

HP 3000 Computer Systems
MPE Quick Reference Guide



HP Part No. 32033-90023
Printed in U.S.A. 1989

Second Edition
E1089

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced, or translated to another language without the prior written consent of Hewlett-Packard Company.

Copyright © 1989 by Hewlett-Packard Company

Print History

The following table lists the printings of this document, together with the respective release dates for each edition. The software version indicates the version of the software product at the time this document was issued. Many product releases do not require changes to the document. Therefore, do not expect a one-to-one correspondence between product releases and document editions.

Edition	Date	Software Version
First Edition	May 1986	G.01.00
Update #1	August 1986	G.02.B0
Update #2	June 1987	G.03.00
Second Edition	October 1989	G.03.08

Documentation Map

MPE V Manual Plan

INTRODUCTORY LEVEL:

**GENERAL INFORMATION
Manual**
(5953-7553)

**MPE V/E HP 3000:
Fundamental Skills**
(Part No. Pending)

**HP 3000 GUIDE
FOR THE
NEW SYSTEM OPERATOR**
(32033-90021)

**MPE V/E HP 3000:
Advanced Skills**
(Part No. Pending)

**MPE V GENERAL USER'S
Reference Manual**
(32033-90158)

**FCOPY
Reference Manual**
(32212-90003)

STANDARD USER LEVEL:

**MPE V COMMANDS
Reference Manual**
(32033-90006)

**MPE V INTRINSICS
Reference Manual**
(32033-90007)

**MPE V SYSTEM UTILITIES
Reference Manual**
(32033-90008)

**MPE V SEGMENTER
Reference Manual**
(30000-90011)

**MPE V DEBUG/
STACK DUMP
Reference Manual**
(30000-90012)

**MPE V FILE SYSTEM
Reference Manual**
(30000-90236)

ADMINISTRATIVE LEVEL:

**MPE V SYSTEM
OPERATION AND
RESOURCE MANAGEMENT
Reference Manual**
(32033-90005)

**MPE V SECURITY AND
ACCOUNT STRUCTURE**
(32033-90136)

**MPE V STORING AND
RESTORING FILES**
(32033-90133)

**MPE V SYSTEM BACKUP
AND RECOVERY**
(32033-90134)

SUMMARY LEVEL:

**MPE V QUICK
REFERENCE GUIDE**
(32033-90023)

LG200027_009a

Preface

This manual references the MPE operating system and its major subsystems. Each section of this manual corresponds to a manual in the MPE manual set. The sections and their corresponding manuals are listed below:

Commands	<i>MPE Commands Reference Manual (32033-90006)</i>
File System	<i>MPE File System Reference Manual (30000-90236)</i>
Utilities	<i>MPE Utilities Reference Manual (32033-90008)</i>
Intrinsics	<i>MPE Intrinsics Reference Manual (32033-90007)</i>
Segmenter	<i>Segmenter Reference Manual (30000-90011)</i>
Debug	<i>Debug/Stack Dump Reference Manual (30000-90012)</i>
FCOPY	<i>FCOPY/3000 Reference Manual (03000-90064)</i>
SORT-MERGE	<i>SORT-MERGE/3000 Reference Manual (32214-90002)</i>
EDIT	<i>EDIT/3000 Reference Manual (03000-90012)</i>
TurboIMAGE	<i>TurboIMAGE/3000 Reference Manual (32215-90003)</i>
QUERY	<i>QUERY/3000 Reference Manual (30000-90042)</i>
KSAM	<i>KSAM/3000 Reference Manual (30000-90097)</i>
VPLUS	<i>VPLUS/3000 Reference Manual (32209-90001)</i>
Special Terminal Keys	None

The Commands, Intrinsics, and Segmenter sections have “function tables” that group commands\intrinsics functionally.

Conventions

UPPERCASE

In a syntax statement, commands and keywords are shown in uppercase characters. The characters must be entered in the order shown; however, you can enter the characters in either uppercase or lowercase. For example:

COMMAND

can be entered as any of the following:

command Command COMMAND

It cannot, however, be entered as:

comm com_mand comamnd

italics

In a syntax statement or an example, a word in italics represents a parameter or argument that you must replace with the actual value. In the following example, you must replace *filename* with the name of the file:

COMMAND *filename*

bold italics

In a syntax statement, a word in bold italics represents a parameter that you must replace with the actual value. In the following example, you must replace ***filename*** with the name of the file:

COMMAND(***filename***)

punctuation

In a syntax statement, punctuation characters (other than brackets, braces, vertical bars, and ellipses) must be entered exactly as shown. In the following example, the parentheses and colon must be entered:

(*filename*):(*filename*)

underlining

Within an example that contains interactive dialog, user input and user responses to prompts are indicated by underlining. In the following example, yes is the user's response to the prompt:

Do you want to continue? >> yes

{ }

In a syntax statement, braces enclose required elements. When several elements are stacked within braces, you must select one. In the following example, you must select either ON or OFF:

COMMAND { ON
 OFF }

Conventions (continued)

[]	In a syntax statement, brackets enclose optional elements. In the following example, <i>OPTION</i> can be omitted: <code>COMMAND filename [OPTION]</code> When several elements are stacked within brackets, you can select one or none of the elements. In the following example, you can select <i>OPTION</i> or <i>parameter</i> or neither. The elements cannot be repeated. <code>COMMAND filename [OPTION parameter]</code>
[empty]	No entry specifies the job/session input device \$STDIN for the FROM file or the job/session list device \$STDLIST for the TO file.
[...]	In a syntax statement, horizontal ellipses enclosed in brackets indicate that you can repeatedly select the element(s) that appear within the immediately preceding pair of brackets or braces. In the example below, you can select <i>parameter</i> zero or more times. Each instance of <i>parameter</i> must be preceded by a comma: <code>[,parameter][...]</code> In the example below, you only use the comma as a delimiter if <i>parameter</i> is repeated; no comma is used before the first occurrence of <i>parameter</i>: <code>[parameter][, ...]</code>
...	In a syntax statement, horizontal ellipses enclosed in vertical bars indicate that you can select more than one element within the immediately preceding pair of brackets or braces. However, each particular element can only be selected once. In the following example, you must select <i>A</i>, <i>AB</i>, <i>BA</i>, or <i>B</i>. The elements cannot be repeated. <code>{ A B } ... </code>
...	In an example, horizontal or vertical ellipses indicate where portions of an example have been omitted.
Δ	In a syntax statement, the space symbol Δ shows a required blank. In the following example, <i>parameter</i> and <i>parameter</i> must be separated with a blank: <code>(parameter)Δ(parameter)</code>
,	In commands, the comma can replace <i>IN</i>, <i>TO</i>, and <i>BY</i>.
	The symbol  indicates a key on the keyboard. For example,  represents the carriage return key or  represents the shift key.

Conventions (continued)

CTRL <i>character</i>	CTRL <i>character</i> indicates a control character. For example, CTRL Y means that you press the control key and the Y key simultaneously.
expr	An expression may be substituted where any numeric field is allowed. An expression can include any of the operators: +, -, /, or *. Use () to override hierarchy. The operands in an expression are assumed to be octal numbers unless indicated as follows: # Prefix to decimal value; e.g., #10 is decimal value 10. " " Enclose ASCII characters for which an octal value is indicated; for example, "A" indicates octal value 101. ' ' Enclose memory location of which the contents are indicated; e.g., 'DB+4' is the contents of location DB+4. \$ Prefix indicating contents of a register; e.g., \$P is the current value of P. !(a:b) Extract bit field where a is the starting bit and b is the number of bits; e.g., \$P(10:10) is bits 8 through 15 of the contents of register P.
<i>segment</i>	In commands, <i>segment</i> refers to logical segment number obtained from PMAP for user program files or from last column of LMAP for Segmented Library Procedures.
:	Indicates indirect addressing when following a parameter used as an offset to a memory location base.
base prefixes	The prefixes %, #, and \$ specify the numerical base of the value that follows: %num specifies an octal number. #num specifies a decimal number. \$num specifies a hexadecimal number. If no base is specified, decimal is assumed.
bits (<i>bit:length</i>)	When a parameter contains more than one piece of data within its bit field, the different data fields are described in the format bits (<i>bit:length</i>), where <i>bit</i> is the first bit in the field and <i>length</i> is the number of consecutive bits in the field. For example, bits (13:3) indicates bits 13, 14, and 15: <pre> most significant least significant -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- 0 13 14 15 -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- bits (0:1) bits (13:3) </pre>

Conventions (continued)

Mnemonics

Mnemonic	Meaning
BA	Byte array
BP	Byte pointer
D	Double by Reference
DA	Double array
DR	Double by reference
DV	Double by value
I	Integer by reference
IA	Integer array
IV	Integer by value
L	Logical by reference
LA	Logical array
LP	Logical Procedure
LV	Logical by value
R	Real

Contents

1. Commands	
Command Functions	1-1
Capabilities Required	1-4
Commands	1-7
() COMMAND LOGON	1-7
ABORT	1-7
ABORTIO/=ABORTIO	1-7
ABORTJOB/=ABORTJOB	1-8
ACCEPT	1-8
ALLOCATE	1-8
ALLOW	1-8
ALTACCT	1-9
ALTGROUP	1-9
ALTJOB	1-9
ALTLOG	1-9
ALTSEC	1-10
ALTPOOLFILE	1-10
ALTUSER	1-10
ALTVSET	1-11
ASSOCIATE	1-11
AUTOALLOCATE	1-11
BASIC	1-11
BASICGO	1-11
BASICOMP	1-12
BASICPREP	1-12
BBASIC	1-12
BBASICGO	1-12
BBASICOMP	1-12
BBASICPREP	1-13
BREAKJOB	1-13
BUILD	1-13
BYE	1-13
CACHECONTROL	1-14
CHANGELOG	1-14
COBOL	1-14
COBOLGO	1-14
COBOLPREP	1-15
COBOLII	1-15
COBOLHIGO	1-15
COBOLIPREP	1-15
COMMENT	1-16
CONSOLE	1-16

CONTINUE	1-16
DATA	1-16
DEALLOCATE	1-16
DEBUG	1-17
DELETESPOOLFILE	1-17
DISALLOW	1-17
DISASSOCIATE	1-17
DISCRPS	1-18
DISMOUNT	1-18
DOWN	1-18
DOWNLOAD	1-18
DSTAT	1-18
EDITOR	1-19
ELSE	1-19
ENDIF	1-19
:EOD	1-19
:EOF:	1-19
EOJ	1-20
FCOPY	1-20
FILE	1-20
FOREIGN	1-21
FORTGO	1-21
FORTPREP	1-21
FORTTRAN	1-21
FREERIN	1-21
FTN	1-22
FTNGO	1-22
FTNPREP	1-22
FULLBACKUP	1-22
GETLOG	1-22
GETRIN	1-23
GIVE	1-23
HEADOFF	1-23
HEADON	1-23
HELLO	1-23
HELP	1-24
IF	1-24
JOB	1-24
JOBFENCE	1-24
JOBPRI	1-25
JOBSECURITY	1-25
LDISMOUNT	1-25
LIMIT	1-25
LISTACCT	1-25
LISTEQ	1-26
LISTF	1-26
LISTFTEMP	1-26
LISTGROUP	1-26
LISTLOG	1-27
LISTUSER	1-27
LISTVS	1-27

LMOUNT	1-27
LOG	1-28
=LOGOFF	1-28
=LOGON	1-28
MOUNT	1-28
NEWACCT	1-28
NEWGROUP	1-29
NEWUSER	1-29
NEWVSET	1-29
OPENQ	1-29
OUTFENCE	1-30
PARTBACKUP	1-30
PASCAL	1-30
PASCALGO	1-30
PASCALPREP	1-30
PASSWORD	1-31
PREP	1-31
PREPRUN	1-31
PTAPE	1-32
PURGE	1-32
PURGEACCT	1-32
PURGEGROUP	1-32
PURGEUSER	1-32
PURGEVSET	1-33
RECALL/=RECALL	1-33
REDO	1-33
REFUSE	1-33
RELEASE	1-33
RELLOG	1-34
RENAME	1-34
REPLY/=REPLY	1-34
REPORT	1-34
RESET	1-34
RESETACCT	1-35
RESETDUMP	1-35
RESTORE	1-35
RESUME	1-35
RESUMEJOB	1-36
RESUMELOG	1-36
RESUMESPOOL	1-36
RPG	1-36
RPGGO	1-37
RPGPREP	1-37
RUN	1-37
SAVE	1-37
SECURE	1-38
SEGMENTER	1-38
SET	1-38
SETCATALOG	1-38
SETDUMP	1-38
SETJCW	1-39

SETMSG	1-39
SHOWALLOCATE	1-39
SHOWALLOW	1-39
SHOWCACHE	1-40
SHOWCATALOG	1-40
SHOWCOM	1-40
SHOWDEV	1-40
SHOWIN	1-40
SHOWJCW	1-41
SHOWJOB	1-41
SHOWLOG	1-41
SHOWLOGSTATUS	1-41
SHOWME	1-42
SHOWOUT	1-42
SHOWQ	1-42
SHOWTIME	1-42
=SHUTDOWN	1-42
SHUTQ	1-43
SPEED	1-43
SPL	1-43
SPLGO	1-43
SPLPREP	1-43
STARTCACHE	1-44
STARTSESS	1-44
STARTSPOOL	1-44
STOPCACHE	1-44
STOPSPOOL	1-45
STORE	1-45
STREAM	1-45
STREAMS	1-45
SUSPENDSPOOL	1-46
SWITCHLOG	1-46
SYSDUMP	1-46
TAKE	1-46
TELL	1-46
TELLOP	1-47
TUNE	1-47
UP	1-47
VINIT	1-47
VMOUNT	1-47
VSUSER	1-48
WARN	1-48
WELCOME	1-48

2. File System	
System Defined Files	2-1
File Codes	2-2
Carriage Control Directives	2-4
Carriage Control Effect Summary	2-7
File Access and Security	2-8
Default Security for Accounts, Groups, and Users	2-8
Net Default Access to Files	2-9
Capabilities	2-9
Default Capabilities	2-9
FOPEN Options	2-10
FOPEN Aoptions	2-11
3. Utilities	
ASOCTABL5	3-1
DISKED5	3-1
DPAN5	3-2
FREE5	3-3
LISTDIR5	3-3
LISTLOG5	3-4
LISTEQ5	3-4
MEMLOGAN	3-5
MEMTIMER	3-6
PATCH	3-6
RECOVER5	3-6
SADUTIL	3-7
SLPATCH	3-9
SPOOK5	3-9
VINIT	3-11
4. Intrinsic	
Intrinsic Function Table	4-1
Intrinsic	4-3
ABORTSESS	4-3
ACTIVATE	4-3
ADJUSTUSLF	4-3
ALTDSEG	4-3
ALMANAC	4-4
ARITRAP	4-4
ASCH	4-4
BEGINLOG	4-4
BINARY	4-5
CALENDAR	4-5
CATCLOSE	4-5
CATOPEN	4-5
CATREAD	4-5
CAUSEBREAK	4-6
CLEANUSL	4-6
CLOCK	4-6
CLOSELOG	4-6
COMMAND	4-7

