36]
- DMSXSU229E Unsupported OS dataset, error nn [RC=nn]
- DMSXWS915E Maximum number of windows already defined [RC=13]
- DMSXWS927E The virtual screen must contain at least 5 lines and 20 columns. [RC=24]
Messages with MEMBER option:

DMSXMB007E  File fn ft fm is not fixed, 80-character records [RC = 32]
DMSXMB033E  File fn ft fm is not a library [RC = 32]
DMSXMB039E  No entries in library fn ft fm [RC = 32]
DMSXMB167S  Previous MACLIB function not finished [RC = 88]
DMSXMB622E  Insufficient free storage for reading map [RC = 104]

Messages with UPDATE options:

DMSXUP007E  File fn ft fm is not fixed, 80-character records [RC = 32]
DMSXUP174W  Sequence error introduced in output file: seqno1 to seqno2 [RC = 8]
DMSXUP179E  Missing or duplicate MACS card in control file fn ft fm [RC = 32]
DMSXUP183E  Invalid {CONTROL/AUX} file control card [RC = 32]
DMSXUP184W  /S not first card in update file—ignored [RC = 12]
DMSXUP185W  Non numeric character in sequence field seqno [RC = 12]
DMSXUP186W  Sequence number [seqno] not found [RC = 12]
DMSXUP207W  Invalid update file control card [RC = 12]
DMSXUP210W  Input file sequence error: seqno1 to seqno2 [RC = 4]
DMSXUP570W  Update ft specified in the UNTIL option field not found
DMSXUP597E  Unable to merge updates containing ./S cards [RC = 32]

Return Codes:

0     Normal
6     Subcommand rejected in the profile due to LOAD error or QUIT subcommand has been issued in macro called from the last file in the ring.
20    Invalid character in filename or filetype
24    Invalid parameters, or options
28    Source file not found (UPDATE MODE), library not found (MEMBER option) or file XEDTEMP CMSUT1 already exists
32    Error during updating process; file is not a library, library has no entries, or file is not fixed 80-char. records
36    Corresponding disk not accessed
88    File is too large and does not fit into storage or previous MACLIB function was not finished or Maclib limit exceeded
100   Error reading the file into storage
104   Insufficient storage available to read library map
Use the XMITMSG command (TRANSMIT MESSAGE) to retrieve a message from a CMS message repository file or your own message repository file. You supply a message identifier and substitution information, and XMITMSG gets the message that you requested. XMITMSG can be used in a REXX, EXEC 2, or CMS environment.

The format of the XMITMSG command is:

```
XMITMSG msgnumber [sublist] [(options... [)])]
```

**Options:**
- `[FORmat nn]`  
- `[LlINE nn]`  
- `[LETter a]`
- `[APPLID applid]`  
- `[CALLER name]`  
- `[VAR]`
- `[COMPress]`  
- `[NOCOMPress]`
- `[HEAder]`  
- `[NOHEADer]`
- `[DISPlay]`  
- `[NODISPlay]`
- `[ERRMSG]`
- `[SYSLANG]`

**where:**

```markdown
**msgnumber**
```

is a one to four digit number used to locate its associated message text in the repository.

```markdown
**sublist**
```

specifies the substitutions to be done on the message.

Any numeric substitution is assumed to be a dictionary substitution, and the substitution is retrieved from the repository. If the substitution is in either single or double quotes, then it is assumed to be a literal substitution. Literal substitutions must not contain blanks or parentheses. Any other substitution in the list is assumed to be a variable name, and the value of the substitution is retrieved from the exec. If the value cannot be retrieved, then the substitution is assumed to be null.

A maximum of 20 substitutions is allowed.
Options:

**FORmat** *nn*

specifies a two-digit format number. It is used to identify different versions of the same message which have the same message number. The numbering of formats is from 01 to 99. The default is 01. A format of 00 is not allowed.

**LINE** *nn*

specifies a two-digit line number that identifies each line of a multi-line message. The numbering of lines is from 01 to 99. You may also specify an asterisk for the line number; this specifies that all lines for a certain message number and format are to be retrieved. The default line number is asterisk. A line number of 00 is not allowed. Each line may be up to 240 characters long.

**LETter** *a*

specifies the severity letter of the message. Message severity is provided already within the message repository module; this parameter is used only when you want to override the provided severity.

**APPLID** *applid*

specifies the name of the application from which the message is issued. The application id is displayed in the message header.

**CALLER** *name*

overrides the default CALLER name that will go in the message header.

For execs, the default CALLER name is either:

1. the first three characters of the exec name, if these characters are different from the application id, or else
2. the next three characters of the exec name.

If you issue XMITMSG from the CMS command line, the default CALLER name is “???”.

**VAR**

copies the message into variables and returns these variables to the exec. **VAR** is only valid when issued from an exec. The complete message is copied into the variable ‘MESSAGE’, with the first line in ‘MESSAGE.1’, the second in ‘MESSAGE.2’, etc..

The number of lines in the message is copied into ‘MESSAGE.0’. If you do not specify NOHEADer, then the message header, i.e. DMSxxxxnnn or DMSxxxnnnn, is also part of the copied information. The length of the message is not returned to the exec.

If the message is to be displayed by the message facility and returned to the exec, then you must specify DISP. Otherwise, the default is NODISP when you specify VAR.
XMITMSG

**COMPress**
removes multiple blanks in the message text, including those preceding and following a substitution field. This is the default for blank compression.

**NOCOMPress**
does not remove multiple blanks in the message text, including those preceding and following a substitution field.

**HEADer**
specifies that the message header is created and displayed with the message, or returned in MESSAGE.nn with the message text if VAR was specified.

**NOHEADer**
Specifies that the message header is not displayed on the terminal, or that it is not returned in MESSAGE.nn with the message text if VAR was specified. You may not specify this option with the ERRMSG option.

**DISPlay**
specifies that the message is displayed on the terminal, regardless of the CP EMSG setting. This is the default option unless you specify VAR.

**NODISPlay**
specifies that the message is not displayed on the terminal, regardless of the CP EMSG setting. This is the default option when you specify VAR.

**ERRMSG**
specifies that the message line is displayed according to the CP EMSG setting. If EMSG is set to:

- **ON** - the entire message is displayed, header plus text
- **OFF** - messages are not displayed
- **TEXT** - only the text portion is displayed
- **CODE** - only the header is displayed

The message header consists of the following:

```
xxxxmmnnns  or  xxxmmmmnnns
```

*where:*

- **xxx** is the application id ("DMS" for CMS)
- **mmm** is the CALLER name
- **nnn** or **nnnn** is the message number
- **s** is the severity code
Code | Message   
---|---------
E  | Error    
I  | Information    
R  | Response    
S  | Severe    
T  | Terminal    
W  | Warning    

You cannot specify ERRMSG with the NOHEADER option.

**SYSLANG**

specifies that the system default language is to be used to issue the message, regardless of the language you are currently using. You may only specify this option when the application id is DMS.

**Usage Notes:**

1. To learn how to create your own message repository refer to the *VM/SP CMS for System Programming*.

2. You should have a copy of the message repository you want to access -- that way you can see the message numbers, formats, lines, and substitution positions.

3. You can use XMITMSG from CMS to display a repository message on your screen; this is useful when you want to verify the content of a repository.

   *Note:* You cannot use the VAR option when invoking XMITMSG from CMS.

4. To issue messages from assembler programs, see the APPLMSG macro in the *VM/SP CMS Macros and Functions Reference*

5. If you are writing an editor macro and you want messages to be issued by the macro and not by XMITMSG, use the VAR and ERRMSG options. To prevent the message from being displayed in CMS rather than the editing environment, use the SET CMSTYPE HT command before the XMITMSG command, followed by SET CMSTYPE RT.
**Examples:**

Assume the message repository for the application MYAPPL1 (applid = MYA) contains these messages:

```
08750101E Attempt to divide by &1 is invalid
08750201E Attempt to &2 by &1 is invalid
08760101E Error &1. rc = &3.
08770101E This is a multi-line message. NOCOMP must be specified
08770102E to keep the return codes lined up on the next line.
08770103E RC 1 = &1. RC 2 = &2.
```

and these dictionary items:

```
90250101 divide
90260101 reading from &2
90270101 tape
```

Following is an example of a REXX EXEC called DIVIDE that displays error messages when it attempts to divide by zero:

```
/* Example using XMITMSG in a REXX EXEC */
ARG DIVR
TEN = 10
IF DIVR = 0 THEN
  DO
    ANS = TEN / DIVR
    EXIT
  END
ELSE
  DO
    ------ issue error message; see cases below ------
  END
```

**Case One:**

This call accesses the repository to display MYA message 875, format 1, with '0' (the value in DIVR) being the substitution:

```
'XMITMSG 875 DIVR (DISP FOR 1 APPLID MYA COMP')
```

Here is what is displayed:

```
DMSDIV875E Attempt to divide by 0 is invalid
```

*Note:* The variable DIVR must be included *inside* the quotes.

**Case Two:**

This call uses a dictionary item 9025 as a second substitution in MYA message 875, format 2:
'XMITMSG 875 DIV 9025 (DISP FOR 2 APPLID MYA COMP')

The same message is displayed as in Case One:

DMSDIV875E Attempt to divide by 0 is invalid

Case Three:

This call illustrates the use of the VAR option. Message 877 in MYA is accessed with two substitution values, 16 and RC2. Blanks are NOT compressed, so spacing is preserved.

RC2 = 8
'XMITMSG 877 "16" RC2 (APPLID MYA NOCOMP VAR'

The message is not yet displayed; instead, each line of the message is placed in a variable, 'MESSAGE.n'. The LINE parameter defaults to '*', meaning all lines of the message go into variables.

This code in the REXX program:

```
DO I = 1 TO MESSAGE.O  /* MESSAGE.O = the number of message lines */
   SAY MESSAGE.I
END
```

displays the message:

DMSDIV877E This is a multi-line message. NOCOMP must be specified to keep the return codes lined up on the next line.
RC 1 = 16.
RC 2 = 8.

Case Four:

This call again shows the use of the VAR option on MYA message 877, but only line 2 of the message is accessed:

'XMITMSG 877 (APPLID MYA NOCOMP VAR LINE 2'

This line in the REXX program:

```
SAY MESSAGE.1
```

then displays the following message line:

DMSDIV877E to keep the return codes lined up on the next line.
XMITMSG

Messages and Return Codes:

The possible error messages for specifying a command format incorrectly are listed on page 24.

DMSMGX065E  option option specified twice  [RC = 24]
DMSMGX066E  option1 and option2 are conflicting options  [RC = 24]
DMSMGX080E  Invalid numtype number  [RC = 24]
DMSMGX109S  Virtual storage capacity exceeded  [RC = 104]
DMSMGX405E  Invalid or missing message number  [RC = 24]
DMSMGX408E  Number of substitutions exceeds 20  [RC = 24]
DMSMGX631E  XMITMSG must be invoked from an EXEC 2 or REXX EXEC or as a CMS command  [RC = 24]
Border Commands

Border commands are single-character commands for windowing that you may enter in the corners of the border.

The commands are processed as soon as they are entered.

The border commands are:

- **B** Scrolls the window backward
- **C** Clears the window of scrollable data
- **D** Drops the window
- **F** Scrolls the window forward
- **H** Hides the window
- **L** Scrolls the window to the left
- **M** Changes the location of the window
- **N** Minimizes the window
- **O** Restores the window
- **P** Pops the window
- **R** Scrolls the window to the right
- **S** Changes the size of the window
- **X** Maximizes the window

General Usage Notes:

1. Use the SET BORDER command to control the border around a window. The borders are ON by default.

2. The border commands can be placed in any of the four corners of the window border. For example:
are valid ways to enter border commands. You can type multiple border commands in several windows before pressing the ENTER key.

3. When multiple commands are entered, the order of execution is from left-to-right and top-to-bottom on the physical screen regardless of the order of the windows. For example:

executes command B first, then command F, then command R, then command P, and finally command X.

**General Error Messages:**

DMSWBX931E Invalid border command: *character*

For a general discussion and examples of how to use border commands, see the *VM/SP CMS User's Guide*.
B

Use the B border command to scroll the window backward.

The format of the B command is:

```
   B
```

Usage Notes:

1. If the window has been cleared (see the C Border command), the B command scrolls you to the bottom of the virtual screen.

2. If the window is positioned in the middle of the virtual screen and you scroll backward using the B command which would result in scrolling past the top, the window is repositioned at the virtual screen top and stops. If the window is positioned at the virtual screen top and you scroll backward, the window is repositioned at the virtual screen bottom and stops.

Error Messages:

None.
BORDER COMMANDS

C

Use the C border command to clear the window of all scrollable data.

The format of the C command is:

```
C
```

Usage Notes:

1. If the window you want to clear is variable size, the window is not displayed when the screen is refreshed.

2. The C command has no effect if the window is displaying a virtual screen which was defined with less than two data lines. In addition, in a System Product editor session, you cannot clear the window that the System Product editor is using.

Error Messages:

None.
D

Use the D border command to place a window beneath all other windows.

The format of the D command is:

| D |

Usage Notes:

When using full-screen CMS, you cannot drop a window that is showing the virtual screen indicated in the status area message. For example, if the CMS window is showing the CMS virtual screen, and the status area message instructs you to “Scroll for more information in vscreen CMS,” you cannot drop the CMS window. You can drop any other window.

Error Messages:

None.
**BORDER COMMANDS**

<table>
<thead>
<tr>
<th><strong>F</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the F border command to scroll the window forward.</td>
</tr>
<tr>
<td>The format of the F command is:</td>
</tr>
</tbody>
</table>

| F |

**Usage Notes:**

1. If the window has been cleared, the F command scrolls the window to the top of the virtual screen.

2. If the window is positioned in the middle of the virtual screen and you scroll forward using the F command which results in scrolling past the bottom, the window is repositioned at the virtual screen bottom and stops. If the window is positioned at the virtual screen bottom and you scroll forward, the window is cleared. A subsequent scroll forward positions the window at the virtual screen top. If the window being scrolled is variable size and it is positioned at the bottom of the virtual screen, the window is not displayed at the next refresh when it is scrolled forward (see the CLEAR WINDOW command).

**Error Messages:**

None.
Usage Notes:

None.

Error Messages:

None.
**BORDER COMMANDS**

<table>
<thead>
<tr>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the L border command to scroll the window left two-thirds the size of the window or to the left edge of the virtual screen to which the window is connected.</td>
</tr>
</tbody>
</table>

The format of the L command is:

```
  L
```

**Usage Note:**

The window is scrolled up to a maximum of two-thirds the width of the window. If you try to scroll beyond column 1 of the virtual screen, the window is placed in column 1 of the virtual screen.

**Error Messages:**

None.
Use the M border command to move the window corner where the command was typed to the location of the cursor when the interrupt occurred.

The format of the M command is:

```
M
```

Usage Note:

The window's size and location must be such that, excluding borders, the entire window fits on the physical screen.

Error Messages:

DMSWBX922E  Window does not fit entirely on the screen
**BORDER COMMANDS**

**N**

Use the N border command to minimize the window.

The format of the N command is:

```
N
```

**Usage Note:**

The O command returns the window to its size and location prior to the minimize.

**Error Messages:**

None.
Use the O border command to restore a maximized or minimized window to its original size and location.

The format of the O command is:

```
O
```

**Usage Notes:**

None.

**Error Messages:**

None.
BORDER COMMANDS

P

Use the P border command to place a window on top of all other windows.

The format of the P command is:

```
P
```

Usage Notes:

None.

Error Messages:

None.
**R**

Use the R border command to scroll the window right two-thirds the size of the window or to the right edge of the virtual screen to which the window is connected.

The format of the R command is:

```
R
```

**Usage Note:**

The window is scrolled up to a maximum of two-thirds the width of the window. If you try to scroll beyond the last column of the virtual screen, the window is scrolled to the last column of the virtual screen.

**Error Messages:**

None.
BORDER COMMANDS

S

Use the S border command to change the size of the window based on the position of the cursor. The corner where the command was typed is moved to the location of the cursor when the interrupt occurred.

The format of the S command is:

```
S
```

Usage Note:

The window's size and location must be such that, excluding borders, the entire window fits on the physical screen.

Error Messages:

- DMSWBX922E  Window does not fit entirely on the screen
- DMSWBX930E  Cursor is not in a valid location
**BORDER COMMANDS**

<table>
<thead>
<tr>
<th>X</th>
</tr>
</thead>
</table>

Use the X border command to maximize a window.

The format of the X command is:

```
X
```

**Usage Notes:**

1. The O command returns the window to its size and location prior to the maximize.

2. A maximized window is positioned at line 1, column 1 of the physical screen.

3. A variable size window that is maximized still retains its variable size properties. Thus, depending on how many lines exist in the virtual screen to which the window is connected, the window may appear to be less than full screen size when it is displayed on the physical screen.

   For example, if a variable size window is connected to line 1 of a virtual screen which contains three data lines, when it is maximized:

   - it moves to line 1, column 1; or line -1, column 1, of the physical screen;
   - its width is the size of the physical screen; and,
   - it contains only three data lines.

4. When you maximize a window so that it fills the entire screen and covers all other windows, you may not be able to enter commands. The WM window is automatically displayed, and the WMPF keys and command line are available to manipulate the window. Use the RESTORE WINDOW command to restore the window and the DROP WINDOW command to exit the WM environment.

**Error Messages:**

None.
Immediate Commands

You can issue an Immediate command from the terminal only after causing an attention interruption by pressing the Attention key (or its equivalent, such as the Enter key). These commands are processed as soon as they are entered. The HT and RT Immediate commands are also recognized when they are stacked in an EXEC procedure, and the HT Immediate command can be appended to a CMS command preceded by a logical line end symbol (#). Any program execution in progress is suspended until the Immediate command is processed.

None of the Immediate commands issue responses.

The Immediate commands are:

- HB - Halt batch execution
- HI - Halt interpretation
- HO - Halt tracing
- HT - Halt typing
- HX - Halt execution
- RO - Resume tracing
- RT - Resume typing
- SO - Suspend tracing
- TE - Trace end
- TS - Trace start

You can define your own Immediate commands by using any of the following:

- the IMMCMD macro in an assembler language program.
- the IMMCMD command within an EXEC (CMS EXEC, EXEC 2, System Product Interpreter).
- NUCXLOAD command with the IMMCMD option specified.

For a general discussion on Immediate commands, see the VM/SP CMS User's Guide.
HB

Use the HB (Halt batch execution) command to stop the execution of a CMS batch virtual machine at the end of the current job.

The format of the HB command is:

| HB |

Usage Notes:

1. If the batch virtual machine is running disconnected, it must be reconnected.

2. When the HB command is executed, CMS sets a flag such that at the end of the current job, the batch processor generates accounting information for the current job and then logs off the CMS batch virtual machine.
IMMEDIATE COMMANDS

HI

Use the HI (Halt Interpretation) command to cause all currently executing System Product Interpreter or EXEC 2 programs or macros to terminate execution without destroying the environment (as HX would).

The format of the HI command is:

```
HI
```
HO

Use the HO (Halt tracing) command during the execution of a command or one of your programs to stop the recording of trace information. Program execution continues to its normal completion, and all recorded trace information is spooled to the printer.

The format of the HO command is:

| HO |   |
IMMEDIATE COMMANDS

HT

Use the HT (Halt typing) command to suppress all terminal output generated by any CMS command or your program that is currently executing.

The format of the HT command is:

```
HT
```

Usage Notes:

1. Program execution continues. When the ready message is displayed, normal terminal output resumes. Use the RT command to restore typing or displaying.

2. CMS error messages having a suffix letter of S or T cannot be suppressed.

3. When using full-screen CMS, HT purges any nonpriority output that is in the queue for the virtual screen to which message class CMS is routed. (For more information on message class routing, see the ROUTE command). All priority output is displayed, such as messages and warnings through IUCV, ECHO output, and output from applications that use the LINEWRIT macro coded with PRIOR = YES.
HX

Use the HX (Halt execution) command to stop the execution of any CMS or CMS/DOS command or program, close any open files or I/O devices, and return to the CMS command environment.

The format of the HX command is:

| HX |

Usage Notes:

1. HX clears all file definitions made via the FILEDEF or DLBL commands, including those entered with the PERM option.

2. The HX command is executed when the next SVC or I/O interruption occurs: therefore a delay may occur between keying HX and the return to CMS. All terminal output generated before HX is processed is displayed before the command is executed.

3. HX does not clear user storage.
IMMEDIATE COMMANDS

RO

Use the RO (Resume tracing) command, during the execution of a command or one of your programs, to resume the recording of trace information that was temporarily suspended by the SO command. Program execution continues to its normal completion, and all recorded trace information is spooled to the printer.

The format of the RO command is:

<table>
<thead>
<tr>
<th>RO</th>
<th></th>
</tr>
</thead>
</table>

732 VM/SP CMS Command Reference
RT

Use the RT (Resume typing) command to restore terminal output from an executing CMS command or one of your programs that was previously suppressed by the HT command.

The format of the RT command is:

| RT |

Usage Note:

Program execution continues, and displaying continues from the current point of execution in the program. Any terminal output that is generated after the HT command is issued and up to the time the RT command is issued is lost. Execution continues to normal program completion.
SO

Use the SO (Suspend tracing) command during the execution of a command or one of your programs to temporarily suspend the recording of trace information. Program execution continues to its normal completion and all recorded trace information is spooled to the printer.

The format of the SO command is:

```
SO
```

Usage Note:

To resume tracing, issue the RO command.
TE

Use the TE (Trace End) command to stop all tracing of your System Product Interpreter or EXEC 2 program or macro.

The format of the TE command is:

```
TE
```
Use the TS (Trace Start) command to start tracing of your System Product Interpreter or EXEC 2 program or macro.

The format of the TS command is:

```
TS
```
This part describes the formats, operands, and defaults of the HELP facility format words. In each of the format word descriptions, the default values are those that are implied when you enter a format word with no operands or parameters. For example, the default operand of the .FO (FORMAT MODE) format word is “on”. Therefore, the format lines

```
.fo
.fo on
```

are equivalent, and in the format box of the .FO format word, the “on” operand is underscored.

HELP format words are used in HELP files when you want HELP to do output formatting when the file is processed. Figure 36 gives a summary of the HELP facility format words. All format words, except .CM, .CS, and .MT, are used expressly with the HELPCONV command. The HELPCONV module converts the specified file into a formatted HELP file, leaving the .CM, .CS, and .MT control words in the file. The output file has the filetype changed from ‘HELPxxxx’ to ‘$HLPxxxx’.
**HELP and HELP CONV Format Words**

<table>
<thead>
<tr>
<th>FORMAT WORD</th>
<th>OPERAND FORMAT</th>
<th>FUNCTION</th>
<th>BREAK</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>.BX (BOX)</td>
<td>V1 V2...Vn OFF</td>
<td>Draws horizontal and vertical lines around subsequent output text in blank columns.</td>
<td>Yes</td>
<td>Draws a horizontal line.</td>
</tr>
<tr>
<td>.CM (COMMENT)</td>
<td>Comments</td>
<td>Places comments in a file for future reference.</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>.CS (CONDITIONAL SECTION)</td>
<td>n/keyword ON/OFF</td>
<td>Allows conditional inclusion of input in the formatted output.</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>.FO (FORMAT MODE)</td>
<td>ON/OFF</td>
<td>Causes concatenation of input lines, and left and right justification of output.</td>
<td>Yes</td>
<td>On</td>
</tr>
<tr>
<td>.IL (INDENT LINE)</td>
<td>n</td>
<td>+n</td>
<td>-n</td>
<td>Indents only the next line the specified number of spaces.</td>
</tr>
<tr>
<td>.IN (INDENT)</td>
<td>n</td>
<td>+n</td>
<td>-n</td>
<td>Specifies the number of spaces subsequent text is to be indented.</td>
</tr>
<tr>
<td>.MT (MENU TYPE)</td>
<td>component</td>
<td>Correlates a component to a menu file when the component is not to be derived from the filename. For files other than menu files, .MT is seen as a comment.</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>.OF (OFFSET)</td>
<td>n</td>
<td>+n</td>
<td>-n</td>
<td>Provides a technique for indenting all but the first line of a section.</td>
</tr>
<tr>
<td>.SP (SPACE)</td>
<td>n</td>
<td>Specifies the number of blank lines to be inserted before the next output line.</td>
<td>Yes</td>
<td>1</td>
</tr>
<tr>
<td>.TR (TRANSLATE)</td>
<td>s t</td>
<td>Specifies the final output representation of any input character.</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Figure 36. HELP and HELP CONV Format Word Summary
.BX (BOX)

The BOX format word defines and initializes a horizontal rule for output and defines vertical rules for subsequent output lines.

The format of the .BX format word is:

```
.BX  [ v1 OFF v2 ] [... vn ]
```

where:

- **v1-vn**
  - are the positions at which you want to place vertical rules in output text. This format of the format word initializes the box and draws a horizontal line with vertical descenders at the columns indicated. Subsequently entering the .BX format word with no operands causes HELPCONV to create a horizontal line with vertical bars at the columns indicated. The maximum value that may be entered for operands v1-vn is 239.

- **Off**
  - causes HELP to finish drawing the box by printing a horizontal line with vertical ascenders at the columns specified in a previous .BX format word.

1. The .BX format word describes an overlay structure for subsequent text that is processed by HELPCONV. After the '.BX v1 v2 ... ' line is processed, HELPCONV continues processing output lines as usual. However, before a line is printed, HELPCONV places vertical bars in the columns indicated by v1, v2, and so on, unless a column is already occupied by a data character. In this case, HELPCONV does not place a vertical bar in the column.

2. The .BX control word causes a break in the text.

3. The terminal output characters for boxes are formed with dashes (-), vertical bars (|), and plus signs (+).

4. You can specify a .BX format word with different columns while a box is being drawn. When this happens, HELPCONV puts in vertical ascenders for all the old columns and vertical descenders for all the new columns. The vertical rules then appear in all subsequent output lines in the new columns designated.
HELP and HELPCONV Format Words

5. The column specification for the .BX format word uses a different rule than is used elsewhere in HELPCONV. In some control words the numbers in the format word represent not columns but displacements. For example the HELPCONV format word .IN 5 means that a blank character should be expanded to enough blanks to fill up through column 5; the next word starts in column 6. In the .BX control word, .BX 5 means to put vertical rules in column 5. Thus, you can use the same numbers for a .IN control word as for a .BX control word, and the vertical bar will appear in the column immediately preceding the first word on that line.

Example:

Consider the HELP file called 'MARYHADA' that looks like this:

```
.fo off
.bx 1 43
.in 5
Mary had a little lamb,
Whose fleece was white as snow,
And everywhere that Mary went,
The lamb was sure to go.
.bx off
```

This file, when processed by HELPCONV, creates the following output:

```
Mary had a little lamb,
Whose fleece was white as snow,
And everywhere that Mary went,
The lamb was sure to go.
```
.CM (COMMENT)

Use the COMMENT format word to place comments within a HELP file.

The format of the .CM format word is:

<table>
<thead>
<tr>
<th>.CM</th>
<th>comments</th>
</tr>
</thead>
</table>

*where:*

*comments*

may be anything; this input line is not used in formatting the output.

**Usage Notes:**

1. The .CM format word enables you to store comments in the HELP files for future reference. Comment lines are retained in the formatted file and do not appear when the HELP file is displayed.

2. You can use comments to store unique identifications to be used to locate a specific region of the file during editing.

3. This format word acts as a break.

**Example:**

`.CM Remember to change the date.`

The line above is seen only when editing the HELP file, and it reminds you to change the date used in the text.
HELP and HELPCONV Format Words

.CS (CONDITIONAL SECTION)

The CONDITIONAL SECTION format word identifies to HELP the section of the input file that is to be displayed based on the specified HELP command option.

The format of the .CS format word is:

```
.CS [n keyword] [ON OFF]
```

where:

- \( n \) is a number from 0 to 7, which corresponds to the conditional section of the HELP file.
- \( \text{keyword} \) is the name of the conditional section. It may be:
  - BRIEF
  - DESCRIB
  - FORMAT
  - PARMS
  - OPTIONS
  - NOTES
  - ERRORS
  - RELATED
  These names correspond to the conditional section code number.

- \( \text{on} \) marks the beginning of conditional section \( n \).
- \( \text{off} \) marks the end of conditional section \( n \).

Usage Notes:

1. The .CS format word enables you to identify the specific sections of the input file that are directly associated with the HELP facility command “options.”

   You can then specify which section(s) of the HELP file are to be displayed by using the HELP command options: BRIEF, DESCRIB, FORMAT, PARMS, OPTIONS, NOTES, ERRORS, and RELATED.
HELP and HELPConv Format Words

If you choose to implement any HELP command files using the HELP command options, either the format word `.CS n ON` or the `.CS keyword ON` is required in the file. You must use the following form:

Top of file

`.CS 0 on` (or `.CS BRIEF on`)  
(Text for BRIEF option)
`.CS 0 off` (or `.CS BRIEF off`)  
`.CS 1 on` (or `.CS DESCRIPT on`)  
(Text for DESCRIPT option)
`.CS 1 off` (or `.CS DESCRIPT off`)  
`.CS 2 on` (or `.CS FORMAT on`)  
(Text for FORMAT option)
`.CS 2 off` (or `.CS FORMAT off`)  
`.CS 3 on` (or `.CS PARMS on`)  
(Text for PARMS option)
`.CS 3 off` (or `.CS PARMS off`)  
`.CS 4 on` (or `.CS OPTIONS on`)  
(Text for OPTIONS option)
`.CS 4 off` (or `.CS OPTIONS off`)  
`.CS 5 on` (or `.CS NOTES on`)  
(Text for NOTES option)
`.CS 5 off` (or `.CS NOTES off`)  
`.CS 6 on` (or `.CS ERRORS on`)  
(Text for ERRORS option)
`.CS 6 off` (or `.CS ERRORS off`)  
`.CS 7 on` (or `.CS RELATED on`)  
(Text for RELATED option)
`.CS 7 off` (or `.CS RELATED off`)  

End of file

2. This format word acts as a break.

3. If blank lines or portions of the file are written between the conditional sections (.CS sections), these lines are considered uncontrolled data and will be displayed with the DETAIL information.

Notes:

- Few RELATED HELP files exist, but you can use this option to customize your own HELP files.
- Abbreviations of conditional keywords are not allowed.
HELP and HELPCONV Format Words

.FO (FORMAT MODE)

Use the FORMAT MODE format word to cancel or restore concatenation of input lines and right-justification of output lines.

The format of the .FO format word is:

```
.FO [ON
      OFF]
```

*where:*

ON
restores default HELPCONV formatting, including both justification and concatenation of lines. If you use the .FO format word with no operands, ON is assumed.

OFF
cancels concatenation of input lines and justification of output lines. Subsequent text is printed “as is.”

Usage Notes:

1. When format mode is in effect, lines are formed by shifting words to or from the next line (concatenation) and padding with extra blanks to produce an aligned right margin (justification).

2. This format word acts as a break.

3. When format mode is in effect, a line without any blanks that exceeds the current line length is extended into the right margin. If a line is processed so that only one word fits on the line, the word is left-justified.

4. If no formatting is done by HELPCONV, HELP description files must contain a “.fo off” format word as the first line of the file.

5. HELP MENU/TASK files must contain a “.fo off” format word as the first line of the file if the HELPCONV command is to be used. For files with RELATED sections, the “.fo off” format line must precede the RELATED section if the HELPCONV command is to be used.

6. If the HELP files are used on releases of CMS prior to VM/SP Release 4, add a “.fo off” format word as the first line in the file. You should
also change any ".CS" lines that contain keywords or section numbers not supported in prior releases to comment lines (.CM).

Examples:

1. .FO OFF

   Justification and concatenation are completed for the preceding line or lines, but the following lines are typed exactly as they appear in the file.

2. .FO

   Justification and formatting are resumed with the next input line. Output from this point on in the file is padded to produce an aligned right margin on the output page.
HELP and HELPCONV Format Words

.IL (INDENT LINE)

Use the INDENT LINE format word to indent the next line only a specified number of characters.

The format of the .IL format word is:

```
 .IL  [  n   ]
    [ + n ]
    [ - n]
```

where:

n specifies the number of character spaces to shift the next line from the current margin. +n specifies that text is shifted to the right, and -n shifts text to the left.

Usage Notes:

1. The .IL format word provides a way to indent the next output line. The line is shifted to the right or the left of the current margin (which includes any indent or offset values in effect).

2. This format word acts as a break.

3. The .IL format word is useful for beginning new paragraphs.

4. When successive .IL format words are encountered without intervening text, or when you specify positive or negative increments for .IL format words entered without intervening text, the indent amount is modified to reflect the last .IL encountered; that is, the increments are added together. Thus the lines:

```
 .il 4  
 .il +6
```

result in the next line being indented 10 spaces.

5. When you use the .IL format word with a negative value (undenting), an error message is generated if the resulting amount would cause a shift to the left of character position one.
HELP and HELPCONV Format Words

.IN (INDENT)

Use the INDENT format word to change the left margin displacement of HELP output.

The format of the .IN format word is:

```
.IN
  [ n ]
  [+n]
  [-n]
  [ 0 ]
```

where:

\( n \)

specifies the number of spaces to be indented. If omitted, 0 is assumed, and indentation reverts to the left margin. If you use \(+n\) or \(-n\), the current left margin increases or decreases by the amount specified.

Usage Notes:

1. The .IN format word resets the current left margin. This indentation remains in effect for all following lines until another .IN format word is encountered. "in 0" cancels the indentation, and output continues at the original left margin setting.

2. The value of \( n \) represents the number of blank spaces left before text margins. Thus, "in 5" sets the left margin at column 6, leaving 5 blank spaces at the left.

3. This format word acts as a break.

4. The .IN format word cancels any .OF (OFFSET) setting. The "of 0" request cancels the current offset, but leaves the left margin specified by the .IN format word unchanged.
.MT (MENU TYPE)

Use the MENU TYPE format word to specify the component of a menu. The format of the .MT format word is:

```
.MT component
```

where:

`component` is the component used by the menu.

Usage Notes:

1. The .MT format word is used to assist in the creation of menus that are a subset of another menu.
2. This format word acts as a break.

Example:

A menu that contains a list of REXX functions might be called FUNCTION HELP MENU. In this case, the HELP files for the individual functions can only be located if they are duplicated under the filetype "HELPFUNC." The .MT control word defines a component id to override that derived from the filename. The FUNCTION menu could be:

```
.MT REXX
```

to specify that the menu contains a list of REXX commands and thus will be found under the filetype "HELPREXX."
HELP and HELP_CONV Format Words

.OF (OFFSET)

Use the OFFSET format word to indent all but the first line of a block of text.

The format of the .OF format word is:

```
.OF  [  n  ]
   [  +n  ]
   [  -n  ]
   [  0  ]
```

*where:*

\( n \)

specifies the number of spaces to be indented after the next line is formatted. If omitted, 0 is assumed, and indentation reverts to the original margin setting. If you use \( +n \) or \( -n \), the current offset value increases or decreases the specified amount, and a new offset is started.

**Usage Notes:**

1. The .OF format word does not take effect until after the next line is formatted. The indentation remains in effect until a .IN (INDENT) format word or another OFFSET control word is encountered.

   You can use the .OF format word within a section that is also indented with the .IN format word. Note that .IN settings take precedence over .OF, however, and any .IN request causes a previous offset to be cleared.

   If you want to start a new section with the same offset as the previous section, you need only repeat the .OF \( n \) request.

2. This format word acts as a break.

3. You can use the .IL (INDENT LINE) format word to shift only the next line to the left or right of the current margin.
HELP and HELPCONV Format Words

Example:

1. Starting an offset:

   .of 10
   The line immediately following the .OF format word is
   printed at the current left margin. All lines
   thereafter (until the next indent or offset
   request) are indented ten spaces from the
   current margin setting. These two examples
   were processed with OFFSET control words in
   the positions shown.

2. Ending an offset:

   .of

   The effect of any previous .OF request is canceled, and all output after
   the next line continues at the current left margin setting.
Use the SPACE LINES format word when you want blank lines to appear between text lines of output.

The format of the .SP format word is:

\[
.SP \quad \begin{bmatrix} n \\ 1 \end{bmatrix}
\]

where:

\( n \)

specifies the number of blank lines to be inserted in the output. If omitted, 1 is assumed.
.TR (TRANSLATE CHARACTER)

The TRANSLATE CHARACTER format word allows you to specify the output representation of each character in the source text. For example, you could specify that all exclamation points in the file appear as blanks in the output.

The format of the .TR format word is:

```
.TR [s t ]
```

where:

- `s` is a source character under consideration. It may be a single character or a two-character hexadecimal code.

- `t` is the intended output representation of the source character. It may be a single character or a two-character hexadecimal code.

Usage Notes:

1. After formatting of an input source line has been completed and immediately before actual output, each character of the output line may be translated to a different output code.

2. Since format words are only processed internally, they are never translated in the file.

3. Translate character specifications remain in effect until explicitly respecified.

4. A .TR format word with no operands causes the translation table to be reinitialized and all previously specified translations to be reset.

5. The .TR format word does not cause a break. If you have a section of text that has translation characters in effect, followed by a .TR to reset the translations, the last line of the text may not yet have been printed. In this case, that last line is not translated.
HELP and HELPCONV Format Words

Example:

.tr 40 ?

This causes all blanks in the file to be typed as question marks (?) on output.
Chapter 4. DEBUG Subcommands

This part describes the subcommands that are available to you when you use the debug environment to test and debug your programs. The debug environment is entered when:

- The DEBUG command is issued from the CMS environment. (The DEBUG command is described in "Chapter 2. CMS Commands.")
- An external interruption occurs. (An external interruption is caused by the CP EXTERNAL command.)
- A breakpoint (instruction address stop) is encountered during program execution. (Breakpoints are set with the DEBUG subcommand BREAK.)

When the debug environment is entered, the contents of all general registers, the channel status word (CSW), and the channel address word (CAW) are saved so they may be examined and changed before being restored when leaving the debug environment. If debug is entered via an interruption, the old program status word (PSW) for that interruption is also saved. If DEBUG is the first command entered after an abnormal termination (abend) occurs, the contents of all general registers, the CSW, the CAW, and the old PSW are available from the time of the abend.

For hints on debugging your programs using the CMS debug environment, consult the VM/SP CMS User's Guide and the VM Diagnosis Guide.
DEBUG Subcommands

BREAK

Use the BREAK subcommand to stop execution of a program or module at a specific instruction location called a breakpoint.