CREATE	4-7
CREATEPROCESS	4-7
CTranslate	4-7
DASCI	4-8
DATELINE	4-8
DBINARY	4-8
DEBUG	4-8
DLSIZE	4-9
DMOVIN	4-9
DMOVOUT	4-9
DUMPPMAP	4-9
ENDLOG	4-10
EXPANDUSLF	4-10
FATHER	4-10
FCARD	4-10
FCHECK	4-11
FCLOSE	4-11
FCONTROL	4-11
FDELETE	4-11
FDEVICECONTROL	4-12
FERRMSG	4-12
FFILEINFO	4-12
FGETINFO	4-12
FINDJCW	4-13
FINDPMAPADDR	4-13
FINDPMAPNAME	4-13
FINTEXT	4-13
FINTSTATE	4-14
FLABELINFO	4-14
FLOCK	4-14
FLUSHLOG	4-14
FMTCALENDAR	4-15
FMTCLOCK	4-15
FMTDATE	4-15
FOPEN	4-15
FPARSE	4-16
FPOINT	4-16
FREAD	4-16
FREADBACKWARD	4-16
FREADDIR	4-17
FREADLABEL	4-17
FREADSEEK	4-17
FREEDSEG	4-17
FREELOCRIN	4-17
FRELATE	4-18
FRENAME	4-18
FSETMODE	4-18
FSPACE	4-19
FUNLOCK	4-19
FUPDATE	4-19
FWRITE	4-19

FWRITEDIR	4-19
FWRITELABEL	4-20
GENMESSAGE	4-20
GETDSEG	4-20
GETINFO	4-20
GETJCW	4-21
GETLOCRIN	4-21
GETORIGIN	4-21
GETPRIORITY	4-21
GETPRIVMODE	4-22
GETPROCID	4-22
GETPROCINFO	4-22
GETUSERMODE	4-22
HPACDINFO	4-23
HPACDPUT	4-23
INITUSLF	4-23
IODONTWAIT	4-23
IOWAIT	4-23
JOBINFO	4-24
KILL	4-24
LOADPROC	4-24
LOCKGLORIN	4-24
LOCKLOCRIN	4-25
LOCRINOWNER	4-25
LOGINFO	4-25
LOGSTATUS	4-25
MAIL	4-26
MYCOMMAND	4-26
NLAPPEND	4-26
NLCOLLATE	4-26
NLCONVCLOCK	4-27
NLCONVCUSTDATE	4-27
NLCONVNUM	4-27
NLFMTCALENDAR	4-27
NLFMTCLOCK	4-28
NLFMTCUSTDATE	4-28
NLFMTDATE	4-28
NLFMTLONGCAL	4-28
NLFMTNUM	4-28
NLGETLANG	4-29
NLINFO	4-29
NLKEYCOMPARE	4-29
NLNUMSPEC	4-29
NLREPCHAR	4-29
NLSCANMOVE	4-30
NLSWITCHBUF	4-30
NLTRANSLATE	4-30
OPENLOG	4-30
PAUSE	4-31
PRINT	4-31
PRINTFILEINFO	4-31

PRINTOP	4-31
PRINTOPREPLY	4-31
PROCINFO	4-32
PROCTIME	4-32
PTAPE	4-32
PUTJCW	4-32
QUIT	4-32
QUITPROG	4-33
READ	4-33
READX	4-33
RECEIVEMAIL	4-33
RESETCONTROL	4-33
RESETDUMP	4-34
SEARCH	4-34
SENDMAIL	4-34
SETDUMP	4-34
SETJCW	4-34
STACKDUMP	4-35
STARTSESS	4-35
SUSPEND	4-35
SWITCHDB	4-35
TERMINATE	4-36
TIMER	4-36
UNLOADPROC	4-36
UNLOCKGLORIN	4-36
UNLOCKLOCRIN	4-37
WHO	4-37
WRITELOG	4-37
XARITRAP	4-37
XCONTRAP	4-38
XLIBTRAP	4-38
XSYSTRAP	4-38
ZSIZE	4-38

5. Segmenter

Initiate SEGMENTER	5-1
Commands	5-2
ADDRL	5-3
ADDSL	5-3
AUXUSL	5-3
BUILDRL	5-3
BUILDSL	5-3
BUILDUSL	5-4
CEASE	5-4
CLEANS�	5-4
CLEANUSL	5-4
COPY	5-4
COPYSL	5-5
COPYUSL	5-5
EXIT	5-5
HELP	5-5

HIDE	5-5
LISTAUX	5-6
LISTPMAP	5-6
LISTRL	5-6
LISTSL	5-6
LISTUSL	5-6
NEWSEG	5-7
PREPARE	5-7
PURGERBM	5-7
PURGERL	5-7
PURGESL	5-8
REDO	5-8
REVEAL	5-8
RL	5-8
SETFPMAP	5-8
SHOW	5-9
SL	5-9
USE	5-9
USL	5-9
Intrinsics	5-9
ADJUSTUSLF	5-10
CLEANUSLF	5-10
DUMPPMAP	5-10
EXPANDUSLF	5-10
FINDPMAPADDR	5-10
FINDPMAPNAME	5-11
INITUSLF	5-11
6. Debug	
Operation	6-1
Command Operation	6-1
Access	6-2
System Messages	6-2
Errors	6-2
Breakpoint Commands	6-3
*AP	6-3
*AS	6-3
B	6-3
BT	6-4
B@	6-4
*B	6-4
*BA	6-5
*BAX	6-5
*BS	6-5
C	6-5
C@	6-6
*C	6-6
*C@	6-6
*C	6-6
E@	6-7
E	6-7

R	6-7
Display Commands	6-8
D	6-8
*D A	6-9
*D CO	6-9
*D COX	6-9
*D DA	6-9
*D DX	6-9
*D EA	6-10
*D SY	6-10
DR	6-10
*DR	6-10
*DV	6-11
Listing Commands	6-11
L	6-11
L0	6-11
*L	6-12
Memory/Register Modification Commands	6-12
M	6-12
MR	6-13
\$register	6-13
Calculation Display Command	6-13
=expr	6-13
Trace Command	6-14
T	6-14
Segment Freeze Commands	6-15
*F	6-15
*U	6-15
Segment and Register Contents	6-16
Stack Marker Format	6-17
Status Register	6-17
Condition Codes	6-18
PMAP Format	6-19
LMAP Format	6-20
 7. FCOPY	
Initiate FCOPY	7-1
Commands	7-1
Keywords	7-2
Examples	7-3
 8. SORT/MERGE	
Initiate SORT	8-1
SORT Commands	8-1
A	8-1
DATA	8-1
E	8-1
EX	8-2
I	8-2
K	8-2
L	8-2

O	8-2
RESET	8-3
SH	8-3
V	8-3
:	8-3
SORT Formal File Designators	8-3
SORT Intrinsic	8-4
SORTINIT	8-4
SORTINPUT	8-4
SORTOUTPUT	8-4
SORTEND	8-4
SORTSTAT	8-4
SORTTITLE	8-4
SORTERRORMESS	8-4
SORTINITIALF	8-4
SORTINITIAL	8-5
Initiate MERGE	8-5
MERGE Commands	8-5
A	8-5
DATA	8-5
E	8-5
EX	8-5
I	8-5
K	8-6
L	8-6
O	8-6
RESET	8-6
SH	8-6
V	8-6
:	8-6
MERGE Formal File Designators	8-7
MERGE Intrinsic	8-7
MERGEINIT	8-7
MERGEOUTPUT	8-7
MERGEEND	8-7
MERGESTAT	8-7
MERGETITLE	8-7
MERGEERRORMESS	8-8
MERGE	8-8
9. Edit	
Initiate EDITOR	9-1
Commands	9-1
A	9-1
B	9-1
C	9-2
CO	9-2
D	9-2
E	9-2
F	9-2
G	9-3

H	9-3
I	9-3
J	9-3
K	9-3
L	9-4
M	9-4
N	9-4
O	9-4
P	9-4
Q	9-5
R	9-5
S	9-5
T	9-5
U	9-6
V	9-6
W	9-6
X	9-6
Y	9-6
Z	9-7
:	9-7

10. TurboIMAGE

Initiate DBSCHEMA	10-1
Commands	10-1
Schema Structure	10-2
Initiate DBLOAD	10-3
Initiate DBRECOV	10-4
Commands	10-4
Initiate DBRESTOR	10-5
Initiate DBSTORE	10-5
Initiate DBUNLOAD	10-6
Initiate DBUTIL	10-7
Commands	10-7
Intrinsics	10-10
Calling IMAGE Procedures:	10-10
DBBEGIN	10-10
DBCLOSE	10-11
DBCONTROL	10-11
DBDELETE	10-11
DBEND	10-11
DBERROR	10-11
DBEXPLAIN	10-11
DBFIND	10-12
DBGET	10-12
DBINFO	10-12
DBLOCK	10-12
DBMEMO	10-13
DBOPEN	10-13
DBPUT	10-13
DBUNLOCK	10-14
DBUPDATE	10-14

IMAGE Error Messages	10-15
--------------------------------	-------

11. QUERY

Initiate QUERY	11-1
Commands	11-1
A	11-1
AD	11-1
AS	11-2
B	11-2
C	11-2
CL	11-2
CO	11-2
D	11-2
DBLIST	11-3
DE	11-3
DEF	11-3
DEL	11-3
E	11-3
F	11-3
FO	11-6
H	11-6
J	11-7
LANGUAGE=	11-7
L	11-7
LISTREDO	11-8
M	11-8
MULTID	11-8
MU	11-8
O	11-9
PA	11-9
P	11-9
REDO	11-9
REL	11-9
REN	11-9
REPL	11-10
R	11-10
S	11-13
SA	11-13
SET	11-13
SH	11-13
SUB	11-13
T	11-13
TRANSE	11-13
TRANSM	11-14
UN	11-14
U ADD	11-14
U DELETE	11-14
U REPLACE	11-14
U	11-14
V	11-14
X	11-14

12. KSAM	
Commands	12-1
BUILD / B	12-1
ERASE	12-2
EXIT / E	12-2
HELP / H	12-2
KEYDUMP / KD	12-2
KEYINFO / KI	12-2
KEYSEQ / KS	12-3
PURGE	12-3
RENAME / R	12-3
SAVE / S	12-3
VERIFY / V	12-3
Intrinsics	12-4
FCHECK	12-4
FCLOSE	12-4
FCONTROL	12-4
FERRMSG	12-4
*FFINDBYKEY	12-4
*FFINDN	12-5
FGETINFO	12-5
*FGETKEYINFO	12-5
FLOCK	12-5
FOPEN	12-5
FPOINT	12-6
FREAD	12-6
*FREADBYKEY	12-6
*FREADC	12-6
FREADDIR	12-6
FREADLABEL	12-6
*FREMOVE	12-7
FSETMODE	12-7
FSPACE	12-7
FUNLOCK	12-7
FUPDATE	12-7
FWRITE	12-7
FWRITELABEL	12-8
HP32208	12-8
BASIC Procedures	12-10
COBOL Procedures	12-11
FORTRAN Procedures	12-13
SPL Intrinsics	12-13

13. VPLUS	
Initiate FORMSPEC	13-1
FORMSPEC Function Keys	13-1
Initiate REFSPEC	13-14
Initiate REFORMAT	13-19
Intrinsics	13-20
COMAREA	13-23
Initiate ENTRY	13-30
14. Special Terminal Keys	
15. ASCII Instruction Set	
Byte Position	15-2
Index of OP Code Groups	15-4
00 Stack OPs	15-6
01 Shifts/Branches	15-7
02 Moves/Immediates	15-8
03 I/O Linkage Control	15-10
Memory Reference	15-11
File Labels	15-12
Status Register	15-13
Code Segment Table	15-14
Interrupts/Traps	15-15
00 Stack Ops	15-16
01 Shifts/Branches	15-17
02 Moves/Immediates	15-18
Two-Word "Hardware" Instructions	15-20
Two-Word I/O Instructions	15-20
03 I/O Linkage Control	15-21
Memory Reference	15-22
Status Register	15-23
Code Segment Table	15-24
Interupts/Traps	15-25
Memory Conversion Table	15-26



Commands

Command Functions

Table 1-1. Command Functions

Function	Commands		
ACCOUNTS	ALTACCT LISTACCT	NEWACCT PURGEACCT	REPORT RESETACCT
BACKUP	FULLBACKUP PARTBACKUP	SYSDUMP RESTORE	STORE
COMMANDS	()	REDO	
COPY	DSCOPY	FCOPY	
DATA	ACCEPT DATA	REFUSE STREAM	STREAMS
DATA COMMUNICATION	DSCONTROL DSCOPY	DSLINE REMOTE	REMOTE HELLO
DATE & TIME	SHOWTIME		
DELETE	DELETESPOOLFILE PURGE	PURGEACCT PURGEGROUP	PURGEUSER PURGEVSET
DEVICES	ABORTIO CONSOLE	DOWNLOAD PTAPE	SPEED
DEVICE CONTROL	ACCEPT ASSOCIATE DISASSOCIATE	DOWN GIVE REFUSE	TAKE UP
DEVICE STATUS	SHOWCOM	SHOWDEV	
DEVICEFILE STATUS	DATA	SHOWIN	SHOWOUT
DISK	BUILD DSTAT	DISCRPS FOREIGN	VINIT
DISK CACHING	CACHECONTROL SHOWCACHE	STARTCACHE STOPCACHE	
FILES	BUILD FILE LISTEQ LISTF	LISTFTMP PURGE RESET RENAME	RESTORE SAVE STORE
GROUPS	ALTGROUP LISTGROUP	NEWGROUP PURGEGROUP	REPORT
JOBS	ALTJOB COMMENT CONTINUE ELSE	ENDIF EOJ IF JOB	SET STREAM

Table 1-1. Command Functions (continued)

Function	Commands		
JOB CONTROL	ABORTJOB ACCEPTALTJOB BREAKJOB JOBFENCE	JOBPRI JOBSECURITY LIMIT LOGOFF LOGON	REFUSE RESUMEJOB SHUTDOWN STREAMS
JOB STATUS	SHOWJOB	SHOWME	
JOB CONTROL WORDS	SETJCW	SHOWJCW	
LOGGING(SYSTEM)	RESUMELOG	SHOWLOG	SWITCHLOG
LOGGING(USER)	ALTLOG CHANGELOG GETLOG	LISTLOG LOG RELLOG	SHOWLOGSTATUS
MESSAGES	RECALL REPLY TELL	TELOP SETMSG WARN	WELCOME
OUTPUT (HEADER/TRAILER)	HEADOFF	HEADON	
PRINTING	Refer to "SPOOLING"		
PRIVATE VOLUMES	ALTACCT ALTGROUP ALTVSET DISMOUNT LISTVS LDISMOUNT	LMOUNT MOUNT NEWVSET NEWACCT NEWGROUP PURGEACCT	PURGEGROUP PURGEVSET REPORT VINIT VMOUNT VSUSER
PROGRAM & PROCEDURE ALLOCATION	ALLOCATE AUTOALLOCATE	DEALLOCATE SHOWALLOCATE	
PROGRAMMATIC SESSIONS	STARTSESS		
PROGRAMMING LANGUAGES	BASIC BASICCOMP BASICGO BASICPREP BBASIC BBASICGO BBASICCOMP BBASICPREP COBOL COBOLGO	COBOLPREP COBOLII COBOLIIGO COBOLIIPREP FORTGO FORTPREP FORTRAN FTN FTNGO FTNPREP	PASCAL PASCALGO PASCALPREP RPG RPGGO RPGPREP SPL SPLGO SPLPREP

1-2 Commands

Table 1-1. Command Functions (continued)

Function	Commands		
PROGRAMS (RUNNING)	ABORT RESUME	RUN PREP	PREPRUN
RESOURCE I.D. NUMBERS (RIN)	GETRIN	FREERIN	
SECURITY	ACCEPT ALLOW ALTSEC	DISALLOW JOBSECURITY RELEASE	SHOWALLOW SECURE VSUSER
SESSIONS	() ABORT BYE :EOD	:EOF HELLO HELP RESUME	SHOWME STARTSESS
SESSION CONTROL	ABORTJOB ACCEPT ALTJOB	JOBFENCE LIMIT LOGOFF	LOGON REFUSE SHUTDOWN
SPOOLING	ALTSPoolFILE DELETESPoolFILE OPENQ	OUTFENCE RESUMESPool SHUTQ	STARTSPool STOPSPool SUSPENDSPool
SUBSYSTEMS	DEBUG EDITOR DSCOPY	FCOPY HELP SEGMENTER	SETDUMP RESETDUMP VINIT
SYSTEM MODIFICATION & BACKUP	FULLBACKUP PARTBACKUP	SYSDUMP RESTORE	STORE
SYSTEM PERFORMANCE MEASUREMENT	CACHECONTROL DISCRPS JOBPRI	SHOWQ STARTCACHE STOPCACHE	TUNE
USER DEFINED COMMANDS (UDC)	SETCATALOG	SHOWCATALOG	
USERS	ALTUSER LISTUSER	NEWUSER PURGEUSER	PASSWORD REPORT

Capabilities Required

Table 1-2. Capabilities Required for Commands

Command	Console	OP	SM	AM	Other
ABORTIO	X				
ABORTJOB	X		X	X	
ACCEPT	X				
ALLOCATE		X			
ALLOW	X				
ALTACCT			X		
ALTGROUP			X	X	
ALTJOB	X				
ALTLOG					LG
ALTSEC					Creator
ALTSPoolFILE	X	X	X		
ALTUSER			X	X	
ALTVSET					CV
ASSOCIATE					Must be given access to the device in ASOCTBLE
BREAKJOB	X				
CACHECONTROL		X	X		
CHANGELOG		X			
CONSOLE	X	X	X		
DEALLOCATE		X			
DEBUG					PM
DELETESPOOLFILE	X				
DISALLOW	X				
DISCRPS	X				
DOWN	X				
DOWNLOAD	X				
FOREIGN	X				
FULLBACKUP		X			

Table 1-2. Capabilities Required for Commands (continued)

Command	Console	OP	SM	AM	Other
GETLOG					LG
GETRIM					Password needed
GIVE	X				
HEADOFF	X				
HEADON	X				
JOBFENCE	X				
JOBPRI		X			
JOBSECURITY	X				
LDISMOUNT	X				
LIMIT	X				
LISTACCT			X	X	
LISTGROUP			X	X	
LISTLOG					LG
LISTUSER			X	X	
LISTVS					UV or CV
LMOUNT	X				
LOG		X			
=LOGON	X				
=LOGOFF	X				
NEWACCT			X		
NEWGROUP			X	X	
NEWUSER			X	X	
NEWVSET					CV
OPENQ	X				
OUTFENCE	X				
PARTBACKUP		X			
PURGEACCT			X		
PURGEGROUP			X	X	
PURGEUSER			X	X	
PURGEVSET					CV
RECALL	X				
REFUSE	X				

Table 1-2. Capabilities Required for Commands (continued)

Command	Console	OP	SM	AM	Other
RELEASE					
RENAME					Must be the creator
REPLY	X				
REPORT					
RESETACCT			X		
RESTORE					Depends on file set
RESUMEJOB	X				
RESUMELOG	X	X			
RESUMESPOOL	X				
SECURE					Must be the creator
SHOWCOM	X				
SHOWLOG	X	X			
SHOWQ		X			
=SHUTDOWN	X				
SHUTQ	X				
STARTCACHE		X	X		
STARTSESS					PS
STARTPOOL	X				
STOPCACHE		X	X		
STOPSPPOOL	X				
STORE					Depends on file set
STREAMS	X				
SUSPENDSPOOL	X				
SWITCHLOG	X	X			
SYSDUMP		X			
TAKE	X				
TUNE		X	X		
UP	X				
VINIT		X	X		
VMOUNT	X				
WARN	X				
WELCOME	X				

Commands

() COMMAND LOGON

Begins a session, executes the MPE V/E command enclosed in parentheses, and ends the session upon completion of the command.

```
( [ : ] commandname ) [ sessionname , ] username [ / userpass ] . acctname [ / acctpass ]
[ , groupname [ / grouppass ] ] [ ; TERM=termtype ] [ ; TIME=cpusecs ] [ ; PRI=
{ BS
  CS
  DS
  ES } ]
[ { ; INPRI=inputpriority } ]
[ { ; HIPRI } ]
```

ABORT

Aborts the current program or operation.

```
ABORT
```

ABORTIO/=ABORTIO

Aborts one pending I/O request for a device.

```
ABORTIO ldev
=ABORTIO ldev
```

ABORTJOB/=ABORTJOB

Aborts a job or session.

$$\begin{aligned} \text{ABORTJOB} & \left\{ \begin{array}{l} \#Jnnn \\ \#Snnn \\ [jobname] user.acct \end{array} \right\} \\ \\ \text{=ABORTJOB} & \left\{ \begin{array}{l} \#Jnnn \\ \#Snnn \\ [jobname,] user.acct \end{array} \right\} \end{aligned}$$

ACCEPT

Permits a designated device to accept jobs, sessions and/or data.

$$\text{ACCEPT} \left[\begin{array}{l} \text{JOBS,} \\ \text{DATA,} \end{array} \right] ldev$$

ALLOCATE

Permanently loads a program or procedure into main memory.

$$\text{ALLOCATE} \left[\begin{array}{l} \text{PROCEDURE,} \\ \text{PROGRAM,} \end{array} \right] name$$

ALLOW

Grants a user access to a specific operator command.

$$\begin{aligned} & \text{ALLOW FILE=formaldesignator[;SHOW]} \\ \\ \text{ALLOW} & \left\{ \begin{array}{l} @.@ \\ user.@ \\ @.acct \\ user.acct \end{array} \right\} ; \text{COMMANDS=command[,command, ...]} \end{aligned}$$

ALTACCT

Changes the attributes of an existing account.

```
ALTACCT acctname [ ;PASS=[password] ] [ ;FILES=[filespace] ] [ ;CPU=[cpu] ]
[ ;CONNECT=[connect] ] [ ;CAP=[capabilitylist] ] [ ;ACCESS=[fileaccess] ]
[ ;MAXPRI=[subqueueuname] ] [ ;LOCATTR=[localattribute] ] [ ;VS=[volset: { ALT
SPAN } ] ]
[ ;USERPASS={ { REQ } }
{ OPT } ] ]
```

Note: The USERPASS parameter is only available if the HP Security Monitor (HP30392) has been installed.

ALTGROUP

Changes one or more attributes of a group.

```
ALTGROUP groupname[.accountname] [ ;PASS=[password] ] [ ;CAP=[capabilitylist] ]
[ ;FILES=[filespace] ] [ ;CPU=[cpu] ] [ ;CONNECT=[connect] ]
[ ;ACCESS=[fileaccess] ] [ ;VS=[volset: { ALT
SPAN } ] [ ; ... ] ]
```

ALTJOB

Alters the attributes of waiting or scheduled jobs.

```
ALTJOB { #Jnnn } [ ;INPRI=inputpriority ] [ ;OUTDEV={ ldev
#Snnn } { devclass } ]
```

ALTLOG

Alters the attributes of an existing user logging identifier.

```
ALTLOG logid [ ;LOG=logfile { ,DISC
,TAPE
,SDISC
,CTAPE } ] [ ;PASS=password ] [ { ;AUTO
;NOAUTO } ]
```

ALTSEC

Alters a file's security provisions and manipulates the access control definitions (ACDs) associated with files and devices.

$$\begin{aligned} & \text{ALTSEC } \textit{objectname} \left[\begin{array}{c} \{ \text{FILENAME} \} \\ \text{LDEV} \\ \text{DEVCLASS} \end{array} \right] ; \{ [\text{ACCESS=}] (\textit{fileaccess} [; \textit{fileaccess} [; \dots]]) \} \\ & \left\{ \begin{array}{l} \text{NEWACD} = \left\{ \begin{array}{l} (\textit{acd pair} [; \textit{acd pair} [; \dots]]) \\ \textit{filereference} \end{array} \right\} \\ \text{COPYACD} = \left\{ \begin{array}{l} \textit{objectname} \left[\begin{array}{c} \{ \text{FILENAME} \} \\ \text{LDEV} \end{array} \right] \\ \text{LDEV} \end{array} \right\} \\ \text{ADDPAIR} = \left\{ \begin{array}{l} (\textit{acd pair} [; \textit{acd pair} [; \dots]]) \\ \textit{filereference} \end{array} \right\} \\ \text{REPPAIR} = \left\{ \begin{array}{l} (\textit{acd pair} [; \textit{acd pair} [; \dots]]) \\ \textit{filereference} \end{array} \right\} \\ \text{DELPAIR} = \left\{ \begin{array}{l} (\textit{user spec} [; \textit{user spec} [; \dots]]) \\ \textit{filereference} \end{array} \right\} \end{array} \right\} \{ \text{DELACD} \} \end{aligned}$$

ALTSPoolFILE

Alters the characteristics of an output spool file.

$$\text{ALTSPoolFILE} \left\{ \begin{array}{c} \{ \#0nnn \} \\ \textit{ldev1} \end{array} \right\} \left\{ \begin{array}{l} ; \text{PRI} = \textit{outputpriority} \\ ; \text{COPIES} = \textit{numcopies} \\ ; \text{DEV} = \left\{ \begin{array}{c} \textit{ldev2} \\ \textit{devclass} \end{array} \right\} \\ ; \text{DEFER} \end{array} \right\} [; \dots]$$

ALTUSER

Alters attributes currently defined for a user.

$$\begin{aligned} & \text{ALTUSER } \textit{username} [, \textit{acctname}] [; \text{PASS} = [\textit{password}]] [; \text{CAP} = [\textit{capabilitylist}]] \\ & [; \text{MAXPRI} = [\textit{subqueue name}]] [; \text{LOCATTR} = [\textit{localattribute}]] \\ & [; \text{HOME} = [\textit{homegroup name}]] [; \text{USERPASS} = \left\{ \begin{array}{c} \text{REQ} \\ \text{OPT} \end{array} \right\} [, \text{EXPIRED}]] \end{aligned}$$

Note: The USERPASS parameter is only available if the HP Security Monitor has been installed.

ALTVSET

Modifies the definitions of volume sets.

```
ALTVSET vname [ ;ADDCLASS=vcname:vname[ , vname[ , ... ] ] ]  
[ ;EXPANDCLASS=vcname:vname[ , vname[ , ... ] ] ]  
[ ;EXPANSET=vname:type[ , vname:type[ , ... ] ] ]
```

ASSOCIATE

Gives system operator control of a device to a user.

```
ASSOCIATE devclass
```

AUTOALLOCATE

Disables or enables the automatic allocation and deallocation of programs.

```
AUTOALLOCATE { OFF }  
               { ON  }
```

BASIC

Interprets a BASIC program. BASIC is not part of the HP 3000 fundamental operating system and must be purchased separately.

```
BASIC [ commandfile ] [ , [ inputfile ] [ , listfile ] ]
```

BASICGO

Compiles, prepares, and executes a BASIC program. BASIC is not part of the HP 3000 fundamental operating system and must be purchased separately.

```
BASICGO [ commandfile ] [ , listfile ]
```

BASICOMP

Compiles a BASIC program. BASIC is not part of the HP 3000 fundamental operating system and must be purchased separately.

```
BASICOMP [ commandfile ] [ , [ uslfile ] [ , listfile ] ]
```

BASICPREP

Compiles and prepares a BASIC program. BASIC is not part of the HP 3000 fundamental operating system and must be purchased separately.

```
BASICPREP [ commandfile ] [ , [ progfile ] [ , listfile ] ]
```

BBASIC

Interprets an HP Business BASIC program. HP Business BASIC is not part of the HP 3000 fundamental operating system and must be purchased separately.

```
BBASIC [ commandfile ] [ , [ inputfile ] [ , listfile ] ]
```

BBASICGO

Compiles, prepares, and executes an HP Business BASIC program. HP Business BASIC is not part of the HP 3000 fundamental operating system and must be purchased separately.

```
BBASICGO infile [ , listfile ]
```

BBASICOMP

Compiles an HP Business BASIC program. HP Business BASIC is not part of the HP 3000 fundamental operating system and must be purchased separately.

```
BBASICOMP infile [ , [ uslfile ] [ , listfile ] ]
```

BBASICPREP

Compiles and prepares an HP Business BASIC program. HP Business BASIC is not part of the HP 3000 fundamental operating system and must be purchased separately.