The format of the BREAK subcommand is:

```
BReak  id  [symbol]  [hexloc]
```

where:

id
is a decimal number, from 0 to 15, which identifies the breakpoint. A maximum of 16 breakpoints may be in effect at one time; if you specify an identification number that is already set for a breakpoint, the previous breakpoint is cleared and the new one is set.

symbol
is a name assigned to the storage location where the breakpoint is set. The operand symbol, if used, must have previously been set using the DEFINE subcommand.

hexloc
is the hexadecimal storage location (relative to the current origin) where the breakpoint is to occur. hexloc must be on a halfword boundary and its value added to the current origin must not exceed your virtual machine size.

Usage Notes:

1. To set breakpoints before beginning program execution, enter the debug environment with the DEBUG command after you load the program into storage. After setting the breakpoints, use the RETURN subcommand to leave the debug environment and issue the START command to begin program execution. For example:

```
load myprog
debug
break 1 20016
break 2 20032
return
start
```
2. When you assign hexloc to a breakpoint, you must know the current origin (set with the ORIGIN subcommand). The hexloc you specify is added to the current origin to determine the breakpoint address.

3. When a breakpoint is found during program execution, the message:

SYSDBG728I DEBUG ENTERED BREAKPOINT yy AT xxxxxx

is displayed at the terminal. To resume program execution, use the GO subcommand.

4. Breakpoints are cleared after they are encountered. Thus, if a breakpoint is encountered during a program loop you must reset the breakpoint if you want to interrupt execution the next time that address is encountered.

5. When you set a breakpoint, the halfword at the address specified is replaced with B2Ex, where x represents the identification number you assigned. After the breakpoint is encountered during execution, B2Ex is replaced with the original operation code.

6. You should set breakpoints only at valid operation code addresses. The BREAK subcommand does not check to see whether or not the specified location contains a valid operation code.

7. If you reference a virtual storage address that is in a shared segment, you are given a nonshared copy of the segment and you receive the message:

SYSTEM sysname REPLACED WITH NON-SHARED copy

8. If you have set a BREAK point at a vector instruction, the contents of the vector status register may not represent what your program will see. This happens because the first time the vector instruction is executed, the vector environment has been disabled and a vector operation exception occurs. CMS enables the vector environment, clears the vector status register, and then reissues the instruction.

To determine whether CMS will clear the vector status register when it encounters this instruction, display control register zero. If the vector control bit is on (CR0.14, 00020000 bit), CMS has already determined that this is a vector application and the clearing will not occur.

Responses:

None.
CAW

Use the CAW subcommand to display at the terminal the contents of the CAW (channel address word) as it existed at the time the debug environment was entered.

The format of the CAW subcommand is:

```
CAW
```

Usage Notes:

1. Issue the CAW subcommand to check that the command address field contains a valid CCW address, or to find the address of the current CCW so you can examine it.

2. The three low-order bits of the command address field must be zeros in order for the CCW to be on a doubleword boundary. If the CCW is not on a doubleword boundary or if the command address specifies a location protected from fetching or outside the storage of a particular user, the Start I/O instruction causes the status portion of the CSW (channel status word) to be stored with the program check or protection check bit on. In this event, the I/O operation is not initiated.

Responses:

The CAW, located at storage location X'48', is displayed. Its format is:

```
<table>
<thead>
<tr>
<th>KEY</th>
<th>0000</th>
<th>Command Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3 4</td>
<td>7 8</td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

<table>
<thead>
<tr>
<th>Bits</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-3</td>
<td>The protection key for all commands associated with Start I/O. The protection key in the CAW is compared to a key in storage whenever a reference is made to storage.</td>
</tr>
<tr>
<td>4-7</td>
<td>This field is not used and must contain binary zeros.</td>
</tr>
</tbody>
</table>
8-31  The command address field contains the storage address (in hexadecimal representation) of the first CCW (channel command word) associated with the next or most recent Start I/O.
HIGHLY CONFIDENTIAL

DEBUG Subcommands

CSW

Use the CSW subcommand to display at the terminal the contents of the CSW (channel status word), as it existed at the time the debug environment was entered.

The format of the CSW subcommand is:

```
CSW
```

Usage Notes:

1. The CSW indicates the status of the channel or an input/output device, or the conditions under which an I/O operation terminated. The CSW is formed in the channel and stored in storage location X'40' when an I/O interruption occurs. If I/O interruptions are suppressed, the CSW is stored when the next Start I/O, Test I/O, or Halt I/O instruction is executed.

2. Whenever an I/O operation abnormally terminates, issue the CSW subcommand. The status and residual count information in the CSW is very useful in debugging. Also, use the CSW to calculate the address of the last executed CCW (subtract eight bytes from the command address to find the address of the last CCW executed).

Responses:

The contents of the CSW are displayed at the terminal in hexadecimal representation. Its format is:

```
KEY 0000 Command Address Status Byte Count
    0  3  4  7  8          31  32  47  48  63
```

Bits Contents

0-3 The protection key is moved to the CSW from the CAW. It shows the protection key at the time the I/O operation started. The contents of this field are not affected by programming errors detected by the channel or by the condition causing termination of the operation.
This field is not used and must contain binary zeros.

The command address contains a storage address (in hexadecimal representation) that is eight bytes greater than the address of the last CCW executed.

The status bits indicate the conditions in the device or channel that caused the CSW to be stored.

The residual count is the difference between the number of bytes specified in the last executed CCW and the number of bytes that were actually transferred. When an input operation is terminated, the difference between the original count in the CCW and the residual count in the CSW is equal to the number of bytes transferred to storage; on an output operation, the difference is equal to the number of bytes transferred to the I/O device.
Use the DEFINE subcommand to assign a symbolic name to a specific storage address. Once a symbolic name is assigned to a storage address, that symbolic name can be used to refer to that address in any of the other DEBUG subcommands.

The format of the DEFINE subcommand is:

```
DEFINE  symbol  hexloc  [bytecount 4]
```

**where:**

**symbol**

is the name to be assigned to the storage address derived from the second operand, hexloc. Symbol may be from one to eight characters long, and must contain at least one nonhexadecimal character. Any symbolic name longer than eight characters is left-justified and truncated on the right after the eighth character.

**hexloc**

is the hexadecimal storage location, in relation to the current origin, to which the name specified in the first operand (symbol), is assigned.

**bytecount**

is a decimal number, between 1 and 56 inclusive, which specifies the length in bytes of the field whose name is specified by the first operand (symbol) and whose starting location is specified by the second operand (hexloc). When bytecount is not specified, 4 is assumed.

**Usage Notes:**

1. Issuing the DEFINE subcommand creates an entry in the debug symbol table. The entry consists of the symbol name, the storage address, and the length of the field. A maximum of 16 symbols can be defined in the debug symbol table at any given time.

2. When a DEFINE subcommand specifies a symbol that already exists in the debug symbol table, the storage address derived from the current request replaces the previous storage address. Several symbols may be assigned to the same storage address, but each of these symbols constitutes one entry in the debug symbol table. The symbols remain
defined until they are redefined or until an IPL subcommand loads a new copy of CMS.

3. When you assign a symbolic name to a storage location, you must know the current origin (set by the ORIGIN subcommand). The hexloc you specify is added to the current origin to create the entry in the symbol table used by DEBUG subcommands. If you change the current origin, existing entries are not changed.

4. You can use symbolic names to refer to storage locations when you issue the DEBUG subcommands BREAK, DUMP, GO, ORIGIN, STORE, and X.

Responses:

None.
Use the DUMP subcommand to print part or all of your virtual storage on the printer. First, a heading:

ident FROM starting location TO ending location

is printed. Next, the general registers 0-7 and 8-15, and the floating-point registers 0-6 are printed, followed by the PSW, CSW, and CAW. Then the specified portion of virtual storage is printed with the storage address of the first byte in the line printed at the left, followed by the alphanumeric interpretation of 32 bytes of storage.

The format of the DUMP subcommand is:

```
DUMP [ symbol1 hexloc1 ] [ symbol2 hexloc2 ] [ ident ]
```

where:

symbol1
is the name assigned (via the DEFINE subcommand) to the storage address that begins the dump.

hexloc1
is the hexadecimal storage location, in relation to current origin, that begins the dump.

symbol2
is the name assigned (via the DEFINE subcommand) to the storage address that ends the dump.

hexloc2
is the hexadecimal storage location, in relation to the current origin, that ends the dump.

* indicates that the dump ends at your virtual machine’s last virtual storage address.

ident
is any name (up to eight characters) that identifies the dump.
Usage Notes:

1. If you issue the DUMP subcommand with no operands, 32 bytes of storage are dumped, starting at the current origin.

2. The first and second operands must designate storage addresses that do not exceed your virtual machine storage size. Also, the storage address derived from the second operand must be greater than the storage address derived from the first operand.

Responses:

None.
DEBUG Subcommands

GO

Use the GO subcommand to exit from the debug environment and begin program execution.

The format of the GO subcommand is:

\[
\text{GO} \quad [\text{symbol} \quad \text{hexloc}]
\]

*where:*

- **symbol** is the symbolic name assigned to the storage location where you want execution to begin.
- **hexloc** is the hexadecimal location, in relation to the current origin, where you want execution to begin.

**Usage Notes:**

1. When you issue the GO subcommand, the general registers, CAW (channel address word), and CSW (channel status word) are restored either to their contents upon entering the debug environment, or, if they have been modified, to their modified contents. Then the old PSW is loaded and becomes the current PSW. Execution begins at the instruction address contained in bits 40-63 of the PSW.

2. When you specify symbol or hexloc with the GO subcommand, the specified address replaces the instruction address in the old PSW, so execution will begin at that address. If you entered the debug environment with the DEBUG command, you must specify an address with the GO subcommand.

3. The address you specify must be within your virtual machine and it must contain a valid operation code.
Responses:

Program execution is resumed.
GPR

Use the GPR subcommand to display the contents of one or more general registers at the terminal.

The format of the GPR subcommand is:

```
GPR  [reg1 [reg2]]
```

where:

- **reg1** is a decimal number (from 0-15 inclusive) indicating the first or only general register whose contents are to be displayed.

- **reg2** is a decimal number (from 0-15 inclusive) indicating the last general register whose contents are to be displayed. “reg2” must be larger than “reg1.”

Responses:

The register or registers specified are displayed, in hexadecimal representation:

```
xxxxxxxx
    .
    .
    .
```
HX

Use the HX subcommand to leave the debug environment, regardless of the reason the debug environment was entered.

The format of the HX subcommand is:

| HX |

Responses:

If you entered the debug environment following a program interruption, you receive the message:

CMS
to indicate a return to the CMS environment. If you entered the debug environment by issuing the DEBUG command, you receive the message:

DMSABN148T SYSTEM ABEND 2E4 CALLED FROM xxxxxx

where xxxxxx is the address of the debug routine.
DEBUG Subcommands

ORIGIN

Use the ORIGIN subcommand to set an origin or base address to be used in the debug environment.

The format of the ORIGIN subcommand is:

```
ORigin [symbol
      hexloc
      0]
```

where:

symbol

is a symbolic name that was previously assigned (via the DEFINE subcommand) to a storage address.

hexloc

is a hexadecimal location within the limits of your virtual storage. If you do not explicitly set an origin, then it has a value of 0.

Usage Notes:

1. When the ORIGIN subcommand specifies a symbol, the debug symbol table is searched. If a match is found, the value corresponding to the symbol becomes the new origin. When a hexadecimal location is specified, that value becomes the origin. In either case, the operand cannot specify an address greater than your virtual storage size.

2. Any origin set by an ORIGIN subcommand remains in effect until another ORIGIN subcommand is issued, or until you obtain a new copy of CMS. Whenever a new ORIGIN subcommand is issued, the value specified in that subcommand overlays the previous origin setting. If you obtain a new copy of CMS (via IPL), the origin is set to 0 until a new ORIGIN subcommand is issued.

3. You can use the ORIGIN subcommand to set the origin to your program’s base address, and then refer to actual instruction addresses in your program, rather than to virtual storage locations.
Responses:

None.
DEBUG Subcommands

PSW

Use the PSW subcommand to display the contents of the PSW (program status word).

The format of the PSW subcommand is:

| PSW |

Usage Notes:

1. If the debug environment was entered because of a program interruption, the program old PSW is displayed. If the debug environment was entered because of an external interruption, the external old PSW is displayed. If the debug environment was entered for any other reason, the following is displayed in response to the PSW subcommand:

01000000xxxxxxxx

where the 1 in the first byte means that external interruptions are allowed and xxxxxxx is the hexadecimal storage address of the debug program.

2. The PSW contains some information not contained in storage or registers but required for proper program execution. In general, the PSW is used to control instruction sequencing and to hold and indicate the status of the system in relation to the program currently executing. For a description of the PSW, refer to VM/SP CMS for System Programming.

Responses:

The PSW is displayed in hexadecimal representation.
RETURN

Use the RETURN subcommand to exit from the debug environment and enter the CMS command environment.

The format of the RETURN subcommand is:

```
RETurn
```

Usage Note:

The RETURN subcommand is valid only when the debug environment was entered via the DEBUG command.

Responses:

The CMS ready message indicates that control has been returned to the CMS environment.
DEBUG Subcommands

SET

Use the SET subcommand to change the contents of the control words and general registers.

The format of the SET subcommand is:

| SET | CAW hexinfo | CSW hexinfo [hexinfo] | PSW hexinfo [hexinfo] | GPR reg hexinfo [hexinfo] |

**where:**

**CAW hexinfo**
stores the specified information (hexinfo) in the CAW (channel address word) that existed at the time the debug environment was entered.

**CSW hexinfo [hexinfo]**
stores the specified information (hexinfo [hexinfo]) in the CSW (channel status word) that existed at the time the debug environment was entered.

**PSW hexinfo [hexinfo]**
stores the specified information (hexinfo [hexinfo]) in the old PSW (program status word) for the interruption that caused the debug environment to be entered.

**GPR reg hexinfo [hexinfo]**
stores the specified information (hexinfo [hexinfo]) in the specified general register (reg).

**Usage Notes:**

1. The SET subcommand can only change the contents of one control word at a time. For example, you must issue the SET subcommand three times:

   set caw hexinfo
   set csx hexinfo [hexinfo]
   set psw hexinfo [hexinfo]

   to change the contents of the three control words.
2. The SET subcommand can change the contents of one or two general registers each time it is issued. When four or fewer bytes of information are specified, only the contents of the specified register are changed. When more than four bytes of information are specified, the contents of the specified register and the next sequential register are changed. For example, the SET subcommand:

```
set gpr 2 xxxxxxxx
```

changes only the contents of general register 2. But, the SET subcommand:

```
set gpr 2 xxxxxxxx xxxxxxxx
```

changes the contents of general registers 2 and 3.

3. Each hexinfo operand should be from one to four bytes long. If an operand is less than four bytes and contains an uneven number of hexadecimal digits (representing half-byte information), the information is right-justified and the left half of the uneven byte is set to zero. If more than eight hexadecimal digits are specified in a single operand, the information is left-justified and truncated on the right after the eighth digit.

4. The number of bytes that can be stored using the SET subcommand varies depending on the form of the subcommand. With the CAW form, up to four bytes of information may be stored. With the CSW, GPR, and PSW forms, up to eight bytes of information may be stored, but these bytes must be represented in two operands of four bytes each. When two operands of information are specified, the information is stored in consecutive locations (or registers), even if one or both operands contain less than four bytes of information.

Responses:

None. To display the contents of control words or registers after you modify them, you must use the CAW, CSW, PSW, and GPR subcommands.
STORE

Use the STORE subcommand to store up to 12 bytes of hexadecimal information in any valid virtual storage location.

The format of the STORE subcommand is:

```
STore { symbol } hexinfo [hexinfo [hexinfo]]
```

**where:**

**symbol**

is the symbolic name assigned (via the DEFINE subcommand) to the storage address where the first byte of specified information is to be stored.

**hexloc**

is the hexadecimal location, relative to the current origin, where the first byte of information is to be stored.

**hexinfo**

is the hexadecimal information, four bytes or less in length (that is, two to eight hexadecimal digits), to be stored.

**Usage Notes:**

1. If an operand is less than four bytes long and contains an uneven number of hexadecimal digits (representing half-byte information), the information is right-justified and the left half of the uneven byte is set to zero. If more than eight hexadecimal digits are specified in a single operand, the information is left-justified and truncated on the right after the eighth digit.

2. The STORE subcommand can store a maximum of 12 bytes at one time. By specifying all three information operands, each containing four bytes of information, the maximum 12 bytes can be stored. If less than four bytes are specified in any or all of the operands, the information given is arranged into a string of consecutive bytes, and that string is stored starting at the location derived from the first operand.
For example, if you have defined a four-byte symbol named FENCE that currently contains X'FFFFFFFF' and you enter:

store fence 0

FENCE contains X'00FFFFFF'.

Responses:

None.

To display the contents of a storage location after you have modified it, you must use the X subcommand.
Use the X subcommand to examine and display the contents of specific locations in virtual storage.

The format of the X (examine) subcommand is:

```
X

symbol n

hexloc n
```

where:

symbol n

is the name assigned (via the DEFINE subcommand) to the storage address of the first byte to be displayed. n is a decimal number from 1 to 56 inclusive, that specifies the number of bytes to be examined. If a symbol is specified without a second operand, the length attribute associated with that symbol in the debug symbol table specifies the number of bytes to be examined.

hexloc n

is the hexadecimal location, in relation to the current origin, of the first byte to be examined. If hexloc is specified without a second operand, four bytes are displayed.

Usage Note:

The address represented by symbol or hexloc must be within your virtual machine storage size.

Responses:

The requested information is displayed at the terminal in hexadecimal format.
Appendix A. VSE/VSAM Functions Not Supported in CMS

Refer to the publication *Using VSE/VSAM Commands and Macros* for a description of Access Method Services functions available under VSE, and, therefore, under CMS. This knowledge of Access Method Services is assumed throughout this publication.

All of VSE/VSAM is supported by CMS, except for the following:

- Non-VSAM data sets with data formats that are not supported by CMS/DOS (for example, BDAM and ISAM files are not supported).

- The SHAREOPTIONS operand is not supported for cross system or cross partition sharing in CMS/DOS (that is, DASD sharing is not supported).

- Space Management for SAM Feature

- Backup/Restore Feature

If an AMSERV input file to VSE/VSAM Access Method Services contains the control statement “DELETE” with “IGNOREERROR,” the PRINT option on the AMSERV command must be used to send the output to the virtual printer.
Appendix B. OS/VS Access Method Services and VSAM Functions Not Supported

In CMS, an OS user is defined as a user that has not issued the command:

```
SET DOS ON (VSAM)
```

OS users can use all of the Access Method Services functions that are supported by VSE/VSAM, with the following exceptions:

- Non-VSAM data sets with data formats that are not supported by CMS/DOS (for example, BDAM and ISAM files are not supported).

- The SHAREOPTIONS operand is not supported for cross system or cross partition sharing in CMS/DOS (that is, DASD sharing is not supported).

- Do not use the AUTHORIZATION (entrypoint) operand in the DEFINE and ALTER commands unless your own authorization routine exists on the DOS core image library, the private core image library, or in a CMS DOSLIB file. In addition, results are unpredictable if your authorization routine issues an OS SVC instruction.

- The OS Access Method Services GRAPHICS TABLE options and the TEST option of the PARM command are not supported.

- The filename in the FILE (filename) operands is limited to seven characters. If an eighth character is specified, it is ignored.

- The OS access method services CNVTCAT and CHKLIST commands are not supported in VSE/VSAM access method services. In addition, all OS access method services commands that support the 3850 Mass Storage System are not supported in DOS/VS access method services.

- Figure 37 is a list of OS operands, by control statement, that are not supported by the CMS interface to VSE/VSAM Access Method Services.

If any of the unsupported operands or commands in Figure 37 are specified, the AMSERV command terminates and displays an appropriate error message.

When you use the PRINT, EXPORT, IMPORT, IMPORTRA, EXPORTRA, and REPRO control statements for sequential access method (SAM) data sets, you must specify the ENVIRONMENT operand with the required DOS options (that is, PRIME DATA DEVICE, BLOCKSIZE, RECORDSIZE, or RECORDFORMAT). You must have previously issued a DLBL for the SAM file.
AMSERV can write SAM data sets only to a CMS disk, but can read them from DOS, OS, or CMS disks.

<table>
<thead>
<tr>
<th>OS ACCESS METHOD SERVICES CONTROL STATEMENT</th>
<th>OPERANDS NOT SUPPORTED IN CMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTER</td>
<td>EMPTY/NOEMPTY</td>
</tr>
<tr>
<td></td>
<td>SCRATCH/NOSCRATCH</td>
</tr>
<tr>
<td></td>
<td>DESTAGEWAIT/NODESTAGEWAIT</td>
</tr>
<tr>
<td></td>
<td>STAGE/BIND/CYLINDERFAULT</td>
</tr>
<tr>
<td>DEFINE</td>
<td>ALIAS</td>
</tr>
<tr>
<td></td>
<td>EMPTY/NOEMPTY</td>
</tr>
<tr>
<td></td>
<td>GENERATIONDATAGROUP</td>
</tr>
<tr>
<td></td>
<td>PAGESPACE</td>
</tr>
<tr>
<td></td>
<td>SCRATCH/NOSCRATCH</td>
</tr>
<tr>
<td></td>
<td>DESTAGEWAIT/NODESTAGEWAIT</td>
</tr>
<tr>
<td></td>
<td>STAGE/BIND/CYLINDERFAULT</td>
</tr>
<tr>
<td></td>
<td>TO/FOR/OWNER\textsuperscript{11}</td>
</tr>
<tr>
<td>DELETE</td>
<td>ALIAS</td>
</tr>
<tr>
<td></td>
<td>GENERATIONDATAGROUP</td>
</tr>
<tr>
<td></td>
<td>PAGESPACE</td>
</tr>
<tr>
<td>EXPORT</td>
<td>OUTDATASET</td>
</tr>
<tr>
<td>IMPORT</td>
<td>INDATASET</td>
</tr>
<tr>
<td></td>
<td>OUTDATASET</td>
</tr>
<tr>
<td></td>
<td>IMPORTA</td>
</tr>
<tr>
<td>LISTCAT</td>
<td>ALIAS</td>
</tr>
<tr>
<td></td>
<td>GENERATIONDATAGROUP</td>
</tr>
<tr>
<td></td>
<td>LEVEL</td>
</tr>
<tr>
<td></td>
<td>OUTFILE\textsuperscript{12}</td>
</tr>
<tr>
<td></td>
<td>PAGESPACE</td>
</tr>
<tr>
<td>PRINT</td>
<td>INDATASET</td>
</tr>
<tr>
<td></td>
<td>OUTFILE\textsuperscript{12}</td>
</tr>
<tr>
<td>REPRO</td>
<td>INDATASET</td>
</tr>
<tr>
<td></td>
<td>OUTDATASET</td>
</tr>
</tbody>
</table>

\textbf{Figure 37. OS Access Method Service Operands NOT Supported in CMS}

OS users can use a subset of OS/VSAM Assembler Language Macros in their assembler language programs. The OS/VSAM Assembler language macros supported for use in CMS are contained in the OSVSAM MACLIB that is distributed with the VM/SP. The macros and options that are \textit{not} supported are shown in Figure 38.

\textsuperscript{11} The TO/FOR/OWNER operands are supported for the access method services interface, but are not supported for the DEFINE NONVSAM control statement.

\textsuperscript{12} The OUTFILE operand is supported by the access method services interface, but is not supported for the LISTCAT and PRINT control statements.
<table>
<thead>
<tr>
<th>OS/VSAM Macro</th>
<th>Options that are Not Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACB</td>
<td>BSTRNO = number</td>
</tr>
<tr>
<td></td>
<td>CATALOG = YES/NO</td>
</tr>
<tr>
<td></td>
<td>CRA = SCRA/UCRA</td>
</tr>
<tr>
<td></td>
<td>MACRF = CFX</td>
</tr>
<tr>
<td>EXLST</td>
<td>UPAD = address</td>
</tr>
<tr>
<td>GENCB BLK = ACB</td>
<td>AM = VSAM</td>
</tr>
<tr>
<td></td>
<td>BSTNRO = number</td>
</tr>
<tr>
<td></td>
<td>CATALOG = YES/NO</td>
</tr>
<tr>
<td></td>
<td>CRA = SCRA/UCRA</td>
</tr>
<tr>
<td></td>
<td>MACRF = CFX</td>
</tr>
<tr>
<td>GENCB BLK = EXLST</td>
<td>AM = VSAM</td>
</tr>
<tr>
<td>GENCB BLK = RPL</td>
<td>MSGAREA = address</td>
</tr>
<tr>
<td></td>
<td>MSGLEN = number</td>
</tr>
<tr>
<td></td>
<td>OPTCD = NWAITX</td>
</tr>
<tr>
<td>MODCB BLK = ACB</td>
<td>AM = VSAM</td>
</tr>
<tr>
<td></td>
<td>BSTRNO = number</td>
</tr>
<tr>
<td></td>
<td>CATALOG = YES/NO</td>
</tr>
<tr>
<td></td>
<td>CRA = SCRA/UCRA</td>
</tr>
<tr>
<td></td>
<td>MACRF = CFX</td>
</tr>
<tr>
<td>MODCB BLK = RPL</td>
<td>MSGAREA = address</td>
</tr>
<tr>
<td></td>
<td>MSGLEN = number</td>
</tr>
<tr>
<td></td>
<td>OPTCD = NWAITX</td>
</tr>
<tr>
<td>RPL</td>
<td>MSGAREA = address</td>
</tr>
<tr>
<td></td>
<td>MSGLEN = number</td>
</tr>
<tr>
<td></td>
<td>OPTCD = NWAITX</td>
</tr>
<tr>
<td>SHOWCB (ACB)</td>
<td>FIELDS = BFRFND, BSTRNO, BUFRDS, ENDRBA, HALCRBA, NUIW, UIW</td>
</tr>
<tr>
<td>SHOWCB (RPL)</td>
<td>FIELDS = MSGAREA, MSGLEN</td>
</tr>
<tr>
<td>TESTCB (ACB)</td>
<td>CATALOG = YES</td>
</tr>
<tr>
<td></td>
<td>CRA = SCRA</td>
</tr>
<tr>
<td></td>
<td>MACRF = CFX</td>
</tr>
<tr>
<td></td>
<td>BSTRNO = number</td>
</tr>
<tr>
<td></td>
<td>ENDRBA = number</td>
</tr>
<tr>
<td>TESTCB (RPL)</td>
<td>MSGAREA = address</td>
</tr>
<tr>
<td></td>
<td>MSGLEN = number</td>
</tr>
</tbody>
</table>

Figure 38. Options of OS/VSAM Macros Not Supported in CMS
Appendix C. Edit Subcommands and Macros

This appendix describes the formats and operands of the EDIT subcommands and macros. EDIT subcommands are valid only in the environment of the CMS editor or in CMS editor migration mode, which is invoked with the EDIT command. The EDIT command format is described in "Chapter 2. CMS Commands."

The editor has two modes of operation: edit mode and input mode. Whenever the EDIT command is issued, edit mode is entered; when the INPUT or REPLACE subcommands are issued with no operands, input mode is entered. In input mode, all lines you enter are written into the file you are editing. To return to edit mode from input mode, you must enter a null line (one that has no data on it).

For a functional description of the CMS editor and information on how to use it, consult the VMjSP CMS User’s Guide.

For a functional description of CMS editor migration mode and information on how to use it, see the EDIT command in this book and the VMjSP System Product Editor Command and Macro Reference.

For a summary of the default settings assumed by the editor for CMS reserved filetypes, see "Appendix D. Edit Reserved Filetype Defaults."

EDIT Subcommands

The EDIT subcommands are listed in alphabetical order for easy reference. Each subcommand description includes the format, a list of operands (if any), usage notes, and responses. For those subcommands that operate somewhat differently on a 3270 display terminal than on a typewriter terminal, an additional discussion, "Display Mode Considerations," is added.

Subcommands that are valid only with 3270 display terminals, namely SCROLL, SCROLLUP, and FORMAT have the notation “(3270 only)” next to the subcommand names. The FORWARD and BACKWARD subcommands, which were designed for use with 3270 terminals but can be issued at any terminal, have the notation “(primarily 3270)” next to the subcommand names.
ALTER

Use the ALTER subcommand to change a specific character to another character, one that may not be available on your terminal keyboard. The ALTER subcommand allows you to reference characters by their hexadecimal values.

The format of the ALTER subcommand is:

```
ALTER char1 char2 [ n * [G] ]
```

where:

char1
specifies the character to be altered. It may be specified either as a single character or as a pair of hexadecimal digits (00 through FF).

char2
specifies the character to which char1 is to be altered. It may be specified either as a single character or as a pair of hexadecimal digits.

n
indicates the number of lines to be searched for the specified character. If you specify an asterisk (*), all lines in the file, beginning with the current line, are searched. If this option is omitted, then only the current line is searched.

G
requests the editor to alter every occurrence of char1 in the lines specified. If G or * is not specified, only the first occurrence of char1 in each line specified is altered.

Usage Notes:

1. If char2 is a hexadecimal value that cannot be represented on your terminal, it may appear as a blank, for example:

   ```
   input XSLC
   alter X 02
   SLC
   
   Column 1 contains an X'02', which cannot be displayed.
   ```

2. Use the ZONE subcommand if you want only particular columns searched for a specific character.
Responses:

When verification is on, altered lines are displayed at your terminal.

Display Mode Considerations

When you request a global change on a 3270, the display is changed only once, to reflect the final position of the current line pointer. The editor displays a message to indicate the number of lines changed:

nnnn LINE(S) CHANGED

NO

AUTOSAVE

Use the AUTOSAVE subcommand to set, reset, or display the automatic save function of the editor. When the automatic save function is in effect, the editor automatically issues the SAVE subcommand each time the specified number of changes or insertions are made.

The format of the AUTOSAVE subcommand is:

| AUTOSAVE | \[n \[OFF\]\] |

where:

\(n\)

is a decimal number between 1 and 32767, indicating the frequency of the automatic save function. One SAVE subcommand is issued for every \(n\) lines that are changed, deleted, or added to the file.

OFF

turns off the automatic save function. This is the initial setting.

Usage Notes:

1. Each line affected by the $MOVE macro is treated as one update. However, all changes caused by a single CHANGE, DELETE, DSTRING, GETFILE, or OVERLAY subcommand are treated as a single update, no matter how many lines are affected.

2. If you are editing a file on a read-only disk, and an automatic save request occurs, the message:

SET NEW FILEMODE AND RETRY
is issued. You can enter CMS subset and access the disk in read/write mode, or use the FMODE subcommand to change the filemode to the mode of a read/write disk. If you were in input mode, you are placed in edit mode.

3. The message “SAVED” is displayed at the terminal each time the save operation occurs.

Responses:

If you issue the AUTOSAVE subcommand with no operands, the editor displays the current setting of the automatic save function.

**BACKWARD (Primarily 3270)**

Use the BACKWARD subcommand to move the current line pointer towards the beginning of the file you are editing.

The format of the BACKWARD subcommand is:

```
BAckward [n]
```

*where:*

- $n$ is the number of records backward you wish to move the current line pointer. If $n$ is not specified, the current line pointer is moved backward one line, toward the top of the file.

**Usage Note:**

The BACKWARD subcommand is equivalent to the UP subcommand; it is provided for the convenience of 3270 users.

Responses:

When verification is on, the current line on the screen contains the record located by the BACKWARD $n$ value. If $n$ exceeds the number of records above the current line, TOF is displayed on the current line.

On a typewriter terminal the new current line is typed if verification is on.
**BOTTOM**

Use the BOTTOM subcommand to make the last line of the file the new current line.

The format of the BOTTOM subcommand is:

```
Bottom
```

**Usage Note:**

Use the BOTTOM subcommand followed by the INPUT subcommand to begin entering new lines at the end of a file.

**Responses:**

When verification is on, the last line in the file is displayed.

**Display Mode Considerations**

If the BOTTOM subcommand is issued at a 3270 display terminal in display mode, EOF: is displayed on the line following the current line, preceded by the last records of the file; the rest of the screen's output area is blank.

**CASE**

Use the CASE subcommand to indicate how the editor is to process uppercase and lowercase letters.

The format of the CASE subcommand is:

```
CASE [ M ]
```

**where:**

- **M** indicates that the editor is to accept any mixture of uppercase and lowercase letters for the file as they are entered at the terminal.
U
indicates that the editor is to translate all lowercase letters to
uppercase letters before the letters are entered into the file. U is the
default value for all filetypes except MEMO and SCRIPT.

Responses:
If you enter the CASE subcommand with no operand, the current setting is
displayed at the terminal.

Display Mode Considerations
If you specify CASE M when using a 3270 that does not have the lowercase
feature (RPQ), you can key in lowercase characters, but they appear on the
screen as uppercase characters.

CHANGE
Use the CHANGE subcommand to change a specified group of characters to
another group of characters of the same or a different length. You may use
the CHANGE subcommand to change more than one line at a time.

The format of the CHANGE subcommand is:

```
CHANGE [/string1] [/string2] [/ [n [* [G]]]]
```

*where:
/* (diagonal)
signifies any unique delimiting character that does not appear in the
character strings involved in the change.

string1
specifies a group of characters to be changed (old data). string1 may
be a null string.

string2
specifies the group of characters that are to replace string1 (new data).
string2 may be a null string; if omitted, it is assumed null.

n or *
indicates the number of lines to be searched, starting at the current
line. If * is entered, the search is performed until the end of the file is
reached. If this option is omitted, then only one line is searched.
G or *
requests the editor to change every occurrence of string1 in the lines specified. If G or * is not specified, only the first occurrence of string1 in each line specified is changed. If string1 is null, G or * may not be specified.

Usage Notes:

1. The first nonblank character following the CHANGE subcommand (or any of its truncations) is considered the delimiter. For example:

   c.VM/SP.CMS.*

   changes the first occurrence of VM/SP to CMS on every line from the current line to the end of the file.

2. If string2 is omitted, it is assumed to be a null string. For example:

   THIS ISN'T THE LINE.
   change /n
   THIS IS THE LINE.

   A null string causes a character deletion. If string1 is null, characters are inserted at the beginning of the line. For example:

   THIS IS THE LINE.
   change //SO /
   SO THIS IS THE LINE.

3. To change multiple occurrences of the same string on one line, enter:

   change/string1/string2/ 1 *

4. The CHANGE subcommand can be used on typewriter terminals to display, without changing, any lines that contain the information specified in string1. Enter:

   change /string1/string1/ * *

5. Use the ZONE subcommand to indicate which columns are to be searched for string1. If string1 is wider than the current zone, you receive the message:

   ZONE ERROR

   and you should either reenter the CHANGE subcommand or change the zone setting.

6. If the character string inserted causes the data line to extend beyond the truncation column or the zone column, any excess characters are truncated. (See the description of the TRUNC subcommand for additional information on truncation.)

7. You should use the ALTER subcommand when you want to change a single character to some special character (one that is not available on your keyboard).
8. When the IMAGE subcommand is set with the CANON operand, backspace characters at the beginning or end of string1 are ignored.

9. To stack a CHANGE subcommand with no operands from a fixed-length EXEC, you should use the &STACK control statement.

Responses:

When verification is on, every line that is changed is displayed.

Display Mode Considerations

If you issue the CHANGE subcommand without operands at a 3270 display terminal in display mode, the following occurs:

1. The record pointed to by the current line pointer appears in the user input area of the display. If the line is longer than the current truncation setting, it is truncated.

2. You can then alter the record in the user input area by retyping part or all of the line, or by using the Insert, Delete, or Erase EOF keys to insert or delete characters.

3. When the line is modified, press the Enter key. This causes the record in the user input area to replace the old record at the current line in the output display area.

If you bring a line down to the user input area and decide not to change it, press the Erase Input key and then the Enter key, and the line is not changed.

When a line is moved to the user input area, all nonprintable characters (including tabs, backspaces, control characters, and so on) are stripped from the line. Also, any characters currently assigned to VM/SP logical line editing symbols (#, @, ¢, “) are reinterpreted when the line is reentered. You should issue an explicit CHANGE subcommand to change lines containing special characters.

The CHANGE subcommand is treated as an invalid subcommand if it is issued without operands at a typewriter terminal or at a 3270 display terminal that is not in display mode.

When you request a global change on a 3270 terminal, the display is changed only once, to reflect the final position of the current line pointer. The editor displays, in the message area of the display screen:

\{ nnnn \} LINE(S) CHANGED

NO to indicate the number of lines that were updated. If the change request resulted in the truncation of any lines, the message is displayed as:

nnnn LINE(S) CHANGED nnnn LINE(S) TRUNCATED
If the change request moves the current line pointer beyond the end of the file, the word EOF: is displayed on the current line, preceded by the last records of the file. The rest of the output area is blank.

**CMS**

Use the CMS subcommand to cause the editor to enter the CMS subset mode, where you may execute those CMS commands that do not need to use the main storage being used by the editor.

The format of the CMS subcommand is:

```
CMS
```

**Usage Notes:**

1. In CMS subset, you can execute any CMS command that is nucleus-resident or that executes in the transient area. Refer the section entitled “CMS Command Execution Characteristics” on page 9 to determine if a command is nucleus-resident or if it executes in the transient area.

   To return to edit mode, use the CMS subset command RETURN.

2. If you attempt to execute a CMS command that requires main storage, you receive the message:

   ```
   INVALID SUBSET COMMAND
   ```

   Results are unpredictable at this point. You should not attempt to execute any program that executes in the user program area. Using the LOAD, INCLUDE (RESET), FETCH, START, and RUN commands could load programs that would overlay the editor’s storage area and its contents. Use these commands only for programs that execute in the transient area.

3. In an edit macro, if you attempt to use a command that is invalid in the CMS subset, you receive a return code of -0002.

4. If you attempt to execute a CMS command that fails because of insufficient storage, your EDIT session may abnormally terminate. You should save input you have entered before you enter CMS subset mode.

5. Combining EDIT and XEDIT, such as executing XEDIT as a CMS Subset Command of EDIT, may not give you the INVALID SUBSET message, but it can result in abends or unpredictable results.
Responses:

After you issue the CMS subcommand, you receive the message:

CMS SUBSET

to indicate that you are in CMS subset mode. On a display terminal, the screen is cleared before the editor issues this message; the display of the file is restored when you enter the RETURN command.

DELETE

Use the DELETE subcommand to delete one or more lines from a file, beginning with the current line. The line immediately following the last line deleted becomes the new current line.

The format of the DELETE subcommand is:

\[
\text{DELETE} \quad [n] \\
\]

where:

\( n \)

indicates the number of lines to be deleted, starting at the current line. If an asterisk (*) is entered, the remainder of the file is deleted. If \( n \) is omitted, only one line is deleted.

Responses:

None. If you delete the last line in the file, or if you issue the DELETE subcommand when the current line pointer is already at the end of the file, the editor displays the message:

EOF:

Display Mode Considerations

If you delete a record when using a display terminal in display mode, the editor rewrites the output display area with the records above the current line pointer unchanged. The record at the current line pointer and the remaining records on the screen move up by one, and a new record (if one exists) moves into the bottom of the output display area.
DOWN

Use the DOWN subcommand to advance the current line pointer forward in the file. The line pointed to becomes the new current line.

The format of the DOWN subcommand is:

\[
\text{DOn} \quad \begin{array}{c}
\text{[n]} \\
\text{[1]}
\end{array}
\]

*where:*

\[n\]

indicates the number of lines to advance the pointer, starting at the current line. If \(n\) is not specified, the current line pointer is advanced one line.

**Usage Note:**

DOWN is equivalent to the NEXT and FORWARD subcommands.

**Responses:**

When verification is on, the new current line is displayed at the terminal; if the end of the file is reached, the message:

\[\text{EOF:}\]

is displayed.

DSTRING

Use the DSTRING subcommand to delete one or more lines beginning with the current line, down to, but not including, the first line containing a specified character string. The current line is not checked for the character string.

The format of the DSTRING subcommand is:

\[
\text{DString} \quad \begin{array}{c}
\text{[/string[/]]}
\end{array}
\]

*where:*

\[/\text{string[/]}\]


Usage Note:

The zone set by the ZONE subcommand or the default zone setting is checked for the presence of the character string. A character string with a length greater than the current zone setting causes the error message ZONE ERROR.

Responses:

If the character string is not found by the end of the file, no deletions occur, the current line pointer is unchanged, and the message:

STRING NOT FOUND, NO DELETIONS MADE

is displayed.

Display Mode Considerations

If verification is on when the DSTRING subcommand is issued at a display terminal in display mode, the screen is changed to reflect the deletions from the file.

FILE

Use the FILE subcommand to write the edited file on disk and, optionally, override the file identifier originally supplied in the EDIT command.

The format of the FILE subcommand is:

FILE  [fn  [ft  [fm]]]

where:

fn

indicates the filename for the file. If filename is omitted, filetype and filemode cannot be specified, and the existing filename, filetype, and filemode are used.
Usage Notes:

1. When you specify a file identifier, any existing file that has an identical fileid is replaced. If the file being edited had been previously written to disk, that copy of the file is not altered.

2. You can change the filename and filemode during the editing session using the FNAME and FMODE subcommands.

Responses:

The CMS ready message indicates that the file has been written to disk and control is returned to the CMS environment.

FIND

Use the FIND subcommand to locate a line based on its initial character string.

The format of the FIND subcommand is:

```
FIND [line]
```

where:

- `line` is any character string, including blanks and tabs, that you expect to find beginning in column 1 of an input record. At least one non-blank character must be specified. If `line` is not specified or the line contains only blanks, the current line pointer is moved down one line.

Usage Notes:

1. Only one blank can be used as a delimiter following the FIND subcommand; additional blanks are considered part of the character string.

2. If the image setting is ON, the editor expands tab characters to the appropriate number of blanks before searching for the line.

3. If the current line pointer is at the bottom of the file when the FIND subcommand is issued the search begins at the top of the file.
**Responses:**

When verification is on, the line is displayed at the terminal. If the line is not found, the message:

```
EOF:
```

is displayed and you may use the REUSE (=) subcommand to search again, beginning at the top of the file.

**FMODE**

Use the FMODE subcommand to display or change the filemode of a file.

The format of the FMODE subcommand is:

```
FMODE [fm]
```

where:

*fm*

indicates the filemode that is to replace the current filemode setting. You can specify a filemode letter (A-Z) or a filemode letter and number (0-6). If you specify a filemode letter, the existing filemode number is retained.

**Usage Notes:**

1. The specified filemode is used the next time a FILE, SAVE, or automatic save request is issued. If the file being edited had been previously filed or saved, that copy of the file remains unchanged.

2. If the disk specified by filemode already contains a file with the same filename and filetype, that file is replaced when a FILE, SAVE, or automatic save request is issued; no warning message is issued.

3. If the filemode specified is that of a read-only disk, then when an attempt is made to file or save the file, the editor displays an error message.

**Responses:**

If you enter the FMODE subcommand without specifying fm, the editor displays the current filemode.
Display Mode Considerations

When you specify a new filemode with the FMODE subcommand, the editor writes the new filemode in the filemode field at the top of the screen.

FNAME

Use the FNAME subcommand to display or change the filename of a file.

The format of the FNAME subcommand is:

<table>
<thead>
<tr>
<th>FName</th>
<th>[fn]</th>
</tr>
</thead>
</table>

*where:*

*fn* indicates the filename that is to replace the current filename.

Usage Notes:

1. The specified filename is used the next time a FILE, SAVE, or automatic save request is issued. If the file being edited had been previously filed or saved, that copy of the file remains unchanged.

2. If a file already exists with the specified filename and the same filetype and filemode, that file is replaced; no warning message is issued.

3. You can use the FNAME subcommand when you want to make multiple copies of a file, with different filenames, without terminating your edit session.

Responses:

If you enter the FNAME subcommand without specifying fn, the editor displays the current filename.

Display Mode Considerations

When you issue the FNAME subcommand specifying a new filename, the editor writes the new name in the filename field at the top of the screen.
FORMAT (3270 only)

Use the FORMAT subcommand to change the mode of a local or remote 3270 terminal from display to line or line to display mode.

The format of the FORMAT subcommand is:

```
FORMat { DISPLAY 
         LINE }
```

where:

DISPLAY

specifies that a full screen display of data is to occur. Subcommands do not appear as part of the data displayed.

LINE

specifies that the display station is to operate as a typewriter terminal. Every line you enter is displayed on the screen; the screen looks like a typewriter terminal's console sheet.

Usage Notes:

1. Line mode is the default for remote 3270s. If you are using a remote 3270 in display mode, and you enter the INPUT subcommand, you are placed in line mode while you enter input. When you return to edit mode, the full screen display is restored.

2. The FORMAT subcommand is treated as invalid under any of the following conditions:
   a. The NODISP option of the EDIT command was used to invoke the editor.
   b. The edit session was initiated on a typewriter terminal. (The session may optionally be continued on a 3270 after a reconnection.)

   To obtain a full screen display, you must save your file and restart your edit session.

3. The column settings for the VERIFY, TRUNC, and ZONE subcommands remain unchanged when you issue the FORMAT subcommand.
FORWARD (Primarily 3270)

Use the FORWARD subcommand to move the current line pointer towards the end of the file you are editing.

The format of the FORWARD subcommand is:

```
FOrward [ n ]
```

where:

- \( n \)
  - is the number of records you wish to move forward in the file being edited. If \( n \) is not specified, 1 is assumed.

Usage Note:

The FORWARD subcommand is equivalent to the DOWN and NEXT subcommands; it is provided for the convenience of 3270 users.

Responses:

When verification is on, the new current line is displayed. If the number specified exceeds the number of lines remaining in the file, the current line pointer is positioned at EOF.

GETFILE

Use the GETFILE subcommand to insert all or part of a specific CMS file into a file you are editing.

The format of the GETFILE subcommand is:
where:

\( fn \)

is the filename of the file that contains the data to be inserted into the file you are editing. When an asterisk (*) is specified, the filename of the file you are editing is assumed.

\( ft \)

is the filetype of the file that contains the data to be inserted. If \( ft \) is not specified or when an asterisk (*) is specified, the filetype of the file you are editing is assumed.

\( fm \)

is the filemode of the file that contains the data to be inserted. If \( fm \) is not specified or when an asterisk (*) is specified, all of your accessed disks are searched for the file.

\( firstrec \)

indicates the number of the first record you want to copy.

\( numrec \)

indicates the number of lines to be inserted, starting with the line specified by \( firstrec \). If \( numrec \) is not specified, or specified as *, then the remainder of the file between \( firstrec \) and the end of the file is inserted.

### Usage Notes:

1. The GETFILE operand list is positional; if you omit one operand, you cannot specify any operands that follow. Thus, if you want to specify \( firstrec \) and \( lastrec \), you must specify the filetype and filemode of the file.

2. The last line inserted becomes the new current line.

3. If the length of the records in the file containing the data to be inserted exceeds that of the file being edited, an error message is displayed, and the GETFILE is not executed; if shorter, the records are padded to the record length of the file being edited and inserted in the file.

4. If you use the GETFILE subcommand to insert lines into a VSBASIC file, use the RENUM subcommand to resequence the file.
5. If the editor fills up available storage while executing a GETFILE request, it may not be able to copy all of the file. You should determine how many records were actually copied, and then write the current file on disk.

**Responses:**

When verification is on, the last line inserted into the file is displayed. If the end of the file has been reached, the message:

**EOF REACHED**

is displayed, followed by the display of the last line inserted.

**IMAGE**

Use the IMAGE subcommand to control how the editor should handle backspaces and tab characters or to display the current image setting.

The format of the IMAGE subcommand is:

```
IMAGE [ON  
     OFF  
    CANON]
```

**where:**

**ON**

specifies that any text entered while in input mode or as a line of data following a FIND, INPUT, OVERLAY, or REPLACE subcommand, is expanded into a line image; backspaces are removed and tabs are replaced by blanks.

Text entered in the form of delimited strings, as in CHANGE, LOCATE, and ALTER, is not expanded; tabs and backspaces are treated in the same way as other characters.

**IMAGE** **ON** is the default for all filetypes except SCRIPT.

**OFF**

specifies that tabs and backspaces are treated as data characters in the same way as other characters. They are not deleted, translated, expanded, or reordered.

**CANON**

specifies that backspaces may be used to produce compound characters such as underscored words, headings, or phrases. Before they are inserted in the file, compound characters are ordered, with
backspaces arranged singly between the characters that overlay each other; the overlaying characters are arranged according to their EBCDIC values. Tab characters are handled as for IMAGE OFF.

CANON is the default for SCRIPT files.

Usage Notes:

1. When the image setting is ON, tab characters are expanded to an appropriate number of blanks, according to the current settings of the TABSET subcommand. The TABSET command has no effect if the image setting is either OFF or CANON.

2. When the image setting is on, backspaces are handled as follows:
   - Backspace characters act in a similar manner to the logical character delete symbol, in deleting the previous characters if a sufficient number of other characters or blanks follow the backspace characters. However, backspace characters that immediately follow a command name are interpreted as separator characters and do not delete any part of the command name.
   - If a backspace character is the last character in the input line, it is ignored.

Responses:

When you issue the IMAGE subcommand with no operand, the current IMAGE setting is displayed.

INPUT

Use the INPUT subcommand to insert a single line into a file, or, if no data line is specified, to leave edit mode and enter input mode.

The format of the INPUT subcommand is:

```
Input [line]
```

where:

`line`

specifies the input line to be entered into the file. It can contain blanks and tabs; if you enter at least two blanks following the INPUT subcommand and no additional text, a blank line is inserted into the file.
Usage Notes:

1. Each line that is inserted into the file becomes the new current line.

2. When you are using line-number editing (LINEMODE LEFT or LINEMODE RIGHT) you cannot use the INPUT subcommand to insert a single line of data; use the nnnnn subcommand.

3. To stack an INPUT subcommand in order to enter input mode from a fixed-length EXEC, you should use the &STACK control statement.

Responses:

When you issue the INPUT subcommand without operands, and verification is on, the editor displays:

```
INPUT:
```

All subsequent lines you enter are written into the file, until you enter a null line to return to edit mode.

Display Mode Considerations

1. When you insert lines while using a local display terminal in display mode, the editor writes each record on the current line. The old current line and all records above it move up one line, except for the topmost record formerly on line 2, which is deleted from the screen.

2. If you are using a remote display terminal in display mode and you issue the INPUT subcommand with no text, the terminal is forced into line mode. The display of the file on the screen disappears and the word INPUT: appears. As you enter input lines, they appear in the output display area. When you leave input mode by entering a null line, the remote terminal returns to display mode. The display of the file reappears on the screen, with the lines you have just entered in their proper place in the file.

3. When you are entering data in input mode at a display terminal that is in line mode, a tab character generated by a program function (PF) key only generates one character, and appears as one character on the screen. That is, the line does not appear spaced according to the tab settings.

LINEMODE

Use the LINEMODE subcommand to set, cancel, or display the status of line-number editing. When you use line-number editing, you can input, locate, and replace lines by referencing their record numbers. Line-number editing is the default for VSBASIC and FREEFORT files.

The format of the LINEMODE subcommand is:
where:

LEFT
L
initializes line-number editing and places sequence numbers on the left, in columns 1 through 5, right-justified and padded with blanks; the near zone is set to 7. If the filetype is FREEFORT, columns 1 through 8 are used for serial numbers; the near zone is set to 9.

You should never use left-handed line-number editing for files in which data must occupy columns 1 through 6, for example ASSEMBLE files.

RIGHT
R
initializes line-number editing and places sequence numbers on the right, in columns 76 to 80, right-justified and padded with zeroes. The end zone and truncation columns are set to 72.

This operand is valid only for files with fixed-length 80-character records.

OFF

cancels line-number editing and (if you were using left-handed line-number editing) resets the first logical tab setting to column 1. The VERIFY, TRUNC, and ZONE subcommand settings remain unchanged. Serialization may still be in effect. OFF is the default for all filetypes except VSBASIC and FREEFORT.

Note: If you enter LINEMODE OFF while editing a FREEFORT file, line-number editing cannot be resumed for the remainder of the edit session.

Usage Notes:

1. When you enter input mode while you are using line-number editing, you are prompted with a line number to enter each line. The default prompting increment is 10; you may change it using the PROMPT subcommand.

If you enter input mode after using the nnnnn subcommand to position the current line pointer, the prompted line number is the next higher multiple of the current prompting increment or an adjusted line number, whichever is smaller. The adjusted line number is determined according to the following formula:
pppp = 1 + cccc + nnnn - cccc (Any fractional remainder is dropped.)

where:

pppp is the prompt line number.

cccc is the current line number.

nnnn is the next sequential line number in the file.

2. When you are prompted on a typewriter terminal, enter your input line on the same line as the prompted line number. If you are using right-handed line-number editing, on a typewriter terminal or on a display terminal in line mode, the serial numbers are not redisplayed in columns 76 to 80 (unless you use the VERIFY subcommand to increase the verification setting). When a line is displayed in edit mode, the line numbers always appear on the left even though they are on the right in the disk copy of the file. Whether or not the line numbers are displayed on the right depends on the current verification setting.

3. You cannot use the INPUT or REPLACE subcommands to input a single data line when you are using line-number editing; use the nnnnn subcommand instead.

4. When you initialize line-number editing for files that already exist, the editor assumes that the records are in the proper format and numbered in ascending order.

5. If you want to place serial numbers in columns 76 through 80, but you do not wish to use line-number editing, use the SERIAL subcommand.

Responses:

When you issue the LINEMODE subcommand with no operands, the current setting is displayed.

Display Mode Considerations

When you use line-number editing on a display terminal in display mode, the prompting numbers in input mode appear on line 2 of the display screen, in the editor message area. Enter your input lines in the user input area. Regardless of whether you are using right- or left-handed line-number editing, the line numbers always appear in their true position in the file.
LOCATE

Use the LOCATE subcommand to scan the file beginning with the next line for the first occurrence of a specified character string.

The format of the LOCATE subcommand is:

| Locate | /[string[/]] |

where:

/ (diagonal)

signifies any unique delimiting character that does not appear in the string. The delimiter may be any nonblank character. The closing delimiter is optional.

string

specifies any group of characters to be searched for in the file.

Usage Notes:

1. If the beginning delimiter is /, you can omit the subcommand name LOCATE. If you enter only:

   /

   on a line, the current line pointer is moved down one line.

2. If string is null or blank, the search is successful on the first line encountered. If the line pointer is at the end of the file when the LOCATE subcommand is issued, scanning starts from the top of the file.

3. Use the ZONE subcommand when you want the editor to search only a specific column. If you specify a character string longer than the current zone width, the editor issues the message ZONE ERROR.

Responses:

When verification is on, the line containing the specified string is displayed. If the string is not found, the messages:

NOT FOUND
EOF:

are displayed, and you may use the REUSE (=) subcommand to request that command be repeated, beginning at the top of the file.
LONG

Use the LONG subcommand to request the editor to respond to invalid subcommand lines with the long form of the ?EDIT message.

The format of the LONG subcommand is:

| LONG |

Usage Note:

When the LONG subcommand is in effect (it is the default), the editor responds to invalid subcommands with the message:

?EDIT: line ...

Responses:

None.

NEXT

Use the NEXT subcommand to advance the line pointer a specified number of lines toward the end of the file. The line pointed to becomes the new current line.

The format of the NEXT subcommand is:

| Next | [ n ] |

where:

n

indicates the number of lines to move the line pointer. If “n” is omitted, then the pointer is moved down only one line.
Usage Note:

NEXT is equivalent to DOWN and FORWARD.

Responses:

When verification is on, the new current line is displayed. If the end of the file is reached, the message:

EOF:

is displayed.

OVERLAY

Use the OVERLAY subcommand to selectively replace one or more character strings in the current line with the corresponding nonblank characters in the line being keyed in.

The format of the OVERLAY subcommand is:

```
OVERLAY [line]
```

where:

```
line
```

specifies an input line that replaces corresponding character positions in the current line. On a typewriter terminal, if you enter the OVERLAY subcommand with no data line, the input record remains unchanged.

Usage Notes:

1. Blank characters in the input line indicate that the corresponding characters in the current line are not to be overlaid. For example:

   CHARMIE
   o   L
   CHARLIE

   Blanks in columns 3, 4, 5, and 6 of the OVERLAY line indicate that columns 1, 2, 3, and 4 of the current line are not to be changed. (At least one blank must follow the OVERLAY subcommand, which can be truncated as O).

2. This subcommand may be entered at a typewriter terminal by typing the letter “o,” followed by a backspace, followed by the overlaying characters. This sets up the correct alignment on the terminal.
3. An underscore in the overlaying line must be used to place a blank into the corresponding position of the current line. Thus, an underscore cannot be placed (or replaced) in a line.

OVERLAY should be used with care on lines containing underscored words or other compound characters.

4. To perform a global overlay operation, issue the REPEAT subcommand just prior to issuing the OVERLAY subcommand. For example, when you enter:

```
repeat *
overlay X
```

an X is placed in the leftmost column of each record in the file, beginning with the current line. The leftmost column, for files with the IMAGE setting ON, is determined by the first logical tab setting.

Responses:

When verification is on, the line is displayed at the terminal after it has been overlaid.

Display Mode Considerations

In addition to using the OVERLAY subcommand in the normal way, you may also issue the OVERLAY subcommand with no operands. The next line you enter is treated as overlay data. To cancel the overlay request, press the Erase Input key and then the Enter key.

PRESERVE

Use the PRESERVE subcommand to save the settings of various EDIT subcommands until a subsequent RESTORE subcommand is issued.

The format of the PRESERVE subcommand is:

```
PREServe
```

Usage Note:

Settings are saved for the following subcommands:

<table>
<thead>
<tr>
<th>CASE</th>
<th>LONG</th>
<th>TABSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMODE</td>
<td>PROMPT</td>
<td>TRUNC</td>
</tr>
</tbody>
</table>
Responses:

None.

PROMPT

Use the PROMPT subcommand to change the prompting increment for input line numbers when you are using line-number editing.

The format of the PROMPT subcommand is:

```
PROMPT [ n ]
```

where:

- `n` specifies the prompting increment; the default value is 10. The value of `n` should not exceed 32,767.

Responses:

When you issue the PROMPT subcommand with no operands, the current setting is displayed.

QUIT

Use the QUIT subcommand to terminate the current editing session and leave the previous copy of the file, if any, intact on the disk.

The format of the QUIT subcommand is:

```
QUIT
```
Usage Notes:

1. You can use the QUIT subcommand when you have made a global change that introduced errors into your file; or whenever you discover that you have made errors in editing a file and want to cancel your editing session.

   If a SAVE subcommand or automatic save request has been issued, the file remains as it was when last written.

2. The QUIT subcommand is a convenient way to terminate an edit session when you enter an incorrect filename on the EDIT command line, or when you edit a file merely to examine, but not to change, its contents.

Responses:

The CMS ready message indicates that control has been returned to CMS.

RECFM

Use the RECFM subcommand to indicate to the editor whether the record format of the file is fixed-length or variable-length, or to display the current RECFM setting.

The format of the RECFM subcommand is:

```
RECFM [F V]
```

where:

F

indicates fixed-length records.

V

indicates variable-length records.

Usage Notes:

1. V is assumed by default for all new EXEC, LISTING, FREEFORT, VSBDATA, and SCRIPT files. Usually, a variable-length format file occupies a smaller amount of disk space because trailing blanks are deleted from each line before it is written onto disk. When variable-length VSBDATA files are written to disk, however, trailing blanks are not truncated (to allow VSBDATA file to span records).
2. When you use the RECFM subcommand to change the format of a file from fixed-length to variable-length records, trailing blanks are removed when the file is written to disk; when you are changing variable-length records to fixed-length, all records are padded to the record length.

Responses:

When you use the RECFM subcommand without specifying F or V, the current setting is displayed.

Display Mode Considerations

When you specify a new record format with the RECFM subcommand, the editor writes the new record format in the format field at the top of the screen.

RENUM

Use the RENUM subcommand to recompute the line numbers for VSBASIC and FREEFORT source files.

The format of the RENUM subcommand is:

```
RENum [ strtno ] [ incrno ]
```

where:

strtno

indicates the number from which you wish to start renumbering your file. Because RENUM renumbers the whole file from beginning to end, the number you specify as strtno becomes the statement number of the first statement in the newly renumbered file. This number may not exceed 99999 for VSBASIC files or 99999999 for FREEFORT files. The default start number value is 10 and the specified start number must not be zero.

incrno

indicates the increment number value by which you wish to renumber your file. This value may not exceed 99999 for VSBASIC files or 99999999 for FREEFORT files. The default for incrno is strtno, the first sequence number in the renumbered file, and the specified incrno must not be zero.
Usage Notes:

1. If you do not specify strtno and incrno, the default value for both is 10. If you specify only strtno, incrno defaults to the same value as strtno.

2. The current line pointer remains as it was before you entered the RENUM subcommand regardless of whether or not RENUM completes successfully. If you are editing a VSBASIC file, the file to be renumbered must either originate from a read/write disk or you must issue an FMODE subcommand to change the file destination to a read/write disk.

3. All VSBASIC statements that use statement numbers for operands are updated to reflect the new line numbers. The VSBASIC statements with line number operands are:

   - CLOSE
   - CLOSEFILE
   - DELETE
   - EXIT
   - GET
   - GOSUB
   - GOTO
   - IF
   - ON
   - OPEN
   - OPENFILE
   - PRINT USING
   - PUT
   - READFILE
   - REREADFILE
   - RESET
   - RESETFILE
   - REREAD FILE
   - REREAD FILE
   - REWRITE FILE
   - WRITEFILE

4. If any error occurs during the RENUM operation, the editor terminates the RENUM operation and the file being edited remains unchanged.

Responses:

When verification is on, the message EDIT: indicates that the RENUM subcommand completed processing.

REPEAT

Use the REPEAT subcommand to execute the immediately following OVERLAY subcommand (or an X or Y subcommand assigned to invoke OVERLAY) for the specified number of lines or to the end of the file.

The format of the REPEAT subcommand is:

```
REPEAT [n]

n = 1
```
where:

\( n \)

indicates the number of times to repeat the OVERLAY request that immediately follows, beginning with the current line. An asterisk (*) indicates that the request is to be repeated until the end of the file is reached. If neither \( n \) nor * is specified, then only one line is handled. The last line processed becomes the new current line.

Usage Notes:

1. If the next subcommand issued after the REPEAT subcommand is not an OVERLAY subcommand, the REPEAT subcommand is ignored.

2. For an example of a REPEAT subcommand followed by an OVERLAY subcommand, see the discussion of the OVERLAY subcommand.

Responses:

None.

REPLACE

Use the REPLACE subcommand to replace the current line with a specified line or to delete the current line and enter input mode.

The format of the REPLACE subcommand is:

| Replace | [line] |

where:

\( \text{line} \)

specifies an input line that is to replace the current line. If a line is specified, then the editor puts it into the file in place of the current line. If no line is specified, the editor deletes the current line and enters input mode (see Usage Note 2 for exception).

Usage Notes:

1. If the LINEMODE subcommand with a LEFT or RIGHT operand is in effect, then issuing the REPLACE subcommand specifying a line is not valid. If the REPLACE subcommand is used without any operands when LINEMODE is set to LEFT or RIGHT, you are prompted for the next available line number; the first data line you enter replaces the current line number.
2. If you use the REPLACE subcommand with no operands to enter input mode, and the next line you enter is a null line, then the current line is not deleted, and you are returned to edit mode.

3. To stack a REPLACE subcommand in order to enter input mode from a fixed-length EXEC, you should use the &STACK control statement.

Responses:

When verification is on and you issue the REPLACE subcommand with no data line, the message:

INPUT:

indicates that your virtual machine is in input mode.

RESTORE

Use the RESTORE subcommand to restore the settings of EDIT subcommands to their values when the PRESERVE subcommand was last issued or to their default values if a PRESERVE subcommand has not been issued.

The format of the RESTORE subcommand is:

| REStore |

Usage Note:

The settings are restored for the following subcommands:

<table>
<thead>
<tr>
<th>CASE</th>
<th>LONG</th>
<th>TABSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMODE</td>
<td>PROMPT</td>
<td>TRUNC</td>
</tr>
<tr>
<td>FNAME</td>
<td>RECFM</td>
<td>VERIFY</td>
</tr>
<tr>
<td>IMAGE</td>
<td>SERIAL</td>
<td>ZONE</td>
</tr>
<tr>
<td>LINEMODE</td>
<td>SHORT</td>
<td></td>
</tr>
</tbody>
</table>

Responses:

None.
RETURN

Use the RETURN subcommand to return to edit mode from the CMS subset environment. RETURN is not an EDIT subcommand, but is listed here as a companion to the CMS subcommand.

The format of the RETURN command is:

```
RETURN
```

Responses:

When verification is on, the editor responds:

```
EDIT:
```

to indicate that your virtual machine is in edit mode.

REUSE (=)

Use the REUSE subcommand (which can also be specified as =) to stack last in, first out (LIFO) the last EDIT request, except for REUSE or a question mark, and then execute the stacked subcommands.

The format of the REUSE (or =) subcommand is:

```
[REUSE] [subcommand]
```

where:

subcommand specifies any valid EDIT subcommand.

Usage Notes:

1. If the subcommand you enter on the REUSE subcommand line is an invalid subcommand, the editor clears the stack.

2. You can also enter more than one equal sign (=) on a single line, to stack the last issued subcommand more than once. For example:
locate /xyz/
XYZ IS MY FAVORITE
= = = =
I FIRST MET XYZ
XYZ’S NAME IS DERIVED
LAST SAW XYZ
EOF:

the LOCATE subcommand is stacked four times, and then the editor, reading from the stack, executes the four stacked subcommands.

3. You can do the following if you issue a CHANGE subcommand before positioning your current line pointer:

\[ c/xx/yy \]
NOT FOUND
\[ = l/x/ \]
LINE XXXX
LINE YYYY

In this example, the CHANGE request was issued and string1 was not found. The REUSE subcommand stacks the CHANGE subcommand and stacks a LOCATE subcommand in front of it. The LOCATE subcommand is read and executed, followed by the CHANGE subcommand.

4. You can stack an INPUT or REPLACE subcommand in front of a data line you mistakenly entered in edit mode, for example:

roses are red, violets are blue
?EDIT: ROSES ARE RED, VIOLETS ARE BLUE
= input
INPUT:
without cms
i would be, too.

The = subcommand stacks the INPUT subcommand in front of the data line. Reading from the stack, the editor executes the INPUT subcommand, then reads in, as the first line of data, the line beginning with ROSES. The file contains:

ROSES ARE RED, VIOLETS ARE BLUE
WITHOUT CMS
I WOULD BE, TOO.

Responses:

Responses are those that are issued to the stacked subcommands.
SAVE

Use the SAVE subcommand to write the file that is currently being edited onto the disk, without returning control to CMS, and optionally to change the file identifier.

The format of the SAVE subcommand is:

```
SAVE [fn [ft [fm]]]
```

where:

- `fn` indicates the filename of the file to be saved. If you specify only `fn`, then the filetype and filemode are the same.
- `ft` indicates the filetype of the file to be saved.
- `fm` indicates the filemode of the file to be saved.

Usage Notes:

1. If you specify a new file identifier, any existing file with the same file identifier is replaced; no message is issued. The file being edited, if previously written to disk, is not altered.

2. To write a file on disk and terminate the editing session, use the FILE subcommand.

3. If you want to save the contents of a file at regular intervals, use the AUTOSAVE subcommand.

Responses:

When verification is on, the editor displays:

```
EDIT:
```

to indicate the SAVE request completed successfully and you may continue to enter EDIT subcommands.
**SCROLL/SCROLLUP (3270 only)**

Use the SCROLL and SCROLLUP subcommands to scan the contents of a file on a display screen.

SCROLL causes the editor to scan forward through the file; SCROLLUP causes the editor to scan backward through the file.

The format of the SCROLL and SCROLLUP subcommands is:

```
{ Scroll
S[ croll] U[p] }
```

where:

\[ n \]

is a number from 1 to 255 that specifies the number of successive screens of data to be displayed. If an asterisk (*) is specified, the entire file, from the current line to the end or beginning of the file, is displayed. If \( n \) is not specified, 1 is the default.

**Usage Notes:**

1. The SCROLLUP subcommand can be specified by any combination of the truncation of SCROLL and UP; the minimum truncation is SU.

2. The number of lines shifted forward or backward depends on the current verification setting. If the verification setting is 80 characters or less, then a scroll request displays a file in increments equal to the number of lines that can be displayed in the output display area of the screen. If the verification setting is more than 80 characters, then a SCROLL request displays a file in increments equal to half the number of lines that can be displayed in the output area.

Therefore, a single SCROLL on a 3270 Model 2 display terminal is the equivalent of DOWN 20 or DOWN 10, depending on the record length, and SCROLLUP is the equivalent of UP 20 or UP 10.

3. When you use the SCROLL or SCROLLUP subcommands to display more than one screenful, each display is held for one minute, and the screen status area indicates MORE... . To hold the screen display longer, press the Enter key.

To halt scrolling before all the requested screenfuls are displayed, enter the HT Immediate command and press the Cancel key twice.
4. When you begin scrolling from the top of the file, the first screenful contains only the first seven lines. When you scroll to the end of the file, the last screen may duplicate lines displayed in the previous screen.

Responses:

The screen display is shifted forward or backward.

SERIAL

Use the SERIAL subcommand to control the serialization of records in columns 73 through 80.

The format of the SERIAL subcommand is:

```
SERial OFF
    { ON [ incr ]
      ALL seq [ 10 ] }
```

where:

OFF
    indicates that neither serialization numbers nor identifiers are to be placed in columns 73-80.

ON
    indicates that the first three characters of the filename are to be used in columns 73-75 as an identifier.

ALL
    indicates that columns 73-80 are to be used for serialization numbers.

seq
    specifies a three-character identification to be used in columns 73-75.

incr
    specifies the increment for the line number in columns 76-80 (or 73-80). This number also becomes the first line number. If incr is not specified, then 10 is assumed.
Usage Notes:

1. The SERIAL subcommand is valid only for files with fixed-length, 80-character records. To renumber VSBASIC or FREEFORT files, use the RENUM subcommand.

2. The serialization setting is ON, by default, for the following filetypes:

   ASSEMBLE    PLI
   COBOL       PLIOPT
   DIRECT      UPDATE
   FORTRAN     UPDTxxxx
   MACRO

3. When serialization is in effect, records in a file are resequenced each time a FILE, SAVE, or AUTOSAVE request is issued. If you are using line-number editing, you must issue the subcommand:

   linemode off

   before issuing a FILE or SAVE subcommand if you wish the records to be resequenced.

Responses:

If you issue the SERIAL subcommand in a file with a zone column greater than 72, the message:

   END ZONE SET TO 72

is displayed, to indicate that the zone has been changed. If the zone column is 72 or less, but the truncation column is greater than 72, the message:

   TRUNC SET TO 72

is displayed.

SHORT

Use the SHORT subcommand to request the editor to respond to invalid subcommand lines with the short form of the ?EDIT message.

The format of the SHORT subcommand is:

**SHORT**
Usage Notes:

1. When the SHORT subcommand is in effect, the editor responds:
   - to an invalid subcommand line, and responds:
   - $ to an invalid macro request.

2. To resume displaying the long form of the ?EDIT message, use the LONG subcommand.

Responses:

None.

STACK

Use the STACK subcommand to stack data lines or EDIT subcommands in the console stack for subsequent reading.

The format of the STACK subcommand is:

```
STACK [ n 
  subcommand ]

where:

n
indicates the number of lines to be stacked beginning with the current line. If a number or a subcommand is not specified, then one line is assumed by default. A maximum of 25 lines can be stacked.

If the current line pointer is at the top of the file, then n-1 lines are stacked. If fewer than n lines remain in the file, only the lines remaining are stacked.

subcommand
specifies an EDIT subcommand to be stacked.

0
stacks a null line.
Usage Notes:

1. STACK subcommands are used to write edit macros, to stack lines from a file so that they can be moved around, or to stack additional subcommands.

2. All lines stacked with the STACK subcommand are stacked FIFO (first in, first out).

3. The length of input lines that are stacked is determined by the current TRUNC setting. The maximum length, however, is 130 characters.

Responses:

None. If you issue the STACK subcommand to stack an EDIT subcommand line, the stacked subcommand is executed immediately; responses are those to the stacked subcommands, if any.

TABSET

Use the TABSET subcommand to set logical tab stops for a file.

The format of the TABSET subcommand is:

<table>
<thead>
<tr>
<th>TABset</th>
<th>nl [n2 ... nn]</th>
</tr>
</thead>
</table>

where:

nl [n2... nn] indicates column positions for logical tab settings. You may specify up to 25 numbers, separated from each other by at least one blank. nl indicates the first column in the file that may contain data.

Usage Notes:

1. The editor assigns the following tab settings by default:

<table>
<thead>
<tr>
<th>Filetypes</th>
<th>Default Tab Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM3705</td>
<td>1, 10, 16, 31, 36, 41, 46, 69, 72, 80</td>
</tr>
<tr>
<td>ASSEMBLE</td>
<td>2, 6, 11, 16, 21, 26, 31, 36, 41, 46, 61, 71, 80</td>
</tr>
<tr>
<td>MACRO</td>
<td>1, 7, 10, 15, 20, 25, 30, 80</td>
</tr>
<tr>
<td>UPDATE</td>
<td>1, 7, 10, 15, 20, 25, 30, 80</td>
</tr>
<tr>
<td>UPDTxxxx</td>
<td>1, 7, 10, 15, 20, 25, 30, 80</td>
</tr>
<tr>
<td>AMSERV</td>
<td>1, 7, 10, 15, 20, 25, 30, 80</td>
</tr>
<tr>
<td>FORTRAN</td>
<td>1, 7, 10, 15, 20, 25, 30, 80</td>
</tr>
</tbody>
</table>

Appendix C. Edit Subcommands and Macros
## Responses:

None.

### TOP

Use the TOP subcommand to move the line pointer to the top of the file. The null top line becomes the current line.

The format of the TOP subcommand is:

```
TOP
```

## Responses:

When verification is on, the message:

```
TOP:
```

is displayed.
Display Mode Considerations

When you are using a display terminal, if you specify TOP and verification is on, the current line (see Figure 29) contains the characters TOF (indicating the top of the file), the lines preceding it are blank, and the rest of the screen's output display area contains the first lines of the file.

TRUNC

Use the TRUNC subcommand to change the truncation column of records or to display the current truncation column setting.

The format of the TRUNC subcommand is:

<table>
<thead>
<tr>
<th>TRUNC</th>
<th>[ n ]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[ * ]</td>
</tr>
</tbody>
</table>

where:

n indicates the column at which truncation is to occur. If n is specified as an asterisk (*), the truncation column is set to the record length for the filetype.

Usage Notes:

1. The editor assigns the following truncation setting by default:

<table>
<thead>
<tr>
<th>Truncation Column</th>
<th>Filetypes</th>
</tr>
</thead>
<tbody>
<tr>
<td>71</td>
<td>ASSEMBLE, MACRO, UPDATE, UPDTxxxx</td>
</tr>
<tr>
<td>72</td>
<td>AMSERV, COBOL, DIRECT, FORTRAN, PLI, PLIOPT</td>
</tr>
</tbody>
</table>

   Record Length All Others

2. The truncation value is used by the INPUT, REPLACE, STACK, and OVERLAY subcommands also, and, for display terminals in display mode, the CHANGE subcommand when it is used with no operands.

3. If your virtual machine is in input mode and you enter a line that is longer than the current truncation setting, the message:

   \textit{TRUNCATED}

   is displayed along with a display of the truncated line. Your virtual machine is still in input mode.
Responses:

When you enter the TRUNC subcommand with no operands, the editor displays the current setting.

**TYPE**

Use the TYPE subcommand to display all or any part of a file at the terminal.

The format of the TYPE subcommand is:

```
Type  [ m  n ]
   [   * * ]
```

**where:**

- **m** indicates the number of lines to be displayed, beginning with the current line. An asterisk (*) indicates all lines between the current line and the end of the file. If m is omitted, only one line is displayed. If the number of lines specified exceeds the number remaining in the file, displaying stops at the end of the file.

- **n** indicates the column at which displaying is to stop, overriding the current end column for verification. If n is specified as an asterisk (*), it indicates that displaying is to take place for the full record length.

**Usage Notes:**

1. Use the TYPE subcommand to display lines when you are editing a file with verification off.

2. If you display one line, the current line pointer does not move; if you display more than one line, the current line is positioned at the last line displayed, or at the end of the file if you specified an asterisk (*).

3. If you have set an end verification column to a value less than the record length, and you want to display an entire record, enter:
   
   ```
type 1 *
   ```

4. If you do not specify an end column, the length of the line(s) displayed is determined by the current end verification setting. If you are using right-handed line-number editing on a typewriter terminal or a display terminal in line mode, the line numbers are displayed on the left.
Responses:

The requested lines are displayed.

Display Mode Considerations

Since the TYPE subcommand was designed for printing terminals, it is of
marginal value on a display terminal, except when you use line mode.
However, if the display screen is interrupted by communication from the
control program (CP), you should use the TYPE subcommand to restore the
full screen display.

UP

Use the UP subcommand to reposition the current line pointer toward the
beginning of the file.

The format of the UP subcommand is:

\[
\text{Up} \quad \left[ n \right]
\]

where:

\[ n \]

indicates the number of lines the pointer is to be moved toward the
beginning of the file. If a number is not specified, then the pointer is
moved up only one line. The line pointed to becomes the new current
line.