BBASICPREP *infile* [, [*progfile*] [, [*listfile*]]

BREAKJOB

Suspends an executing job.

BREAKJOB #*Jnnn*

BUILD

Creates and immediately allocates a new empty file on disk.

BUILD *filereference* [;REC=[*resize*] [, [*blockfactor*] [, [{ F
U
V }] [, [{ BINARY
ASCII }]]]]]
[{ ;CCTL
;NOCCTL }] [;TEMP] [;DEV=[[*dsdevice*] #] [*device*]] [;CODE=*filecode*]
[;DISC=[*numrec*] [, [*numextents*] [, [*initialloc*]]]] [{ ;RIO
;NORIO }] [{ ;MSG
;CIR }]

BYE

Ends an interactive session.

BYE

CACHECONTROL

Tunes the performance of disk caching. Disk caching software is not included in the MPE V/E fundamental operating system and must be purchased separately.

$$\text{CACHECONTROL} \left\{ \begin{array}{l} \text{SEQUENTIAL}=\text{sequentn} \\ \text{;RANDOM}=\text{randn} \\ \text{;BLOCKWRITE}=\left\{ \begin{array}{l} \text{YES} \\ \text{NO} \end{array} \right\} \end{array} \right\} [; \dots]$$

CHANGELOG

Changes the user logging file without stopping or interrupting the logging process.

CHANGELOG *logid* [;DEV=*device*]

COBOL

Compiles a COBOL program on the COBOL 68 compiler. COBOL is not part of the HP 3000 fundamental operating system and must be purchased separately.

COBOL [*textfile*] [, [*uslfile*] [, [*listfile*] [, [*masterfile*] [, *newfile*]]]

COBOLGO

Compiles, prepares, and executes a COBOL program on the COBOL 68 compiler. COBOL is not part of the HP 3000 fundamental operating system and must be purchased separately.

COBOLGO [*textfile*] [, [*listfile*] [, [*masterfile*] [, *newfile*]]]

COBOLPREP

Compiles and prepares a COBOL program on the COBOL 68 compiler. COBOL is not part of the HP 3000 fundamental operating system and must be purchased separately.

```
COBOLPREP [ textfile ] [ , [ progfile ] [ , [ listfile ] [ , [ masterfile ] [ , newfile ] ] ] ]
```

COBOLII

Compiles a COBOLII program on the COBOL '74 compiler. COBOLII is not part of the HP 3000 fundamental operating system and must be purchased separately.

```
COBOLII [ textfile ] [ , [ uslfile ] [ , [ ; listfile ] [ , [ masterfile ] [ , newfile ] ] ] ] [ ; INFO=quotedstring ]
[ ; WKSP=workspacename ]
```

COBOLIIGO

Compiles, prepares, and executes a COBOLII program on the COBOL '74 compiler. COBOLII is not part of the HP 3000 fundamental operating system and must be purchased separately.

```
COBOLIIGO [ textfile ] [ , [ listfile ] [ , [ masterfile ] ] ] [ ; INFO=quotedstring ]
[ ; WKSP=workspacename ]
```

COBOLIIPREP

Compiles and prepares a COBOLII program on the COBOL '74 compiler. COBOLII is not part of the HP 3000 fundamental operating system and must be purchased separately.

```
COBOLIIPREP [ textfile ] [ , [ progfile ] [ , [ listfile ] [ , [ masterfile ] [ , newfile ] ] ] ]
[ ; WKSP=workspacename ] [ ; INFO=quotedstring ]
```

COMMENT

Inserts a comment into the command stream.

COMMENT [*text*]

CONSOLE

Changes the system console from its current device to another job-accepting terminal.

CONSOLE [*ldev*]

CONTINUE

Overrides job errors so that the job continues executing.

CONTINUE

DATA

Enters data into the system from any device file except \$STDIN.

DATA [*jsname*,] *username* [*/userpass*] . *acctname* [*/acctpass*] [; *filename*]

DEALLOCATE

Deallocates a program or procedure previously loaded into memory with the ALLOCATE command.

DEALLOCATE [PROGRAM,]
[PROCEDURE,] *name*

DEBUG

Invokes the MPE V/E debug facility.

DEBUG

DELETESPOOLFILE

Deletes a spool file from disk.

DELETESPOOLFILE $\left\{ \begin{array}{l} \#0nnn \\ \#Innn \\ ldev \end{array} \right\}$



DISALLOW

Prohibits access to a specific operator command.

DISALLOW FILE=*formaldesignator*[;SHOW] DISALLOW $\left\{ \begin{array}{l} @. @ \\ user. @ \\ @. acct \\ user. acct \end{array} \right\}$
;COMMANDS=*command*[, *command*, ...]

DISASSOCIATE

Removes control of a device from the user.

DISASSOCIATE *devclass*

DISCRPS

Enables or disables the rotational position sensing (RPS) feature on a specified logical device. It requires a special firmware upgrade CS-80 disk drive.

`DISCRPS ldev { ,ENABLE
 ,DISABLE }`

DISMOUNT

Causes a volume set that was explicitly mounted by the user to be dismounted.

`DISMOUNT [{ *
 vcname } [.groupname [.acctname]]]`

DOWN

Removes a device from normal system use.

`DOWN ldev`

DOWNLOAD

Downloads format information to a line printer.

`DOWNLOAD ldev [,filename
 ,MARGIN=nn] [, ...]`

DSTAT

Displays the current status of the disk drives on the system.

`DSTAT [ldev
 ALL]`

:EOF:

EDITOR

Runs EDIT/3000, which is used to create and manipulate ASCII files.

EDITOR [*listfile*]

ELSE

Provides an alternate execution sequence for an IF statement.

ELSE

ENDIF

Terminates an IF block.

ENDIF

:EOD

Denotes end-of-data on an input stream and terminates a set of data initialized by the DATA command.

:EOD

:EOF:

Simulates a physical end-of-file on an input stream from any device. The colons (:) are a required part of this command.

:EOF:

EOJ

Ends a batch job.

EOJ

FCOPY

Runs the FCOPY subsystem.

FCOPY [*fcopycommand*]

FILE

Declares the file attributes to be used when a file is opened. You may use this declaration, informally known as a file equation, to override programmatic or system default file specifications. With the addition of shared parameters from the NS/3000 AdvanceNet or the DS/3000 subsystems, the declaration may specify a formal file designator that you may use to access a remote file or device in a subsequent command or intrinsic. NS/3000 AdvanceNet and DS/3000 are not part of the HP 3000 fundamental operating system and must be purchased separately.

$$\text{FILE } \textit{formaldesignator} = \left\{ \begin{array}{l} * \textit{formaldesignator} \\ = \$\text{NULL} \\ = \$\text{NEWPASS} \\ = \$\text{OLDPASS} \\ = \$\text{STDIN} \\ = \$\text{STDINX} \\ = \$\text{STDLIST} \\ = \textit{filereference} [: \textit{nodespec}] [, \textit{filedomain}] \end{array} \right\}$$

[; DEV = [[*envname*] *] [*device*] [, *outpri*] [, *numcopies*]] [; VTERM]
[; ENV = *envfile* [: *nodespec*]] [; *option*] [; *access*] [; *disposition*]

FOREIGN

Directs MPE V/E to treat the disk volume mounted on the specified logical device as foreign.

FOREIGN *ldev*

FORTGO

Compiles, prepares, and executes a FORTRAN/3000 program. FORTRAN/3000 is not part of the HP 3000 fundamental operating system and must be purchased separately.

FORTGO [*textfile*] [, [*listfile*] [, [*masterfile*] [, *newfile*]]] [*;INFO=quotedstring*]

FORTPREP

Compiles and prepares a FORTRAN/3000 program. FORTRAN/3000 is not part of the HP 3000 fundamental operating system and must be purchased separately.

FORTPREP [*textfile*] [, [*progfile*] [, [*listfile*] [, [*masterfile*] [, *newfile*]]]] [*;INFO=quotedstring*]

FORTRAN

Compiles a FORTRAN/3000 program. FORTRAN/3000 is not part of the HP 3000 fundamental operating system and must be purchased separately.

FORTRAN [*textfile*] [, [*uslfile*] [, [*listfile*] [, [*masterfile*] [, *newfile*]]]] [*;INFO=quotedstring*]

FREERIN

Releases a global resource identification number (RIN).

FREERIN *rin*

FTN

Compiles a FORTRAN 77 program. FORTRAN 77 is not part of the HP 3000 fundamental operating system and must be purchased separately.

```
FTN [ textfile ] [ , [ ustfile ] [ , listfile ] ] [ ;INFO=quotedstring ]
```

FTNGO

Compiles, prepares, and executes a FORTRAN 77 program. FORTRAN 77 is not part of the HP 3000 fundamental operating system and must be purchased separately.

```
FTNGO [ textfile ] [ , listfile ] [ ;INFO=quotedstring ]
```

FTNPREP

Compiles and prepares a FORTRAN 77 program. FORTRAN 77 is not part of the HP 3000 Fundamental Operating System and must be purchased separately.

```
FTNPREP [ textfile ] , [ progfile ] [ , listfile ] [ ;INFO=quotedstring ]
```

FULLBACKUP

Performs a backup of MPE V/E and all the files on the system.

```
FULLBACKUP [ dumptape ] [ , auxlistfile ]
```

GETLOG

Establishes a user logging identifier on the system.

```
GETLOG logid ; LOG=logfile  $\left[ \begin{array}{c} \{ , \text{DISC} \\ , \text{TAPE} \\ , \text{SDISC} \\ , \text{CTAPE} \} \end{array} \right] \left[ ; \text{PASS}=\textit{password} \right] \left[ \begin{array}{c} \{ ; \text{AUTO} \\ ; \text{NOAUTO} \} \end{array} \right]$ 
```

GETRIN

Acquires a global resource identification number (RIN) and assigns a password to it.

GETRIN *rinpassword*

GIVE

Assigns a downed device to diagnostics.

GIVE *ldev*

HEADOFF

Stops header/trailer output to a device.

HEADOFF *ldev*

HEADON

Resumes header/trailer output to a device.

HEADON *ldev*

HELLO

Initiates an interactive session.

```

HELLO [sessionname,] username[/userpass] . acctname[/acctpass]
      [,groupname[/grouppass]] [;TIME=cpusecs] [;PRI=
      { BS
        CS
        DS
        ES } ]
      [ { ;INPRI=inputpriority } ] [;TERM={ termtype } ]
      [ ;HIPRI ] [;TERM={ termname } ]

```

HELP

Accesses the HELP subsystem.

HELP $\left[\begin{array}{l} \text{HELP} \\ \text{tablecontents} \\ \text{command} [[,] \text{keyword}] \\ \text{ALL} \\ \text{EXIT} \end{array} \right]$

IF

Used to control the execution sequence of a job.

IF [() *logezpr* ()] THEN

JOB

Identifies a job to be executed in conjunction with the STREAM command in batch mode.

JOB [*jobname* ,] *username* [/ *userpass*] . *acctname*
[/ *acctpass*] [, *groupname* [/ *grouppass*]] [; TIME = *cpusecs*]
 $\left[\begin{array}{l} \text{;PRI} = \left\{ \begin{array}{l} \text{BS} \\ \text{CS} \\ \text{DS} \\ \text{ES} \end{array} \right\} \end{array} \right] \left[\left\{ \begin{array}{l} \text{;INPRI} = \text{inputpriority} \\ \text{;HIPRI} \end{array} \right\} \right] [\text{;RESTART}]$
 $\left[\text{;OUTCLASS} = [\text{device}] [, \text{outputpriority} [, \text{numcopies}]] \right] \left[\text{;TERM} = \left\{ \begin{array}{l} \text{termtype} \\ \text{termname} \end{array} \right\} \right]$

JOBFENCE

Defines the minimum input priority that a job or session must have in order to execute.

JOBFENCE *priorityfence*

JOBPRI

Sets or changes the default execution priority for batch jobs and sets a maximum execution priority for batch jobs.

JOBPRI [*maxsubqueue*] [*,defaultsubqueue*]

JOBSECURITY

Designates what level of user may request resources and control the execution of jobs.

JOBSECURITY $\left\{ \begin{array}{l} \text{HIGH} \\ \text{LOW} \end{array} \right\}$

LDISMOUNT

Performs the logical dismount of a private volume set or class.

LDISMOUNT *vsname.group.account*

LIMIT

Limits the number of concurrently running jobs and/or sessions.

LIMIT $\left\{ \begin{array}{l} \text{numberjobs} \\ \text{, numbersessions} \\ \text{numberjobs, numbersessions} \end{array} \right\}$

LISTACCT

Lists the attributes for one account, with AM (account manager) capability, or for any or all accounts, with SM (system manager) capability.

LISTACCT $\left[\left\{ \begin{array}{l} @ \\ acct \end{array} \right\} \right] [, listfile]$

LISTEQ

Displays all active file equations for a job or session.

`LISTEQ [ listfile ]`

LISTF

Lists descriptions of one or more permanent disk files.

`LISTF [ fileset ]  $\left\{ \begin{array}{l} , 0 \\ , 1 \\ , 2 \\ , -1 \\ , -2 \end{array} \right\}$  [ ; listfile ]`

LISTFTEMP

Lists the descriptions of one or more temporary disk files for a job or session.

`LISTFTEMP [ fileset ] [ , listlevel ] [ ; listfile ]`

LISTGROUP

Prints a list of attributes for groups.

`LISTGROUP  $\left\{ \begin{array}{l} group.acct \\ group \\ @.acct \\ @ \\ @.@ \end{array} \right\}$  [ , listfile ]`

LISTLOG

Lists currently active user logging identifiers on the system and whether log file changing has been enabled.

```
LISTLOG [ logid [ ;PASS ] ]
```

LISTUSER

Lists attributes currently assigned to users.

```
LISTUSER { user.acct
            user
            @.acct
            @
            @.@
          } [ ,listfile ]
```

LISTVS

Produces a formatted listing of volume set definitions.

```
LISTVS [ vslist ] [ { { ,0
                       ,1
                       ,2
                     } } ] [ ;listfile ]
```

LMOUNT

Performs a logical mount of a private volume set.

```
LMOUNT vsname.group.account [ ;GEN=[genindex] ]
```

LOG

Starts, restarts, or stops user logging.

```
LOG logid      { ,RESTART  
                  ,START  
                  ,STOP }
```

=LOGOFF

Aborts all executing jobs and/or sessions and prevents any further logon.

```
=LOGOFF
```

=LOGON

Enables the processing of jobs and/or sessions following an =LOGOFF command.

```
=LOGON
```

MOUNT

Requests the system operator to mount a volume set.

```
MOUNT { *  
        vsname } [ .groupname [ .acctname ] ] [ ;GEN=[genindex] ]
```

NEWACCT

Creates a new account and an account manager and PUB group associated with it.

```
NEWACCT acctname,mgrname [ ;PASS=[password] ] [ ;FILES=[filespace] ]  
[ ;CPU=[cpu] ] [ ;CONNECT=[connect] ] [ ;CAP=[capabilitylist] ]  
[ ;ACCESS=[fileaccess] ] [ ;MAXPRI=[subqueuename] ] [ ;LOCATTR=[localattribute] ]  
[ ;VS=[volset:SPAN] ] [ ;USERPASS={ REQ  
                                   OPT }
```

Note: The USERPASS parameter is available only if the HP Security Monitor (HP30392) has been installed.

NEWGROUP

Creates a new group within an account.

```
NEWGROUP groupname [ .accountname ] [ ;PASS=[ password ] ] [ ;FILES=[ filespace ] ]
[ ;CPU=[ cpu ] ] [ ;CONNECT=[ connect ] ] [ ;CAP=[ capabilitylist ] ]
[ ;ACCESS=[ fileaccess ] ] [ ;VS=[ volset[:SPAN] ] ]
```

NEWUSER

Defines a new user.

```
NEWUSER username [ .acctname ] [ ;PASS=[ password ] ] [ ;CAP=[ capabilitylist ] ]
[ ;MAXPRI=[ subqueueuname ] ] [ ;LOCATTR=[ localattribute ] ]
[ ;HOME=[ homegroupname ] ] [ ;USERPASS={ REQ } [ , EXPIRED ] ]
```

Note: The USERPASS parameter is available only if the HP Security Monitor has been installed.

NEWVSET

Defines private volume sets and classes.

```
NEWVSET vsname;MEMBERS=vsname:type[ , vsname:type ... ]
[ ;CLASS=vcname:vsname[ , vsname ... ] ]
```

OPENQ

Opens the spool queue for a specified logical device or device class.

```
OPENQ { ldev
       devclass }
```

OUTFENCE

Defines the minimum priority that an output spool file needs in order to be printed.

`OUTFENCE outputpriority [ ;LDEV=ldev]`

PARTBACKUP

Performs a backup of MPE V/E and all of the files that have been modified since the last full backup.

`PARTBACKUP [ dumptape ] [ ,auxlistfile]`

PASCAL

Compiles a Pascal program. Pascal/3000 is not part of the HP 3000 fundamental operating system and must be purchased separately.

`PASCAL [ textfile ] [ , [ uslfile ] [ ,listfile ] ] [ ;INFO=quotedstring]`

PASCALGO

Compiles, prepares, and executes a Pascal/3000 program. Pascal is not part of the HP 3000 fundamental operating system and must be purchased separately.

`PASCALGO [ textfile ] [ ,listfile ] [ ;INFO=quotedstring]`

PASCALPREP

Compiles and prepares a Pascal/3000 program. Pascal is not part of the HP 3000 fundamental operating system and must be purchased separately.

`PASCALPREP [ textfile ] [ ,progfile ] [ ,listfile ] [ ;INFO=quotedstring]`

PASSWORD

Allows you to establish or change your own password.

PASSWORD

PREP

Prepares a program from a user subprogram library (USL) file into a program file.

```
PREP uslfile, progrid [ ;ZERODB ] [ ;CAP=capabilitylist ] [ ;PMAP ] [ ;RL=filename ]
[ ;MAXDATA=segsz ] [ ;PATCH=patchsize ] [ ;STACK=stacksize ] [ ;NOSYM ] [ ;DL=dlsz ]
[ { ;FPMAP } ]
[ { ;NOFPMAP } ]
```

PREPRUN

Prepares and executes a compiled program.

```
PREPRUN uslfile, entrypoint [ ;NOPRIV ] [ ;RL=filename ] [ ;PMAP ] [ ;NOCB ]
[ ;DEBUG ] [ ;INFO=quotestring ] [ ;LMAP ] [ ;STDIN= { *formaldesig } ] [ ;ZERODB ]
[ ;STDLIST= { *formaldesig } ] [ ;MAXDATA=segsz ] [ ;PATCH=patchsize ]
[ ;PARM=parameternum ] [ ;NOSYM ] [ ;STACK=stacksize ] [ { ;FPMAP } ]
[ ;DL=dlsz ] [ ;LIB= { G } ] [ ;CAP=capabilitylist ]
[ ;P ]
[ ;S ]
```

PTAPE

Reads a paper tape without X-OFF control.

PTAPE filename

PURGE

Deletes a file from the system.

PURGE filereference [,TEMP]

PURGEACCT

Removes an account and its groups and users from the system directory or from the specified volume set's directory.

PURGEACCT acctname [;VS=*volset*]

PURGEGROUP

Removes a group (and all files belonging to it) from the system, or from the specified volume set directory.

PURGEGROUP groupname [.*accountname*] [;VS=*volset*]

PURGEUSER

Removes a user from an account.

PURGEUSER user [.*accountname*]

PURGEVSET

Deletes an existing volume set.

PURGEVSET *vsname*

RECALL/=RECALL

Displays all pending console **REPLY** messages.

RECALL

REDO

Allows you to edit a command entry.

REDO

REFUSE

Disables jobs, sessions and/or data on a designated device.

REFUSE $\left[\begin{array}{l} \text{JOBS,} \\ \text{DATA,} \end{array} \right] [ldev]$

RELEASE

Removes all security provisions for a file with the exception of any access control definitions (ACDs). The file remains unsecured until the **SECURE** command is issued.

RELEASE *filereference*

RELLOG

Removes a logging identifier from the system.

RELLOG *logid*

RENAME

Changes the identity (filename, lockword, and/or group name) of a disk file.

RENAME *oldfilereference, newfilereference* [,TEMP]

REPLY/=REPLY

Replies to pending resource requests at the console.

REPLY *pin* ,*reply*

REPORT

Displays accounting information for the logon account and group.

REPORT [*groupset*] [,*listfile*] [;VS=*volset*]

RESET

Cancels file equations.

RESET { *formaldesignator* }
 @

RESETACCT

Zeroes the running counts of CPU time or connect time accumulated by an account and by all groups within that account.

```
RESETACCT [ [ @ ] [ [ CPU ] ] ]
           [ [ acct ] [ , [ CONNECT ] ] ]
```

RESETDUMP

Disables the stack dump facility.

```
RESETDUMP
```

RESTORE

Returns files that have been stored on magnetic tape or serial disk to the system.

```
RESTORE[restorefile] [ ; { fileset[-fileset] [ , fileset[-fileset] ] [ , ... ] } ]
                    [ ; !indirectfile ]
[ ; DEV=device ] [ ; SHOW=[ showparm [ , showparm [ , ... ] ] ] ] [ ; FILES=maxfiles ]
[ ; { LOCAL
  ; GROUP=groupname } [ ; ... ] ] [ ; CREATE [ = { GROUP
  ; ACCT=acctname } [ ; ... ] ] ] [ ; { ACCT
  ; CREATOR } [ , ... ] ] ]
[ ; CREATOR [=username] ] [ ; { KEEP
  ; NOKEEP } ] [ ; { OLDDATE
  ; NEWDATE } ] [ ; ONERR[or]= { QUIT
  ; SKIP } ] ]
[ RESTORESET=tape|listspec ] [ ; STARTHERE ] [ ; COPYACD ]
```

RESUME

Resumes execution of a suspended operation.

```
RESUME
```

RESUMEJOB

Resumes a suspended job.

RESUMEJOB #Jnnn

RESUMELOG

Resumes system logging following suspension caused by an error.

RESUMELOG

RESUMESPOOL

Resumes suspended spooler output to a spooled device.

RESUMESPOOL ldev;BACK $\left[\begin{array}{l} \text{nnn FILES} \\ \text{nnn PAGES} \end{array} \right]$

RESUMESPOOL ldev;FORWARD $\left[\begin{array}{l} \text{nnn FILES} \\ \text{nnn PAGES} \end{array} \right]$

RESUMESPOOL ldev;BEGINNING

RPG

Compiles an RPG program. RPG is not part of the HP 3000 fundamental operating system and must be purchased separately.

RPG [*textfile*] [, [*uslfile*]] [, [*listfile*]] [, [*masterfile*]] [, [*newfile*]]]]

RPGGO

Compiles, prepares, and executes an RPG program. RPG is not part of the HP 3000 fundamental operating system and must be purchased separately.

```
RPGGO [ textfile ] [ , [ listfile ] [ , [ masterfile ] [ , newfile ] ] ]
```

RPGPREP

Compiles and prepares an RPG program. RPG is not part of the HP 3000 fundamental operating system and must be purchased separately.

```
RPGPREP [ textfile ] [ , [ progfile ] [ , [ listfile ] [ , masterfile ] [ , newfile ] ] ]
```

RUN

Executes a prepared program.

```
RUN progfile[ , entrypoint ] [ ;NOPRIV ] [ ;LIB= { G
P
S } ] [ ;LMAP ] [ ;NOCB ] [ ;DEBUG ]
[ ;INFO=quotestring ] [ ;MAXDATA=segsz ] [ ;STDIN= { *formaldesig
fileref
$NULL } ] ]
[ ;PARM=parameternum ] [ ;STDLIST= { *formaldesig
fileref [ ,NEW ]
$NULL } ] ] [ ;STACK=stacksize ]
[ ;DL=dlsz ]
```

SAVE

Saves a file in the permanent system file domain.

```
SAVE { $OLDPASS, newfilereference }
      { tempfilereference }
```

SECURE

Restores security provisions for a file that were previously suspended by a RELEASE command.

SECURE filereference

SEGMENTER

Calls the MPE Segmenter.

SEGMENTER [listfile]

SET

Allows a job using a spooled \$STDLIST to mark its standard list device for deletion when the job terminates.

SET STDLIST= $\left\{ \begin{array}{l} \text{DELETE} \\ \text{SAVE} \end{array} \right\}$

SETCATALOG

Causes the command interpreter to search a catalog of user- defined commands (UDCs) and establish an entry for each command, or to deactivate previously set catalogs.

SETCATALOG [catfilename[/lockword] [, catfilename[/lockword]] ...] ... [;SHOW]
[; { SYSTEM }] [;USER=user[.acct]]
[; { ACCOUNT }]

SETDUMP

Enables the stack dump facility and initiates it at the time of an abort.

SETDUMP $\left[\left\{ \begin{array}{l} \text{DB} \\ \text{ST} \\ \text{QS} \end{array} \right\} [, \dots] \right] [; \text{ASCII}]$

SETJCW

Creates or assigns a value to a job control word (JCW).

$$\text{SETJCW } jcwname \text{ delimiter value } \left[\left\{ \begin{array}{c} + \\ - \end{array} \right\} value \right]$$

SETMSG

Enables or disables the receipt of user or operator messages at the standard list device.

$$\text{SETMSG } \left\{ \begin{array}{c} \text{OFF} \\ \text{ON} \end{array} \right\}$$

SHOWALLOCATE

Displays status information about ALLOCATE and/or AUTOALLOCATE.

$$\text{SHOWALLOCATE } \left[\begin{array}{l} \text{STATUS } [, listfile] \\ \text{ALLOCATE } \left[\begin{array}{l} , fileset, listfile \\ \text{AUTO } , fileset, listfile \\ \text{ALL} \end{array} \right] , fileset, listfile \end{array} \right]$$

SHOWALLOW

Displays the operator commands that have been allowed.

$$\text{SHOWALLOW } \left\{ \begin{array}{l} user.acct \\ user.@ \\ @.acct \\ @.@ \end{array} \right\}$$

SHOWCACHE

Displays a summary of disk caching performance. Disk caching software is not included in the fundamental operating system, and must be purchased separately.

SHOWCACHE

SHOWCATALOG

Lists user-defined command (UDC) files.

SHOWCATALOG [*listfile* ;USER=*user*[*.acct*]]

SHOWCOM

Displays status information about a communication device.

SHOWCOM *ldev*[;ERRORS] [;RESET]

SHOWDEV

Reports the status of input/output devices and information about any associated access control definitions (ACDs).

SHOWDEV [*ldev*
 classname] [;ACD]

SHOWIN

Reports the status of input device files.

SHOWIN [#*Innn*
 STATUS
 SP
 item[; *item*[; ...]]]

SHOWJCW

Displays the current state of one or more job control words (JCWs).

SHOWJCW [*jcwname*]

SHOWJOB

Displays the status information about jobs and/or sessions.

SHOWJOB $\left[\begin{array}{l} [\#] Snnn \\ [\#] Jnnn \\ STATUS \\ SCHED \\ item[; item[; \dots]] \end{array} \right] [; *listfile]$

SHOWLOG

Displays the number of the system's current log file and the percentage of disk space used.

SHOWLOG

SHOWLOGSTATUS

Displays status information about currently opened user logging files assigned to a logging identifier.

SHOWLOGSTATUS [*logid*]

SHOWME

Reports the status of your job or session.

SHOWME

SHOWOUT

Displays the status of output device files.

SHOWOUT $\left\{ \begin{array}{l} \#nnn \\ \text{STATUS} \\ \text{SP} \\ \text{item[;item[; ...]]} \end{array} \right\}$

SHOWQ

Displays process scheduling data and the contents of each subqueue.

SHOWQ

SHOWTIME

Prints current time and date.

SHOWTIME

=SHUTDOWN

Initiates a shutdown of MPE V/E.

=SHUTDOWN

SHUTQ

Closes the spool queue for the specified logical device or device class.

$$\text{SHUTQ} \left\{ \begin{array}{l} \text{lddev} \\ \text{devclass} \end{array} \right\}$$

SPEED

Changes the operating speed of the terminal.

SPEED *newinspeed, newoutspeed*

SPL

Compiles an SPL program. SPL is not part of the HP 3000 fundamental operating system and must be purchased separately.

SPL [*textfile*] [, [*usfile*]] [, [*listfile*]] [, [*masterfile*]] [, [*newfile*]]]]
[; *INFO=quotedstring*]

SPLGO

Compiles, prepares, and executes an SPL program. SPL is not part of the HP 3000 fundamental operating system and must be purchased separately.