Usage Note:

UP is equivalent to BACKWARD.

Responses:

When verification is on, the line pointed to is displayed at your terminal. If
the UP subcommand causes the current line pointer to move beyond the
beginning of the file, the following message is displayed:

TOF:
VERIFY

Use the VERIFY subcommand to set or display the current verification setting.

The format of the VERIFY subcommand is:

```
Verify [ ON [ startcol ] endcol ]
```

where:

ON
specifies that lines located, altered, or changed are displayed, and changes between edit and input mode are indicated. ON is the initial setting.

OFF
specifies that lines that are located, altered, or changed are not displayed, and changes between edit and input mode are not indicated.

`startcol`
indicates the column in which verification is to begin, when verification is on. The default is column 1. `startcol` must not be greater than the record length nor greater than `endcol`.

`endcol`
indicates the last column to be verified, when verification is on. `endcol` must not be greater than the record length. If `endcol` is specified as an asterisk (*), each record is displayed to the end of the record.

Usage Notes:

1. If you issue the VERIFY subcommand with only one operand, that operand is assumed to be the `endcol` operand. For example, if you issue VERIFY 10, verification occurs in columns 1 through 10.

2. The editor assigns the following settings, by default:

<table>
<thead>
<tr>
<th>Filetypes</th>
<th>Verification End Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMSERV, ASSEMBLE, COBOL, DIRECT, FORTRAN, MACRO, PLI, PLIOPT, UPDATE, UPDTxxxx</td>
<td>Column 72</td>
</tr>
</tbody>
</table>
Responses:

If you issue the VERIFY subcommand with no operands, the current startcol and endcol settings are displayed, regardless of whether verification is on or off.

X or Y

Use the X or Y subcommands to assign a given EDIT subcommand to be executed whenever X or Y is entered, or to execute the previously assigned subcommand a specified number of times.

The format of the X and Y subcommands is:

\[
\{ X \} \quad \begin{array}{l}
\{ Y \} \\
\text{subcommand} \\
\quad n \\
\quad 1
\end{array}
\]

where:

subcommand

indicates any EDIT subcommand line. The editor assumes that you have specified a valid EDIT subcommand, and no error checking is done.

n

indicates the number of times the previously assigned subcommand is to be executed. If X or Y is entered with no operands, 1 is assumed.

Usage Notes:

1. Advancement of the current line pointer depends upon the EDIT subcommand that has been assigned to X or Y. If a number or a subcommand is not specified, the previously assigned subcommand is executed once.

2. X and Y are initially set to null strings. If you enter X or Y without having previously assigned a subcommand to it, the editor issues the ?EDIT error message.
3. You can use the X and Y subcommands in many instances where you must repeat a subcommand line many times while editing a file, but the situation does not lend itself to a global request. For example, if you assign X to a LOCATE and Y to a CHANGE subcommand, issue:

```
x
```

to execute the LOCATE request, and after examining the line, you can change it and continue searching, by entering the Y subcommand followed by the X subcommand:

```
y#x
```
or just continue searching:

```
x
```

**Responses:**

Responses are issued for the EDIT subcommands that are assigned to X and Y, in accordance with the current verification setting.

**ZONE**

Use the ZONE subcommand to specify the columns of each record (starting position and ending position) to be scanned when the editor searches for a character string or to display the current ZONE settings.

The format of the ZONE subcommand is:

```
Zone [ firstcol lastcol ]

  firstcol
  lastcol

where:

firstcol
  indicates the near zone column of each record to be scanned. If firstcol is specified as an asterisk (*), the default is column 1.

lastcol
  indicates the end zone column of each record to be scanned. If lastcol is specified as an asterisk (*), the default is the record length.
```
Usage Notes:

1. The editor assigns the following settings by default:

<table>
<thead>
<tr>
<th>Filetype</th>
<th>Near Zone (column)</th>
<th>End Zone (column)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASSEMBLE, MACRO UPDATE, UPDTxxx</td>
<td>1</td>
<td>71</td>
</tr>
<tr>
<td>AMSERV, PLI, PLIOPT</td>
<td>2</td>
<td>72</td>
</tr>
<tr>
<td>COBOL, DIRECT, FORTRAN</td>
<td>1</td>
<td>72</td>
</tr>
<tr>
<td>BASIC, VSBASIC</td>
<td>7</td>
<td>Record Length</td>
</tr>
<tr>
<td>FREEFORT</td>
<td>9</td>
<td>Record Length</td>
</tr>
<tr>
<td>Others</td>
<td>1</td>
<td>Record Length</td>
</tr>
</tbody>
</table>

2. The ZONE settings are used by the ALTER, CHANGE, and LOCATE subcommands to define the columns that will be scanned. If you specify a character string longer than the zone, you receive the message:

ZONE ERROR

and the subcommand is not executed.

3. If you issue a CHANGE subcommand that increases the length of a line beyond the end zone setting, the line is truncated.

4. You can use the ZONE subcommand to protect data in particular columns, for example:

```
edit newfile memo
NEW FILE:
EDIT:
zone
  1 80
zone 10 20
input the zone is now set for columns 10-20

EDIT:
change /o/*/
the zone is n*w set for columns 10-20
```

Note that the LOCATE and CHANGE subcommands operated on the word now, not the word zone, because scanning started in position 10, not in position 1.
Responses:

When you enter the ZONE subcommand without specifying zone settings, the editor displays the current setting.

? (QUESTION MARK)

Use the ? subcommand to display the last EDIT subcommand executed except for a REUSE (=) or ? (question mark) subcommand.

The format of the ? subcommand is:

```
?  
```

Usage Note:

After an X, Y, or = subcommand, the last EDIT subcommand is the subcommand that was executed as a result of issuing the X or Y subcommand.

Display Mode Considerations

When you issue the ? subcommand using a 3270 in display mode, the last EDIT subcommand that was executed is redisplayed in the user input area. Press the Enter key to execute it again; you may modify the line before reentering it.

nnnnn

Use the nnnnn subcommand to enter and locate lines when you are using line-number editing.

The format of the nnnnn subcommand is:

```
{{nnnn  
nnnnnnn}} [ text ]
```

where:
nnnnn
indicates a line number between 0 and 99999 if the filetype is BASIC or VSBASIC, or a line number between 0 and 99999999 if the filetype is FREEFORT.

text
specifies a line of text to be inserted into the file at the specified line number. If a line with that number already exists, it is replaced. If no text line is specified, the current line pointer is positioned at the line number specified.

Usage Note:

The nnnnn subcommand is valid only when you are using line-number editing; that is, you have issued the LINEMODE subcommand using the RIGHT or LEFT operand. Line-number editing is the default for VSBASIC and FREEFORT files.

Responses:

When you issue the nnnnn subcommand with no operands, the line with the specified line number is displayed. If the line is not found, the editor displays the message:

LINE NOT FOUND

and the current line pointer is set at the largest line number that does not exceed nnnnn.

EDIT Macros

Edit macros are CMS EXEC files that execute sequences of EDIT subcommands. The following edit macros are supplied with VM/SP for your convenience.

$DUP

Use the $DUP to duplicate the current line.

The format of the $DUP macro is:

\[
\text{$DUP} \quad \begin{bmatrix} n \\ 1 \end{bmatrix}
\]

where:

n
indicates the number of times you want to duplicate the line; the maximum value you can specify is 25. If n is omitted, the current line is duplicated once.

**Usage Notes:**

1. The last copy of the line duplicated becomes the new current line.

2. If you use the logical line end symbol (#) to stack additional subcommands on the same line with the $DUP edit macro those subcommands are cleared from the console stack and the message:

   STACKED LINES CLEARED BY $DUP

   is issued. The stacked subcommand(s) are not executed.

3. Because it uses console functions, $DUP cannot be used when duplicating records containing binary zeros or nonprintable characters. Truncated duplicate records will result.

4. When using line-number editing, you can insert duplicate lines between existing numbered lines if the interval between line numbers is large enough. Execution of $DUP stops after the last valid line number has been assigned. You can renumber your file to increase the interval between line numbers.

5. Because it uses the STACK EDIT subcommand, $DUP can duplicate a maximum of 130 characters in one line. Longer lines are truncated.

**Responses:**

The last line duplicated (the new current line) is displayed.

**$MOVE**

Use the $MOVE edit macro to move one or more lines from one place in a file to another place.

The format of the $MOVE macro is:

| $MOVE | n  | { UP m  | DOWN m | TO label  |

*where:*
indicates the number of records you want to move, beginning with the current line. The maximum number of lines you can move is 25.

UP m indicates that you want to move the lines toward the top of the file, m lines above the current line.

DOWN m indicates that you want to move the lines toward the end of the file, m lines below the last line you are going to move.

TO 'label indicates that you want the lines inserted following the specified label. The label must be one to eight uppercase characters and must start in column 1.

Usage Notes:

1. The last line moved becomes the new current line.

2. If the label is not found or if the DOWN value exceeds the number of lines remaining before end of file, the lines are inserted at the end of the file. If the UP value exceeds the number of lines remaining before top of file, the lines are inserted at the top of the file.

3. If you use the logical line end symbol (#) to stack additional subcommands on the same line with the $MOVE request, those subcommands are cleared from the console stack and the message:

STACKED LINES CLEARED BY $MOVE

is displayed. The stacked subcommands are not executed.

4. Because it uses console functions, $MOVE will truncate duplicated records containing binary zeros or nonprintable characters.

5. Because it uses the STACK EDIT subcommand, $MOVE can move a maximum of 130 characters in one line. Longer lines are truncated.

Responses:

When verification is on, the last line moved is displayed.
### Appendix D. Edit Reserved Filetype Defaults

<table>
<thead>
<tr>
<th>Filetype</th>
<th>RECFM</th>
<th>LRECL</th>
<th>ZONE</th>
<th>TRUNC</th>
<th>VERIFY</th>
<th>SERIAL</th>
<th>TABS</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>F</td>
<td>80</td>
<td>1</td>
<td>*</td>
<td>1</td>
<td>*</td>
<td>OFF</td>
<td>1,6,11,16,21,26,31,36,41,46,51,61,71,81,91,101,111,121,131</td>
</tr>
<tr>
<td>AMSERV</td>
<td>F</td>
<td>80</td>
<td>2</td>
<td>72</td>
<td>72</td>
<td>1</td>
<td>72</td>
<td>OFF</td>
</tr>
<tr>
<td>ASSEMBLE</td>
<td>F</td>
<td>80</td>
<td>1</td>
<td>71</td>
<td>71</td>
<td>1</td>
<td>72</td>
<td>ON</td>
</tr>
<tr>
<td>ASM3705</td>
<td>F</td>
<td>80</td>
<td>1</td>
<td>71</td>
<td>71</td>
<td>72</td>
<td>ON</td>
<td>1,10,16,31,36,41,46,69,72,80</td>
</tr>
<tr>
<td>BASIC</td>
<td>F</td>
<td>80</td>
<td>7</td>
<td>*</td>
<td>1</td>
<td>*</td>
<td>L/L</td>
<td>7,10,15,20,25,30,80</td>
</tr>
<tr>
<td>BASDATA</td>
<td>F</td>
<td>80</td>
<td>7</td>
<td>*</td>
<td>1</td>
<td>*</td>
<td>L/L</td>
<td>7,10,15,20,25,30,80</td>
</tr>
<tr>
<td>COBOL</td>
<td>F</td>
<td>80</td>
<td>1</td>
<td>72</td>
<td>72</td>
<td>1</td>
<td>72</td>
<td>ON</td>
</tr>
<tr>
<td>DIRECT</td>
<td>F</td>
<td>80</td>
<td>1</td>
<td>72</td>
<td>72</td>
<td>1</td>
<td>72</td>
<td>ON</td>
</tr>
<tr>
<td>EXEC</td>
<td>V</td>
<td>80</td>
<td>1</td>
<td>*</td>
<td>1</td>
<td>1</td>
<td>*</td>
<td>OFF</td>
</tr>
<tr>
<td>FREEFORT</td>
<td>V</td>
<td>81</td>
<td>9</td>
<td>*</td>
<td>1</td>
<td>1</td>
<td>L/L</td>
<td>9,15,18,23,28,33,38,80</td>
</tr>
<tr>
<td>FORTRAN</td>
<td>F</td>
<td>80</td>
<td>1</td>
<td>72</td>
<td>72</td>
<td>1</td>
<td>72</td>
<td>ON</td>
</tr>
<tr>
<td>LISTING</td>
<td>V</td>
<td>121</td>
<td>1</td>
<td>*</td>
<td>1</td>
<td>1</td>
<td>*</td>
<td>OFF</td>
</tr>
<tr>
<td>MACRO</td>
<td>F</td>
<td>80</td>
<td>1</td>
<td>71</td>
<td>71</td>
<td>71</td>
<td>72</td>
<td>ON</td>
</tr>
<tr>
<td>MEMO</td>
<td>F</td>
<td>80</td>
<td>1</td>
<td>*</td>
<td>1</td>
<td>1</td>
<td>*</td>
<td>OFF</td>
</tr>
<tr>
<td>PL/I</td>
<td>F</td>
<td>80</td>
<td>2</td>
<td>72</td>
<td>72</td>
<td>1</td>
<td>72</td>
<td>ON</td>
</tr>
<tr>
<td>OPT</td>
<td>F</td>
<td>80</td>
<td>2</td>
<td>72</td>
<td>72</td>
<td>1</td>
<td>72</td>
<td>ON</td>
</tr>
<tr>
<td>SCRIPT</td>
<td>V</td>
<td>132</td>
<td>1</td>
<td>*</td>
<td>1</td>
<td>*</td>
<td>OFF</td>
<td>(IMAGE setting is CANON.)</td>
</tr>
<tr>
<td>UPDATE</td>
<td>F</td>
<td>80</td>
<td>1</td>
<td>71</td>
<td>71</td>
<td>72</td>
<td>ON</td>
<td>1,10,16,31,36,41,46,69,72,80</td>
</tr>
<tr>
<td>UPDxxxx</td>
<td>F</td>
<td>80</td>
<td>1</td>
<td>71</td>
<td>71</td>
<td>72</td>
<td>ON</td>
<td>1,10,16,31,36,41,46,69,72,80</td>
</tr>
<tr>
<td>VSBIASIC</td>
<td>F</td>
<td>80</td>
<td>7</td>
<td>*</td>
<td>1</td>
<td>1</td>
<td>L/L</td>
<td>7,10,15,20,25,30,80</td>
</tr>
<tr>
<td>VSBDATA</td>
<td>V</td>
<td>132</td>
<td>1</td>
<td>*</td>
<td>1</td>
<td>*</td>
<td>OFF</td>
<td>1,6,11,16,21,26,31,36,41,46,51,61,71,81...131</td>
</tr>
</tbody>
</table>

* indicates that the ZONE, TRUNC, or VERIFY setting is equal to the current record length.
L/L indicates that the LINEMODE setting is LEFT, with serial numbers on the left.

Figure 39. Default EDIT Subcommand Settings for CMS Reserved Filetypes
This appendix describes the formats, usage rules, and default values for CMS EXEC control words, including:

- Control statements
- Built-in functions
- Special variables

An EXEC procedure is a CMS file that contains a sequence of CP and CMS commands and/or EXEC control statements. Control statements determine the logic flow for EXEC, provide terminal communications, and may be used to manipulate CMS disk files. For an introduction to the EXEC facilities, and for complete tutorial information, including examples, consult the VM/SP CMS User's Guide. Refer to the VM/SP System Product Interpreter Reference for information on the System Product Interpreter, or to the VM/SP EXEC 2 Reference for information on EXEC 2.

EXEC procedures may be invoked with the EXEC command, described in "Chapter 2. CMS Commands." You may also execute an EXEC procedure by specifying its filename, as long as the implied EXEC (IMPEX) function is in effect. You can determine the setting of IMPEX by entering QUERY IMPEX. You can change the setting of IMPEX with the SET command.

### The Assignment Statement

Use the assignment statement in an EXEC procedure to assign a value to a variable symbol. Variable symbols may be tested and manipulated to control the execution of an EXEC procedure.

The format of the assignment statement is:

```
& variable = { string ae function X'xxxxxx }
```

*where:*
&variable
indicates the variable symbol that is assigned the specified value. A variable may contain a maximum of eight alphameric and national characters, including the initial ampersand, which is required. Except in the EXEC special variables &* and &DISK*, a variable must not contain any special characters.

string
is a data item of up to eight characters. It may also be a variable symbol or null. Whether a numeric string is treated as numeric or character data depends on how it is used in the EXEC. If a string containing variable symbols expands to more than eight characters, it is truncated. If the string consists of eight X'FF' characters, the variable is set to a null string.

ae
is an arithmetic expression consisting of a sequence of data items that possess positive or negative integral values and are separated by plus or minus signs:
\[ &1 - 4 + &CALC - 6 \]

function
is an EXEC built-in function followed by at least one token.

X'xxxxxx
indicates up to six hexadecimal digits to be converted to decimal before assignment. For example:
\[ &A = X'CO \]
results in &A having the decimal value 192.

Hexadecimal conversion is not performed unless you have used the &HEX ON control statement.

Variable Substitution

All variable symbols occurring in executable statements are substituted before the statement is executed. An executable statement is (1) a CMS command line, or (2) an EXEC control statement (including assignment statements).

Variable substitution is performed on all symbols on the left-hand side of an assignment statement, except the leftmost variable. For example:
\[ &I = 2 \]
\[ &X&I = 5 \]
sets &X2 to 5.

If a variable on the left-hand side of an assignment statement has already been assigned a value, it is replaced by the new value specified in the assignment statement.
If the special form, \texttt{X'&symbol}, is used, the specified symbol is converted to its hexadecimal equivalent. For example:

\begin{verbatim}
&\texttt{A} = 192
&\texttt{TYPE X'&A}
\end{verbatim}

results in the display:

\texttt{CO}

If a variable symbol that has not been defined is used in an executable statement the symbol is set to a null token and ignored. In some instances this may cause an EXEC processing error.

\textbf{Tokens}

All executable statements in an EXEC are scanned into eight-character tokens, and padded or truncated as necessary. Tokens are formed of words delimited by blanks and parentheses. If there is no blank before or after a parenthesis, one is added in either case. If more than one blank separates a word or a parenthesis from another, the extra blanks are removed from the line. For example, the line:

\begin{verbatim}
&\texttt{TYPE THIS IS AN EXAGGERATED (MESSAGE}
\end{verbatim}

scans as:

\begin{verbatim}
&\texttt{TYPE THIS IS AN EXAGGERA ( MESSAGE}
\end{verbatim}

Variable symbols are substituted after each line is scanned, and each token is scanned repeatedly until all symbols in it are substituted.

In an executable statement, a token beginning with the character X'FF" (or a variable to which such a token is assigned as a value) usually prevents the processing of data following it on the same line. However, if an assignment statement sets a variable to eight X'FF" characters, data following the variable in an executable statement is processed.

\textbf{&ARGS}

Use the \texttt{&ARGS} control statement to redefine the value of one or more of the special variables, \&1 through \&30.

The format of the \texttt{&ARGS} control statement is:

\begin{verbatim}
&\texttt{ARGS} [arg1 [arg2 ... [arg30] ] ]
\end{verbatim}

where:
Usage Notes:

1. To enter an argument list from the terminal, use the &READ ARGS control statement.
2. An &ARGS control statement resets the values of the &INDEX, &, and $ special variables.

&BEDEMMSG

Use the &BEDEMMSG control statement to introduce one or more unscanned lines to be edited as VMjSP error messages. The list of lines to be displayed must be terminated by an &END control statement, which must appear beginning in column 1.

The format of the &BEDEMMSG control statement is:

| &BEDEMMSG | [ALL] |

where:

ALL

specifies, for fixed-length EXEC files, that the entire line (to a maximum of 130 characters) is to be displayed.

Usage Notes:

1. To qualify for error message editing, the first data item on each line following the &BEDEMMSG control statement must be seven characters long, in the format:

```
mmmnnns
```

where:

mmmnnns is a six-character message identification you can supply for the error message. Standard VMjSP error messages use a three-character module code (mmm) and a three-character message number (nnn).
s indicates the severity code. The following codes qualify the message for error message editing:

<table>
<thead>
<tr>
<th>Code</th>
<th>Message Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Informational</td>
</tr>
<tr>
<td>E</td>
<td>Error</td>
</tr>
<tr>
<td>W</td>
<td>Warning</td>
</tr>
</tbody>
</table>

When the severity code is E, I, or W, the message is displayed in accordance with the CP EMSG setting (ON, OFF, CODE, or TEXT). You can change this setting with the CP SET command, described in VM/SP CP Command Reference.

2. When you use the &BEGEMSG control statement to display error messages, the character string “DMS” is inserted in front of the seven-character message identification. For example, if the EMSG setting is ON, the lines:

   &BEGEMSG
   TEST01E INSURMOUNTABLE ERROR
   &END

result in the display:

   DMSTEST01E INSURMOUNTABLE ERROR

   Note: Since the maximum length of a line that you can display at your terminal is 130 characters, the insertion of the characters DMS will cause lines greater than 127 characters long to be truncated.

3. Messages that are displayed as the result of an &BEGEMSG control statement are not scanned by the EXEC interpreter. Therefore, no variable substitution is performed and no data items are truncated. To display variable data, use the &EMSG control statement.

---

**&BEGPUNCH**

Use the &BEGPUNCH control statement to delimit the beginning of a list of one or more data lines to be spooled to your virtual punch. The list of lines to be punched is terminated by the control statement &END, which must occur beginning in column 1.

The format of the &BEGPUNCH control statement is:

| &BEGPUNCH | [ALL] |

*where:*
ALL
specifies that data occupying columns 73 through 80 should be punched. If ALL is not specified, input records are truncated at column 72 and columns 73 through 80 of the output record are padded with blanks.

Usage Notes:

1. Lines that are punched as the result of an &BEGPUNCH control statement are not scanned by the EXEC interpreter. Therefore, no variable substitution is performed and no data items are truncated. To punch variable data, you must use the &PUNCH control statement.

2. When you are finished punching lines in an EXEC procedure, you should use the CP CLOSE command to close your virtual punch.

&BEGSTACK

Use the &BEGSTACK control statement to delimit the beginning of a list of one or more data lines to be placed in the console stack. The list of lines to be stacked is terminated by the control statement &END which must occur beginning in column 1.

The format of the &BEGSTACK control statement is:

```
&BEGSTACK [ FIFO ] [ ALL ]
```

where:

FIFO
specifies that the lines that follow are to be stacked on a first in, first out basis. This is the default value.

LIFO
specifies that the lines that follow are to be stacked on a last in, first out basis.

ALL
specifies, for fixed-length EXEC files, that the entire line (to a maximum of 130 characters) is to be stacked. If ALL is not specified, the lines are truncated in column 72.
Usage Notes:

1. Lines that are stacked as the result of an &BEGSTACK control statement are not scanned by the EXEC interpreter. Therefore, no variable substitution is performed, and data items are not truncated. To stack variable data, you must use the &STACK control statement.

2. To stack a null line in an EXEC file you must use the &STACK control statement. A null line following an &BEGSTACK control statement is interpreted as a line of blanks. To stack an INPUT, REPLACE, or CHANGE subcommand to enter input mode from a fixed-length EXEC, you should use the &STACK control statement.

&BEGTYPE

Use the &BEGTYPE control statement to delimit the beginning of a list of one or more data lines to be displayed at the terminal. The list of lines to be displayed is terminated by the control statement &END, which must occur beginning in column 1.

The format of the &BEGTYPE control statement is:

```
&BEGTYPE [ALL]
```

where:

ALL

specifies, for fixed-length EXEC files, that data occupying columns 73 through 130 is to be displayed. If ALL is not specified, the lines are truncated at column 72.

Usage Note:

Lines that are displayed as the result of an &BEGTYPE control statement are not scanned by the EXEC interpreter. Therefore, no variable substitution is performed, and data items are not truncated. To display variable data, you must use the &TYPE control statement.

&CONTINUE

Use the &CONTINUE control statement to instruct the EXEC interpreter to process the next statement in the EXEC file.

The format of the &CONTINUE control statement is:
Usage Note:

&CONTINUE is generally used with an EXEC label (for example, -LAB &CONTINUE) to provide a branch address for &ERROR, &GOTO, and other branching statements. &CONTINUE is the default action taken when an error is detected in processing a CMS command.

&CONTROL

Use the &CONTROL control statement to specify the amount of data to be displayed in the execution summary of an EXEC.

The format of the &CONTROL control statement is:

\[
&CONTROL \quad [ \text{OFF} \quad \text{ERROR} \quad \text{CMS} \quad \text{ALL} ] \quad [ \text{MSG} \quad \text{NOMSG} \quad \text{TIME} \quad \text{NOTIME} \quad \text{PACK} \quad \text{NOPACK} ]
\]

where:

OFF
- suppresses the display of CMS commands and EXEC control statements as they execute and of any return codes that may result from CMS commands.

ERROR
- displays only those CMS commands that result in an error and also displays the error message and the return code.

CMS
- displays each CMS command as it is executed and all nonzero return codes.

ALL
- displays CMS commands and EXEC executable statements as they execute as well as any nonzero return codes from CMS commands.
MSG
does not suppress the “FILE NOT FOUND” message if it is issued by the following commands when they are invoked from an EXEC procedure: ERASE, LISTFILE, RENAME, or STATE.

NOMSG
suppresses the “FILE NOT FOUND” message if it is issued when the ERASE, LISTFILE, RENAME, or STATE commands are invoked from an EXEC procedure.

TIME
includes the time-of-day value with each CMS command printed in the execution summary; for example:

14:36:30 TYPE A B

This operand is effective only if CMS or ALL is also specified.

NOTIME
does not include the time-of-day value with CMS commands printed in the execution summary.

PACK
packs the lines of the execution summary so that surplus blanks are removed from the displayed lines.

NOPACK
does not pack the lines of the execution summary.

Usage Notes:

1. The execution summary may consist of CP and CMS commands, responses, error messages, and return codes, as well as EXEC control statements and assignment statements. When EXEC statements are displayed, they are displayed in their scanned format, with all variable symbols substituted.

2. Each operand remains set until explicitly reset by another &CONTROL statement that specifies a conflicting operand. When &CONTROL is used with no operands, all operands are reset to their default values.

3. There is no global setting for &CONTROL. When an EXEC is nested within another EXEC, the execution summary is controlled by the nested EXEC’s &CONTROL setting. When control returns to the outer EXEC, the original &CONTROL setting is restored.
&EMSG

Use the &EMSG control statement to display a line of tokens to be edited as a VM/SP error message.

The format of the &EMSG control statement is:

<table>
<thead>
<tr>
<th>&amp;EMSG</th>
<th>mmmnnns [tok1 ... [tokn]]</th>
</tr>
</thead>
</table>

where:

mmnnnn

is a six-character identification you may supply for the error message. Standard VM/SP messages are coded using a three-character module code (mmm) and a three-character message number (nnn).

s

indicates the severity code. The following codes qualify the message for error message editing:

<table>
<thead>
<tr>
<th>Code</th>
<th>Message Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Information</td>
</tr>
<tr>
<td>E</td>
<td>Error</td>
</tr>
<tr>
<td>W</td>
<td>Warning</td>
</tr>
<tr>
<td>R</td>
<td>Response</td>
</tr>
</tbody>
</table>

tok1...[tokn]

is the text of the message to be displayed.

Usage Notes:

1. When the severity code is I, E, or W, the message is displayed in accordance with the CP EMSG setting (ON, OFF, CODE, or TEXT). You can change the setting with the CP SET command, described in VM/SP CP Command Reference.

2. When an &EMSG code is displayed, it is prefixed with DMS. For example, the statement:

   &EMSG ERROR1E INVALID ARGUMENT

   displays as follows when the EMSG setting is ON:

   DMSERROR1E INVALID ARGUMENT

3. To display an error message with unsubstituted data, or to display a line with words of more than eight characters, use the &BEGEMSG control statement.
**&END**

Use the &END control statement to terminate a list of one or more lines that began with an &BEGEMSG, &BEGPUNCH, &BEGSTACK, or &BEGTYPE control statement.

The format of the &END control statement is:

```
&END
```

The word "&END" must be entered beginning in column 1.

**&ERROR**

Use the &ERROR control statement to specify the action to be taken when a CMS command results in an error and returns with a nonzero return code. Put the &ERROR statement prior to the command where the action will be required.

The format of the &ERROR control statement is:

```
&ERROR [executable-statement]
&CONTINUE
```

*where:*

*executable-statement*

specifies any executable statement, which may be an EXEC control statement or assignment statement or a CMS command. If you specify an EXEC control statement that transfers control to another line in the EXEC, execution continues at the specified line. Otherwise, execution continues with the line following the CMS command line that caused the error.

**Usage Notes:**

1. If your EXEC does not contain an &ERROR control statement, then the default is &CONTINUE; that is, EXEC processing is to continue with the line following the CMS command that caused the error. You can use &ERROR &CONTINUE to reset a previous &ERROR statement.

2. The words following an &ERROR control statement are not scanned until a CMS command returns a nonzero return code. Therefore, if you specify an invalid EXEC statement, the error is not detected until a
CMS command failure triggers the &ERROR statement. If the &ERROR statement executes a CMS command that also results in an error, EXEC processing is terminated.

&EXIT

Use the &EXIT control statement to terminate processing the EXEC file. If the exit is taken from a first-level EXEC procedure, control passes to CMS. If the exit is taken from a nested EXEC procedure, control passes to the calling EXEC procedure.

The format of the &EXIT control statement is:

```
&EXIT [ return-code ]
```

where:

return-code

specifies a numeric value, which may be a variable symbol, to be used as the return code from this EXEC. If the return code is not specified, it defaults to 0.

Usage Notes:

1. If control is returned to CMS, the CMS ready message indicates the return code value. Thus, the statement:

   &EXIT 12

results in the ready message:

   R(00012):T=0/02 15:32:34

2. If you specify:

   &EXIT &RETCODE

the return code value displayed is the return code from the most recently executed CMS command.
&GOTO

Use the &GOTO control statement to transfer control to a specific location in the EXEC procedure. Execution then continues at the location that is branched to.

The format of the &GOTO control statement is:

```
&GOTO {
  TOP
  line-number
  -label
}
```

where:

TOP
transfers control to the first line of the EXEC file.

line-number
transfers control to a specific line in the EXEC file.

-label
transfers control to a specific label in the EXEC file. A label must begin with dash (-), and it must be the first token on a line. The remainder of the line may contain an executable statement or it may be null.

Usage Notes:

1. Scanning for an EXEC label starts on the line following the &GOTO statement, goes to the end of the file, then to the top of the file, and (if unsuccessful) ends on the line above the &GOTO statement. If more than one statement in the file has the same label, the first one encountered by these rules satisfies the search.

2. To provide a branch up or down a specific number of lines in the EXEC, use the &SKIP control statement.

&HEX

Use the &HEX control statement to initiate or inhibit hexadecimal conversion in an EXEC procedure.

The format of the &HEX control statement is:
where:

ON
indicates that tokens beginning with the string X' are to be interpreted as hexadecimal notation.

OFF
indicates that no hexadecimal conversion is to be done by EXEC. OFF is the default setting.

Usage Notes:

1. You should use the &HEX control statement when you want to display a hexadecimal value. For example:

   &HEX ON
   &TYPE X'40
   &HEX

   results in the display:

   28

   If you did not use the &HEX ON control statement, the &TYPE statement would result in the display:

   X'40

2. To convert a hexadecimal value to its decimal equivalent, use an assignment statement.

3. The VM/SP CMS User's Guide should be consulted for details and examples of correct usage of EXEC control statements with &HEX ON in effect.

&IF

Use the &IF control statement to test a condition in an EXEC procedure and to perform a particular action if the result is true. If the test is invalid, execution continues with the statement following the &IF control statement.

The format of the &IF statement is:
&IF 

\[
\begin{array}{c}
\{ \text{token1} \} \\
\&$
\{ \text{token2} \}
\end{array}
\]

where:

\[\text{token1}\]
\[\text{token2}\]

may be numeric constants, character strings, or EXEC variable symbols. All variable symbols are substituted before the &IF statement is executed.

\&$

tests all of the arguments entered when the EXEC was invoked. If at least one of the arguments satisfies the specified condition, the &IF statement is true.

\&*

tests all of the arguments entered when the EXEC was invoked. All of the entered arguments must meet the specified condition in order for the &IF statement to be true.

\textbf{operator}

indicates the test to be performed on the tokens. If both tokens are numeric, an arithmetic test is performed. Otherwise, a logical (alphabetic) test is performed. The comparison operators, listed below, may be specified either in symbolic or mnemonic form:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>= or EQ</td>
<td>equals</td>
</tr>
<tr>
<td>(\neq) or NE</td>
<td>not equal</td>
</tr>
<tr>
<td>&lt; or LT</td>
<td>less than</td>
</tr>
<tr>
<td>(\leq) or LE</td>
<td>less than or equal to (not greater than)</td>
</tr>
<tr>
<td>&gt; or GT</td>
<td>greater than</td>
</tr>
<tr>
<td>(\geq) or GE</td>
<td>greater than or equal to (not less than)</td>
</tr>
</tbody>
</table>

\textbf{executable-statement}

is any valid EXEC executable statement which may be a CMS command, an EXEC control statement, or an assignment statement. You may also specify another &IF statement; the number of nested &IF statements allowed according to the following criteria:

CMS EXEC file: a maximum of 32 tokens is allowed for a variable length file.

EXEC 2 file: the record length of the file.
Usage Notes:

1. The values &* and &$ are reset when an &ARGS or &READ ARGS control statement is executed. They are not changed when you reset a specific numeric variable (&1 through &30).

2. If a variable symbol used in an &IF control statement is undefined, the EXEC interpreter cannot properly compare it. In cases where a variable may be null, or to check for a null symbol, you should use a concatenation character when you write the &IF statement; for example:

   &IF .&1 EQ . &GOTO -NOARGS

   tests for a null symbol &1.

3. If the symbols &* or &$ are null because no arguments were entered, the entire &IF statement is treated as a null statement.

&LOOP

Use the &LOOP control statement to describe a loop in an EXEC procedure, including the conditions for exit from the loop. The format of the &LOOP control statement is:

&LOOP

where:

\( n \)

is a positive integer from 0 to 4095 that indicates the number of executable and nonexecutable lines in the loop. These lines must immediately follow the &LOOP statement.

\(-label\)

specifies that all of the lines following the &LOOP statement down to, and including the line with the specified label, are to be executed in the loop. The first character of the label must be a hyphen, and it must be the first token on a line. The remainder of the line may contain an executable statement, or it may be null.

\( m \)

is a positive integer from 0 to 4095 that indicates the number of times the loop is to be executed.
condition
specifies the condition that must be met. The syntax of the exit
condition is the same as that in the &IF statement, that is:

\[
\begin{align*}
tok1 & \quad \text{operator} \quad tok2 \\
& \quad \&\$ \quad \&\$
\end{align*}
\]

Usage Notes:

1. When loop execution is complete, control passes to the next statement
following the end of the loop.

2. The condition is always tested before the loop is executed. If the
specified condition is met, then the loop is not executed. For example,
the statement:

\[
&\text{LOOP 3 } \&\text{COUNT } = 100
\]

specifies that the next three lines are interpreted until the value of
&COUNT is 100.

3. Loops may be nested up to four levels deep. All nested loops may end at
the same label.

4. A loop is closed when the requirements for termination specified in the
&LOOP statement are met, or when control is transferred outside the
scope of the loop (via &GOTO or &SKIP).

&PUNCH

Use the &PUNCH control statement to punch a line of tokens to the virtual
punch. The format of the &PUNCH control statement is:

\[
&\text{PUNCH} \quad [tok1 \ [tok2 \ldots \ [tokn]]]
\]

where:

\[
tok1 \ [tok2 \ldots \ [tokn]]
\]

specifies the tokens to be punched. All tokens are padded or
truncated to eight characters. The punched line is right-padded with
blanks to fill an 80-column card. If no tokens are specified, a line
consisting of 80 blank characters is punched.
**Usage Notes:**

1. Lines punched with the &PUNCH control statement are scanned by the EXEC interpreter and variable symbols are substituted before the line is punched. In fixed-length EXEC files, only the first 72 characters of the record are scanned. To punch one or more lines of unscanned data, use the &BEGPUNCH or &BEGPUNCH ALL control statement.

2. When you have finished punching lines in an EXEC procedure, you can use the CP command CLOSE to close the spool punch file and release it for processing.

**&READ**

Use the &READ control statement to read one or more lines from the console stack. The lines may contain data or executable statements. The format of the &READ control statement is:

```
&READ [n
       1
       ARGS
       VARS [&var1 [&var2 ... [&varn]]]]
```

**where:**

- **n** reads the next n lines from the stack or the terminal and treats them as if they had been in the EXEC file. Reading from the terminal stops when n lines have been read, or when an &LOOP statement or a statement that transfers control is encountered. If an &READ statement is encountered, the number of lines to be read by it is added to the number outstanding.

- **1** If n is not specified, the default 1 is assumed, and the EXEC continues processing after reading a single line.

- **ARGS** reads a single line, assigns the entered tokens to the special variables &1, &2, ..., &n, and resets the special variables &INDEX, &, and &$. If any of the tokens is specified as a percent sign (%) or begins with the character X'FF', the corresponding argument is set to blanks.

- **VARS [&var1 [&var2 ... [&varn]]]** reads a single line and assigns the tokens entered to the variable symbols &var1, &var2, ..., &varn (up to 17).
These variables are scanned in the same way as though they appeared on the left-hand side of an assignment statement. If no variable names are specified, any data read from the terminal is lost.

If any of the tokens is specified as a percent sign (%) or begins with the character 'X'FF', the corresponding variable is set to blanks.

**Usage Note:**

You can test the special variable &READFLAG to determine whether the next &READ statement will result in a physical read to your terminal (the value of &READFLAG is CONSOLE) or in reading a line from the console stack (the value of &READFLAG is STACK).

**&SKIP**

Use the &SKIP control statement to cause a specified number of lines in the EXEC file to be skipped.
The format of the &SKIP control statement is:

```
&SKIP [ n ]
```

*where:*

- *n* specifies the number of lines to be skipped:
  - If *n* is greater than 0, the specified number of lines are skipped. Execution continues on the line following the skipped lines. If the value of *n* surpasses the number of lines remaining in the file, the EXEC terminates processing.
  - If *n* is equal to 0, no lines are skipped, and execution continues with the next line.
  - If *n* is less than 0, execution continues with the line that is *n* lines above the current line. An attempt to skip beyond the beginning of the file results in an error exit from the EXEC.
  - The *n* may be coded as a variable symbol. 1 is the default value that is used when no value is specified for *n*. 

Appendix E. CMS EXEC Control Statements 859
Usage Note:

To pass control to a particular label in an EXEC procedure, use the &GOTO control statement. The &GOTO control statement provides more flexibility when you want to update your EXEC procedures. The &SKIP statement, however, is more efficient, in terms of execution time.

&SPACE

Use the &SPACE control statement to display a specified number of blank lines at your terminal.

The format of the &SPACE control statement is:

\[
&\text{SPACE} \quad [ \quad n \quad ]
\]

\textit{where:}

\( n \)

specifies the number of blank lines to be displayed at the terminal. If no number is specified, &SPACE 1 is assumed by default.

Usage Note:

You may want to use the &SPACE control statement to control the format of the execution summary that displays while your EXEC executes.

&STACK

Use the &STACK control statement to stack a single data line in the console stack. Stacked lines may be read by the EXEC, by CMS, or by the CMS editors.

The format of the &STACK control statement is:

\[
&\text{STACK} \quad [ \quad \text{FIFO} \quad ] \quad [ \quad \text{tok1} \quad \text{tok2}... \quad \text{tokn} \quad ]
\]

\[
\quad \text{LIFO}\quad \text{HT} \quad \text{RT}
\]

\textit{where:}
**FIFO**

specifies that the line is to be stacked in a first in, first out sequence. FIFO is the default.

**LIFO**

specifies that the line is to be stacked in a last in, first out sequence.

\[ tok1 \ [tok2 \ldots \ [tokn]] \]

specify the tokens to be stacked. If no tokens are specified, a null line is stacked. The tokens are in expanded form.

**HT**

stacks the CMS Immediate command HT (halt typing), which is executed immediately. All CMS terminal display from the EXEC, except for CMS error messages with a suffix letter of “S” or “T,” is suppressed until the end of the file or until an RT (resume typing) command is read.

**RT**

stacks the CMS Immediate command RT (resume typing), which is executed immediately. If CMS terminal display has been suppressed as the result of an HT (halt typing) request, display is resumed.

**Usage Notes:**

1. Lines stacked with the &STACK control statement are scanned by the EXEC interpreter and variable symbols are substituted before the line is stacked. To stack one or more unscanned lines, use the &BEGSTACK or &BEGSTACK ALL control statement.

2. You must use the &STACK control statement when you want to stack a null line.

3. The commands SET CMSTYPE HT and SET CMSTYPE RT perform the same functions as &STACK HT and &STACK RT.

4. To stack an equal sign, you must assign it to an EXEC variable (“&A = ”, for example) and use the result with the &STACK control word (&STACK &A).

**&TIME**

Use the &TIME control statement to request timing information to be displayed at the terminal after each CMS command that is executed.

The format of the &TIME control statement is:
&TIME

| ON | OFF | RESET | TYPE |

**where:**

**ON**
resets the processor's time before every CMS command, and prints the timing information on return. If the &CONTROL control statement is set to CMS or ALL, the display of the timing information is followed by a blank line.

**OFF**
does not automatically reset the processor's time before every CMS command, nor does it print the timing information on return.

**RESET**
performs an immediate reset of the processor's time.

**TYPE**
displays the current timing information (and resets the processor's time).

**Usage Notes:**

1. When timing information is displayed, it is in the format:

   T=x.xx/y.yy hh:mm:ss

   **where:**

   x.xx is the virtual processor's time used since it was last reset in the current EXEC file.

   y.yy is the total of the processor's time used since it was last reset in the current EXEC file.

   hh:mm:ss is the actual time of day in hours:minutes:seconds.

2. The processor's time is set to zero before the execution of the first statement in the EXEC file, and is again set to zero (reset) whenever timing information is printed.
&TYPE

Use the &TYPE control statement to display a line of tokens at the terminal.

The format of the &TYPE control statement is:

| &TYPE | [tok1 [tok2 ... [tokn]]] |

where:

tok1 [tok2 ... [tokn]]
specify the tokens to be displayed. All tokens are padded or truncated to eight characters. If no tokens are specified, a null line is displayed.

Usage Note:

Lines displayed with the &TYPE control statement are scanned by the EXEC interpreter and variable symbols are substituted before the line is displayed. To display one or more unscanned lines, use the &BEGTYPE or &BEGTYPE ALL control statements.

Built-in Functions

You can use the EXEC built-in functions to assign and manipulate variable symbols. With the exception of &LITERAL, built-in functions may be used only on the right-hand side of an assignment statement, as follows:

&mix = &CONCAT &1 &2

Built-in functions may not be combined with arithmetic expressions.

Each of the built-in functions (&CONCAT, &DATATYPE, &LENGTH, &LITERAL, and &SUBSTR) is described separately.

&CONCAT

Use the &CONCAT function to concatenate two or more tokens and assign the result to a variable symbol.

The format of the &CONCAT function is:
**&variable = &CONCAT tok1 [tok2 ... [tokn]]**

**where:**

- **&variable**
  - is the variable symbol whose value is determined by the &CONCAT function.

- **tok1 [tok2...[tokn]]**
  - specifies the tokens that are to be concatenated into a single token; for example:
    ```
    &A = **
    .
    .
    &B = &CONCAT XX &A 45
    &TYPE &B
    ```
  - results in the printed line:
    ```
    XX**45
    ```

**Usage Note:**

If the concatenated token is longer than eight characters, the data is left-justified and truncated on the right.

**&DATATYPE**

Use the &DATATYPE function to determine whether the value of the specified token is alphabetic or numeric data.

The format of the &DATATYPE function is:

- **&variable = &DATATYPE token**

**where:**

- **&variable**
  - is the variable symbol whose value is determined by the &DATATYPE function.

- **token**
  - specifies the target token that is to be examined for alphabetic or numeric data. The result of the &DATATYPE function has the value NUM or CHAR, depending on the data type of the specified token. For example:
&CHECK = &DATATYPE  ABC
&TYPE &CHECK

results in the display:

CHAR

A null token is considered character data.

&LENGTH

Use the &LENGTH function to determine the number of characters in a token.

The format of the &LENGTH function is:

\[
&variable = &LENGTH \ token
\]

where:

&variable

is the variable symbol whose value is determined by the &LENGTH function.

token

specifies the target token that is to be examined for nonblank characters. The result of the &LENGTH function is the number of nonblank characters in the specified token. For example:

&LEN = &LENGTH ALPHA
&TYPE &LEN

results in the display:

5

&LITERAL

Use the &LITERAL function to inhibit variable substitution on the specified token.

The &LITERAL function may appear in any EXEC control statement, as follows:
where:

&LITERAL

&LITERAL token

where:

token

specifies the token whose literal value is to be used without substitution. For example:

&X = **
&type &LITERAL &X EQUALS &X

results in the printed line:

&X EQUALS **

&SUBSTR

Use the &SUBSTR function to extract a character string from a specified token and to assign the substring to a variable symbol.

The format of the &SUBSTR function is:

&variable = &SUBSTR token i j

where:

&variable

is the variable symbol whose value is determined by the &SUBSTR function.

token

is the token from which the character string is to be extracted.

i

specifies the character position in the token of the first character to be used in the substring.

j

specifies the number of characters in the string. If omitted, the remainder of the token is used.
Usage Note:

The values of i and j (if given) must be positive integers. For example:

\texttt{\&A = \&SUBSTR ABCDE 2 3}  
\texttt{\&TYPE \&A}

results in the printed line:

\texttt{BCD}

Special Variables

Special variables are variable symbols that are assigned values by the EXEC interpreter, and that you can test or display in your EXEC procedures. In some cases, you may assign your own values to EXEC special variables; these cases are noted in the variable descriptions.

\&n

The \&n special variable represents the numeric variables \&1 through \&30. When an EXEC is invoked, the numeric variables from \&1 through \&30 are initialized according to the arguments that are passed to the EXEC file (if any).

The numeric variables can be reset by either an \&ARGS or \&READ ARGs control statement; when fewer than 30 arguments are set or reset, the remainder of the \&n variables are set to blanks.

A particular argument can be set to blanks by assigning it a percent sign (%) when invoking the EXEC procedure, in an \&ARGS control statement, or in an \&READ ARGs control statement. An argument is also set to blanks if it begins with the character X'FF' and is specified when invoking the EXEC procedure or in an \&READ ARGs control statement. You may set the values of specific arguments using assignment statements. Any value of \n, however, that is greater than 30 or less than 0 is rejected by the EXEC interpreter.

\&* and \&\$

These variables can be used to perform a collective test on all of the arguments passed to the EXEC procedure. \&* and \&\$ may only be used in the \&IF and \&LOOP control statements and are described under the description of the \&IF control statement. You may not assign values to the special variables \&* and \&\$.
&0

The &0 special variable contains the filename of the EXEC file. You may test and manipulate this variable.

&DISKx

You can use the &DISKx special variable to determine whether a disk is an OS, DOS, or CMS disk. x represents the mode letter at which the disk is accessed. For example, if you access an OS disk with a mode letter of C, then the special variable &DISKC has a value of OS. The possible values for the &DISKx special variable are OS (for an OS disk), DOS (for a DOS disk), CMS (for a CMS disk), and NA (when the disk is not accessed).

You may set or change the values of an &DISKx special variable. If you do so, you are no longer able to test the status of the disk at mode x.

&DISK*

The &DISK* special variable contains the one-character mode letter of the first read/write disk in the CMS search order. If you have no read/write disks accessed, this special variable contains the value NONE.

You may assign a value to the &DISK* special variable for your own use; if you do so, however, you will not be able to use it to obtain the filemode letter of a read/write disk.

&DISK?

You can use the &DISK? special variable in an EXEC to determine which read/write disk that you have accessed has the most space on it. If you have no read/write disks accessed or if the accessed disk is full, &DISK? contains the value NONE.

You may assign a value to the &DISK? special variable for your own use; if you do so, however, you will no longer be able to locate the read/write disk with the most space.

&DOS

The &DOS special variable contains one of the two character values ON or OFF, depending on whether the CMS/DOS environment is active. If you have issued the command:

set dos on
then the &DOS special variable contains the value ON.

You may set or change the value of the &DOS special variable for your own use; if you do so, however, you will not be able to test whether the CMS/DOS environment is active.

&EXEC

The &EXEC special variable is the filename of the EXEC file. You cannot set this variable explicitly but you can examine and test it.

&GLOBAL

The &GLOBAL special variable contains the recursion level of the EXEC currently executing. Since the EXEC interpreter can handle up to 199 levels of recursion, the value of &GLOBAL ranges from 1 to 199. You cannot set this variable explicitly, but you can examine and test it.

&GLOBALn

The &GLOBALn special variable represents the variables &GLOBAL0 through &GLOBAL9. You can set these variables only to integral numeric values. They are all initially set to 1. Unlike other EXEC variables, these can be used to communicate between different recursion levels of the EXEC interpreter.

&INDEX

The &INDEX special variable contains the number of arguments passed to the EXEC procedure. Since up to 30 arguments can be passed to an EXEC procedure, the value of &INDEX can range from 0 through 30.

Although you cannot set this variable explicitly, it is reset by an &ARGS or &READ ARGS control statement. &INDEX can be examined to determine the number of active arguments in the EXEC procedure.

&LINENUM

The &LINENUM special variable contains the current line number in the EXEC file. You cannot explicitly set this variable but you can examine and test it.
&READFLAG

The &READFLAG special variable contains one of two literal values: CONSOLE or STACK. If there are stacked lines in the program stack or terminal input buffer &READFLAG contains the value STACK and the next read request results in a line being read from the stack. If not, then the next read request results in a physical read to the terminal, and the value of &READFLAG is CONSOLE. You cannot explicitly set this variable but you can examine and test it.

&RETCODE

The &RETCODE special variable contains the return code from the most recently executed command. &RETCODE can contain only integral numeric values (positive or negative), and is set after each CMS command is executed. You can examine, test, and change this variable but changing it is not recommended.

&TYPEFLAG

The &TYPEFLAG special variable contains one of two literal values: RT (resume typing) or HT (halt typing). It contains the value HT when terminal display has been suppressed by the Immediate command HT. It contains the value RT when the terminal is displaying output. You cannot explicitly set this variable, but you can examine and test it.
Summary of Changes

Structural Changes

This book has been restructured for VM/SP Release 5. CMS Functions and CMS Macro Instructions have been moved to a new book, the *VM/SP CMS Macros and Functions Reference*, SC24-5284. This book has been restructured as follows:

Chapter 1. Introduction and General Concepts
Chapter 2. CMS Commands
Chapter 3. HELP and HELPCONV Format Words
Chapter 4. DEBUG Subcommands

In addition, six appendixes contain the following information:

Appendix A. VSE/VSAM Functions Not Supported in CMS
Appendix B. OS/VS Access Method Services and VSAM Functions Not Supported
Appendix C. Edit Subcommands and Macros
Appendix D. Edit Reserved Filetype Defaults
Appendix E. CMS EXEC Control Statements

Technical Changes

Summary of Changes
for SC19-6209-4
for VM/SP Release 5

How to Obtain Prior Editions of This Publication

To obtain editions of this publication that pertain to earlier releases of VM/SP, you must use the pseudo-number assigned to the respective edition when ordering. For:

Release 4, order ST00-1583
Release 3, order ST00-1357
Release 2, order SQ19-6209
Release 1, order ST19-6209
Summary of Changes
for SC19-6209-4
for VM/SP Release 5

New Commands for Release 5 of VM/SP

The following CMS commands are new for this release.

CLEAR VSCREEN
   Erases data in the virtual screen by overwriting the data buffer with nulls.

CLEAR WINDOW
   Scrolls past all data in the virtual screen to which the window is connected
   so that no scrollable data is displayed in the window.

CMSSERV
   Starts IBM Cooperative Processing communications on VM/SP.

CONVERT COMMANDS
   Converts a CMS file containing Command Syntax Definition Language
   (CSDL) statements into an internal form for the parsing facility.

CURSOR VSCREEN
   Positions the cursor on specified line and column in a virtual screen.

DEFINE VSCREEN
   Create a virtual screen.

DEFINE WINDOW
   Creates a window with the specified name, size, and position on the physical
   screen.

DELETE VSCREEN
   Removes a virtual screen definition.

DELETE WINDOW
   Removes a window definition.

DROP WINDOW
   Moves a window down in the order of displayed windows.

GENMSG
   Converts a message repository file into an internal form.

GET VSCREEN
   Writes data from a CMS file to the specified virtual screen.

HIDE WINDOW
   Prevents the specified window from being displayed, and, optionally,
   connects the window to a virtual screen.

MAXIMIZE WINDOW
   Expands a window to the physical screen size.

MINIMIZE WINDOW
   Reduces the size of the window to one line.
MOREHELP
Obtains either additional or related information about the latest valid HELP command you issued.

PARSECMD
Calls the parsing facility from within an exec.

POP WINDOW
Moves a window up in the order of displayed windows.

POSITION WINDOW
Changes the location of a window on the physical screen.

PUT SCREEN
Copies the physical screen and writes the image to a CMS file.

PUT VSCREEN
Writes the data from the scrollable data area of a virtual screen to a CMS file.

RESTORE WINDOW
Return a maximized or minimized window to its size and location prior to the maximize or minimize.

ROUTE
Directs data of a particular message class to a virtual screen.

SCROLL
Moves a window to a new location on the virtual screen to which it is connected.

SHOW WINDOW
Places a window on top of all other displayed windows and connects the window to a virtual screen.

SIZE WINDOW
Changes the number of lines and columns for a specified window.

VALIDATE
Verifies the syntax of a file identifier and verifies whether or not a disk is accessed.

WAITREAD VSCREEN
Used from an EXEC, updates the virtual screen with data, refreshes the physical screen, and waits for the next attention interrupt.

WAITT VSCREEN
Updates the virtual screen with data.

WRITE VSCREEN
Enters information in a virtual screen. Information is queued to a virtual screen and is displayed the next time the screen is refreshed.

XMITMSG
Retrieves a message from a CMS message repository file or your own message repository file.
Border Commands
Manipulate windows when entered on the corners of window borders. Several single-character border commands have been added to manipulate windows in the full-screen CMS environment:

B Scrolls the window backward
C Clears the window of scrollable data
D Drops the window
F Scrolls the window forward
H Hides the window
L Scrolls the window to the left
M Changes the location of the window
N Minimizes the window
O Restores the window
P Pops the window
R Scrolls the window to the right
S Changes the size of the window
X Maximizes the window

Modified Commands for Release 5 of VM/SP

The following CMS commands have been modified this release.

ACCESS
Addition of the SAVEONLY and NOSAVE options that allow you to specify whether a disk is accessed with a saved copy of the file directory or with a file directory in user storage.

DEFAULTS
Defaults options for HELP can be set for a BRIEF or DETAIL layer of information.

EXECDROP
Addition of the SHARED option that drops an exec that is located in an Installation DCSS.

EXECIO
Addition of the BUFFER option allows you to specify the length of the CP command response expected from a CP operation. The length may be from 1 to 2^31-1 characters (bytes).

EXECMAP
Addition of the SHARED option allows you to list execs that are located in an Installation DCSS.

EXECUPDT
Comments can be removed from an exec source file with the COMMENTS option. The ETMODE option specifies that the source file contains DBCS characters and that shift-in and shift-out should be paired while comments are removed.

FILEDEF
Addition of the ALT option allows you to specify an alternate tape drive. The SYSPARM option is added for OS simulation standard label tape processing exits.

FORMAT
The BLKSIZE option defaults to a block size that optimizes the I/O and data storage for the particular device.
GLOBAL
The GLOBAL command has been enhanced to allow you to list up to 63 libraries from one of the supported library types.

HELP
Addition of the BRIEF, DETAIL, and RELATED options allow to specify different levels, or layers, of information. The EXTEND option uses the HELP search order when you issue the HELP command from the editing environment and the component is not specified.

INCLUDE
The HIST option saves the history information (comments) from text files. The NOHIST does not save the history information from text files.

LOAD
The HIST option saves the history information (comments) from text files. The NOHIST does not save the history information from text files.

PRINT
Addition of the OVERSIZE option allows you to print files that have records larger than the carriage size of the virtual printer and files that have a SPECIAL status of YES.

QUERY
APL, BORDER, CHARMODE, CMSPF, CURSOR, DISPLAY, DOSLIB, FULLREAD, FULLSCREEN, HIDE, INSTSEG, KEY, LIBRARY, LOADLIB, LOCATION, LOGFILE, MACLIB, NONDISP, REMOTE, RESERVED, SHOW, TEXT, TXTLIB, TRANSLATE, VSCREEN, WINDOW, and WMPF.

SET
ABBREV, APL, BORDER, CHARMODE, CMSPF, FULLREAD, FULLSCREEN, INSTSEG, LANGUAGE, LANGLIST, LINEND, LOCATION, LOGFILE, NONDISP, REMOTE, RESERVED, TEXT, TRANSLATE, VSCREEN, WINDOW, and WMPF.

TXTLIB
The FILENAME option indicates that all the filenames specified will be used as the membernames for their respective entries in the TXTLIB file instead of the first CSECT in the file's text deck.

UPDATE
The UPDATE command is enhanced to list up to 29 macro library (MACLIB) filenames.

XEDIT
The WINDOW option specifies the window and virtual screen that the System Product editor uses to display the file being edited.

Migration of CMS Commands into the Nucleus
The following CMS commands are now nucleus resident commands:

- COPYFILE
- GLOBALV
- IDENTIFY
- PRINT
CMS Command Search Order

Document the search for translations and abbreviations in CMS Command Search Order description.

CMS Command Syntax Error Messages

For some CMS commands, syntax error messages are issued by the CMS Parsing Facility.

Integration of Functional Enhancements to Release 4

Document support of the following:
- IBM 3380 Direct Access Storage Device, Models AE4 and BE4
- IBM 3380 Direct Access Storage Device Support for VSE/VSAM
- CMS Vector Processing
- CMS Loader

Miscellaneous

- This major revision incorporates minor technical and editorial changes.
- A keyword operand is added to the .CS HELPCONV format word
- The DDR command is moved to the VM/SP CP for System Programming.

Summary of Changes
for SC19-6209-3
for VM/SP Release 4

New Commands for Release 4 of VM/SP

The following CMS commands are new for this release.

EXECDROP
   Purges storage resident EXECs.

EXECLOAD
   Load EXECs into storage.

EXECMAP
   Provides a list of storage resident EXECs.

EXECSTAT
   Provides the status of a specific EXEC.

HELPCONV
   Converts a specified file into a formatted HELP file, leaving the .CS, .CM, and .MT control words in the file.
MACLIST

Displays a list of all members in a specified MACLIB, with the ability to edit and issue commands from the list.

Commands and Macros that have been Modified

The following commands and macros have been modified for this release.

**DDR command**
Addition of the MODE 38K option for use with the 3480 Magnetic Tape Subsystem. Supports the use of the COMPACT option on the OUTPUT control statement for the DUMP function.

**DEFAULTS command**
Defaults options can be set up for the HELP and MACLIST commands.

**EXECIO**
Addition of the VAR and STEM options that allow you to use the EXECIO command directly with REXX or EXEC 2 variables.

**FILEDEF**
Addition of the 18TRACK option for the TAPn operand.

**HELP command**
Addition of MESSAGE, MSG, and TASKS operands. Addition of DESCRIPTION, FORMAT, PARMS, OPTIONS, NOTES, ERRORS, SCREEN, and NOSCREEN options that allow you to select specific areas of a command HELP file to be displayed.

**INCLUDE command**
Addition of the RLDSAVE option that instructs the CMS loader to save the relocation information from text files.

**LABELDEF command**
Addition of VOLID ? and VOLID SCRATCH operands.

**LOAD**
Addition of the RLDSAVE option that instructs the CMS loader to save the relocation information from text files.

**MACLIB command**
Individual member names can now be specified with the MAP operand. Addition of the STACK, LIFO, FIFO, and XEDIT options that allow you to stack the MAP output.

**TAPE**
Addition of the 18TRACK option that allows you to specify an 18 track tape that is used with the 3480 Magnetic Tape Subsystem. The TRANSFER IMMEDIATE and TRANSFER BUFFERED options specify the write mode for the 3480 Magnetic Tape Subsystem.

**XEDIT command**
New MEMBER option allows you to specify the name of a macro library member to be edited.

**TAPECTL macro**
Addition of the BLKBUFF operand to be used with the LOCLBLK and RDBLKID functions for the 3480 Magnetic Tape Subsystem.

**WRTAPE**
ADDITION of TRAN BUFF and TRAN IMMED operands to specify the write mode when used with the 3480 Magnetic Tape Subsystem.
Migration of CMS Commands into the Nucleus

The following CMS commands are now nucleus resident commands:

- ACCESS
- DLBL
- FILEDEF
- RELEASE
- SET

CMS Command Search Order

Document the search for storage resident EXECs in the CMS Command Search Order description.

Integration of Functional Enhancements to Release 3

Document support of the following:

- IBM 3800 Printing Subsystems, Models 1 and 3.
- IBM 3370 Direct Access Storage Device, Models A2 and B2.
- 3290 Information Panel.
- IBM 4248 Printer.

New VM/SP Component

Document support of the Interactive Problem Control System (IPCS) component of VM/SP.

Miscellaneous

- This major revision incorporates minor technical and editorial changes.
- Document support of the IBM 3480 Magnetic Tape Subsystem
- Document changes to XEDIT screens for messages that appear in mixed case.
- Document the new .MT (MENU TYPE) HELP_CONV format word.

Other Changes

- MOVE EDIT SUBCOMMANDS to appendix
- MOVE EXEC CONTROL STATEMENTS to appendix
- Book now has 6 chapters (formerly referred to as sections).
  1. INTRO
  2. CMS Commands
  3. CMS Functions
  4. CMS Macros
  5. Help Format Words
6. DEBUG subcommands

- and six appendixes

(EDIT SUBCOMMANDS and EXEC control statement have been moved here).

Summary of Changes
for SC19-6209-2
for VM/SP Release 3

New CMS Commands
The following CMS commands are new for this release:

CATCHECK - Allows CMS VSAM users to invoke the VSE/VSAM Catalog Check Service Aid to verify a complete catalog structure.

EXECOS - Resets the OS and VSAM environments under CMS without returning to the interactive environment.

EXECUPDT - Used to apply updates to a System Product Interpreter source program and create an executable version of the program.

IMMCMMD - Establishes or cancels Immediate commands from within an EXEC.

RESERVE - Allocates all available blocks of a 512-, 1K-, 2K-, or 4K-byte block formatted minidisk to a unique CMS file.

New Immediate Commands
HI - Halt Interpretation terminates execution of all currently executing System Product interpreter or EXEC 2 EXECs without destroying the environment as HX would.

TE - Trace End stops all tracing of your System Product interpreter or EXEC 2 programs or macros.

TS - Trace Start starts tracing your System Product interpreter or EXEC 2 programs or macros.

New CMS Function
DISKID - Obtains information on the physical organization of RESERVEd minidisk.

New CMS Macro Instructions
ABNEXIT - Sets or clears ABEND exit routines.

IMMCMMD - Declares, clears, or queries Immediate commands.

WAITECB - Waits on an Event Control Block (ECB) or a list of ECBs.

Miscellaneous
- The SET command establishes, turns off, or resets the following functions. The QUERY command determines the status of these functions.

EXECTRAC - Turns the tracing on or off for a System Product interpreter program.

IMESCAPE - Specifies whether or not an escape character is required to execute Immediate commands and allows you to specify an escape character.
• A list of CMS commands and their execution characteristics is provided.

• When in the FILELIST, RDRLIST, and SENDFILE environments:
  – You can sort the files with the synonyms provided.
  – The ENTER key executes the commands typed on the file lines.

• Document support of the 512-byte block size for CMS formatted minidisks.

• This revision incorporates minor technical and editorial changes.
Glossary of Terms and Abbreviations

Some of the following convenience terms are used throughout this publication:

- Throughout this publication, the term "VM/SP" refers to the VM/SP program package when you use it in conjunction with VM/370 Release 6. The terms "CP," "CMS," and "IPCS" refer to the VM/370 components enhanced by the functions included in the VM/SP package. Any references to "RSCS," unless otherwise noted, are to the VM/370 component unchanged by the VM/SP package.

When you install and use VM/SP in conjunction with the VM/370 Release 6 System Control Program (SCP), it becomes a functional operating system that provides extended features to the following components of VM/370 Release 6:

- Control Program (CP)
- Conversational Monitor System (CMS)
- Interactive Problem Control System (IPCS).

VM/SP adds no additional functions to the Remote Spooling Communications Subsystem (RSCS) component of VM/370. However, you can appreciably expand the capabilities of these components in a VM/SP system by installing the RSCS Networking program product (5748-XP1). Note that VM/SP has an enhanced interactive problem control system (VM/SP IPCS) component. This component replaces the unmodified VM/370 interactive problem control system. Detail of this major component are found in the VM/SP Interactive Problem Control System Guide, SC24-5260.

Note: For VM/SP users, VM/SP IPCS is more effective than the IPCS Extension Program Product (5748-SA1).

- The term "CMS/DOS" refers to the functions of CMS that become available when you issue the command:

  set dos on

CMS/DOS is a part of the normal CMS system, and is not a separate system. Users who do not use CMS/DOS are sometimes referred to as OS users, since they use the OS simulation functions of CMS.

- Unless otherwise noted, the term "VSE" refers to the combination of the DOS/VSE system control program and the VSE/Advanced Functions program product.

In certain cases, the term DOS is still used as a generic term. For example, disk packs initialized for use with VSE or any predecessor DOS or DOS/VS system may be referred to as DOS disks.

The DOS-like simulation environment provided under the CMS component of the VM/System Product, continues to be referred to as CMS/DOS.


- The term "CMS files" refers exclusively to files that are in the format used by CMS file system commands. VSAM and OS data sets and DOS files are not compatible with the CMS file format and cannot be manipulated using CMS file system commands.

- The terms "disk" and "virtual disk" are used interchangeably to indicate disks that are in your CMS virtual machine configuration. Where necessary, a distinction is made between the CMS-formatted disks and disks in OS or DOS format.

- The term "CMS console stack" refers to the combination of the program stack and the terminal input buffer.

The following terms in this publication refer to the indicated support devices:

- "2305" refers to IBM 2305 Fixed Head Storage, Models 1 and 2.
• "270x" refers to IBM 2701, 2702, and 2703 Transmission Control Units or the Integrated Communications Adapter (ICA) on the System/370 Model 135.

• "3270" refers to a series of display devices, namely, the IBM 3275, 3276, 3277, 3278, and 3279 Display Stations, and the 3290 Information Panel. A specific device type is used only when a distinction is required between device types.

Information about display terminal usage also applies to the IBM 3138, 3148, and 3158 Display Consoles when used in display mode, unless otherwise noted.

Any information pertaining to the IBM 3284 or 3286 Printer also pertains to the IBM 3287, 3288, and 3289 printers, unless otherwise noted.

• "3330" refers to the IBM 3330 Disk Storage Models 1, 2, or 11; and the 3350 Direct Access Storage operating in 3330/3333 Model 1 or 3330/3333 Model 11 compatibility mode.

• "3340" refers to the IBM 3340 Disk Storage, Models A2, B1, and B2, and the 3344 Direct Access Storage Model B2.

• "3350" refers to the IBM 3350 Direct Access Storage Models A2 and B2 in native mode.

• "3370" refers to the IBM 3370 Direct Access Storage Models A1, A2, B1, and B2.

• "3375" refers to the IBM 3375 Direct Access Storage Device.

• "3380" refers to the IBM 3380 Direct Access Storage Device.

• "3704," "3705," or "3704/3705" refers to IBM 3704 and 3705 Communications Controllers.

• "3705" refers to the 3705 I and the 3705 II unless otherwise noted.

• "2741" refers to the IBM 2741 and the 3767, unless otherwise specified.

• "3066" refers to the IBM 3066 System Console.

• "3480" refers to the IBM 3480 Magnetic Tape Subsystem.

• "3800" refers to the IBM 3800 Printing Subsystems, Models 1 and 3. A specific device type is used only when a distinction is required between device types.

• "4248" refers to the IBM 4248 Printer.

For a glossary of VM/SP terms, see the Virtual Machine/System Product Library Guide, Glossary, and Master Index, GC19-6207.

This section explains some of the terms and acronyms that appear in this manual.

ADCON. Is an A-type address constant used in the calculation of storage addresses.

CMS (Conversational Monitor System). A component of Virtual Machine/System Product (VM/SP) that is a conversational operating system designed to run under Control Program (CP).

console stack. Refers collectively to the program stack and the terminal input buffer.

CP (Control Program). A component of Virtual Machine/System Product (VM/SP) that controls the resources of the real machine.

discontiguous saved segment. An area of storage beyond the address of your virtual machine address space (not contiguous with your virtual storage) where segments are loaded as needed.

ECB. Event Control Block

extended PLIST (untokenized parameter list). Consists of four addresses that indicate the extended form of the command as it was entered at the terminal.

IPL. Initial program load.

look-aside entry. A nucleus resident routine becomes a look-aside entry after it has been executed.

module. A nonrelocatable file whose external references have been resolved.

nucleus. That part of CP or CMS that is resident in main storage.

program stack. Temporary storage for lines (or files) being exchanged by programs that execute in CMS.

PLIST. Parameter list.

REXX language. Restructured Extended Executor language used in System Product interpreter programs.

SID code. Support Identification code
**terminal input buffer.** Holds lines entered at your terminal until CMS processes them.

**tokenized PLIST (parameter list).** A string of doubleword aligned parameters occupying successive double words.

**virtual machine.** A functional equivalent of a real machine.

**VM/SP (Virtual Machine/System Product).** A licensed program that controls virtual machines.
Prerequisite Publications

In addition to the VM/SP CMS User's Guide, prerequisite information is contained in the following publications:

- For information about the terminal that you are using, including procedures for gaining access to the VM/SP system and logging on, see the Virtual Machine/System Product Terminal Reference, GC19-6206.

- If you are using an IBM 3767 Communications Terminal, the IBM 3767 Operator's Guide, GA18-2000, is a prerequisite.

- The CP commands that are available to you for all CP privilege classes are described in Virtual Machine/System Product CP Command Reference, SC19-6211.

A new user of CMS might refer to the VM/SP: CMS Primer, SC24-5236, for introductory tutorial information on using CMS.

If you are going to use an IBM Licensed Program compiler under CMS, you should have available the appropriate Licensed Program documentation. These publications are listed in Virtual Machine/System Product Introduction, GC19-6200.

Corequisite Publications

The Virtual Machine/System Product System Messages and Codes, SC19-6204, describes all of the error messages and system responses produced by the CMS commands and EDIT and DEBUG subcommands referenced in this publication. It also lists the error messages issued by the EXEC processors during execution of your EXEC procedures. The Virtual Machine/System Product System Messages Cross Reference, SC19-6204, contains the cross reference lists formerly contained in the appendix of SC19-6204.

If you are alternating between CMS and other operating systems in virtual machines running under VM/SP, you should consult Virtual Machine/System Product Running Guest Operating Systems, GC19-6212.

For information on the CMS Functions and CMS Macro Instructions refer to the Virtual Machine/System Product CMS Macros and Functions Reference.

For information on the VM/SP System Product Editor refer to the:

Virtual Machine/System Product

For information about the System Product Interpreter, see the:

**Virtual Machine/System Product**


For information on EXEC 2 refer to **VM/SP EXEC 2 Reference, SC24-5219**.

**Supplemental Publications**

For general information about the VM/SP system, see **Virtual Machine/System Product Introduction, GC19-6200**.

Additional descriptions of various CMS functions and commands which are normally used by system support personnel are described in:

**Virtual Machine/System Product**

*Installation Guide, SC24-6237*
*Operator's Guide, SC19-6202*
*Planning Guide and Reference, SC19-6201*
*CMS for System Programming, SC24-5286*
*CP for System Programming, SC24-5285*
*Transparent Services Access Facility Reference, SC24-5287*
*System Facilities for Programming, SC24-5288*
*VM Diagnosis Guide, LY24-5241*

For information on the terms used in VM/SP and for assistance in locating specific information, refer to the **VM/SP Library Guide and Master Index, GC19-6207**.

Information on VM/SP IPCS commands, which are invoked under CMS, is contained in **VM Diagnosis Guide, LY24-5241**.

Details on the CMS CPEREP, a command used to generate output reports from VM/SP error recording records, are contained in:

**Virtual Machine/System Product OLTSEP and Error Recording Guide, SC19-6205**.

For more details on the operands used with CPEREP, refer to:

**OS/VS, DOS/VSE, VM/370 Environmental Recording, Editing, and Printing (EREP) Program, GC28-0772**.

For messages issued by CMS CPEREP, see:

**OS/VS, DOS/VSE, VM/370 EREP Messages, GC38-1045**.

For information on IBM GAM/SP, refer to:

**OS/VS Graphic Programming Services (GPS) for IBM 2250 Display Unit and IBM 3250 Graphics Display System, SC27-6971**
*CMS GAM/SP User's Guide, LC33-0126*

Details on the CMS IOCP command are contained in:

**Input/Output Configuration Program User's Guide and Reference, GC28-1027.**
For information about OS/VS tape label processing, refer to:

*OS/VS Tape Labels, GC26-3795.*

**Ready References for VM/SP Users**

There are six publications available as ready reference material when you use VM/SP. They are:

**Virtual Machine/System Product**

- *Commands (General User), SX20-4401.*
- *Commands (other than General User), SX20-4402.*
- *EXEC 2 Language, SX24-5124*
- *Quick Reference, SX20-4400*
- *SP Editor Command Language, SX24-5122*
- *System Product Interpreter Reference Summary, SX20-5126*


**For VSAM and Access Method Services Users**

CMS support of Access Method Services is based on VSE and VSE/VSAM. The control statements that you can use are described in *Using VSE/VSAM Commands and Macros, SC24-5144.* The *VM/SP CMS User's Guide* contains details on how to use this support. Error messages produced by the Access Method Services program, and return codes and reason codes are listed in *VSE/VSAM Messages and Codes, SC24-5146.*

For additional information refer to the *VSE/VSAM Programmer's Reference, SC24-5145.*


**For CMS/DOS Users**

The CMS ESERV command invokes the VSE ESERV program, and uses, as input, the control statements that you would use in VSE. These control statements are described in *Guide to the DOS/VSE Assembler, GC33-4024.*

Linkage editor control statements, used when invoking the linkage editor under CMS/DOS, are described in *VSE System Control Statements, SC33-6095,* and *OS/VS Linkage Editor and Loader, GC26-3813.*

Batch DL/I application programs can be written and tested in the CMS/DOS environment. See *VM/SP CMS User's Guide,* and *DL/I DOS/VS General Information, GH20-1246,* for details.