SPLGO [*textfile*] [, [*listfile*]] [, [*masterfile*]] [, [*newfile*]]]] [; *INFO=quotedstring*]

SPLPREP

Compiles and prepares an SPL program. SPL is not part of the HP 3000 fundamental operating system and must be purchased separately.

SPLPREP [*textfile*] [, [*progfile*]] [, [*listfile*]] [, [*masterfile*]] [, [*newfile*]]]]
[; *INFO=quotedstring*]

STARTCACHE

Enables caching on a single disk. Disk caching software is not included with the fundamental operating system and must be purchased separately.

`STARTCACHE ldev`

STARTSESS

Creates a session on the specified device, provided that the command's user has programmatic sessions (PS) capability.

```
STARTSESS ldev [ sessionname, ] user [ /userpass ] .acct [ /acctpass ]  
  
[ , group [ /grouppass ] ] [ ;TERM={ termtype  
                                     termname } ] [ ;TIME=cpusecs ] [ ;PRI={ BS  
                                     CS  
                                     DS  
                                     ES } ]  
  
[ { ;INPRI=inputpriority } ] [ ;NOWAIT ]
```

STARTSPOOL

Initiates the spooler process for a device.

```
STARTSPOOL { ldev [ ;SHUTQ ] }  
            { devclass }
```

STOPCACHE

Disables caching on a single disk. Disk caching software is not included in the fundamental operating system, and must be purchased separately.

`STOPCACHE ldev`

STOPSPPOOL

Terminates spooling to a specified device or device class.

```
STOPSPPOOL { ldev[;OPENQ] }
            { devclass }
```

STORE

Copies disk files onto magnetic tape or serial disk. Use the RESTORE command to copy files back to disk.

```
STORE [ { fileset[-filesset][,filesset[-filesset]] [, ... ] } ] [ { ;*storefile } ]
      [ ;!indirectfile ]
      [ ;SHOW[=showparm[,showparm[, ... ]]] ] [ ;FILES=mazfiles ]
      [ ;DATE[ { <=accdate } ] ] [ ;ONERR[OR]={ QUIT } ] [ ;PURGE ]
      [ ;PROGRESS[=#minutes] ] [ ;STORESET=tapelist-spec ] [ ;INTER[leave] ]
      [ ;COPYACD ]
```

STREAM

Spools batch jobs or data from a session or job. The optional time-related parameters of the STREAM command may be used to schedule jobs.

```
STREAM filename [,char] [ ;AT=timespec ] [ ;DAY={ day-of-week
                                                    day-of-month
                                                    days-until-month } ]
      [ ;DATE=mm/dd/yy ] [ ;IN= [ days ] [,hours[,minutes] ] ]
```

STREAMS

Enables or disables the streams device: it permits users to submit job and/or data streams, or prevents them from doing so.

```
STREAMS { ldev }
         { OFF }
```

SUSPENDSPOOL

Suspends output to a spooled device.

```
SUSPENDSPOOL ldev [ ;FINISH ]
```

SWITCHLOG

Closes the current system log file, then creates and opens a new one.

```
SWITCHLOG
```

SYSDUMP

Starts the configurator dialog, and copies all system and user files to magnetic tape or serial disk and creates a backup of your system, user files, and configuration data.

```
SYSDUMP [  $\begin{matrix} \$\text{NULL} \\ \text{dumpfile} \end{matrix}$  ] [ , auxlistfile ]
```

TAKE

Releases a device previously assigned to diagnostics so that it may be brought online.

```
TAKE ldev
```

TELL

Sends a message to another session.

```
TELL  $\left\{ \begin{matrix} [ \# ] Snnn \\ [ sessionname, ] username.acctname \\ @ \\ @.acctname \\ @S \end{matrix} \right\} [ text ]$ 
```

TELOP

Sends a message to the system console.

TELOP [*text*]

TUNE

Changes the filter and/or priority limits of circular subqueues. It also changes the value of the memory manager algorithm threshold.

TUNE [*minclockcycle*]; $\left\{ \begin{array}{l} \text{CQ} \\ \text{DQ} \\ \text{EQ} \end{array} \right\} = [\text{base}[, \text{limit}[, \text{min}[, \text{maz}[; \dots]]]]]$

UP

Returns a particular device to its normal function on the system by cancelling a DOWN command that had been issued for the device.

UP *ldev*

VINIT

Runs the VINIT subsystem to format, initialize, and label private volumes, serial disks, and foreign disks. The prompt in the VINIT subsystem is ">".

VINIT [*listdevice*]

VMOUNT

Enables or disables MPE V/E's private volumes facility.

VMOUNT $\left\{ \begin{array}{l} \text{ON} [, \text{AUTO}] \\ \text{OFF} \end{array} \right\} [; \text{ALL}]$

VSUSER

Lists all users of a currently mounted, nonsystem volume set.

VSUSER [*vsname*]

WARN

Sends an urgent message to jobs or sessions.

$$\text{WARN} \left\{ \begin{array}{l} \text{q} \\ \left[\begin{array}{l} \# \\ \# \end{array} \right] \text{Jnn} \\ \left[\begin{array}{l} \# \\ \# \end{array} \right] \text{Snnn} \\ \text{jsname, } \text{user.acct} \end{array} \right\} [; \text{message}]$$

WELCOME

Defines the welcome message.

WELCOME [*welcfile*]



File System

System Defined Files

MPE reserves certain file designators for system defined files. System defined files are reserved words that refer to a specific type of system file.

\$STDIN	Refers to the device that you used to initiate your current session or job. The device is normally a terminal for a session and spoolfile for a job. Data entries in this file should not have a colon in column 1. (A colon in column 1 indicates the end-of-data). Use the :EOD command to delimit data.
\$STDINX	Is the same as \$STDIN, except that a colon in the first column does not indicate the end of data. Thus \$STDINX may contain commands as well as data. Interactive programs and subsystems often use \$STDINX to reference the terminal as an input file. Use :EOD or :EOF: to indicate the end of data.
\$STDLIST	Is the device designated as the session or job output device, the device MPE uses to respond to your commands. This device is normally a terminal for sessions and line printer for jobs.
\$NULL	Is a file designator that is used to tell MPE to read from or write to a nonexistent file as though the I/O operation were successful. This file is usually used to discard output.
\$NEWPASS	Is a temporary disk file. MPE uses \$NEWPASS to store information during the execution of a program. When a program closes \$NEWPASS, the system automatically changes its name to \$OLDPASS.
\$OLDPASS	Is a temporary disk file containing the contents of the last \$NEWPASS file closed. When a \$NEWPASS file is renamed \$OLDPASS, the system deletes the previous version of \$OLDPASS.

You can use \$NEWPASS and \$OLDPASS when compiling and preparing a program. MPE compilers write object code to \$NEWPASS during compilation. When compilation is complete, the compiler closes \$NEWPASS and the system renames the object code (USL) file \$OLDPASS. When you prepare the USL file (\$OLDPASS), the system stores prepared (executable) code in the \$NEWPASS file. When preparation is complete, the system closes \$NEWPASS and renames the executable code file \$OLDPASS. Use the SAVE command to save the program stored in \$OLDPASS to a permanent file.

File Codes

Table 2-1. File System File Codes

Integer	Mnemonic	Description
1024	USL	User Subprogram Library
1025	BASD	BASIC Data
1026	BASP	BASIC Program
1027	BASFP	BASIC Fast Program
1028	RL	Relocatable library
1029	PROG	Program File
1031	SL	Segmented Library
1035	VFORM	View Form File
1036	VFAST	View Fast Forms File
1037	VREF	View Reformat File
1040	XL SAV	Cross Loader ASCII File (SAVE)
1041	XLBIN	Cross Loader Relocated Binary File
1042	XL DSP	Cross Loader ASCII File (DISPLAY)
1050	EDITQ	Edit Quick File
1051	EDTCQ	Edit KEEPQ File (COBOL)
1052	EDTCT	Edit TEXT File (COBOL)
1054	TDPDT	TDP Diary File
1055	TDPQM	TDP Proof Marked QMARKED
1056	TDPP	TDP Proof Marked non-COBOL File
1057	TDPCP	TDP Proof Marked COBOL File
1058	TDPQ	TDP Workfile
1059	TDPXQ	TDP Workfile (COBOL)
1060	RJEPN	RJE Punch File
1070	QPROC	QUERY Procedure File
1080	KSAMK	KSAM Key File
1083	GRAPH	GRAPH Specification File
1084	SD	Self-describing File
1090	LOG	User Logging Log File
1100	WDOC	HPWORD Document
1101	WDICT	HPWORD Hyphenation dictionary
1102	WCONF	HPWORD Configuration File
1103	W2601	HP 2601 Environment File
1110	PCELL	IDS/3000 Character Cell File
1111	PFORM	IDS/3000 Form File
1112	PENV	IFS/3000 Environment File
1113	PCCMP	IFS/3000 Compiled Character Cell File
1114	RASTR	Graphics Image in raster Format
1130	OPTLF	OPT/3000 Log File
1131	TEPES	TEPE/3000 Script File
1132	TEPEL	TEPE/3000 Log File
1133	SAMPL	APS/3000 Log File
1139	MPEDL	MPEDCP/DRP Log File
1140	TSR	HPToolset Root File
1141	TSR	HPToolset Data File
1145	DRAW	Drawing File for HPDRAW
1146	FIG	Figure File for HPDRAW
1147	FONT	Reserved
1148	COLOR	Reserved
1149	D48	Reserved

2-2 File System

Table 2-1. File System File Codes (continued)

Integer	Mnemonic	Description
1152	SLATE	Compressed SLATE File
1153	SLATW	Expanded SLATE Workfile
1156	DSTOR	Store File for RAPID/3000 Utility DICTDBU
1157	TCODE	Code File for Transact/3000 Compiler
1158	RCODE	Code File for Report/3000 Compiler
1159	ICODE	Code File for Inform/3000 Compiler
1166	MDIST	HPDESK Distribution list
1167	MTEXT	HPDESK Text
1168	MARPA	ARPA Message File
1169	MARPD	ARPA Distribution List
1170	MCMND	HPDESK Abbreviated Commands File
1171	MFRM	Reserved
1172		Reserved
1173	MEFT	Reserved
1174	MCRPT	Reserved
1175	MSERL	Reserved
1176	UCSF	Reserved
1177	TTYPE	Term Type File
1178	TVFC	Term Vertical Format Control File
1192	NCONF	Network Configuration File
1193	NTRAC	Network Trace File
1194	NLOG	Network Log File
1195	MIDAS	Reserved
1211	ANODE	Reserved
1212	INODE	Reserved
1213	INVRT	Reserved
1214	EXCEP	Reserved
1215	TAVON	Reserved
1216	QUERF	Reserved
1217	DOCDR	Reserved
1226	VC	VC File
1227	DIF	DIF File
1228	LANGD	Language Definition File
1229	CHARD	Character Set Definition File
1230	MGCAT	Formatted Application Message Catalog
1236	BMAP	Reserved
1242	BDATA	BASIC Data
1243	BFORM	BASIC Field Order for VPLUS
1244	BSAVE	BASIC Saved Program File
1245	BCNFG	BASIC Configuration File for Default Options
1258	PFSTA	Pathflow STATIC File
1259	PFDYN	Pathflow DYNAMIC File
1270	RTDCA	Revisable Form DCA Document (DCA = Document Content Architecture)
1271	FFDCA	Final Form DCA Document
1272	DIV	Document Interchange Unit File
1273	PDOC	HPWORD/150 Document
1401	CWPTX	Reserved
1421	MAP	HPMAP/3000 Map Specification File
1422	GAL	Reserved
1425	TTX	Reserved
3333		Reserved

Carriage Control Directives

Table 2-2. Carriage Control Directives

Octal Code	ASCII Symbol	Carriage Action
%40	" "	Single space (with or without automatic page eject).
%53	"+"	No space, return (next printing at column 1). This cannot be used more than once on the 2608 A/S without losing data.
%55	"_"	Triple space (with or without automatic page eject).*
%60	"0"	Double space (with or without automatic page eject).*
%61	"1"	Conditional page eject (form feed) is performed by the software. If the printer is not already at the top of the form, a page eject is performed. Ignored if: Post-space mode: The current request has a transfer count of 0 and the previous request was FOPEN, FCLOSE, or FWRITE, which specified a carriage control directive of %61. Pre-space mode: Both the current request and the previous request have transfer counts of 0, and the current request and previous request are any combination of FOPEN, FCLOSE, or FWRITE specifying a carriage control of %61.
%62		Skip to one line before top of form. This specification is valid only for the HP 2608S and 2563A printers.
%63		A conditional page eject form (form feed) is performed by the printer. If the printer is not already at the top of form, perform a page eject. This specification is valid only for the HP 2608S and 2563A printers.
%2nn		Space nn lines (no automatic page eject); nn is any octal number from 0 through 77.*
%300-%313		Select VFC Channel 1 - 12 (HP 2613, 2617, 2618, 2619).
%300-%317		Select VFC Channel 1 - 16 (HP 2608 A/S).

Table 2-2. Carriage Control Directives (continued)

Octal Code	ASCII Symbol	Carriage Action
%300		Skip to top of form (page eject).
%301		Skip to bottom of form.
%302		Single spacing with automatic page eject.
%303		Skip to next odd line with automatic page eject.*
%304		Skip to next third line with automatic page eject.*
%305		Skip to next 1/2 page.
%306		Skip to next 1/4 page.
%307		Skip to next 1/6 page.
%310		Skip to bottom of form.
%311		User option (HP 2613/17/18/19), skip to one line before bottom of form (HP 2608 A/S).
%312		User option (HP 2613/17/18/19), skip to one line before top of form (2608 A/S).
%313		User option (HP 2613/17/18/19), skip to top of form (HP 2608A).
%314		Skip to next seventh line with automatic page eject.
%315		Skip to next sixth line with automatic page eject.
%316		Skip to next fifth line with automatic page eject.
%317		Skip to next fourth line with automatic page eject.
%320		No space, no return (next printing physically follows this).