For information on VSE and CMS/DOS tape label processing, refer to:
VSE/Advanced Functions Tape Labels, SC24-5212.
The VM/SP Library (Part 1 of 3)

Evaluation
- General Information
  - GC20-1838
- Introduction
  - GC19-6200

Index
- Library Guide, Glossary, and Master Index
  - GC19-6207

Planning
- Planning Guide and Reference
  - SC19-6201
- Running Guest Operating Systems
  - GC19-6212
- Release 5 Guide
  - SC24-5290
- Distributed Data Processing Guide
  - SC24-5241

Installation
- Installation Guide
  - SC24-5237

Applications
- Application Development Guide
  - SC24-5247
- Programmer's Guide to the SRPI for VM/SP
  - SC24-5291

Operation
- Operator's Guide
  - SC19-6202

Reference Summaries
- Commands (General User)
  - SX20-4401
- Commands (Other than General User)
  - SX20-4402
- CMS Primer Summary of Commands
  - SX24-5151
- CMS Primer Line-Oriented Summary of Commands
  - SX24-5159
- SP Editor Command Reference Summary
  - SX24-5122
- Problem Reporting Summary (Poster)
  - SX24-5171
- EXEC 2 Reference Summary
  - SX24-5124
- Summary of End Use Tasks and Commands (Poster)
  - SX24-5173
- Sys.Prod Interpreter Reference Summary
  - SX24-5126

To order all of the Reference Summaries, use order number SBOF-3242
The VM/SP Library (Part 2 of 3)

### End Use

<table>
<thead>
<tr>
<th>Terminal Reference</th>
<th>CMS Primer</th>
<th>CMS Primer for Line-Oriented Terminals</th>
<th>CMS User’s Guide</th>
<th>CMS Command Reference</th>
<th>CMS Macros and Functions Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>GC19-6206</td>
<td>SC24-5236</td>
<td>SC24-5242</td>
<td>SC19-6210</td>
<td>SC19-6209</td>
<td>SC24-5284</td>
</tr>
<tr>
<td>SC24-5220</td>
<td>SC24-5238</td>
<td>SC24-5239</td>
<td>SC24-5219</td>
<td>SC19-6211</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Quick Reference</th>
<th>SX20-4400</th>
</tr>
</thead>
</table>

### Diagnosis

<table>
<thead>
<tr>
<th>System Messages and Codes</th>
<th>System Messages Cross-Reference</th>
<th>Service Routines Program Logic</th>
<th>Problem Reporting Guide</th>
<th>VM Diagnosis Guide</th>
<th>GCS Diagnosis Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC19-6204</td>
<td>SC24-5264</td>
<td>LY20-0890</td>
<td>SC24-5282</td>
<td>LY24-5241</td>
<td>LY24-5239</td>
</tr>
<tr>
<td>LY20-0892</td>
<td>LY24-5220</td>
<td>LY20-0893</td>
<td>LY24-5221</td>
<td>SC19-6205</td>
<td>LX23-0347</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VM CP Internal Trace Table (Poster)</th>
<th>LX24-5202</th>
</tr>
</thead>
</table>

890 VM/SP CMS Command Reference
The VM/SP Library (Part 3 of 3)

Administration

VM System Facilities for Programming
SC24-5288

CP for System Programming
SC24-5285

CMS for System Programming
SC24-5286

TSAF Reference
SC24-5287

GCS Command and Macro Reference
SC24-5250

Auxiliary Communication Support

VTAM Installation and Resource Definition
SC23-0111

VTAM Customization
SC23-0112

VTAM Operation
SC23-0113

VTAM Messages and Codes
SC23-0114

VTAM Reference Summary
SC23-0135

VTAM Programming
SC23-0115

VTAM Diagnosis Guide
SC23-0116

VTAM Diagnosis Reference
LY30-5582

VTAM Data Areas (VM)
LY30-5583

RSCS Networking Version 2 General Information
GH24-5055

RSCS Networking Version 2 Planning and Installation
SH24-5057

RSCS Networking Version 2 Operation and Use
SH24-5058

RSCS Networking Version 2 Diagnosis Reference
LY24-5228

RSCS Networking Version 2 Ref. Summary
SX24-5135

VM/Pass-Through Facility General Information
GC24-5206

VM/Pass-Through Facility Guide and Reference
SC24-5208

VM/Pass-Through Facility Logic
LY24-5208

Bibliography 891
Special Characters

/. (COMMENT) UPDATE control statement 674
/D (DELETE) UPDATE control statement 673
/I (INSERT) UPDATE control statement 672
/R (REPLACE) UPDATE control statement 673
/S (SEQUENCE) UPDATE control statement 671
.RX (BOX) format word 739
.CM (COMMENT) format word 741
.CS (CONDITIONAL SECTION) format word 742
.FO (FORMAT MODE) format word 744
.IL (INDENT LINE) format word 746
.IN (INDENT) format word 747
.MT (MENU TYPE) format word 748
.SP (SPACE LINES) format word 751
.TR (TRANSLATE CHARACTER) format word 752
&* special variable
description 867
in &IF control statement 855
setting 844
&0 special variable 868
&$ special variable
description 867
in &IF control statement 855
setting 844
&ARGS EXEC control statement described 843
&BEGEMSG EXEC control statement
ALL operand 844
description 844
&BEGPUNCH EXEC control statement
ALL operand 845
description 845
&BEGSTACK EXEC control statement
ALL operand 846
description 846
&LENGTH built-in function described 863
&CONCAT built-in function described 863
&CONTINUE EXEC control statement
description 847
used with &ERROR control statement 848
&CONTROL EXEC control statement
ALL operand 848
CMS operand 848

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming

Index 893
testing 859
&RETCODE special variable 870
&SKIP EXEC control statement, description 859
&SPACE EXEC control statement, description 860
&STACK EXEC control statement
description 860
FIFO operand 860
HT operand 861
LIFO operand 861
RT operand 861
stacking CHANGE subcommand 792
stacking INPUT subcommand 805
&SUBSTR built-in function, description 866
&TIME EXEC control statement-
description 861
OFF operand 862
ON operand 862
RESET operand 862
TYPE operand 862
&TYPE EXEC control statement, description 863
&TYPEFLAG special variable 870
$DUP edit macro, description 835
$LISTIO EXEC file
appending information to 304
creating 304
DOWN operand 837
TO operand 837
UP operand 837
* (asterisk)
entered in fileid 6
in ACCESS command 27
in ALTER subcommand 786
in CHANGE subcommand 790
in COPYFILE command 68
in DELETE subcommand 794
in DLBL command 111
in DSERV command 137
in EDIT command 141
in ERASE command 145
in EXEDROP command 155
in EXECIO command 160
in EXECLOAD command 176
in EXECMAP command 179
in EXECSTAT command 184
in FILEDEF command 195
in FILELIST command 212
in FINIS command 224
in GETFILE subcommand 802
in INCLUDE command 279
in LABELDEF command 283
in LISTS command 289
in LISTFILE command 295
in LOAD command 312
in NAMEFIND command 357
in NUCKDROP command 380
in PEEK command 396
in PRINT command 407
in PUNCH command 414
in READCARD command 474
in RENAME command 495
in REPEAT subcommand 816
in SCROLL/SCROLLUP subcommand 821
in START command 624
in STATE and STATEW commands 627
in SYNONYM command 635
in TAPE command 642
in TAPPDS command 654
in TRUNC subcommand 827
in TYPE subcommand 828
in VERIFY subcommand 830
in XEDIT command 698
with DISK option, of CMS QUERY
command 431
with RESET option
of INCLUDE command 279
of LOAD command 312
with ROUTE option, of CMS QUERY
command 447
*COPY statement 331
/ (diagonal)
used in ACCESS command 27
used in EXECUTE command 221, 470
%, used to pass null arguments to EXEC procedure,
description 867
? (question mark)
subcommand described 834
used with DISK option of FILEDEF
command 205
used with DSN option of DLBL command 111
= (equal sign)
in COPYFILE command 68, 69, 73
in RDR command 459
in RENAME command 495
= subcommand 818

A

A option of LISTIO command 303
A-disk accessed after IPLing CMS 28
ABBREV option
of CMS QUERY command 426
of CMS SET command 528
relationship to SYNONYM command 636
abbreviations
of command names 5, 528, 636
querying acceptability of 426
setting acceptability of 528
used with synonyms 636
ABENDs
debbuging your programs 755
effect on DLBL definitions 111
effect on FILEDEF definitions 202
effect on virtual screen definitions 94
effect on window definitions 98
encountered by CMSBATCH command 57
entering DEBUG environment after 755

ABNEXIT Macro
See MACREF
abnormal termination (abend)
deeding your programs 755
effect on DBLDEF definitions 202
effect on virtual screen definitions 94
effect on window definitions 98
encountered by CMSBATCH command 57
entering DEBUG environment after 755

ACCESS command
description 27
ERASE option 28
examples 29
first command after IPL 28
NODISK option 28
NOPROF option 28
NOSAVE option 28
read/only access 29
SAVEONLY option 28
usage with DEFINE command 30
access method services (AMS)
allocating VSAM space
for OS VSAM users 121
in CMS/DOS 115
determine free space extents for 289
invoking in CMS 34
LISTING file created by 34
restrictions
accessing disks with a saved file directory 28
ACK option
of NOTE command 371
of SENDFILE command 516
activating message repository files 573
ADCON-free module, installing as nucleus extension 382
ADD option
of MACLIB command 329
of NOTE command 371
of TXTLIB command 661
ALARM VSCREEN command
description 33
format 33
messages 33
operands
vname 33
responses 33
summary 13
alarm, sounding 33
ALIGN option of ASSEMBLE command 42
alignment of boundaries in assembler program statements 42
ALIGN2 option of LKED command 306
ALL
operand

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming
assembler input ddname 44
command
  ALIGN option 42
  ALOGIC option 39
  BUFSIZE option 43
  DECK option 41
description 38
  DISK option 40
  ESD option 39
  flag option 39
  LIBMAC option 40
  LINECOUN option 39
  LIST option 39
listing control options for 39
  MCALL option 39
  MLOGIC option 40
  NOALIGN option 42
  NODECK option 41
  NOEDS option 39
  NOESD option 40
  NOFLAG option 39
  NOLIBMAC option 40
  NOLIST option 39
  NOMCALL option 39
  NOMLOGIC option 40
  NONUM option 42
  NOOBJECT option 41
  NOPRINT option 40
  NOTERM option 42
  NOSTMT option 42
  NOUNT option 42
  OBJECT option 41
  PRINT option 40
  RENT option 43
  RLD option 40
  SMTT option 42
  TERMINAL option 42
  TEST option 41
  WORKSIZE option 44
  XREF option 40
  YFLAG option 43
filetype
created by the TAPPDS command 654
default CMS editor settings 839
used as input to assembler 39
assembler
  conditional assembly statements, listing 39
  overriding CMS file defaults 44
using under CMS 38
ASSGN command
  DEN option 49
description 47
  IGN option 48
  LOWCASE option 48
  PRINTER option 48
  PUNCH option 48
  READER option 48
  SYSxxx option 47
  TAPn option 48
  TERMINAL option 48
  TRCH option 48
  UA option 48
  UPPCASE option 48
  7TRACK option 48
  9TRACK option 48
assignment
  logical unit, listing 303
  system and programmer, unassigning 493
assignment statements 841
attention interruption, causing 11
ATTN Function
  See MACREF
AUTO option
  of INCLUDE command 280
  of LOAD command 313
automatic
  read function, setting 532
  save function of CMS editor
    canceling 787
    invoking 786
AUTOREAD option
  of CMS QUERY command 426, 428
  of CMS SET command 532
AUTOSAVE subcommand
description 787
  OFF operand 787
auxiliary directory, creating 230
AUXPROC directory, creating 203
B
B border command 713
backspace
  characters, how the editor handles 803
  key, used with OVERLAY subcommand 810
  BACKWARD subcommand described 788
  BASDATA filetype default CMS editor settings 839
  base address for debugging set with ORIGIN subcommand 770
  BASIC filetype default CMS editor settings 839
  BCD characters, convering to EBCDIC 71
  BDAM files, specifying in CMS 199
  blank lines, displaying at terminal during EXEC processing 860
  blanks
    as delimiters 3
    in FIND subcommand 797
  overlaying characters with 810
  trailing
    truncating from variable-length file 813
blip
  characters
These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming

C

C border command 714
calling the parsing facility from an EXEC 393
CANCEL option of NOTE command 371
canceling immediate commands 276
CANON operand of IMAGE subcommand 803
card deck, reading into virtual card reader 477
CARD option of EXECIO command 160
CASE subcommand
description 789
M operand 789
U operand 789
CAT option
of DLBL command 112
element of use 123
use in CMS/DOS 119
catalog, VSAM, verifying structure of 51
CATCHCHECK command described 51
CAW (channel address word)
changing in debug environment 774
displaying in DEBUG environment 758
CAW (channel address word) option
operand of SET subcommand 774
subcommand, description 758
CC option
of EXECIO command 161
of NOTE command 371
of PRINT command 408
CD option of DSERV command 137
CHANGE
option
of DLBL command 112
of FILEDEF command 198
of LABELDEF command 285
subcommand
effect on zone setting 833
stacking with &STACK control
statement 792
subcommand described 790
changing location of windows 405
changing number of lines and columns in a window 617
changing the current language 573
changing virtual screen attributes 605
changing window attributes 607
channel address word (CAW)
changing in debug environment 774
displaying in DEBUG environment 758
format 758
channel status word (CSW)
  changing in debug environment 774
  displaying in debug environment 760
  format 760
CHAR result of &DATATYPE built-in function 864
character
  defining logical line end in CMS 579
  displaying CMS logical line end 442
  for blip string
    displaying 426
    setting 534
  nondisplayable, defining character used in place of 585
  nondisplayable, displaying character used in place of 444
  overlaying, with OVERLAY command 810
  sets used in CMS 4
  special, changing on 3270 792
strings
  assigning to variable symbols 841
  changing 790
  copying 77
  extracting in EXEC procedure 866
  locating 808
  valid in CMS command lines 3
CHARMODE option
  of CMS QUERY command 427
  of CMS SET command 539
CHARS option of SETPRT command 612
CLEAR key in full-screen CMS 403, 561
CLEAR operand of IMMCMD command 276
CLEAR option
  of DLBL command 111
  of FILEDEF command 196
  of INCLUDE command 279
  of LABELDEF command 284
  of LOAD command 311
  of SYNONYM command 635
CLEAR VSCREEN command
  description 53
  format 53
  messages 53
  operands
    vname 53
  summary 13
  usage notes 53
CLEAR WINDOW command
  description 54
  format 54
  messages 55
  operands
    wname 54
  summary 13
  usage notes 54
  clearing a stack 103
  clearing a window 54
  closing files via FINIS command 224
CMDCALL command described 56
CMS (Conversational Monitor System)
  accessing with no virtual disks attached to
    virtual machine 28
  basic description 1
  batch facility
    See CMS batch facility
  command language, basic description 1
  commands
    See CMS (Conversational Monitor System) commands
  files
    See files
  loader
    See loader
  operand of &CONTROL control statement 848
  option of DLBL command 111
  subcommand, description 793
  subset
    See CMS subset
CMS (Conversational Monitor System) commands
  ACCESS 27
  ALARM VSCREEN 33
  AMSERV 34
  ASSEMBLE 38
  ASSGN 47
  AS border commands 711
  CLEAR VSCREEN 53
  CLEAR WINDOW 54
  CMDCALL 56
  CMSBATCH 57
  CMSSERV 59
  COMPARE 61
  CONVERT COMMANDS 63
  CONWAIT 67
  COPYFILE 68
  CP 82
  CURSOR VSCREEN 84
  DEBUG 88
  DEFAULTS 89
  DEFINE VSCREEN 92
  DEFINE WINDOW 97
  DELETE VSCREEN 101
  DELETE WINDOW 102
  DESBUF 103
  DISCARD 218
  DISK 104
  displaying during EXEC processing 848
  DLBL 110
  DOSLIB 126
  DOSLKED 129
  DROP WINDOW 135
  DROPBUF 134
  DSSERV 137
  EDIT 140
  entering 3
  entering by synonym 635
  ERASE 144
  ESERV 147
  ESTATE 627
  ESTATEW 627
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXEC</td>
<td>Execution characteristics</td>
</tr>
<tr>
<td>EXECDROP</td>
<td></td>
</tr>
<tr>
<td>EXECIO</td>
<td></td>
</tr>
<tr>
<td>EXECLOAD</td>
<td></td>
</tr>
<tr>
<td>EXECMAP</td>
<td></td>
</tr>
<tr>
<td>EXECOS</td>
<td></td>
</tr>
<tr>
<td>EXECSTAT</td>
<td></td>
</tr>
<tr>
<td>EXECUPDT</td>
<td></td>
</tr>
<tr>
<td>EXECUTE</td>
<td></td>
</tr>
<tr>
<td>FETCH</td>
<td></td>
</tr>
<tr>
<td>FILEDEF</td>
<td></td>
</tr>
<tr>
<td>FILELIST</td>
<td></td>
</tr>
<tr>
<td>FINIS</td>
<td></td>
</tr>
<tr>
<td>FORMAT</td>
<td></td>
</tr>
<tr>
<td>GENDIR</td>
<td></td>
</tr>
<tr>
<td>GENMOD</td>
<td></td>
</tr>
<tr>
<td>GENMSG</td>
<td></td>
</tr>
<tr>
<td>GLOBAL</td>
<td></td>
</tr>
<tr>
<td>GLOBALV</td>
<td></td>
</tr>
<tr>
<td>HALT</td>
<td>731</td>
</tr>
<tr>
<td>HELP</td>
<td></td>
</tr>
<tr>
<td>HELP2CONV</td>
<td></td>
</tr>
<tr>
<td>HIDE WINDOW</td>
<td>271</td>
</tr>
<tr>
<td>IDENTIFY</td>
<td></td>
</tr>
<tr>
<td>IMMCMND</td>
<td></td>
</tr>
<tr>
<td>INCLUDE</td>
<td></td>
</tr>
<tr>
<td>LABELDEF</td>
<td></td>
</tr>
<tr>
<td>LISTDS</td>
<td></td>
</tr>
<tr>
<td>LISTFILE</td>
<td></td>
</tr>
<tr>
<td>LISTIO</td>
<td></td>
</tr>
<tr>
<td>LKED</td>
<td></td>
</tr>
<tr>
<td>LOAD</td>
<td></td>
</tr>
<tr>
<td>LOADLIB</td>
<td></td>
</tr>
<tr>
<td>LOADMOD</td>
<td></td>
</tr>
<tr>
<td>MACLIB</td>
<td></td>
</tr>
<tr>
<td>MACLIST</td>
<td></td>
</tr>
<tr>
<td>MAKEBUF</td>
<td></td>
</tr>
<tr>
<td>MAXIMIZE</td>
<td>345</td>
</tr>
<tr>
<td>MINIMIZE</td>
<td>347</td>
</tr>
<tr>
<td>MODMAP</td>
<td></td>
</tr>
<tr>
<td>MOREHELP</td>
<td></td>
</tr>
<tr>
<td>MOVEFILE</td>
<td></td>
</tr>
<tr>
<td>NAMEFIND</td>
<td></td>
</tr>
<tr>
<td>NAMES</td>
<td></td>
</tr>
<tr>
<td>NOT</td>
<td>364</td>
</tr>
<tr>
<td>NOTE</td>
<td>370</td>
</tr>
<tr>
<td>NUCLDROP</td>
<td></td>
</tr>
<tr>
<td>NUCLOAD</td>
<td></td>
</tr>
<tr>
<td>NUCXMAP</td>
<td></td>
</tr>
<tr>
<td>OPTION</td>
<td></td>
</tr>
<tr>
<td>OSRUN</td>
<td></td>
</tr>
<tr>
<td>PARSECMD</td>
<td></td>
</tr>
<tr>
<td>PEEK</td>
<td></td>
</tr>
<tr>
<td>POP WINDOW</td>
<td>402</td>
</tr>
<tr>
<td>POSITION WINDOW</td>
<td>405</td>
</tr>
<tr>
<td>PRINT</td>
<td></td>
</tr>
<tr>
<td>PSERV</td>
<td></td>
</tr>
<tr>
<td>PUNCH</td>
<td></td>
</tr>
<tr>
<td>PUT SCREEN</td>
<td></td>
</tr>
<tr>
<td>PUT VSCREEN</td>
<td></td>
</tr>
<tr>
<td>QUERY</td>
<td></td>
</tr>
<tr>
<td>RDR</td>
<td></td>
</tr>
<tr>
<td>RDRLIST</td>
<td></td>
</tr>
<tr>
<td>READCARD</td>
<td></td>
</tr>
<tr>
<td>RECEIVE</td>
<td></td>
</tr>
<tr>
<td>REFRESH</td>
<td></td>
</tr>
<tr>
<td>RELEASE</td>
<td></td>
</tr>
<tr>
<td>RENAME</td>
<td></td>
</tr>
<tr>
<td>RESERVE</td>
<td></td>
</tr>
<tr>
<td>RESTORE</td>
<td>WINDOW 502</td>
</tr>
<tr>
<td>ROUTE</td>
<td></td>
</tr>
<tr>
<td>RUN</td>
<td></td>
</tr>
<tr>
<td>SCROLL</td>
<td></td>
</tr>
<tr>
<td>SENDFILE</td>
<td></td>
</tr>
<tr>
<td>SENTRIES</td>
<td></td>
</tr>
<tr>
<td>SET</td>
<td></td>
</tr>
<tr>
<td>SETPRT</td>
<td></td>
</tr>
<tr>
<td>SHOW WINDOW</td>
<td>615</td>
</tr>
<tr>
<td>SIZE WINDOW</td>
<td></td>
</tr>
<tr>
<td>SORT</td>
<td></td>
</tr>
<tr>
<td>SSERV</td>
<td></td>
</tr>
<tr>
<td>START</td>
<td></td>
</tr>
<tr>
<td>STATE</td>
<td></td>
</tr>
<tr>
<td>STATEW</td>
<td></td>
</tr>
<tr>
<td>SUMMARY</td>
<td>10</td>
</tr>
<tr>
<td>SVCTRACE</td>
<td></td>
</tr>
<tr>
<td>SYNONYM</td>
<td></td>
</tr>
<tr>
<td>TAPE</td>
<td></td>
</tr>
<tr>
<td>TAPEMAC</td>
<td></td>
</tr>
<tr>
<td>TAPPDS</td>
<td></td>
</tr>
<tr>
<td>TELL</td>
<td></td>
</tr>
<tr>
<td>TRANSIENT</td>
<td>9</td>
</tr>
<tr>
<td>TXTLIB</td>
<td></td>
</tr>
<tr>
<td>TYPE</td>
<td></td>
</tr>
<tr>
<td>UPDATE</td>
<td></td>
</tr>
<tr>
<td>UPDATE2</td>
<td></td>
</tr>
<tr>
<td>USER</td>
<td></td>
</tr>
<tr>
<td>VALIDATE</td>
<td></td>
</tr>
<tr>
<td>WAITREAD</td>
<td></td>
</tr>
<tr>
<td>WAITT</td>
<td></td>
</tr>
<tr>
<td>WRITE VSCREEN</td>
<td></td>
</tr>
<tr>
<td>XEDIT</td>
<td></td>
</tr>
<tr>
<td>XMITMSG</td>
<td></td>
</tr>
<tr>
<td>CMS</td>
<td>batch facility</td>
</tr>
<tr>
<td>CMS</td>
<td>editor in migration mode                                                      140</td>
</tr>
<tr>
<td>CMS</td>
<td>EXEC file APPENDING INFORMATION TO 296</td>
</tr>
<tr>
<td>CMS</td>
<td>creating 296</td>
</tr>
<tr>
<td>CMS</td>
<td>format 299</td>
</tr>
<tr>
<td>CMS</td>
<td>files</td>
</tr>
<tr>
<td>CMS Functions</td>
<td>See files</td>
</tr>
<tr>
<td>CMS Functions</td>
<td>See MACREF</td>
</tr>
</tbody>
</table>

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming

Index 899
CMS immediate commands
See immediate commands
CMS LOADLIBs
compressing with LOADLIB command 323
  copying with LOADLIB command 323
  creating with LKED command 306
  executing a load module from 392
  listing with LOADLIB command 323
CMS Macro Instructions
See MACREF
CMS subset
  entering 793
  returning to edit mode 818
CMSAMS saved system name 598
CMSBAM saved system name 598
CMSBATCH command
  description 57
  recursive abends encountered by 57
CMSDOS saved system name 598
CMSENDLEVEL option, of CMS QUERY command 428
CMSLIB assembler macro library ddname 44
CMSPF keys
  canceling 542
  defining to execute a command 541
  displaying definitions 428
  RETRIEVE function 543
  using NOWRITE option 543
CMSPF option
  of CMS QUERY command 428
  of CMS SET command 541
CMSSERV command
  description 59
  format 59
  messages 60
  summary 13
  usage notes 59
CMSTYPE option
  of CMS QUERY command 429
  of CMS SET command 544
CMSUT1 file
  created by DISK LOAD command 106
  created by README command 476
  created by TAPE LOAD command 646
  created by TAPPDS command 655
CMSVSAAM saved system name 598
COBOL
  compilers
  querying options in effect for 444
  specifying options for in CMS/DOS 389
  filetype default CMS editor settings 839
COL option
  ofCOMPARE command 61
  ofTYPE command 666
columns
  comparing disk files by 61
  displaying particular
    with TYPE command 666
    with TYPE subcommand 828
  of data, copying 77
  specifying
    for copy operations 77
    for verification setting 830
    for zone setting 832
columns, changing number in a window 617
COL1 option of TAPPDS command 656
commands
  abbreviating 5
  defaults shown by underscore in command
  format box 6
  entering 3
  entering from the WM window 403
  environment
    CMS 1
    CP 1
  execution, halting 731
  keyboard differences in entering 11
  language, CMS 2
  modules, creating 231
  operands 3
  options 3
  stacking in terminal input buffer 11
  truncating 5
  valid in CMS subset 793
  when to enter 11
COMMANDS ASSEMBLE utility file 65
COMMANDS CMSUT1 utility file 65
COMMANDS CMSUT2 utility file 65
COMMANDS TEXT utility file 65
commands you can enter in the WM window 403
COMMENT (.CM) format word 741
COMMENTS option of EXECUPDT command 187
COMP
  of DOSLIB command 126
  option
    of FETCH command 190
    of MACLIB command 329
COMPARE command
  COL option 61
  description 61
  comparing files 61
  comparison operators in EXEC procedure 854
  compilers used under CMS 2
  compiling a message repository file 236
  components
    of VM/SP 1
    of VM/370 1
COMPRESS option
  of EXECUPDT command 187
  of LOADLIB command 323
COMPSWT Macro
  See MACREF
CONCAT option of FILEDEF command 199
conditional execution
  &IF control statement 854
  &LOOP control statement 856
CONDITIONAL SECTION (.CS) format word 742
connecting a window to a virtual screen 271, 615
console
read, controlling after CMS command execution 532
stack
  clearing 103
  reading data in EXEC procedure 858
stacking lines, &BEGSTACK control statement 846
stacking lines, &STACK control statement 860
stacking lines, STACK subcommand 824
testing whether it is empty 870
terminal input buffer
  clearing 103
CONSOLE value of &READFLAG special variable 870
constants, altering
  with LOAD command 320
  with STORE subcommand 776
continuation character
  on COPYFILE specification list 78
  on COPYFILE translation list 79
Control Program (CP)
  See also CP (Control Program)
  basic description 1
  commands
    See CP (Control Program) commands
Control Program (CP) commands
  description 82
  executing
    in CMS command environment 82
    in EXEC procedure 82
    in jobs for CMS batch facility 82
  implied 568
  when to use 82
control statements
  for access method services 34
  for UPDATE command 670
conventions, notational 5
Conversational Monitor System (CMS)
  accessing with no virtual disks attached to virtual machine 28
  basic description 1
  batch facility
    See CMS batch facility
  command language, basic description 1
  commands
    See CMS (Conversational Monitor System) commands
files
  See files
loader
  See loader
subset
  See CMS subset
Conversational Monitor System (CMS) commands
  ACCESS 27
  ALARM VSCREEN 33
  AMSERV 34
ASSEMBLE 38
ASSGN 47
border commands 711
CLEAR VSCREEN 53
CLEAR WINDOW 54
CMDCALL 56
CMSBATCH 57
CMSSERV 59
COMPARE 61
CONVERT COMMANDS 63
CONWAIT 67
COPYFILE 68
CP 82
CURSOR VSCREEN 84
DEBUG 88
DEFAULTS 89
DEFINE VSCREEN 92
DEFINE WINDOW 97
DELETE VSCREEN 101
DELETE WINDOW 102
DSBUF 103
DISCARD 218
DISK 104
displaying during EXEC processing 848
DLBL 110
DOSLIB 126
DOSLKED 129
DROP WINDOW 135
DROPBUF 134
DSEVR 137
EDIT 140
entering 3
entering by synonym 635
ERASE 144
ESERV 147
ESTATE 627
ESTATEW 627
EXEC 150
EXECDROP 155
EXECIO 158
EXECLOAD 176
EXECMAP 179
EXECOS 182
EXECSTAT 184
EXECUPDT 186
EXECUTE 218
execution characteristics 9
FETCH 190
FILEDEF 193
FILELIST 212
FINIS 224
FORMAT 225
GENDIRT 230
GENMOD 231
GENMSG 236
GLOBAL 243
GLOBALV 246
halting execution 731

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming

Index 901
HELP 258
HELPCONV 269
HIDE WINDOW 271
IDENTIFY 273
IMMCMD 276
INCLUDE 278
LABELDEF 283
LISTDS 289
LISTFILE 295
LISTIO 303
LKED 306
LOAD 311
LOADLIB 323
LOADMOD 327
MACLIB 329
MACLIST 333
MAKEBUF 344
MAXIMIZE WINDOW 345
MINIMIZE WINDOW 347
MODMAP 348
MOREHELP 349
MOVEFILE 352
NAMEFIND 356
NAMES 364
not for general users 19
NOTE 370
NUCXDROP 380
NUCXLOAD 382
NUCXMAP 387
OPTION 389
OSRUN 392
PARSECMD 393
PEEK 396
POP WINDOW 402
POSITION WINDOW 405
PRINT 407
FSERV 412
PUNCH 414
PUT SCREEN 418
PUT VSCREEN 420
QUERY 422
RDR 459
RDRLIST 463
READCARD 474
RECEIVE 481
REFRESH 492
RELEASE 493
RENAME 496
RESERVE 498
RESTORE WINDOW 502
ROUTE 503
RSERV 506
RUN 508
SCROLL 511
search order 7
SENDFILE 515
SENTRIES 526
SET 527
SETPRT 612
SHOW WINDOW 615
SIZE WINDOW 617
SORT 619
SSERV 622
START 624
STATE 627
STATEW 627
summary 10
SVTRACE 630
SYNONYM 635
TAPE 641
TAPEMAC 650
TAPPDS 654
TELL 659
transient area 9
TXTLIB 661
TYPE 665
UPDATE 668
user area 9
valid in CMS subset 793
VALIDATE 682
WAITREAD VSCREEN 684
WAITT VSCREEN 688
WRITE VSCREEN 690
XEDIT 697
XMITMSG 704
CONVERT COMMANDS command
description 63
examples 66
format 63
messages 66
operands
ALL 64
CHECK 64
FIFO 64
fn ft fn 63
LIFO 64
OUTMODE 64
STACK 64
SYSTEM 63
USER 63
summary 13
usage notes 65
converting DLCS files
checking for DLCS coding errors 64
converting for the parsing facility 63
example 66
from an EXEC 65
from XEDIT 65
output files 64
utility files 65
converting message repository files
converting into internal form 236
example 238
output files 237
CONWAIT command
description 67
using 67
COPIES option of SETPRT command 612
COPY
These symbols are used in the index to refer to other VM and VM/SP books:

- MACREF—VM/SP CMS Macros and Functions Reference
- CPPROG—VM/SP CP for System Programming

Index 903
for DLBL command, restrictions for OS
users 113
relating to CMS file 193
to identify VSAM catalogs (CMS/DOS) 118
to identify VSAM catalogs (OS VSAM
tusers) 123
used by assembler 44

DDR Command
See CPPROG

DEBUG
command described 88
subcommand
BREAK 756
CAW 758
CSW 760
DEFINE 762
DUMP 764
GO 766
GPR 768
HX 769
ORIGIN 770
PSW 772
RETURN 773
SET 774
STORE 776
X 778

DECK option
of ASSEMBLE command 41
of OPTION command 389

DEFAULTS command
description 89
LIST option 89
SET option 89
valid CMS command options 89

DEFINE subcommand described 762

DEFINE VSSCREEN command
description 92
examples 95
format 92
messages 96
op
derands
color 93
cols 92
exthi 93
HIGH 93
lines 92
NOHIGH 93
NOPROTECT 93
NOTYPE 93
PROTECT 93
psset 93
robot 92
rttop 92
SYSTEM 94
TYPE 93
USER 94
vname 92
summary 14
usage notes 94,99

DEFINE WINDOW command
description 97
format 97
messages 100
op
derands
BORDER 98
cols 97
FIXED 98
lines 97
NOBORDER 98
NPOP 98
NOTOP 98
POP 98
pscol 97
psline 97
SYSTEM 98
TOP 98
USER 98
VARIABLE 98
vname 97
summary 14
defining a virtual screen 92
defining a window 97

Definition Language for Command Syntax (DLCS)
See DLCS (Definition Language for Command Syntax)

DEL option
of DOSLIB command 126
of MACLIB command 329
of TXTLIB command 661

DELETE
c
control statement, for UPDATE command 673
subcommand, description 794

DELETE VSSCREEN command
description 101
format 101
messages 101
op
derands
vname 101
summary 14
usage notes 101

DELETE WINDOW command
description 102
format 102
messages 102
op
derands
oper 102
summary 14
usage notes 102
deleting a virtual screen definition 101
deleting a window definition 102
deleting, program stack buffer 134

DEN option
of ASSGN command 49
of FILEDEF command 200
of TAPE command 645

DESBUF command described 103

DET option in RELEASE command 493
detaching a disk from virtual machine
configuration 493
**DETAIL HELP**

- described 261
- obtaining using HELP command 258
- obtaining using MOREHELP command 349

**DIRECT**

- filetype default CMS editor settings 839
- directories
  - CMS auxiliary 230
  - CMS file, writing to disk 493
  - of VSE libraries, sorting 138
- DISCARD command
  - use with FILELIST command 218
  - use with PEEK command 400
  - use with RDRLIST command 468
- discontiguous, shared segment, saved system names 598

**DISK**

- command
  - description 104
  - DUMP operand 104
  - LOAD operand 104
  - FULLPROMPT option 105
  - MINPROMPT option 105
  - NOPROMPT option 105
  - NOREPLACE option 105
  - option
    - interactive use with FILEDEF command 205
    - of ASSEMBLE command 40
    - of CMS QUERY command 430
    - of DOSLIB command 127
    - of DOSLKED command 130
    - of DSERV command 138
    - of FILEDEF command 195
    - of LKD command 308
    - of LOADLIB command 324
    - of MACLIB command 330
    - of PSERV command 412
    - of RSENV command 506
    - of SSERV command 622
    - of TAPI command 644
    - of TXTLIB command 662
    - of UPDATE command 670
    - use with FILEDEF command 204
  - REPLACE option 105
  - verifying access 682

**Disk Operating System (DOS)**

- disks, accessing 30
- files
  - listing information 289
  - specifying FILEDEF options for 202

**DISKID Function**

- See MACREF

**DISKR option of EXECIO command 160**

- accessing 27
- detaching 493
- determining

---

These symbols are used in the index to refer to other VM and VM/SP books:

- MACREF—VM/SP CMS Macros and Functions Reference
- CPPROG—VM/SP CP for System Programming

Index 905
DLCS (Definition Language for Command Syntax)
checking for DLCS coding errors 64
converting a DLCS file 63
converting files from an EXEC 65
converting from XEDIT 65
displaying contents of synonym and translation
tables 450
enabling user DLCS definitions 573
example 66
output files from conversion 64
setting translations and synonyms 601
specifying use of translation tables 601
utility files 65
DOS (Disk Operating System)
disks, accessing 30
files
   listing information 289
   specifying FILEDEF options for 202
DOS option
   of CMS QUERY command 434
   of CMS SET command 546
   of GENMOD command 232
DOSLIB
command
   COMP option 126
   DEL option 126
   description 126
   DISK option 127
   MAP option 126
   PRINT option 127
   TERM option 127
files
   adding phases to 131
   fetching phases from 190
   identifying for fetching 243
   listing information about members 126
   output filemode 129
   size considerations 127
   space considerations 131
   which DOSLIBs will be searched 434
option
   of CMS QUERY command 434
   of GLOBAL command 243
DOSLKED command
description 129
DISK option 130
PRINT option 130
TERM option 130
DOSLNCNT option
   of CMS QUERY command 434
   of CMS SET command 548
DOSLNK filetype
   CMS/DOS linkage editor input 129
   creating 130
DOSPART option
   of CMS QUERY command 435
   of CMS SET command 549
DOWN
   operand of $MOVE edit macro 837
   subcommand, description 795
DROP WINDOW command
description 135
format 135
messages 136
operands
   * (asterisk) 135
   = (equal sign) 135
   n 135
   WM 135
   wname 135
summary 14
usage notes 135
DROPBUF command
description 134
using 134
dropping a window 135
dropping EXECs and editor macros from
storage 155
dropping the WM window 135
DSERV command
   ALL operand 137
   CD operand 137
description 137
   DISK option 138
   PD operand 137
   PRINT option 138
   RD operand 137
   SD operand 137
   SORT option 138
   TD operand 137
   TERM option 138
DSN option of DLBL command 111
DSORG option of FILEDEF command 200
DSTRING subcommand described 795
DUMMY option
   of DLBL command
      restrictions for OS VSAM user 120
      using in CMS/DOS 114
   of FILEDEF command 195
DUMP
   option
      of DISK command 104
      of OPTION command 389
      of TAPE command 642
DUP option
   of INCLUDE command 280
   of LOAD command 313
DVOL1 operand of TAPE command 643

E

EBCDIC
display file in 665
   of COPYFILE command 71
EDIT
command
description 140

906 VM/SP CMS Command Reference
These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming
ESERV command described 147
ESTATE command described 627
ESTATEW command described 627
ETMODE option of EXECUPDT command 187
EXCLUDE SYSIN control statement 325
EXDTE operand of LABELDEF command 285
EXEC
  built-in functions
    &CONCAT 863
    &DATATYPE 864
    &LENGTH 865
    &LITERAL 865
    &SUBSTR 866
    when to use 863
command
description 150
implied 568
control statements
  &ARGS 843
  &BEGEMSG 844
  &BEGPUNCH 845
  &BEGSTACK 846
  &BEGTYPE 847
  &CONTINUE 847
  &CONTROL 848
  &EMSG 850
  &END 851
  &ERROR 851
  &EXIT 852
  &GOTO 853
  &HEX 853
  &IF 854
  &LOOP 856
  &PUNCH 857
  &READ 858
  &SKIP 859
  &SPACE 860
  &STACK 860
  &TIME 861
  &TYPE 863
assignment statement 841
displaying during EXEC processing 848
files
  $LISTIO EXEC created by LISTIO
  CMS EXEC file created by LISTFILE
command 296
executing with the RUN command 508
using IMMCMMD from 277
filetype
default CMS editor settings 839
record format 150
option
  of LISTFILE command 296
  of LISTIO command 304
procedures
  branching with &GOTO control statement 853
  branching with &SKIP control statement 859
comparing tokens in 854
concatenating tokens in 863
defining synonyms for 636
ESERV 147
executing 150
exiting from 852
halting terminal output during 861
passing arguments to nested EXEC procedures 869
reading data from terminal during 858
reading from and writing to windows 685
resuming terminal output during 861
RUN EXEC 508
using windows 685
using WRITE VSCREEN 693
special variables
  &* 867
  &$ 867
  &DISK* 868
  &DISK? 868
  &DISKx 868
  &DOS 868
  &EXEC 869
  &GLOBAL 869
  &GLOBALn 869
  &INDEX 869
  &INDEX, setting 844
  &LINENUM 869
  &n 867
  &READFLAG 870
  &RETCODE 870
  &TYPEFLAG 870
  &1 through &30 867
variables for PARSECMD 394
variables for WAITREAD VSCREEN 684
variables returned
code.n 394
code.n values 394
token.n 394
WAITREAD.n 684
EXEC 2 procedures, executing 150
EXECDROP command
description 155
EXECIO command
  CARD option 160
  CC option 161
  CP option 160
  description 158
  DISKR option 160
  DISKW option 160
  EMSG option 161
  machine code 161, 171
  PRINT option 161
  PUNCH option 161
  STEM option 163
  VAR option 163
EXECLOAD command
description 176
908 VM/SP CMS Command Reference
EXECMAP command
description 179
FIFO option 180
LIFO option 180
EXECOS command
description 182
EXECs and editor macros in storage
controlling system searching of Installation
DCSS 571
discontinue use of Installation DCSS 155
listing 179
querying status of Installation DCSS 438
removing 155
EXECSTAT command
description 184
EXECTRAC option
of CMS QUERY command 435
of CMS SET command 551
EXECUPDT command
COMMENTS option 187
COMPRESS option 187
CTL option 187
description 186
ETMODE option 187
HISTORY option 187
NOCOMMENTS option 187
NOCOMPRESS option 187
NOHISTORY option 187
NOSID option 188
NOUPDATE option 188
SID option 187
used with System Product interpreter source
programs 186
using UPDATE options with 188
EXECUTE command
used in FILELIST environment 218
used in RDRLIST environment 468
execution characteristics of CMS commands 9
EXIT operand
exiting the WM window 135
expanding a window size to the physical screen
size 345
extensions
read-only
accessing 27
editing files on 141
releasing 493
EXTENT option
of DLBL command 112
of DLBL for CMS/DOS users 114
of LISTDS command 290
EXTERNAL, CP command 755