Table 2-2. Carriage Control Directives (continued)

Octal Code	ASCII Symbol	Carriage Action
%2 - %37		
%41 - %52		
%54		
%56-%57		Same as %40.
%62-%77		
%104-%177		
%310-%317		(HP 2607)
%314-%317		(HP 2613/17/18/19)
%321-%377		
%400 or %100		Sets post-space movement option; this first prints, then spaces. If previous option was pre-space movement, the driver outputs a line with a skip to VFC Channel 3 (automatic page eject in effect) or a one line advance (equivalent to an octal code of %201 without automatic page eject) to clear the buffer.
%401 or %101		Sets pre-space movement option; this first spaces, then prints.
%402 or %102		Sets single-space option, with automatic page eject (60 lines per page).
%403 or %103		Sets single-space option, without automatic page eject (66 lines per page).

Note



The HP2680 laser printer, uses different codes from other printers, in some cases. Use code %202 to produce double spacing (without automatic page eject) and %203 to produce triple spacing (without automatic page eject). Codes %55 and %60 produce the same results as %303 and %304 - skip to next odd line with automatic page eject.

Carriage Control Effect Summary

FOPEN OR :FILE	FWRITE Control Parameter		
	= 0	= 1	= Greater than 1
Carriage Control Option or CCTL	Resize = 133 Byte 1 0 record = 132 Data output contains 132 characters; the prefix byte is added and contains 0.	Resize = 132 record = 132 Data output contains 132 characters; the carriage control character in the first byte is not printed if output is to a list device.	Resize = 133 Byte 1 Con- trol record = 132 Data output contains 132 characters; the prefix character added is a carriage control character specified by the FWRITE control parameter.
	132 record = 132 Data output contains 132 characters.	132 record = 132 Data output contains 132 characters.	132 record = 132 Data output contains 132 characters.

Effect on Data Output

LQ200154_001

Figure 2-1. Carriage Control Data

File Access and Security

Table 2-3. File Access and Security Descriptions

Mnemonic	File Access (Mode)	Mnemonic	Security (User)
R	Read	ANY	Any user
L	Lock (allows exclusive access to the file)	AC	Member of this account only
A	Append (implicitly specifies L also)	GU	Member of this group only
W	Write (implicitly specifies A and L also)	AL	Account librarian user only
X	Execute	GL	Group librarian user only
S	Save	CR	Creating user only

Default Security for Accounts, Groups, and Users

Table 2-4. Default Security

Type	Access Permitted
SYS account	(R,X:ANY;W,A,L:AC)
Accounts other than SYS	(R,X,W,A,L:AC)
PUB groups in any account	(R,X:ANY;A,W,L,S:AL,GU)
Groups other than PUB	(R,X,S,W,A,L:GU)
Users with no security specified	(R,X,W,A,L:ANY)

Net Default Access to Files

Table 2-5. File Default Security

File Reference	File	Access Permitted	Save Access To Group
<i>Filename .PUB.SYS</i>	Any file in public (PUB) group system (SYS) account.	(R,X:ANY;W,A,L)	AL, GU
<i>Filename .grp.SYS</i>	Any file in any group in SYS account.	(R,W,X,A,L:GU)	GU
<i>Filename .PUB.acct</i>	Any file in PUB group of any account.	(R,X:AC;W,A,L:AL, GU)	AL, GU
<i>Filename .grp.acct</i>	Any file in any group in any account.	(R,W,A,L,X:GU)	GU

Capabilities

Table 2-6. Capabilities

Capability	Mnemonic	Capability	Mnemonic
System Manager	SM	Use Private Volumes	UV
Account Manager	AM	Create Volumes	CV
Account Librarian	AL	Use Communication	
Group Librarian	GL	Software	CS
Diagnostician	DI	Programmatic Sessions	PS
System Supervisor	OP	User Logging	LG
Network Administrator	NA	Process Handling	PH
Node Manager	NM	Extra Data Segments	DS
Permanent Files	SF	Multiple RINs	MR
Access of Nonshareable I/O devices	ND	Privileged Mode	PM
		Interactive Access	IA
		Batch Access	BA

Default Capabilities

Table 2-7. Default Capabilities

Type	Capabilities
Accounts	AM,AL,GL,SF,ND,IA,BA
Groups	IA,BA
Users	SF,ND,IA,BA

FOPEN Options

FOPEN OPTIONS

Bits	(0:2)	(2:3)	(5:1)	(6:1)	(7:1)	(8:2)	(10:3)	(13:1)	(14:2)
Field	Reserved	File Type	Disallow :FILE	MPE Tape Labels	Carriage Control	Record Format	Default File Designator	ASCII BINARY	Domain
Meaning		00 0 = STD 01 1 = KSAM 10 0 = RQ 11 0 = CIR 0 = MSG	0 = Allow :FILE 1 = No :FILE	0 = Non-Labeled Tape 1 = Labeled Tape	0 = NOCCTL 1 = CCTL	00 = Fixed 01 = Variable 10 = Undefined 11 = Spoolfile	000 = FILENAME 001 = \$STDLIST 010 = \$NEWPASS 011 = \$OLDPASS 100 = \$STDIN 101 = \$STDINX 110 = \$NULL	0 = BINARY 1 = ASCII	00 = New File 01 = Old Permanent File 10 = Old Temporary File 11 = Old Permanent or Temporary File

LG200154_002

NOTE: Double lines indicate octal digit boundaries

Figure 2-2. Options Bit Summary

FOPEN Options

FOPEN OPTIONS

Bits	(0:3)	(3:1)	(4:1)	(5:2)	(7:1)	(8:2)	(10:1)	(11:1)	(12:4)
Field	Reserved	File Copy Access	NOWAIT I/O	Multi-Access	Inhibit Buffering	Exclusive Access	Dynamic Locking	Multi-Record Access	Access Type
Meaning		0 = Access Native Mode 1 = Access as Standard Sequential File	1 = NOWAIT 0 = Non-NOWAIT	00 = Non Multi-Access 01 = Only Intra-Job Multi-Access 10 = Inter-Job Multi-Access Allowed	0 = BUF 1 = NOBUF	00 = Default 01 = Exclusive 10 = Semi-exclusive Access Read 11 = Share	0 = No FLOCK Allowed 1 = FLOCK Allowed	0 = No Multi-Record 1 = Multi-Record	000 = Read only 001 = Write only 010 = Write (Save) only 011 = Append only 100 = Read/Write 101 = Update 110 = Execute

NOTE: Double lines indicate octal digit boundaries

L0200154_003

Figure 2-3. Aoptions Bit Summary

Table 2-8. File System Error Messages

VALUE	DESCRIPTION
0	END OF FILE. (FSERR 0)
1	ILLEGAL DB REGISTER SETTING. (FSERR 1)
2	ILLEGAL CAPABILITY. (FSERR 2)
3	REQUIRED PARAMETER IS MISSING. (FSERR 3)
4	DISK FREE SPACE ALLOCATION DISABLED ON ALL DISCS IN DOMAIN. (FSERR 4)
5	DRT NUMBER > 511. (FSERR 5)
6	DEVICE HAS NO AVAILABLE SPARE BLOCKS. (FSERR 6)
7	UNFORMATTED OR UNINITIALIZED MEDIA ON DEVICE. (FSERR 7)
8	ILLEGAL PARAMETER VALUE. (FSERR 8)
9	INVALID FILE TYPE SPECIFIED IN FOPTIONS. (FSERR 9)
10	INVALID RECORD SIZE SPECIFICATION. (FSERR 10)
11	INVALID RESULTANT BLOCK SIZE. (FSERR 11)
12	RECORD NUMBER OUT OF RANGE. (FSERR 12)
13	CAN'T OPEN FILE MULTI-ACCESS, OUT OF FMAVT ENTRIES. (FSERR 13)
16	MORE THAN 255 OPENS OF A FILE. (FSERR 16)
17	MAGNETIC TAPE RUNAWAY. (FSERR 17)
18	DEVICE POWERED UP. (FSERR 18)
19	FORMS CONTROL WAS RESET. (FSERR 19)
"	INVALID OPERATION. (FSERR 20)
!	DATA PARITY ERROR. (FSERR 21)
"	SOFTWARE TIME-OUT. (FSERR 22)
#	END OF TAPE. (FSERR 23)
\$	UNIT NOT READY. (FSERR 24)
%	NO WRITE-RING ON TAPE. (FSERR 25)
&	TRANSMISSION ERROR. (FSERR 26)
'	I/O TIME-OUT. (FSERR 27)
(	TIMING ERROR OR DATA OVERRUN. (FSERR 28)
)	SIO FAILURE. (FSERR 29)
'0'	UNIT FAILURE. (FSERR 30)
'1'	END OF LINE. (FSERR 31)
'2'	SOFTWARE ABORT. (FSERR 32)
'3'	DATA LOST. (FSERR 33)
'4'	UNIT NOT ON-LINE. (FSERR 34)
'5'	DATA-SET NOT READY. (FSERR 35)
'6'	INVALID DISK ADDRESS. (FSERR 36)
'7'	INVALID MEMORY ADDRESS. (FSERR 37)
'8'	TAPE PARITY ERROR. (FSERR 38)
'9'	RECOVERED TAPE ERROR. (FSERR 39)

Table 2-8. File System Error Messages (continued)

VALUE	DESCRIPTION
'@'	OPERATION INCONSISTENT WITH ACCESS TYPE. (FSERR 40)
'A'	OPERATION INCONSISTENT WITH RECORD TYPE. (FSERR 41)
'B'	OPERATION INCONSISTENT WITH DEVICE TYPE. (FSERR 42)
'C'	WRITE EXCEEDS RECORD SIZE. (FSERR 43)
'D'	UPDATE AT RECORD ZERO. (FSERR 44)
'E'	PRIVILEGED FILE VIOLATION. (FSERR 45)
'F'	OUT OF DISK SPACE. (FSERR 46)
'G'	I/O ERROR ON FILE LABEL. (FSERR 47)
'H'	INVALID OPERATION DUE TO MULTIPLE FILE ACCESS. (FSERR 48)
'I'	UNIMPLEMENTED FUNCTION. (FSERR 49)
'P'	NONEXISTENT ACCOUNT. (FSERR 50)
'Q'	NONEXISTENT GROUP. (FSERR 51)
'R'	NONEXISTENT PERMANENT FILE. (FSERR 52)
'S'	NONEXISTENT TEMPORARY FILE. (FSERR 53)
'T'	INVALID FILE REFERENCE. (FSERR 54)
'U'	DEVICE UNAVAILABLE. (FSERR 55)
'V'	INVALID DEVICE SPECIFICATION. (FSERR 56)
'W'	OUT OF VIRTUAL MEMORY. (FSERR 57)
'X'	NO PASSED FILE. (FSERR 58)
'Y'	STANDARD LABEL VIOLATION. (FSERR 59)
' '	GLOBAL RIN UNAVAILABLE. (FSERR 60)
'a'	OUT OF GROUP DISK SPACE. (FSERR 61)
'b'	OUT OF ACCOUNT DISK SPACE. (FSERR 62)
'c'	USER LACKS NON-SHARABLE DEVICE CAPABILITY. (FSERR 63)
'd'	PROGRAM NOT PREPPED WITH MULTI-RIN CAPABILITY. (FSERR 64)
'e'	PUNCH HOPPER EMPTY. (FSERR 65)
'f'	PLOTTER LIMIT SWITCH REACHED. (FSERR 66)
'g'	PAPER TAPE ERROR. (FSERR 67)
'h'	INSUFFICIENT SYSTEM RESOURCES. (FSERR 68)
'i'	I/O ERROR. (FSERR 69)
'p'	I/O ERROR WHILE PRINTING HEADER/TRAILER. (FSERR 70)
'q'	TOO MANY FILES OPEN. (FSERR 71)
'r'	INVALID FILE NUMBER. (FSERR 72)
's'	BOUNDS VIOLATION. (FSERR 73)
't'	NO ROOM LEFT IN STACK SEGMENT FOR ANOTHER FILE ENTRY (FSERR 74)
'w'	NO-WAIT I/O PENDING. (FSERR 77)
'x'	NO NO-WAIT I/O PENDING FOR ANY FILE. (FSERR 78)
'y'	NO NO-WAIT I/O PENDING FOR SPECIAL FILE. (FSERR 79)
80	SPOOLFILE SIZE EXCEEDS CONFIGURATION. (FSERR 80)

Table 2-8. File System Error Messages (continued)

VALUE	DESCRIPTION
81	NO "SPOOL" CLASS IN SYSTEM. (FSERR 81)
82	INSUFFICIENT SPACE FOR SPOOLFILE. (FSERR 82)
83	I/O ERROR ON SPOOLFILE. (FSERR 83)
84	DEVICE UNAVAILABLE FOR SPOOLFILE. (FSERR 84)
85	OPERATION INCONSISTENT WITH SPOOLING. (FSERR 85)
86	SPOOLING INTERNAL ERROR. (FSERR 86)
87	BAD SPOOLFILE BLOCK. (FSERR 87)
88	NONEXISTENT SPOOLFILE. (FSERR 88)
89	POWER FAILURE. (FSERR 89)
90	EXCLUSIVE VIOLATION: FILE BEING ACCESSED. (FSERR 90)
91	EXCLUSIVE VIOLATION: FILE ACCESSED EXCLUSIVELY. (FSERR 91)
92	LOCKWORD VIOLATION. (FSERR 92)
93	SECURITY VIOLATION. (FSERR 93)
94	USER IS NOT CREATOR. (FSERR 94)
95	READ COMPLETED DUE TO BREAK. (FSERR 95)
96	DISK I/O ERROR. (FSERR 96)
97	NO CONTROL Y PIN. (FSERR 97)
98	READ TIME OVERFLOW. (FSERR 98)
99	BOT AND BACKSPACE TAPE. (FSERR 99)
100	DUPLICATE PERMANENT FILE NAME. (FSERR 100)
101	DUPLICATE TEMPORARY FILE NAME. (FSERR 101)
102	I/O ERROR ON DIRECTORY. (FSERR 102)
103	PERMANENT FILE DIRECTORY OVERFLOW. (FSERR 103)
104	TEMPORARY FILE DIRECTORY OVERFLOW. (FSERR 104)
105	BAD VARIABLE BLOCK STRUCTURE. (FSERR 105)
106	EXTENT SIZE EXCEEDS MAXIMUM. (FSERR 106)
107	INSUFFICIENT SPACE FOR USER LABELS. (FSERR 107)
108	INVALID FILE LABEL. (FSERR 108)
109	INVALID CARRIAGE CONTROL. (FSERR 109)
110	ATTEMPT TO SAVE PERMANENT FILE AS TEMPORARY. (FSERR 110)
111	USER LACKS SAVE FILES (SF) CAPABILITY. (FSERR 111)
112	USER LACKS PRIVATE VOLUMES (UV) CAPABILITY. (FSERR 112)
113	VOLUME SET NOT MOUNTED - MOUNT PROBLEM. (FSERR 113)
114	VOLUME SET NOT DISMOUNTED - DISMOUNT PROBLEM. (FSERR 114)
115	ATTEMPTED RENAME ACROSS VOLUME SETS - REJECTED. (FSERR 115)
116	INVALID TAPE LABEL FOPEN PARAMETERS. (FSERR 116)
117	ATTEMPT TO WRITE ON AN UNEXPIRED TAPE FILE. (FSERR 117)
118	INVALID HEADER OR TRAILER TAPE LABEL. (FSERR 118)
119	I/O ERROR POSITIONING TAPE FOR TAPE LABELS. (FSERR 119)
120	ATTEMPT TO WRITE IBM STANDARD TAPE LABEL. (FSERR 120)