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming

Index 909
NL operand 207
NOCHANGE option 198
NOEOV option 201
NSL operand 207
OPTCD option 199
PERM option 198
positioning read/write pointer 199
PRINTER option 195
PUNCH option 195
READ option 195
RECFM option 198
SL operand 206
SUL operand 206
SYSPRM option 207
TAPn option 195
TERMINAL option 195
TRTCH option 199
UPCASE option 201
VOLID operand 207
when to use (OS users) 120
when to use in CMS/DOS 115
XTENT option 199
18TRACK option 200
7TRACK option 200
9TRACK option 200
definitions
  clearing 196
  displaying 204
  option of CMS QUERY command 435, 449
fileid in command syntax 6
FILELIST
  command
    APPEND option 213
    description 212
    DISCARD command, used in FILELIST environment 218
    example 220
    EXECUTE command, used in FILELIST environment 218
    FILELIST option 213
    NOFILELIST option 213
    PROFILE option 213
    setting defaults 213
  option
    of FILELIST command 213
    of SENDFILE command 516
filemode
  changing
    with COPYFILE command 73
    with FMODE subcommand 798
  displaying, FMODE subcommand 798
letter
  establishing 27
  replacing 493
  numbers, changing 496
specifying on READCARD command 474
specifying when receiving a file 481
filename
  changing with FNAME subcommand 799
  of EXEC file testing 869
files
  creating
    with CMS editor 140
    with COPYFILE command 68
    with READCARD command 474
    with the System Product Editor 697
defining for CMS/DOS 110
display new name for, after renaming 495
identifier
  entering on FILEDEF command 204
  entering on LISTDS command 289
  in command syntax 6
inserting lines
  with INPUT subcommand 804
  with UPDATE command 668
listing information about 212
loading
  from tape to disk 642
  from virtual reader to disk 481
modifying 68
moving from device to device 352
numbering lines in 822
overlaying data in
  specifying number of lines to overlay 815
  with COPYFILE command 68
  with OVERLAY subcommand 810
packing 71
  specifying fill character 71
printing
  in hexadecimal format 409
  specifying number of lines per page 408
processed by TAPE command, listing 644
protecting data during edit session 833
punched, restoring to disk
  with DISK LOAD command 104
  with READCARD command 474
  with RECEIVE command 481
punching to virtual card punch 104
reading
  from virtual card reader 104
  receiving from your virtual reader 481
relating to OS ddname 193
renaming 495
renumbering lines in 814
replacing lines in
  with REPLACE subcommand 816
  with UPDATE command 669
replacing old file with new copy 69
sort records in 619
tape, writing to disk 642
transferring with DISK DUMP command 104
unpack 71
verifying existence of
  with ESTATE and STATEW commands 627
  with STATE and STATEW commands 627
verifying syntax of fileid
  with VALIDATE command 682
writing to disk
with AUTOSAVE subcommand 787
with FILE subcommand 796
with SAVE subcommand 820
filetypes, reserved default CMS editor settings 839
FILL option of COPYFILE command 71
FIND subcommand
description 797
effect of image setting 803
FINIS command described 224
first-in first-out (FIFO) stacking in EXEC
procedure 846, 860
fixed-length files, converting to variable-length 76, 813
FLAG option of ASSEMBLE command 39
FLASH option of SETPRT command 612
FMODE
option of LISTFILE command 297
subcommand, description 798
fn ft fm used to represent file identifier 6
FNAME
option of LISTFILE command 297
subcommand, description 799
FOR option of COPYFILE command 70
FORM operand
FORMAT command
command
BLKSIZE option 226
description 225
examples 227
LABEL option 226
NOERASE option 226
performance considerations 227
RECOMP option 226
selecting appropriate blocksize 228
option
of LISTDS command 290
of LISTFILE command 298
subcommand
description 800
DISPLAY operand 800
LINE operand 800
FORMAT MODE (.FO) format word 744
FORTRAN filetype default CMS editor settings 839
FORWARD subcommand described 801
FREE option of LISTDS command 290
FREEFORT
files, renumbering 814
filetype default CMS editor settings 839
FRLABEL option of COPYFILE command 70
FROM option
of COPYFILE command 70
of GENMOD command 231
FSF Macro
See MACREF
FSFBD Macro
See MACREF
FSFCLOSE Macro
See MACREF
See MACREF
FSEQ operand of LABELDEF command 284
FSERASE Macro
See MACREF
FS tape control function 643
FSOPEN Macro
See MACREF
FSPOINT Macro
See MACREF
FSR tape control function 643
FSREAD Macro
See MACREF
FSSTATE Macro
See MACREF
FWRITE Macro
See MACREF
FTYPE option of LISTFILE command 297
full-screen CMS
border commands 711
BRKKEY setting, default changed 555
CLEAR key 403, 561
clearing a virtual screen 53
clearing a window 54
connecting a window to a virtual screen 615
copying the physical screen to a CMS file 418
default connections of windows and virtual
screens 560
default message routing 560
default virtual screens 559
default windows 557
default windows and virtual screens
defined 554
defining a virtual screen 92
defining a window 97
defining fields in a virtual screen 690
deleting a virtual screen definition 101
deleting a window definition 102
dropping a window 135
dropping the WM window 135
entering 554
entering information in a virtual screen 690
erasing data in a virtual screen 53
general information 555
location indicator 581
maximizing windows 345
message routing 503
minimizing windows 347
nesting SUSPEND and RESUME 561
null characters, recognizing 552
PA2 key 403, 561
placing a window at top of display order 615
popping a window 402
positioning the cursor in a virtual screen 84
positioning windows 405
querying status of 436
refreshing a screen 492
refreshing a screen from an EXEC 684
restoring windows 502

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming

Index 911
resuming full-screen CMS 554
returning to line-mode CMS 554
running applications in full-screen CMS 561
screen display, refreshing from an EXEC 684
scrolling process 555
sounding the alarm 33
specifying character attributes in a virtual screen 690
suspending temporarily 554
system defaults 557
updating a virtual screen from an EXEC 684
updating the plane buffers in a virtual screen 690
updating virtual screen with data 688
writing data in a virtual screen 690
writing lines from a file to a virtual screen 241
writing virtual screen data to a CMS file 420
FULLPROMPT option
of DISK command 105
of READCARD command 475
of RECEIVE command 483
FULLREAD option
of CMS QUERY command 435
of CMS SET command 552
FULLSCREEN option
of CMS QUERY command 436
of CMS SET command 554

GEN option
of MACLIB command 329
of TXTLIB command 661
GENDIRT command described 230
general registers
  changing, in debug environment 774
  displaying in debug environment 768
  printing contents of 764
GENMOD command
  ALL option 232
  description 231
  DOS option 232
  FROM option 231
  MAP option 231
  NOMAP option 232
  NOSTR option 232
  OS option 232
  STR option 232
  SYSTEM option 233
  TO option 231
GENMSG command
  description 236
  examples 238
  format 236
  messages 239
  operands

applid 236
CP 236
DBC 237
fn ft fm 236
langid 236
LIST 237
MARGIN 237
NODBCS 237
NOLIST 237
NOOBJECT 237
NOXREF 237
OBJECT 237
XREF 237
summary 15
usage notes 237
GENN operand of LABELEF command 284
GENV operand of LABELEF command 285
GET sub-function of GLOBALV command 251
GET VSCREEN command
  operands
  summary 15
GETFILE subcommand described 801
global changes
  querying
    which DOSLIBs were last specified 434
    which MACLIBs were last specified 443
    which TXTLIBs were last specified 452
  with ALTER subcommand 786
  with CHANGE subcommand 790
  with OVERLAY subcommand 810
GLOBAL command
  description 243
  DOSLIB option 243
  LOADLIB option 243
  MACLIB option 243
  TXTLIB option 243
global variables, creating 246
GLOBALV command
  description 246
  examples 254
  GET sub-function 251
  GRPLIST option 251
  GRPSTACK option 251
  INIT option 248
  LIST option 249
  PURGE option 251
  PUT sub-function 250
  SELECT option 248
  SESSION file 246
  STACK option 249
  STACKR option 250
  use in CMS EXECs 253
GO subcommand described 766
GPR subcommand described 766
GPR
  operand of SET subcommand 774
  subcommand described 768
GRAF option of FILEDEF command 196
GRPLIST option of GLOBALV command 251
GRPSTACK option of GLOBALV command 251
H border command 717
handling remote data transmission 594
HB immediate command described 727
header
card
as READ control card 476
punched by PUNCH command 414
for LISTFILE command 296
format 301
HEADER option
of LISTFILE command 296
of PUNCH command 414
HELP
command
description 258
format 258
messages 268
sample requests 267
setting defaults 90
usage notes 264
format words
.BX (BOX) 739
.CM (COMMENT) 741
.CS (CONDITIONAL SECTION) 742
.FO (FORMAT MODE) 744
.IL (INDENT LINE) 746
.IN (INDENT) 747
.MT (MENU TYPE) 748
.OF (OFFSET) 749
.SP (SPACE LINES) 751
.TR (TRANSLATE CHARACTER) 752
summary 737
Layering options 262
obtaining additional or related information 349
operands
ALL 263
BRIEF 262
component-name command-name 259
DESCRIPT 263
DETAIL 262
ERRORS 263
EXTEND 264
FORMAT 263
HELP 259
menuname MENU 259
MESSAGE message-id 261
MSG 261
NOSCREEN 264
NOTES 263
NOTYPE 264
OPTIONS 263
PARMS 263
RELATED 263
SCREEN 264
taskname TASKS 259
TASKS 258
Other options 264
Subsetting options 263
HELPX command
description 259
HEX option
of PRINT command 409
of TYPE command 666
hexadecimal
conversion in assignment statement 842
display in file 665
printing file in 409
substitution
in EXEC procedure 842
invoking in EXEC procedure 842
suppressing in EXEC procedure 842
values, displaying in EXEC procedure 854
HI (Halt Interpretation) immediate command 728
hidden windows, displaying information about 436
HIDE option
of CMS QUERY command 436
HIDE WINDOW command
description 271
format 271
messages 272
operands
col 271
line 271
vname 271
wname 271
summary 15
usage notes 271
hiding a window 271
HISTORY option of EXECUPDT command 187
HNDEXT Macro
See MACREF
HNDINT Macro
See MACREF
HNDSCV Macro
See MACREF
HO immediate command described 729
HT immediate command described 730
HX immediate command
DEBUG subcommand, description 769
Immediate command 731
effect on DLBL definitions 112
effect on FILEDEF definitions 202
HX immediate command described 731

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming

Index 913
ICS control statement
See include control section (ICS) loader control statement
ID CARD, CP, example 477
ID operand
of TAPEMAC command 651
of TAPPDS command 655
IDENTIFY command
description 273
display user information 274
FIFO option 273
LIFO option 273
STACK option 273
TYPE option 273
IEBPTPCH utility program, creating CMS files from tapes created by 654
IEBUPDTE utility program, creating CMS files from tapes created by 654
IEHMOVE utility program
creating CMS files from tapes created by 654
creating CMS MACLIBs from tapes created by 650
IGN option of ASSGN command 48
with DUMMY data sets 115
IJSYSCL defining in CMS/DOS 114
IJSYST defined 123
in CMS/DOS 118
IJSYSSL defining in CMS/DOS 114
IJSYSRC defining in CMS/DOS 114
IJSYSUC defined 123
in CMS/DOS 118
image setting
effect on FIND subcommand 797
effect on logical tab settings 826
IMAGE subcommand
CANON operand 803
description 803
OFF operand 803
ON operand 803
IMESCAPE option
of CMS QUERY command 437
of CMS SET command 565
IMMCMD command
CLEAR operand 276
description 276
QUERY operand 276
SET operand 276
STATUS operand 276
operand of NUCXLOAD command 383
IMMCMD Macro
See MACREF
immediate commands
canceling 276
creating 276
HB 727
HI 728
HO 729
HT 730
HX 731
RO 732
RT 733
SO 734
summary 11
TE 735
TS 736
IMPCP option
of CMS QUERY command 437
of CMS SET command 567
IMPEX option
of CMS QUERY command 437
of CMS SET command 568
implied
CP function
query status of 437
setting 567
EXEC function
query status of 437
setting 567
INC option of UPDATE command 669
INCLUDE command
AUTO option 280
CLEAR option 279
description 278
DUP option 280
effect on loader tables 577
examples 281
following LOAD command 281
HIST option 281
identify TXTLIBs to be searched 243
INV option 279
LIBE option 280
MAP option 279
NOAUTO option 280
NOCLEAR option 279
NODUP option 280
NOHIST option 281
NOINV option 279
NOLIBE option 280
NOMAP option 279
NOREP option 280
NOTYPE option 279
ORIGIN option 279
REP option 280
RESET option 279
RLDSAVE option 280
SAME option 280
START option 280
TYPE option 279
include control section (ICS) loader control statement 318
INCR option of XEDIT command 700
increment
specifying for line-number editing 812
specifying for sequence numbers in file 822
INDEX (.IN) format word 747
INDEX LINE (.IL) format word 746
INIT option
  of GLOBALV command 248
  of SETPRT command 613
INMOVE default for MOVEFILE command
dname 352
INPUT
  effect of IMAGE setting 803
  on = subcommand line 819
  option
    of CMS QUERY command 438
    of CMS SET command 569
  subcommand
    description 804
    stacking with &STACK 805
input mode 785
  during line number editing 806
  entering 804
  leaving 785
INSERT control statement for UPDATE
  command 672
instructions
  addresses, halting program execution at 756
  altering
    with LOAD command 320
    with STORE subcommand 776
INSTSEG option
  of CMS QUERY command 438
  of CMS SET command 571
Interactive Problem Control System (IPCS) 1
interruptions
  entering debug environment after 755
  handling
INV option
  of INCLUDE command 279
  of LOAD command 312
IPCS (Interactive Problem Control System) 1
issuing a message from a repository 704
ITEMCT option of TAPEMAC command 651

K

KEY option
  of CMS QUERY command 438
  key, displaying last pressed 438
KEYLEN option of FILEDEF command 199

L

L border command 718
LABEL option
  of FORMAT command 226
  of LISTFILE command 298
LABELDEF
  command
    CHANGE option 285
    CLEAR operand 284
    CRDTE operand 285
    description 283
    EXDTE operand 285
    FID operand 284
    FSEQ operand 284
    GENN operand 284
    GENV operand 285
    NOCHANGE option 285
    PERM option 285
    SEC operand 285
    VOLID operand 284
    VOLSEQ operand 284
  operand of CMS QUERY command 439
LABOFF operand of FILEDEF command 206
LANGGEN command 19
LANGLIST option
  of CMS QUERY command 439
LANGMERG command 19
LANGUAGE option
  of CMS QUERY command 440
  of CMS SET command 573
language, changing the current 573
languages, national
  changing the current language 573
  controlling language translation tables 601
  displaying language identifiers 439
  displaying the current language 440
  displaying translations and synonyms in
effect 450
  suppressing translations and translation
  synonyms 601
last-in first-out (LIFO) stacking in EXEC
  procedure 846, 861
LDRTBLS option
  of CMS QUERY command 441
  of CMS SET command 577
LEAVE option
  of FILEDEF command 201
  of TAPE command 645
LEFT operand of LINEMODE subcommand 806
LET option of LKED command 306
LIBE option
  of INCLUDE command 280
  of LKED command 307
  of LOAD command 313
LIBMAC option of ASSEMBLE command 40

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming
libraries
OS, macro libraries
See macro libraries, OS
VSE
assigning logical units 49
obtain information about 137
VSE core image
defining IJSYSCL 114
fetching phases from 190
VSE procedure
copying procedures from 412
displaying directories 137
displaying procedures from 412
printing procedures from 412
punching procedures from 412
VSE relocatable
assigning SYSRLB 507
copying modules from 506
defining IJSYSRL 114
displaying modules from 506
link-editing modules from 129
printing modules from 506
punching modules from 506
VSE source statement
assigning SYSSSLB 623
copying books 622
copying macros from 147
defining IJSYSSL 114
displaying books 622
printing books 622
punching books 622
LIBRARY
loader control statement 317
option, of CMS QUERY command 441
LIFO (last-in/first-out)
operand
of &BEGSTACK control statement 846
of &STACK control statement 861
of CONVERT COMMANDS command 64
option
of IDENTIFY command 273
of NAMEFIND command 357
of NUUXMAP command 387
of RDR command 460
LIMCT option of FILEDEF command 199
line end character
defining 579
displaying in CMS 442
implications when you redefine 579
recognizing 579
LINE operand of FORMAT subcommand 800
line-number editing
displaying line numbers 805
inserting single line 804
left-handed 806
reserializing records in file 822
right-handed 806
setting prompting increment for 812
LINECOUN option
of ASSEMBLE command 39
of PRINT command 408
LIEXEDIT Macro
See MACREF
LINEMODE subcommand
description 805
LEFT operand 806
OFF operand 806
RIGHT operand 806
LINEEND option
of CMS QUERY command 442
of CMS SET command 579
LINEENUM option of NAMEFIND command 357
lines
duplicating, in CMS file 835
image, of record 803
locating by beginning character string 797
mode
of CMS editor 142
of 3270 800
moving, within CMS file 836
number of EXEC statement testing 869
printing
punching in EXEC procedure 857
reading from console stack 824
lines, changing number in a window 617
LINK command, accessing disks after 29
link-editing
in CMS/DOS 129
modules from VSE relocatable libraries 129
TEXT files in storage 311
TXTLIB members 663
linkage editor control statements
OS
read by TXTLIB command 663
required format for TXTLIB command 663
VSE supported in CMS/DOS 130
LIST option
of ASSEMBLE command 39
of DEFAULTS command 89
of GLOBALV command 249
of LKED command 307
of LOADLIB command 323
of OPTION command 389
LISTDS command
description 289
examples 291
EXTENT option 290
FORMAT option 290
FREE option 290
PDS option 290
LISTFILE command
ALLOC option 298
APPEND option 296
ARGS option 296
BLOCKS option 298
DATE option 298
description 295
EXEC option 296
FIFO option 297
FMODE option 297
FNAME option 297
FORMAT option 298
FTYPE option 297
HEADER option 296
LABEL option 298
LIFO option 297
NOHEADER option 296
STACK option 296
TRACE option 296
XEDIT option 297
listing EXECs and editor macros in storage 179
LISTING filetype
created by access method services 34
created by ASSEMBLE command 39
controlling 39
created by ESERV program 147
default CMS editor settings 839
printing 408
LISTIO command
A option 303
ALL option 303
APPEND option 304
description 303
EXEC option 304
PROG option 303
STAT option 304
SYS option 303
SYSxxx option 303
UA option 303
LISTX option of OPTION command 390
literal substitution with XMITMSG command 704
literval values used in EXEC procedure 865
LKED command
ALIGN2 option 306
description 306
DISK option 308
LET option 306
LIBE option 307
LIST option 307
MAP option 307
NAME option 307
NCAL option 306
NE option 307
NOPRINT option 308
NOTERM option 307
OL option 307
OVLY option 307
PRINT option 308
REFR option 307
RENT option 307
REUS option 307
SIZE option 308
TERM option 307
usage 308
XCAL option 307
XREF option 307
load maps
creating 312
with INCLUDE command 279
with LOAD command 312
displaying 312
generated by GENMOD command 231
invalid card images in 312
of MODULE files, displaying 348
replace card images in 279
load point, specifying 279, 312
loader
CMS 314
control statements
ENTRY statement 317
include control section (ICS) statement 318
LIBRARY statement 317
loader terminate (LDT) statement 317
replace (REP) statement 320
set location counter (SLC) statement 319
Set Page Boundary (SPB) statement 321
search order for unresolved references 315
search order, for unresolved references 317
tables
defining storage for 577
displaying number of 441

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming
loader terminate (LDT) loader control statement 317
loading a virtual 3800 printer via SETPRT command 612
LOADLIB command
  COMPRESS option 323
  COPY option 323
description 323
  DISK option 324
  EXCLUDE SYSIN control statement 325
  LIST option 323
  MODIFY option 324
  PRINT option 324
  REPLACE option 324
  SELECT SYSIN control statement 325
  TERM option 324
option
  of CMS QUERY command 442
  of GLOBAL command 243
LOADLIBs, CMS
compressing with LOADLIB command 323
copying with LOADLIB command 323
creating with LKED command 306
executing a load module from 392
listing with LOADLIB command 323
LOADMOD command
  CMS/DOS considerations 327
description 327
LOCATE subcommand
description 808
effect of zone setting 833
location indicator
displaying in a window 581
querying 442
location of windows, changing 405
LOCATION option
  of CMS QUERY command 442
  of CMS SET command 581
log files
controlling updating 583
for messages 561
for warnings 561
querying 443
LOG option
  of NOTE command 372
  of RECEIVE command 482
  of SENDFILE command 517
LOGFILE option
  of CMS QUERY command 443
  of CMS SET command 583
logging of messages 561
logging of warnings 561
logical line end symbol 579
operators, in EXEC procedures 854
record length of CMS file, defaults used by CMS editor 141
units
  assigning 47
  ignoring assignments 48
  listing 303
  unassigning 546
  unassigning in CMS/DOS 49
LONG
  option of NOTE command 372
  subcommand, description 809
looping in EXEC procedure 856
LOWCASE option
  of ASSGN command 48
  of COPYFILE command 71
  of FILEDEF command 201
lowercase letters
  suppressing translation to uppercase 789
  translating to uppercase
    with CASE subcommand 789
    with COPYFILE command 71
    with PRINT command 408
LRECL option
  of COPYFILE command 70
  of EDIT command 141
  of FILEDEF command 198

M

M border command 719
M operand of CASE subcommand 789
machine code 161, 171
MACLIB command
  ADD option 329
  COMP option 329
  DEL option 329
description 329
  DISK option 330
  FIFO option 330
  GEN option 329
  LIFO option 331
  MAP option 330
  PRINT option 330
  reading files created by ESERV program 148
  REP option 329
  STACK option 330
  TERM option 330
  XEDIT option 331
files
  creating 329
  displaying names of MACLIBs to be searched 443
distributed with CMS system 331
  specifying for assembly or compilation 243
option
  of CMS QUERY command 443
  of GLOBAL command 243
MACLIST command
description 333
setting defaults 90
MACREF
See VM/SP CMS Macros and Functions Reference
MACRO
files created by ESERV program 147
filetype
adding to MACLIBs 329
default CMS editor settings 839
invalid records handled by MACLIB command 331
macro definitions
in assembler listing 40
in MACRO files 330
macro libraries
CMS
adding to 329
compacting members of 329
creating 329
deleting members of 329
display information about members in 330
printing members 410
punching members 414
reading OS macro libraries into 650
replacing members of 329
typing members 666
creating
from OS partitioned data sets on tape 650
from tapes created by IEHMOVE utility program 650
identifying for assembly 44, 243
OS
concatenating 199
reading into CMS MACLIBs 650
using in CMS 45
VSE, copying macros from 147
MAKEBUF command
description 344
return code effect on &ERROR statement 344
MAP
filetype
created by DOSLIB command 127
created by DSERV command 138
created by LOAD command 314
created by MACLIB command 330
created by TAPE command 644
created by TXTLIB command 662
option
determined by DOSLIB command 126
determined by GENMOD command 231
determined by INCLUDE command 279
determined by LIKED command 307
determined by LOAD command 312
determined by MACLIB command 330
determined by TXTLIB command 661
maps
created by DOSLIB command 126
created by GENMOD command 231

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming
minidisks
See also disks
counting cylinders on 225
MINIMIZE WINDOW command
description 347
format 347
messages 347
operands
= (equal sign) 347
wname 347
summary 16
usage notes 347
MINPROMPT option
of DISK command 105
of READCARD command 475
of RECEIVE command 483
MLOGIC option of ASSEMBLE command 40
MODE
operand
MODESET option of TAPE command 642
MODIFY option
of LOADLIB command 324
of SETPRT command 613
MODMAP command described 348
MODULE file
creating 231
debugging 327
defining synonyms for 635
executing with RUN command 508
format 231
generating 231
loading into storage for execution 327
mapping 348
VSE, link-editing 129
modules, VSE, link-editing 129
MOREHELP command
description 349
format 349
messages 351
operands
ALL 350
BRIEF 349
DESCRIPT 350
DETAIL 349
ERRORS 350
FORMAT 350
NOTES 350
OPTIONS 350
PARMS 350
RELATED 349
summary 16
usage notes 350
MOVEFILE command
default device attributes 354
description 352
examples 352
PDS option 352
moving a window up in the order of displayed windows 402
moving windows on a virtual screen 511
MSG operand of &CONTROL statement 849
MULT option of DLBL command 112
multi-level update with UPDATE command 676
multiple extents
input files
for UPDATE command 669
with COPYFILE command 74
output files
with COPYFILE command 74
with RENAME command 496
specifying 121
specifying in CMS/DOS 116
multivolume data sets, displaying volumes on which they reside 117
multivolume VSAM extents
identifying with DLBL command 122
in CMS/DOS 117
maximum number of disks 123
in CMS/DOS 118
rules for specifying 122
in CMS/DOS 117

N
N border command 720
NAME option of LKED command 307
NAMEFIND command
description 356
FIFO option 357
FILE option 357
LIFO option 357
LINENUM option 357
NAMES file format 358
NAMES file tags 359
SIZE option 357
simple names file 363
STACK option 357
START option 357
TYPE option 357
XEDIT option 368
NAMES command
description 364
nickname 364
PF keys on NAMES menu 367
sample NAMES screen 369
naming a virtual screen 95
naming CMS files 4
National Language Support
See languages, national
NCAL option of LKED command 306
NE option of LKED command 307
nesting
&IF statements in EXEC procedure 855
EXEC procedures
effect on &CONTROL 849
passing variable data 869
testing recursion level 869
loops in EXEC procedure 857
never-call function, specifying in CMS TEXT
file 317
NEWDATe option
    of COPYFILE command 69
    of RECEIVE command 483
NEWFILE option of COPYFILE command 69
NEXT subcommand described 809
nickname assigned in NAMES file 364
NL operand of FILEDEF command 207
nnnn subcommand described 834
NO option of START command 624
NOACK option
    of NOTE command 371
    of SENDFILE command 516
NOALIGN option of ASSEMBLE command 42
NOMLOGIC option of ASSEMBLE command 39
NOAUTO option
    of INCLUDE command 280
    of LOAD command 313
NOCC option of PRINT command 408
NOCHANGE option
    of DLBL command 112
    of FILEDEF command 198
    of LABELDEF command 285
NOCLEAR option
    of INCLUDE command 279
    of LOAD command 311
    of XEDIT command 698
NOCOL1 option of TAPPDS command 666
NOCOMMENTS option of EXECUPDT command 187
NOCOMPRESS option of EXECUPDT command 187
NOCTL option
    of UPDATE command 669
    of XEDIT command 699
NODECK option of ASSEMBLE command 41
    of OPTION command 389
NODISK option of ACCESS command 28
NODISP option
    of EDIT command
        effect on FORMAT command 800
NODUMP option of OPTION command 389
NODUP option
    of INCLUDE command 280
    of LOAD command 313
NOEND option of TAPPDS command 656
NOEOV option of FILEDEF command 201
NOERASE option of FORMAT command 226
NOERROR option of OPTION command 390
NOESD option of ASSEMBLE command 39
NOFILeLIST option of FILELIST command 213
NOHEADER option
    of LISTFILE command 296
    of PUNCH command 414

NOHISTORY option of EXECUPDT command 187
NOINC option of UPDATE command 669
NOINV option
    of INCLUDE command 279
    of LOAD command 312
NOLIBE option
    of INCLUDE command 280
    of LOAD command 313
NOLIBMAC option of ASSEMBLE command 40
NOLIST option
    of ASSEMBLE command 39
    of OPTION command 389
NOLISTX option of OPTION command 390
NOLOG option
    of NOTE command 372
    of RECEIVE command 482
    of SENDFILE command 517
NOMAP option
    of GENMOD command 232
    of LOAD command 312
NOMAXTEN option of TAPPDS command 657
NOMCALL option of ASSEMBLE command 39
NOMLOGIC option of ASSEMBLE command 40
NOMSG operand, of &CONTROL statement 849
    option of XEDIT command 699
NONDISP option
    of CMS QUERY command 444
    of CMS SET command 585
    non-displayable character
        defining character used in place of 585
        displaying character used in place of 444
    non-relocatable modules in CMS 231
NONSHELl option of CMS SET command 586
nonshared copy
    of named system, obtaining 586
    of saved system, obtaining during debug 757
NONUM option of ASSEMBLE command 42
NOOBJECT option of ASSEMBLE command 41
NOPACK operand of &CONTROL statement 849
NOPDS option of TAPPDS command 656
NOPRINT option
    of ASSEMBLE command 40
    of LKED command 308
    of TAPE command 644
NOPROF option of ACCESS command 28
NOOPOFIL option of XEDIT command 698
NOMPROMPT option
    of DISK command 105
    of REACRD command 475
        of RECEIVE command 483
NOMPROMPT option of COPYFILE command 69
NORENT option of ASSEMBLE command 43
NOREP option
    of INCLUDE command 280
    of LOAD command 313
    of UPDATE command 669
NOREPLACE option

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming

Index 921
of DISK option 105
of READCARD command 475
of RECEIVE command 483
NORLD option of ASSEMBLE command 40
NOSAVE option of ACCESS command 28
NOSCREEN option of XEDIT command 698
NOSEQ8 option
of UPDATE command 669
of XEDIT command 699
NOSID option of EXECUPDT command 188
NOSPECS option of COPYFILE command 70
NOSTMT option of ASSEMBLE command 42
NOSTOR option of UPDATE command 670
NOSTR option of GENMOD command 232
NOSYM option of OPTION command 390
notational conventions 5
NOTE command
ACK option 371
ADD option 371
CANCEL option 371
CC option 371
description 370
LOG option 372
LONG option 372
NOACK option 371
NOLOG option 372
NONOTEBOOK option 372
NOTEBLOCK option 372
PF key settings 375
PROFILE option 373
REPLACE option 373
sending a note 374
SHORT option 373
NOTE option of SENDFILE command 517
NOTEBOOK option
of NOTE command 372
of RECEIVE command 482
NOTERM option
of ASSEMBLE command 42
of LIKED command 307
of OPTION command 390
of UPDATE command 670
NOTEST option of ASSEMBLE command 41
NOTIME operand of &CONTROL statement 849
NOTRC option of PRINT command 408
NOTRUNC option COPYFILE command 71
NOTYPE option
of COPYFILE command 69
of ERASE command 144
of INCLUDE command 279
of LOAD command 312
of RDR command 459
of RENAME command 495
of SENDFILE command 517
NOUPDATE option
of EXECUPDT command 188
of XEDIT command 699
NOUPDIRT option of RENAME command 495
NOWTM option of TAPE command 643
NOXREF option
of ASSEMBLE command 40
of OPTION command 390
NOYFLAG option of ASSEMBLE command 43
NSL operand
of FILEDEF command 297
of TAPEMAC command 660
of TAPPDS command 655
NUCEXT Function
See MACREF
nucleus
CMS protected storage 589
protection feature
displaying status of 444
setting 589
nucleus extensions
cancel an extension 380
obtain information about 387
NUCXDROP command
description 380
NUCLOAD command
description 382
ENDCMD operand 383
IMM CMD operand 383
PUSH operand 383
SERVICE operand 383
SYSTEM operand 383
NUCMAP command
description 387
FIFO option 387
LIFO option 387
STACK option 387
null
arguments in EXEC procedure 867
block, dumping to tape 646
line
stacking in console stack 824
stacking in EXEC 861
to return to edit mode from input mode 785
when entering VSAM extents in CMS/DOS 116
symbols in EXEC statement 856
null characters, recognizing in full-screen CMS 552
NUM result of &DATATYPE built-in function 864
number of characters in token in EXEC procedure 865
NUMBER option of ASSEMBLE command 42
numeric
determining if token contains data 864
substitution with XMITMSG command 704
variables in EXEC procedure 867
O border command 721
object deck, assembler, generating 41
OBJECT option of ASSEMBLE command 41
obtaining additional or related help 349
OFF operand
of &CONTROL statement 848
of &HEX control statement 854
of &TIME control statement 861
of AUTOSAVE subcommand 787
of IMAGE subcommand 803
of LINEMODE subcommand 806
of SERIAL subcommand 822
OFFSET (.OF) format word 749
OL option of LKED command 307
OLD option of SENDIFLE command 521
OLDDATE option
of COPYFILE command 69
of DISK LOAD command 105
of RECEIVE command 483
ON operand
of &HEX control statement 854
of &TIME control statement 861
of IMAGE subcommand 803
of SERIAL subcommand 822
operands, command 3
Operating System (OS)
data sets
defining in CMS 193
listing information 289
disks, accessing 30
environment, resetting 182
linkage editor control cards, adding to TEXT files 663
macro libraries
reading into CMS MACLIBs 650
used in assembly 45
option of GENMOD command 232
partitioned data sets
See partitioned data sets (PDS)
tapes
containing partitioned data sets 656
standard-label processing 657
utility programs
creating CMS files from tapes created by 654
IEBPPTPCH 654
IEBUPTDE 654
IEHMOVE 654
operators, comparison, in EXEC procedure 854
OPTCD option of FILEDEF command 199
OPTION command
DECK option 389
description 389
DUMP option 389
ERRS option 390
LIST option 389
LISTX option 390
NODECK option 389
NODUMP option 389
NOERRS option 390
NOLIST option 389
NOLISTX option 390
NOSYM option 390
NOTERM option 390
NOXREF option 390
SYM option 390
TERM option 390
XREF option 390
48C option 390
60C option 390
option, of CMS QUERY command 444
options
command 3
for DOS/VS COBOL compiler in CMS/DOS, querying 444
for DOS/VS COBOL compiler, specifying 389
LOAD and INCLUDE commands, retaining 280
ORIGIN option
of FETCH command 190
of INCLUDE command 279
of LOAD command 312
subcommand described 770
origin for debug environment
setting 770
used to compute symbol location 763
OS (Operating System)
data sets
defining in CMS 193
listing information 289
disks, accessing 30
environment, resetting 182
linkage editor control cards, adding to TEXT files 663
macro libraries
reading into CMS MACLIBs 650
used in assembly 45
option of GENMOD command 232
partitioned data sets
See partitioned data sets (PDS)
tapes
containing partitioned data sets 656
standard-label processing 657
utility programs
creating CMS files from tapes created by 654
IEBPPTPCH 654
IEBUPTDE 654
IEHMOVE 654
OSRUN
description 392

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming
PARM keyword 392
OUTMODE option of UPDATE command 669
OUTMOVE default for MOVEFILE command
dname 352
OUTPUT option
  of CMS SET command 587
  of CMS QUERY command 444
OVLY option
description 810
effect of image setting 803
  of COPYFILE command 70
  of LKED command 307
PACK operand of &CONTROL statement 849
option of COPYFILE command 71
parameter list
  passed by RUN command 509
  passed by START command 625
  passed to SVC instruction, recorded 630
parent disk of read-only extension, recorded 27
parentheses
  before option list 3
  scanned by EXEC interpreter 843
PARM
  keyword of OSRUN command 392
PARSECMD command
description 393
format 393
messages 395
operands
  APPLID 393
  NOTYPE 393
  STRING 394
  TYPE 393
  uniqueid 393
summary 16
usage notes 394

PARSECMD command
description 393
format 393
messages 395
operands
  APPLID 393
  NOTYPE 393
  STRING 394
  TYPE 393
  uniqueid 393
summary 16
usage notes 394

paring facility
  See also DLCS (Definition Language for
  Command Syntax)
calling from an EXEC 393
variables returned to EXEC 394
partitioned data sets (PDS)
copying files into CMS files 352
copying into partitioned data sets 353
displaying member names 290
listing members of 289
  on tapes, creating CMS files 656
PA1 key in full-screen CMS 561
PA2 key in full-screen CMS 403, 561
PD option of DSERV command 137
PDS (partitioned data sets
copying files into CMS files 352
copying into partitioned data sets 353
displaying member names 290
  listing members of 289
  on tapes, creating CMS files 656
PDS option
  of LISTDS command 290
  of MOVEFILE command 352
  of TAPPDPS command 656
PEEK command
description 396
DISCARD command used with PEEK 400
PF key settings 397
PROFILE option 396
period
  as concatenation character for EXEC
  variables 856
PERM option
  of DLBL command 112
  of FILEDEF command 198
  of LABELDEF command 285
permanent file definitions 198
PF key default settings
  changing for full-screen CMS 541
  changing for the WM window 609
displaying settings for full-screen CMS 428
displaying settings for the WM window 455
  for full-screen CMS 542
  for the WM window 610
  on NAMES menu 367
  on NOTE menu 375
  on PEEK screen 397
  on RDRLIST screen 466
  on SENDFILE menu 519
PF keys
  changing settings for full-screen CMS 541
  changing settings for WM window 609
displaying definitions for full-screen CMS 428
displaying definitions for WM window 455
phase library
  clearing to zeros 132
  CMD/DOS 126
deleting phases from 126
phases
  executing in CMS/DOS 190
  in VSE core image libraries, obtaining
  information about 137
PLI filetype default CMS editor settings 839
PLIOPT filetype default CMS editor settings 839
POP WINDOW command
description 402
format 402
messages 404
operands
  * (asterisk) 402
  n 402
  WM 402
  wname 402
summary 16
usage notes  402
popping a variable size window  402
popping a window  402
popping the WM window  402
POSITION WINDOW command
description  405
format  405
messages  406
operands
  pscol  405
  pslin e  405
  wname  405
summary  16
usage notes  405
positioning the cursor in a virtual screen  84
positioning windows  405
preferred auxiliary files  678
prefixes
  identifying sets of files
    with ACCESS command  29
    with LISTFILE command  299
prefixing error messages issued in EXEC with
DMS  845
PRESERVE subcommand  811
PRINT command
  CC option  408
  description  407
  HEX option  409
  LINECOUN option  408
  MEMBER option  409
  NOCC option  408
  NOTRC option  408
  OVERSIZE option  407
  TRC option  408
  UPCASE option  408
option
  of AMSERV command  34
  of ASSEMBLE command  40
  of DOSLIB command  127
  of DOSLKED command  130
  of DSERV command  138
  of EXECIO command  161
  of LKED command  308
  of LOADLIB command  324
  of MACLIB command  330
  of PREAD command  412
  of RREAD command  506
  of SREAD command  622
  of TAPE command  644
  of TXTLIB command  662
  of UPDATE command  670
PRINTER option
  of ASSGN command  48
  of FILEDEF command  195
PRINTL Macro
  See MACREF
private libraries
  See MACREF
See libraries, VSE
PROC files creating in CMS/DOS  412
procedures, VSE, copying into CMS files  412
processor time displayed in EXEC procedure  861
PROFILE EXEC, suppressing execution of  28
PROFILE option
  of NOTE command  373
  of PEEK command  396
  of RDrLIS T command  463
  of XEDIT command  698
PROG option of LISTIO command  303
program function (PF) key
  See PF key default settings
program stack
  buffer
    creating  344
    eliminating  134
    using WAITRD function to read lines from  344
    determining number of lines in  526
program status word (PSW)
  changing in debug environment  772
  displaying in debug environment  772
programmer logical units
  for job catalogs  114
  listing assignments for in CMS/DOS  303
  restrictions for R and T  50
  valid assignments in CMS/DOS  49
programs
  compilation and execution, with RUN command  508
  entry point
    selection during CMS loader processing  314
    specifying  311
  execution
    halting  731, 756
    in CMS subset  793
    in CMS/DOS  190
    modifying control words  774
    modifying general registers  774
    modifying storage  776
    resuming after breakpoint  766
    with INCLUDE command  278
    with LOAD command  311
    with START command  624
  loading into storage
    while using CMS editor  793
    with INCLUDE command  278
  stack buffer, clearing  103
PROMPT option
  of COPYFILE command  69
  subcommand, description  812
prompting
  increment for line number editing  805
  setting increment  812
PROTECT option
  of CMS QUERY command  444
  of CMS SET command  589

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming

Index 925
PSERV command
  description 412
  DISK option 412
  PRINT option 412
  PUNCH option 412
  TERM option 412
PSW (program status word)
  changing in debug environment 772
  displaying in debug environment 772
  operand of SET subcommand 774
  subcommand described 772
PUNCH
  assembler punch output ddname 44
  command
    description 414
    HEADER card format 415
    HEADER option 414
    MEMBER option 414
    NOHEADER option 414
  option
    of ASSIGN command 48
    of EXECIO command 161
    of FILEDEF command 185
    of PSERV command 412
    of RSERV command 506
    of SSERV command 622
PUNCHC Macro
  See MACREF
  punched files, restoring to disk 104
PURGE option
  of GLOBALV command 251
  of RECEIVE command 482
PUSH option of NUCXLOAD command 383
PUT SCREEN command
  description 418
  format 418
  messages 419
  option
    fn ft fm 418
    summary 17
  usage notes 418
PUT sub-function of GLOBALV command 250
PUT VSCREEN command
  description 420
  format 420
  messages 421
  option
    fn ft fm 420
    fromlin 420
    numlin 420
    vname 420
    summary 17
  usage notes 421

QUERY command (CMS)
  ABBREV option 426
  APL option 426
  AUTOREAD option 426, 428
  BLIP option 426
  BORDER option 427
  CHARMODE option 427
  CMSLEVEL option 428
  CMSPF option 428
  CMSTYPE option 429
  CURSOR option 429
  description 422
  DISK option 430
  DISPLAY option 432
  DLBL option 432
  DOS option 434
  DOSLIB option 434
  DOSLNCNT option 434
  DOSPART option 435
  EXECTRAC option 435
  FIFO option 456
  FILEDEF option 435, 449
  FULLREAD option 435
  FULLSCREEN option 436
  HIDE option 436
  IMESCAPE option 437
  IMPCP option 437
  IMPEX option 437
  INPUT option 438
  INSTSEG option 438
  KEY option 438
  LABELDEF option 439
  LANGLIST option 439
  LANGUAGE option 440
  LDRRTBS option 441
  LIBRARY option 441
  LIFO option 456
  LINEND option 442
  LOADLIB option 442
  LOCATION option 442
  LOGFILE option 443
  MACLIB option 443
  NONDISP option 444
  OPTION option 444
  OUTPUT option 444
  PROTECT option 444
  RDYMSG option 445
  REDTYPE option 445
  RELPAGE option 445
  REMOTE option 446
  RESERVED option 446
  ROUTE option 446
  SEARCH option 447
  SHOW option 448
  STACK option 456
  SYNONYM option 449
SYSNAMES option 449
TEXT option 450
TRANSLATE option 450
TXTLIB option 452
UPSI option 452
VSCREEN option 452
WINDOW option 454
WMPF option 455
QUERY operand of IMMCMD command 276
querying remote data transmission 446
querying status of virtual machine environment 422
QUIT subcommand described 812

R border command 723
RD option of DSERV command 137
RDCARD Macro
See MACREF
RDR command
description 459
FIFO option 459
LIFO option 460
NOTYPE option 459
STACK option 459
use of = 459
RDRLIST command
APPEND option 463
default PF key settings 466
description 463
DISCARD command, used with RDRLIST 468
displaying a file 467
EXECUTE command, used with RDRLIST 468
issuing commands from RDRLIST 466
PROFILE option 463
special symbols 470
synonyms that sort the list 467
RDTAPE Macro
See MACREF
RTERM Macro
See MACREF
RDMYSEQ option
of CMS QUERY command 445
of CMS SET command 590
READ control card
deleting 476
format of 477
read-only
disks, editing files on 787
extensions
editing files on 141
releasing 493
read/write
status of disks
controlling 27

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming
F operand 813
V operand 813
RECOMP option of FORMAT command 226
record format
    of CMS file
        changing 70, 76, 813
        listing 298
        of file, specifying 198
        records that can be punched 415
record length
    default used by CMS editor 141
    modifying 141
    of CMS file
        changing 70
        listing 298
        maximum lengths for PRINT command 409
        specifying truncation setting for input 827
        specifying with FILEDEF command 198
records
    displaying selected positions of 665
    in file, numbering with UPDATE command 668
red type
    for error messages 591
RETYPE option
    of CMS QUERY command 445
    of CMS SET command 591
reducing a window size to one line 347
references
    unresolved
        resolving with INCLUDE command 280
    resolving with LOAD command 315
REFR option of LKED command 307
REFRESH command
    description 492
    format 492
    messages 492
    summary 17
refreshing a screen from an EXEC 684
refreshing a screen in full-screen CMS 492
REGEOQU Macro
    See MACREF
RELATED HELP
    described 261
    obtaining using HELP command 258
    obtaining using MOREHELP command 349
RELEASE command
    description 493
    DET operand 493
relocatable
    libraries (VSE), displaying directories of 137
    modules, link-editing in CMS/DOS 129
relocation dictionary assembler 40
RELPAG option
    of CMS QUERY command 445
    of CMS SET command 592
remote data transmission, handling 594
remote data transmission, querying 446
REMOTE option
    of CMS QUERY command 446
    of CMS SET command 594
removing a virtual screen definition 101
removing a window definition 102
removing EXECs and editor macros from storage 155
RENAME command
    description 495
    NOTYPE option 495
    NOUPDIRT option 495
    TYPE option 495
    UPDIRT option 495
RENT
    option
        of ASSEMBLE command 43
        of LKED command 307
RENUM subcommand described 814
REP option
    of INCLUDE command 280
    of LOAD command 312
    of MACLIB command 329
    of UPDATE command 669
REPEAT subcommand
    description 815
    used with OVERLAY subcommand 816
REPLACE control statement, for UPDATE command 673
    of READCARD command 475
    option
        of COPYFILE command 69
        of DISK command 105
        of LOADLIB command 324
        of NOTE command 373
        of RECEIVE command 483
    subcommand
        description 816
        effect of image setting 803
        restriction while using line-number editing 807
        stacking with &STACK control statement 860
replace (REP)
    image of statement in load map 280
    loader control statement 320
RESERVE command
    description 498
    format of RESERVEd file 498
    of CMS QUERY command 446
    use with DISKID function 498
reserved lines
    defined 596
    displaying number in a window 446
    specifying number in a window 596
RESERVED option
    of CMS SET command 596
RESET
    operand
        operand of &TIME control statement 862
    option
        of INCLUDE command 279
        of LOAD command 312
resetting
OS environment 182
VSAM environment 182
responses, CMS editor, controlling format of 809
RESTORE
subcommand, description 817
RESTORE WINDOW command
description 502
format 502
messages 502
operands
- (equal sign) 502
wname 502
summary 17
restoring a window 502
restrictions
access method services and VSAM
OS/VS users 781
VSE users 781
resuming full-screen CMS 554
retrieving a message from a repository 704
RETURN
subcommand (DEBUG) 773
subcommand (EDIT) 818
return code
from MAKEBUF command effect on &ERROR
statement 344
from SENTRYs effect on EXEC procedure 526
return codes
CMS in EXEC procedure 151
displaying during EXEC processing 848
from access method services 36
from CMS commands, testing in EXEC
procedure 870
from CMS EXEC interpreter 151
from EXEC 2 interpreter 152
from System Product interpreter 153
specifying in EXEC procedure 851
returning to full-screen CMS after suspending 554
REUS option of LKED command 307
REUSE subcommand
description 818
examples 818
REW tape control function 643
REWIND option
of TAPE command 645
RIGHT operand of LINEMODE subcommand 806
RLD option of ASSEMBLE command 40
RLDSAVE option
of INCLUDE command 280
of LOAD command 313
RO immediate command 732
ROUTE command
description 503
format 503
messages 505
of CMS QUERY command 446
operators
* (asterisk) 503
ALARM 503
CMS 503
CP 503
MESSAGE 503
msgclass 503
NETWORK 503
NOALARM 504
NONOTIFY 504
NOTIFY 504
SCIF 503
vname 503
WARNING 503
summary 17
usage notes 504
routing messages 503
RSERV command
description 506
DISK option 506
PRINT option 506
PUNCH option 506
TERM option 506
RT immediate command 733
RUN
description 508
tape control function 643
running applications in full-screen CMS 561

S

S border command 724
S-disk accessed after IPLing CMS 28
SAME option of INCLUDE command 280
SAVE subcommand described 820
saved systems
names
querying 449
setting 598
sharing 598
SAVEONLY option of ACCESS command 28
SCAN option of TAPE command 642
scanning
&ERROR control statement 851
in EXEC procedure 843
screen display, refreshing from an EXEC 684
screen images, copying to a CMS file 418
screen, copying the image to a CMS file 418
screen, displaying characteristics of 432
SCRIPT filetype default CMS editor settings 839
SCROLL command
description 511
format 511
messages 514
operands
BACKWARD 512

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming

Index 929
BOTTOM 512
DOWN 512
FORWARD 512
LEFT 512
NEXT 512
RIGHT 512
TOP 512
UP 512
wname 511
summary 17
usage notes 513
SCROLL subcommand described 821
scrolling a window 403, 511
scrolling process in full-screen CMS 555
SCROLLUP subcommand described 821
SD option of DSERV command 137
SEARCH option of CMS QUERY command 447
search order
for CMS commands 7
for CMS loader 315
for executable phases in CMS/DOS 190
for relocatable modules in CMS/DOS 131
of CMS disks, querying 447
SEC operand of LABELDEF command 285
SELECT option of GLOBALV command 248
SELECT SYSIN control statement 325
SENDFILE command
ACK option 516
default PF key settings on SENDFILE menu 519
description 515
example 526
file format 521
FILELIST option 516
LOG option 517
NEW option 517
NOACK option 516
NOFILELIST option 516
NOLOG option 517
NOTE option 517
NOTYPE option 517
OLD option 517
TYPE option 517
sending
messages 659
notes 374
SENTRIES command
description 526
effect of non-zero return code on EXECS 526
SEQUENCE control statement for UPDATE command 671
sequence numbers
assigned (CMS/DOS) to VSAM extents 116
assigned to VSAM extents 122
SEQ8 option
of UPDATE command 669
of XEDIT command 699
SERIAL subcommand
ALL operand 822
description 822
OFF operand 822
ON operand 822
SERVICE operand of NUCXLOAD command 383
SESSION file of GLOBALV command 246
SET command
description 527
determining status of SET operands 422
list of options 527
messages 527
operands
ABBREV 528
APL 530
AUTOREAD 532
BLIP 534
BORDER 536
CHARMODE 539
CMSPF 541
CMSTYPE 544
DOS 546
DOSLNCTS 548
DOSPART 549
EXECTRAC 551
FULLREAD 552
FULLSCREEN 554
IMESCAPE 565
IMPCP 567
IMPEX 568
INPUT 569
INSTSEG 571
LANGUAGE 573
LDRTBLS 577
LINEND 579
LOCATION 581
LOGFILE 583
NODISP 585
OUTPUT 587
PROTECT 589
RDYMSG 590
REDCYPE 591
RELPAGE 592
REMOTE 594
RESERVED 596
TEXT 599
TRANSLATE 601
UPSI 604
VSCREEN 605
WINDOW 607
WMPF 609
SYSNAME option 598
usage notes 527
set location counter (SLC) loader control statement 319
SET operand
of DEFAULTS command 89
of IMMCMD command 276
Set Page Boundary (SPB) loader control statement 321
SET subcommand (DEBUG)
CAW operand 774

930 VM/SP CMS Command Reference
CSW operand 774
description 774
GPR operand 774
PSW operand 774

SETPRT command
CHARS option 612
COPIES option 612
COPYnr option 612
description 612
FCB option 612
FLASH option 612
INIT option 613
MODIFY option 613
using 613

setting another language 573
setting partition size for CMS/DOS 549

SHARED
option of EXECDROP command 156
of EXECMAP command 180

SHORT
option of NOTE command 373
subcommand described 823

SHOW option
of CMS QUERY command 448

SHOW WINDOW command
description 615
format 615
messages 616
operands
col 615
line 615
vname 615
wname 615
summary 18

usage notes 615

SID option of EXECUPDT command 187

SIDCODE option of XEDIT command 700

SINGLE option of COPYFILE command 72

SIZE option of NAMEFIND command 387

SIZE WINDOW command
description 617
format 617
messages 618
operands
cols 617
lines 617
wname 617

summary 18
usage notes 617

SKIP option
of TAPE command 642

SL operand
of FILEDEF command 206
of TAPEMAC command 650
of TAPPDS command 655

SLC (set location counter) loader control statement 319

SO immediate command 734

SORT
command
description 619
storage requirements 619
option of DSERV command 138
sort fields defined 619
sounding the alarm 33
source file, numbering records with UPDATE command 669
source files assembling
identifying macro libraries 243
for assembler 38
updating with EXECUPDT command 186
updating with UPDATE command 668
source statement libraries, VSE, displaying directories 137
source symbol table, assembler, generating 40
SPACE LINES (.SP) format word 751
space, determine free extents for VSAM 289
special variables
See EXEC, special variables
specification list for COPYFILE command format 78

SPECS option of COPYFILE command 70

SPOOL command
used with DISK DUMP command 106
used with PRINT command 410

SSERV command
description 622
DISK option 622
PRINT option 622
PUNCH option 622
TERM option 622

STACK
option
of CONVERT COMMANDS command 64
of EXECMAP command 180
of GLOBALV command 249
of IDENTIFY command 273
of NAMEFIND command 357
of NUXMAP command 387
of RDR command 459
of RECEIVE command 483
subcommand, description 824
value of &READFLAG special variable 870

stacking
EDIT subcommands 824
in EXEC procedure, testing whether stack contains lines 870
lines in console stack
&BEGSTACK control statement 846
&STACK control statement 860

STACKR option of GLOBALV command 250

START
command
description 624

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming

Index 931
NO option 624
passing arguments 624
option of FETCH command 190
of INCLUDE command 280
of LOAD command 313
of NAMEFIND command 357
starting point for execution of module, setting 311
STAT option of LISTIO command 304
STATE command described 627
STATEW command described 627
status of virtual machine environment, querying 422
STAT operand of LISTIO command 304
STATE command described 627
STATEW command described 627
status of virtual machine environment, querying 422
SVCTRACE command
description 630
output 631
SYM option of OPTION command 390
symbol
debug
defining 762
modifying 776
used to set breakpoints 756
in EXEC procedure
effect of undefined symbols in &IF statement 854
reading from terminal or console stack 858
substituted in EXEC procedure, displaying 848
variable
See variable symbols
symbol table, debug 762
symbolic names assigned to storage locations in debug environment 762
SYNONYM command
clearing 635
description 635
example 637
NOSTD option 635
relationship to SET ABBREV command 636
STD option 635
option, of CMS QUERY command 449
syonym table
clearing 635
defining 636
displaying system synonym tables 450
displaying user synonym tables 450
format for entries in 636
invoking 635
setting system translation synonyms 601
setting user translation synonyms 601
synonyms
for CMS user-written commands
defining 635
displaying 638
example 637
SYS option of LISTIO command 303
SYSCAT assigned in CMS/DOS 118
SYSPARM option of ASSEMBLE command 43
SYSPARM option of ASSEMBLE command 43
SYSPARM assigned in CMS/DOS 49
SYSS
assembler input 44
logical unit assignment in CMS/DOS 47
SYSIPT assigned for ESERV program 147
SYSLST lines per page
displaying number of 434
setting number of 548
SYSTYPE option of CMS SET command 598
SYSSNAMES option of CMS QUERY command 449
SYSPARM option of ASSEMBLE command 43
SYSPARM assigned in CMS/DOS 49
SYST}
operand of NUCXLOAD command 383
option
  of CONVERT COMMANDS command 63
  of EXECDROP command 156
  of EXECLOAD command 177
  of EXECMAP command 179
  of GENMOD command 233
system and programmer logical units entered on
DLBL command 114
system disk
  files available 28
  releasing 493
system logical units
  invalid assignments in CMS/DOS 49
  listing assignments for in CMS/DOS 303
  valid assignments in CMS/DOS 47
System Product interpreter
  error codes 153
  tracing programs interpreted by 551
system residence volume, VSE; specifying 546
SYSTERM option of ASSEMBLE command 42
SYSxxx option
  of ASSGN command 47
  of DLBL command 112
  of LISTIO command 303

T

Table Character Reference byte 408
TABSET subcommand
  affected by IMAGE subcommand 803
  description 825
TAPE command
  BLKSIZE option 643
  control functions
    BSF 643
    BSR 643
    ERG 643
    FSF 643
    FSR 643
    REW 643
    RUN 643
    WTM 643
  DEN option 645
  description 641
  DISK option 644
  DUMP option 642
  dumping null block 646
  DVOL1 option 643
  EOF option 644
  EOT option 644
  LEAVE option 645
  LOAD option 642
  MODESET option 642
  NOPRINT option 644
  NOWTM option 643
  PRINT option 644
  REWIND option 645
  SCAN option 642
  SKIP option 642
  TAPn option 644
  TERM option 644
  TRANSFER BUFFERED option 645
  TRANSFER IMMEDIATE option 645
  TRTCH option 645
  WTM option 643
  WVOL1 option 643
  18TRACK option 645
  7TRACK option 644
  9TRACK option 645
TAPECTL Macro
  See MACREF
TAPEMAC command
  description 650
  ID operand 651
  ITEMCT option 651
  NSL operand 650
  SL operand 650
  TAPn option 651
tapes
  assigning to logical units, in CMS/DOS 47
  backward spacing 643
  control functions
    restrictions when using 646
  creating CMS disk files 654
density of, specifying 645
dumping filenames on 642
dumping and loading files 642
  erasing a defective section 643
files
  created by OS utility programs 654
  created by tape command 641
  writing to disk 641
forward spacing 643
labels
  displaying definitions in effect 439
  displaying VOL1 label 643
  in FILEDEF command 195
  in TAPEMAC command processing 652
  in TAPPDS command processing 655
  specifying descriptive information 283
  writing VOL1 label 643
marks
  writing 643
OS, standard label processing 654
positioning
  after VOL1 label is processed 643
  at specified file 642
  recording technique, specifying 645
  rewinding 643
used for AMSERV input and output 34
  entering ddnames 35
  in CMS/DOS 36
TAPESL Macro
  See MACREF

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming
TAPIN option of AMSERV command 34
TAPn option
  of ASSGN command 48
  of FILEDEF command
description 195
  usage 206
  of TAPE command 644
  of TAPEMAC command 651
  of TAPPDS command 656
TAPOUT option of AMSERV command 35
TAPPDS command
  COL1 option 656
description 654
  END option 656
  ID operand 655
  MAXTEN option 657
  NOCOL1 option 656
  NOEND option 656
  NOMAXTEN option 657
  NOPDS option 656
  NSL operand 655
  PDS option 656
  processing OS standard-label tapes 657
  SL operand 655
  TAPn option 656
  UPDATE option 656
TASKS operand of HELP command 258
TD option of DSERV command 137
TE (Trace End) immediate command 735
TELL command
  change the CP command that TELL uses 660
description 659
  restrictions 660
  using nicknames 659
TERM option
  of DOSLIB command 127
  of DOSLKED command 130
  of DSERV command 138
  of LKED command 307
  of LOADLIB command 324
  of MACLIB command 330
  of OPTION command 390
  of PSERV command 412
  of RSERV command 506
  of SSERV command 622
  of TAPE command 644
  of TXTLIB command 662
  of UPDATE command 670
TERMINAL option
  of ASSEMBLE command 42
  of ASSGN command 48
  of FILEDEF command 195
terminals
  output
  determining if terminal is displaying 870
  halting 730
  halting in EXEC procedure 861
  restoring 733
  restoring in EXEC procedure 861
  reading data from
during EXEC processing 858
  terminate execution of System Product interpreter
  or EXEC 2 EXECs 728
TEST option of ASSEMBLE command 41
TEXT
  assembler output ddname 44
  files
    automatic loading 313
    cards read in by loader 317
    creating with assembler 38
    executing with RUN command 508
    link-editing in CMS/DOS 129
    linking in storage 311
    loading into virtual storage 311
    resolving unresolved references with LOAD
    command 315
  libraries
    See TXTLIB
    of CMS QUERY command 450
    of CMS SET command 599
TEXT character conversion
  activating in CMS 599
  displaying status in CMS 450
TEXT files
  loading into storage for execution 278
  setting starting point for execution 311
TEXT option
time information displayed during EXEC
  processing 861
TIME operand of &CONTROL statement 849
time-of-day
  value, displaying during EXEC processing 849
timers, virtual interval 534
TO
  keyword of SENDFILE command 515
  operand of $MOVE Edit macro 837
  option of GENMOD command 231
TODACCNT Function
  See MACREF
tokens
  comparing in EXEC procedure 843
description 843
TOLABEL option of COPYFILE command 70
TOP
  operand of &GOTO control statement 853
  subcommand, description 826
tracing
  EXEC 2 programs 551
  resuming after temporary halting 732
  start, for System Product interpreter or EXEC 2
    EXEC 736
  suspending for System Product interpreter or
    EXEC 2 EXEC 735
  suspending recording temporarily 734
  SVC instructions 630
  halting 730
  System Product interpreter programs 551
  trailing fill characters, removing from records 71
TRANS option of COPYFILE command 72
TRANSFER BUFFERED option of TAPE command 645
TRANSFER IMMEDIATE option of TAPE command 645
transient area
  CMS commands that execute in 9
  creating modules to execute in 233
  loading programs into 312
transient directories in VSE, displaying 137
TRANSLATE CHARACTER (.TR) format word 752
TRANSLATE option
  of CMS QUERY command 450
  of CMS SET command 601
translate tables
  defining input characters for translation 569
  defining output characters for translation 569
  displaying 438
  displaying system translate tables 450
  displaying user translate tables 450
  setting system translations 601
  setting user translations 601
translation list for COPYFILE command, description 72
TRC option of PRINT command 408
TRTCH
  of ASSGN command 48
  of FILEDEF command 200
  of TAPE command 645
TRUNC
  of commands 5
  option of COPYFILE command 70
  subcommand, description 827
truncation
  column, for input mode 827
  of command names
    querying acceptability of 426
  of input records with CMS editor, default settings 827
  of records in CMS file 70
    during GETFILE subcommand 801
    following CHANGE subcommand 790
  of tokens in EXEC procedure 843
  of trailing blanks from CMS file 70
TS (Trace Start) immediate command 736
two-color ribbon, controlling use of 591
TXTLIB command
  ADD option 661
  DEL option 661
  description 661
  DISK option 662
  FILENAME option 662
  GEN option 661
  MAP option 661
  PRINT option 662
  TERM option 662
  file, searching for unresolved references 278
adding members 661
creating members 661
deleting members 661
determining which TXTLIBs are searched 452
identifying for LOAD and INCLUDE command processing 243
listing members in 661
maximum number of members 663
search for unresolved references 315, 317
searched during INCLUDE command processing 278
searched during LOAD command processing 311
option
  of CMS QUERY command 452
  of GLOBAL command 243
TYPE command
  COL option 666
  description 665
  HEX option 666
  MEMBER option 666
  operand of &TIME control statement 861
option
  of COPYFILE command 69
  of ERASE command 144
  of IDENTIFY command 273
  of INCLUDE command 279
  of LOAD command 312
  of NAMEFIND command 357
  of RENAME command 495
  of SENDFILE command 517
  subcommand, description 828
U
U operand of CASE subcommand 789
UA option
  of ASSGN command 48
  of LISTIO command 303
underscore
char, on OVERLAY subcommand 810
data records, using backspaces 803
UNPACK option of COPYFILE command 71
unresolved references
  during MODULE execution 233
  loader handling of 315
  resolving with INCLUDE command 280
  searching for TEXT files 313
  searching TXTLIBs for 313
UNTIL option of XEDIT command 700
UP
  operand of $MOVE edit macro 837
  subcommand described 829

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming

Index 935
UPCASE option
of ASSGN command 48
of COPYFILE command 71
of FILEDEF command 201
of PRINT command 408

UPDATE
command
control statements 670
CTL option 669
description 668
DISK option 670
error handling for 680
INC option 669
input files 675
multi-level updates, example with auxiliary
control file 676
NOCTL option 669
NOINC option 669
NOREP option 669
NOSEQ8 option 669
NOSTK option 670
NOTERM option 670
OUTMODE option 669
output files 674
PRINT option 670
REP option 669
SEQ8 option 669
STK option 669
STOR option 670
TERM option 670
control statements
COMMENT 674
DELETE 673
INSERT 672
REPLACE 673
SEQUENCE 671
filetype default CMS editor settings 839
option
of TAPPDS command 656
of XEDIT command 699
update log
for UPDATE command operations 670
generating at your terminal 670
updating a virtual screen and associated
window 492
updating a virtual screen from an EXEC 684
UPDIRT option of RENAME command 495
UPDTxxxx filetype default CMS editor
settings 839
uppercase letters
converting to lowercase with COPYFILE
command 71
suppressing translation of lowercase letters 789
UPSI (user program switch indicator)
byte
querying setting of 452
setting 604
option
of CMS SET command 604
option of CMS QUERY command 452
user catalogs 123
identifying
in CMS/DOS 119
user file directory
contents 27
creating 27
updating on disk 493
USER option
of CONVERT COMMANDS command 63
of EXECDROP command 155
of EXECLOAD command 177
of EXECMAP command 179
user program area
commands that execute in 9
user program switch indicator (UPSI)
byte
querying setting of 452
setting 604
option
of CMS SET command 604
option of CMS QUERY command 452
user-defined synonyms, displaying 449
user-written commands
assigning synonyms for 635
creating 231
UWORD operand

VALIDATE command
description 682
examples 682
format 682
messages 683
operands
* (asterisk) 682
fn ft fm 682
responses 683
summary 18
usage notes 682
VAR option of EXECIO command 163
variable data
in EXEC procedure
displaying 863
punching 857
stacking 860
variable length
converting to fixed-length 70
using RECFM subcommand 813
variable size window, displaying 402
variable symbols
assigning values to in EXEC procedure 843
reading from terminal or console stack, in an
EXEC 858
substituting in EXEC procedure 858
testing, in EXEC procedure 854
VARS operand of &READ control statement 858
verification setting, for CMS editor, changing 830
VERIFY subcommand, description 830
verifying a VSAM catalog structure 51
verifying that a disk is accessed 682
verifying the syntax of file identifier 682
virtual disks
See also disks
counting cylinders on 225
initializing 225
resetting number of cylinders on 225
valid addresses for 225
virtual machine
components of 1
c console 1
definition 1
environment, determining status of 422
virtual screen
changing attributes 605
clearing 53
creating 92
data area 84
default connections to windows 560
default virtual screens 559
defining 92
defining fields 690
deleting 101
displaying information about 452
displaying location of cursor 430
entering information 690
field definition character 685
naming 95
numbering of reserved lines 85
positioning the cursor 84
querying 452
reserved area 84
specifying character attributes 690
suppressing updates 606
updating 492
updating from an EXEC 684
updating the associated plane buffers 690
updating with data 688
using character attributes when displaying data 539
writing data 690
writing data to a CMS file 420
writing lines from a file to a virtual screen 241
XEDIT virtual screen 698
Virtual Storage Access Method (VSAM)
catalogs
determining which catalog is searched 123
identifying 123
identifying in CMS/DOS 118
verifying a structure of 51
data set extents displayed 117
determining free space extents 289
environment, resetting 182
files
defining with DLBL command 110
specifying extents 112
master catalog
identifying 123
identifying in CMS/DOS 118
option
of DLBL command 112
of SET DOS ON command 546
restriction
for OS/VS users 779
for VSE users 779
OS/VS users 781
VM/SP (Virtual Machine/System Product) basic
description 1
VOLID operand
of FILEDEF command 207
of LABDEF command 284
VOLSEQ operand of LABELDEF command 284
VSAM (Virtual Storage Access Method)
catalogs
determining which catalog is searched 123
identifying 123
identifying in CMS/DOS 118
verifying a structure of 51
data set extents displayed 117
determining free space extents 289
environment, resetting 182
files
defining with DLBL command 110
specifying extents 112
master catalog
identifying 123
identifying in CMS/DOS 118
option
of DLBL command 112
of SET DOS ON command 546
restriction
for OS/VS users 779
for VSE users 779
OS/VS users 781
VSBASIC
files, renumbering 814
filetype default CMS editor settings 839
VSBDATA filetype default CMS editor settings 839
VSCREEN option
of CMS QUERY command 452
of CMS SET command 605
VSE/VSAM Catalog Check Service Aid,
invoking 51

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming
WAITD Macro
See MACREF
WAITECB Macro
See MACREF
WAITRD Function
See MACREF
WAITREAD VSSCREEN command
description 684
execution described 684
messages 686
operands
  vname 684
summary 18
usage notes 684
using with WRITE VSSCREEN 684
WAITT Macro
See MACREF
WAITT VSSCREEN command
description 688
format 688
messages 689
operands
  * (asterisk) 688
  vname 688
summary 18
usage notes 688
WIDTH option of XEDIT command 698
window
  changing attributes 607
  changing location 405
  changing number of lines and columns 617
  clearing 54
  connecting to a virtual screen 615
  creating 97
  default connections to virtual screens 560
  default windows 557
  defining 97
  defining borders 536
  deleting 102
  displaying a variable size window 615
  displaying information about 454
  displaying location indicator 581
  displaying number of reserved lines 446
  dropping 135
  dropping the WM window 135
  expanding the size to the physical screen size 345
  hidden windows, displaying information about 436
  hiding 271
  placing at top of display order 615
  popping 402
  positioning 405
  reducing a window size to one line 347
  restoring after maximizing or minimizing 502
  scrolling the topmost window 403
  specifying reserved lines 596
  XEDIT window 698
WINDOW option
  of CMS QUERY command 454
  of CMS SET command 607
  querying 454
WINDOW option of XEDIT command 698
WM window
  automatically displayed 403, 556
  CLEAR key 561
  commands you can enter 403
  displaying 402
  dropping 135
  exiting 135
  PA2 key 561
WMPF keys
  canceling 610
  changing settings 609
  displaying definitions 455
  RETRIEVE function 611
  using NOWRITE option 611
WMPF option
  of CMS QUERY command 455
  of CMS SET command 609
WRITE VSSCREEN command
description 690
examples 694
format 690
messages 696
operands
  BLANKS 691
  col 690
  color 691, 692
  DATA 692
  exthi 692
  FIELD 692
  HIGH 691
  INVISIBLE 691
  length 690
  line 690
  NOHIGH 691
  NOPROTECT 691
  NULLS 691
  PROTECT 691
  PSS 693
  psset 692
  RESERVED 691
  text 693
  vname 690
summary 18
usage notes 693
writing lines from a file to a virtual screen 241
writing virtual screen data to a CMS file 420
WRTAPE Macro
See MACREF
WRTERM Macro
See MACREF
WTM
option of TAPE command 643
tape control function 643
WVOL1 operand of TAPE command 643

X

DEBUG subcommand described 778
EDIT subcommand, description 831
X border command 725
XCAL option of LKED command 307
XEDIT command,
  CTL option 699
description 697
  editing a MACLIB member 699
  INCR option 700
  MEMBER option 699
  MERGE option 700
  NOCLEAR option 698
  NOCTL option 699
  NOMSG option 699
  NOPROFIL option 698
  NOSCREEN option 698
  NOSEQ8 option 699
  NOUPDATE option 699
  PROFILE option 698
  SEQ8 option 699
  SIDCODE option 700
  UNTIL option 700
  UPDATE option 699
  usage 700
  WIDTH option 698
  WINDOW option 698
XEDIT option
  of LISTFILE command 297
  of NAMEFIND command 358
XMITMSG command
description 704
examples 708
format 704
messages 710
operands
  * (asterisk) 705
  APPLID 705
  CALLER 705
  COMPRESS 706
  DISPLAY 706
  ERRMSG 706
  FORMAT 705
  HEADER 706
  LETTER 705
  LINE 705
  msgnumber 704
  nn 705
  NOCOMPRESS 706
  NODISPLAY 706
  NOHEADER 706
  sublist 704
  SYSLANG 707
  VAR 705
  summary 19
  usage notes 707
XREF option
  of ASSEMBLE command 40
  of LKED command 307
  of OPTION command 390
XTENT option of FILEDEF command 199

Y

Y subcommand (edit), description 831
Y-disk accessed after IPLing CMS 28
YFLAG option of ASSEMBLE command 43

Z

ZONE subcommand described 832
zone settings for edit sessions 832

Numerics

18-track tapes, specifying on TAPE command 645
18TRACK option of FILEDEF command 200
19E virtual disk address accessed as Y-disk 28
190 virtual disk address accessed as S-disk 28
191 virtual disk address accessed as A-disk 28
192 virtual disk address accessed as D-disk 28
195 virtual disk address formatted by CMS batch facility 57
3480 Magnetic Tape Subsystem used with TAPE command 645
3800 printer, loading a virtual, via SETPRT command 612
48C option of OPTION command 390
512-byte block formatted disk using with RESERVE command 498
60C option of OPTION command 390
7-track tapes, specifying on TAPE command 644
7TRACK option
  of ASSGN command 48
  of FILEDEF command 200
  of TAPE command 644
9-track tapes, specifying on TAPE command 645
9TRACK option
  of ASSGN command 48
  of FILEDEF command 200

These symbols are used in the index to refer to other VM and VM/SP books:
MACREF—VM/SP CMS Macros and Functions Reference
CPPROG—VM/SP CP for System Programming

Index 939
of TAPE command 645
Is there anything you especially like or dislike about this book? Feel free to comment on specific errors or omissions, accuracy, organization, or completeness of this book.

If you use this form to comment on the online HELP facility, please copy the top line of the HELP screen.

--------------------- Help Information line ___ of ___

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you, and all such information will be considered nonconfidential.

Note: Do not use this form to report system problems or to request copies of publications. Instead, contact your IBM representative or the IBM branch office serving you.

Would you like a reply?  ___YES ___NO

Please print your name, company name, and address:

__________________________________________________________________________

__________________________________________________________________________

__________________________________________________________________________

IBM Branch Office serving you: ____________________________________________

Thank you for your cooperation. You can either mail this form directly to us or give this form to an IBM representative who will forward it to us.
Reader's Comment Form

BUSINESS REPLY MAIL
FIRST-CLASS MAIL   PERMIT NO. 40   ARMONK, NY
POSTAGE WILL BE PAID BY ADDRESSEE:

INTERNATIONAL BUSINESS MACHINES CORPORATION
DEPARTMENT G60
PO BOX 6
ENDICOTT NY 13760-9987
Is there anything you especially like or dislike about this book? Feel free to comment on specific errors or omissions, accuracy, organization, or completeness of this book.

If you use this form to comment on the online HELP facility, please copy the top line of the HELP screen.

________________________ Help Information line ___ of ___

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you, and all such information will be considered nonconfidential.

Note: Do not use this form to report system problems or to request copies of publications. Instead, contact your IBM representative or the IBM branch office serving you.

Would you like a reply? __YES__ __NO

Please print your name, company name, and address:

_________________________________________________________________

_________________________________________________________________

_________________________________________________________________

IBM Branch Office serving you: ______________________________________

Thank you for your cooperation. You can either mail this form directly to us or give this form to an IBM representative who will forward it to us.
Is there anything you especially like or dislike about this book? Feel free to comment on specific errors or omissions, accuracy, organization, or completeness of this book.

If you use this form to comment on the online HELP facility, please copy the top line of the HELP screen.

_________ _________ _______ Help Information line ___ of ___

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you, and all such information will be considered nonconfidential.

Note: Do not use this form to report system problems or to request copies of publications. Instead, contact your IBM representative or the IBM branch office serving you.

Would you like a reply? ___YES ___NO

Please print your name, company name, and address:

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

IBM Branch Office serving you:

________________________________________________________________________

Thank you for your cooperation. You can either mail this form directly to us or give this form to an IBM representative who will forward it to us.
Reader's Comment Form

CUT OR FOLD ALONG LINE

BUSINESS REPLY MAIL
FIRST-CLASS MAIL   PERMIT NO. 40   ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE:

INTERNATIONAL BUSINESS MACHINES CORPORATION
DEPARTMENT G60
PO BOX 6
ENDICOTT NY 13760-9987

©