Table 2-8. File System Error Messages (continued)

VALUE	DESCRIPTION
121	TAPE LABEL LOCKWORD VIOLATION. (FSERR 121)
122	TAPE LABEL TABLE OVERFLOW. (FSERR 122)
123	END OF TAPE VOLUME SET. (FSERR 123)
124	ATTEMPT TO APPEND LABELED TAPE. (FSERR 124)
125	EXPIRATION DATE CAN'T BE LATER THAN PRECEDING FILE'S (FSERR 125)
126	CHARACTER SET NUMBER MUST BE BETWEEN 0 AND 31. (FSERR 126)
127	FORM NUMBER MUST BE BETWEEN 0 AND 31. (FSERR 127)
128	LOGICAL PAGE NUMBER MUST BE BETWEEN 0 AND 31. (FSERR 128)
129	VERTICAL FORMAT NUMBER MUST BE BETWEEN 0 AND 31. (FSERR 129)
130	NUMBER OF COPIES MUST BE BETWEEN 1 AND 32767. (FSERR 130)
131	NUMBER OF OVERLAYS MUST BE BETWEEN 1 AND 8. (FSERR 131)
132	PAGE LENGTH PARM MUST BE BETWEEN 12 (=3") AND 68 (=17"). (FSERR 132)
133	PICTURE NUMBER MUST BE BETWEEN 0 AND 31. (FSERR 133)
134	EXTENDED CAPABILITY PARM MUST BE 0 (OFF) OR 1 (ON). (FSERR 134)
137	DEFECTIVE TRACK ON FOREIGN DISK. (FSERR 137)
138	TRACK DOES NOT EXIST ON FOREIGN DISK. (FSERR 138)
139	DELETED RECORD ON IBM DISKETTE. (FSERR 139)
148	INACTIVE RIO RECORD. (FSERR 148)
149	MISSING ITEM NUMBER OR RETURN-VARIABLE. (FSERR 149)
150	INVALID ITEM NUMBER. (FSERR 150)
151	UNDEFINED FILE TYPE. (FSERR 151)
152	UNRECOGNIZED KEYWORD IN FOPEN DEVICE PARAMETER. (FSERR 152)
153	EXPECTED ";" OR CARRIAGE RETURN IN DEVICE PARAMETER (FSERR 153)
154	ENVIRONMENT FILE OPEN ERROR. (FSERR 154)
155	NOT ENVIRONMENT FILE. CHECK FILE CODE OR RECORD SIZE. (FSERR 155)
156	ENVIRONMENT HEADER RECORD INCORRECT. (FSERR 156)
157	UNCOMPILED ENVIRONMENT FILE. (FSERR 157)
158	ERROR READING ENVIRONMENT FILE. (FSERR 158)
159	ERROR CLOSING ENVIRONMENT FILE. (FSERR 159)
160	ERROR DOING FDEVICECONTROL FROM ENVIRONMENT FILE. (FSERR 160)
161	TOO MANY PARAMETERS IN DEVICE STRING - OVERFLOW. (FSERR 161)
162	EXPECTED "=" AFTER KEYWORD IN DEVICE PARAMETER. (FSERR 162)
163	"ENV" BACK REFERENCE IN FILE EQUATION INCORRECT. (FSERR 163)
164	DEVICE PARAMETER TOO LARGE OR MISSING CARRIAGE RETURN (FSERR 164)

Table 2-8. File System Error Messages (continued)

VALUE	DESCRIPTION
165	INVALID DENSITY SPECIFICATION. (FSERR 165)
166	FFILEINFO FAILED IN ACCESSING REMOTE SPOOL FILE. (FSERR 166)
167	SPOOL FILE LABEL ERROR, CAN'T INSERT ENV FILE NAME. (FSERR 167)
168	ITEM NOT SUPPORTED ON REMOTE SYSTEM. (FSERR 168)
170	RECORD IS MARKED FOR DELETION. (FSERR 170)
171	DUPLICATE KEY VALUE. (FSERR 171)
172	NO SUCH KEY. (FSERR 172)
173	TCOUNT PARAMETER LARGER THAN RECORD SIZE. (FSERR 173)
174	CAN NOT GET EXTRA DATA SEGMENT. (FSERR 174)
175	KSAM INTERNAL ERROR. (FSERR 175)
176	ILLEGAL EXTRA DATA SEGMENT LENGTH. (FSERR 176)
177	TOO MANY EXTRA DATA SEGMENTS FOR THIS PROCESS. (FSERR 177)
178	NOT ENOUGH VIRTUAL MEMORY FOR EXTRA DATA SEGMENT. (FSERR 178)
179	THE FILE MUST BE LOCKED BEFORE ISSUING THIS INTRINSIC (FSERR 179)
180	KSAM FILE MUST BE REBUILT. (FSERR 180)
181	INVALID KEY STARTING POSITION. (FSERR 181)
182	FILE IS EMPTY. (FSERR 182)
183	RECORD DOES NOT CONTAIN ALL KEYS. (FSERR 183)
184	INVALID RECORD NUMBER (FFINDN INTRINSIC). (FSERR 184)
185	SEQUENCE ERROR IN PRIMARY KEY. (FSERR 185)
186	INVALID KEY LENGTH. (FSERR 186)
187	INVALID KEY SPECIFICATION. (FSERR 187)
188	INVALID DEVICE SPECIFICATION. (FSERR 188)
189	INVALID RECORD FORMAT. (FSERR 189)
190	INVALID KEY BLOCKING FACTOR VALUE. (FSERR 190)
191	RECORD DOES NOT CONTAIN SEARCH KEY SPECIFIED FOR DELETION (FSERR 191)
192	SYSTEM FAILURE OCCURRED WHILE THE KSAM FILE WAS OPENED (FSERR 192)
193	\$STDIN/\$STDLIST CANNOT BE REDIRECTED TO KSAM FILES (FSERR 193)
194	KSAM FILES NOT ALLOWED FOR GLOBAL APT's. (FSERR 194)
195	GLOBAL FILES CANNOT BE REMOTE FILES. (FSERR 195)
196	LANGUAGE NOT SUPPORTED. (FSERR 196)
197	NATIVE LANGUAGE INTERNAL ERROR. (FSERR 197)
198	INVALID VERSION NUMBER IN KSAM FILE. (FSERR 198)
302	INVALID ITEM NUMBER FOR FDEVICECONTROL (FSERR 302)
303	INVALID ACCESS FOR ITEM NUMBER TO FDEVICECONTROL (FSERR 303)
304	ATTEMPT TO CHANGE TERMINAL PARITY IN 8 BIT MODE (FSERR 304)
305	INVALID FORMAT IN TERMINAL CONFIGURATION FILE (FSERR 305)

Table 2-8. File System Error Messages (continued)

VALUE	DESCRIPTION
306	CHECKSUM ERROR IN TERMINAL CONFIGURATION FILE (FSERR 306)
307	PASSED VALUE TO FDEVICECONTROL LESS THAN MIN (FSERR 307)
308	PASSED VALUE TO FDEVICECONTROL GREATER THAN MAX (FSERR 308)
309	PASSED VALUE TO FDEVICECONTROL IS UNSUPPORTED (FSERR 309)
310	COUNT TO FDEVICECONTROL INSUFFICIENT TO RETURN INFO (FSERR 310)
311	COUNT TO FDEVcnt GREATER THAN AVAILABLE TO STORE INFO (FSERR 311)
312	PASSED SPECIAL CHARACTER HAS PREVIOUSLY DEFINED FUNCTION (FSERR 312)
350	KSAM LOCK TABLE FULL (FSERR 350)
351	TOO MANY RECORD LOCKS IN ONE PROCESS (FSERR 351)
352	RECORD WAS LOCKED ALREADY (FSERR 352)
353	FILE MUST BE UNLOCKED BEFORE ISSUING THIS INTRINSIC (FSERR 353)
354	DATA INTEGRITY VIOLATION (FSERR 354)
355	FILE IS LOCKED BY ANOTHER PROCESS (FSERR 355)
356	LOCK REQUEST REJECTED DUE TO POTENTIAL DEADLOCK (FSERR 356)
357	DATA MAY BE INVALID BECAUSE FILE IS LOCKED BY OTHERS (FSERR 357)

Utilities

ASOCTBL5

To Initiate: `RUN ASOCTBL5.PUB.SYS'`

Commands:

$$\text{devclass} = \left\{ \begin{array}{l} \text{0.0} \\ \text{0.acctname} \\ \text{user.acctname} \\ \text{user.0} \end{array} \right\} \left[\left\{ \begin{array}{l} \text{0.0} \\ \text{0.acctname} \\ \text{user.acctname} \\ \text{user.0} \end{array} \right\} \dots \right]$$

Default: Input from a terminal

Capabilities: System manager (SM) or SAVE and WRITE access to PUB.SYS

Example: To input from an editor file:

```
file input=filename
run asoctl5.pub.sys
```



DISKED5

To Initiate: `RUN DISKED5.PUB.SYS`

Commands:

```
B[ase]  absector
DEBUG
DISC  ldev

D[ump] { relsector[, numsectors] } [ , A ]
        ALL

E[xit]
F[ile]  filename
HELP

L[ist] { ldev
        devclass }

M[odify] sectornum, relwordaddr [, numwords]
W[idth]
```


DISKED5

Defaults: *relsector=0, numsectors=1*
absector=0, \$STDLIST
numwords=1, ldev=1

Capabilities: System manager (SM), privileged mode (PM)

Notes: Formal file designator=DEDILIST

DPAN5

To Initiate: `RUN DPAN5.PUB.SYS[,EIGHTLPI][;PARM\10]`

Commands: None

Default: Session (\$STDLIST) Job (DEV=LP)

Capabilities: Standard (IA, BA)

Formal File Designator: DPANLIST, MDUMP

Examples: To format the dump with eight lines per inch:

`run dpan5.pub.sys,eightlpi`

To interactively specify which system tables to format (there will be a short pause before the user is prompted):

`run dpan5.pub.sys;parm=10`

To redirect output from magnetic tape to another device (for example, serial disk) set up a file equation:

`file mdump; dev=sdisk`
`run dpan5.pub.sys`

To generate a standard dump in batch mode:

`stream dumpjob.pub.sys`

To generate a mini dump in batch mode:

`stream dumpmini.pub.sys`

To generate a dump that can be transferred to other systems:

`stream dumparch.pub.sys`

To use or create a permanent dump file:

`run dpan5.pub.sys;info"filename"`

FREE5

To Initiate: **RUN FREE5.PUB.SYS**

Commands: None

Defaults: **\$STDLIST**

Capabilities: Standard (IA, BA)

Formal File **FREE5OUT**

Designator:

Example: To redirect output to the line printer (LP):

```

file free5out;dev=lp
run free5.pub.sys

```

LISTDIR5

To Initiate: **RUN LISTDIR5.PUB.SYS[;PARM=1]**

Commands: **E[xit]**
DISMOUNT
HELP
LISTACCT [*acctname*] [, *listfile*] [;PASS]
LISTGROUP [*groupname* [. *acctname*]] [, *listfile*] [;PASS]
LISTUSER [*user* [. *acctname*]] [, *listfile*] [;PASS]
LISTF *fname* [. *groupname* [. *acctname*]] [, *listfile*] [;PASS] [;MAP]
LISTSEC *filename* [. *groupname* [. *acctname*]] [, *listfile*] [;PASS]

MOUNT [{ *vsname* } [. *groupname* [. *acctname*]]]
 *
 ;GEN=[*genindex*]]

Capabilities: Standard (IA, BA). Some of the above commands are restricted to users having account manager (AM), system manager (SM) or use private disk volumes (UV) capability.

Formal File **OUT**

Designator:

Example: To redirect the output to the line printer (LP) use a file equation and the parameter PARM=1:

```

file out;dev=lp
run listdir5.pub.sys;parm=1

```

LISTLOG5

To Initiate: **RUN LISTLOG5.PUB.SYS**

Commands: None

Default: LP

Capabilities: System manager (SM)

Formal File LOGLIST

Designator:

Example: To redirect the output to the terminal:

file loglist;dev=\$stdlist

run listlog5.pub.sys

LISTEQ5

To Initiate: **RUN LISTEQ5.PUB.SYS[;PARAM=1]**

Commands: None

Default: \$STDLIST

Capabilities: Standard (IA, BA)

Formal File LIST

Designator:

Notes: You can use the commands LISTEQ and LISTFTEMP for the same purpose.

Example: To redirect output to the line printer (LP) use a file equation and the parameter PARAM=1:

file list;dev=lp

run listeq5.pub.sys;parm=1

MEMLOGAN

To Initiate: `RUN MEMLOGAN.PUB.SYS[;PARM=n]`

Where *n* is:

- 0 The memory log file printed but not cleared.
- 1 Displays and clears the memory log file. (default)
- 2 Displays the memory log file and deletes the file after the next SHUTDOWN command.

Commands None

Capabilities: System manager (SM)

Formal File OUT

Designator:

Example: To redirect output to the line printer (LP):

```
file out;dev=lp  
run memlogan.pub.sys
```

MENTIMER

To Initiate: **RUN MENTIMER.PUB.SYS;PARM=n**

Where *n* is any integer greater than 0, but less than 32767. Denotes the new logging interval in seconds.

Commands: None

Capabilities: System manager (SM)

Default: 60 minutes (Series 39/40/42/44/48/52/58/64/68/70)
 10 minutes (Series 37)

PATCH

To Initiate: **RUN PATCH.PUB.SYS**

Commands: **D segnum, address [,numlocations]**
 M segnum, address [,numlocations]
 DG reloffset [,numwords]
 MG reloffset [,numwords]

Capabilities: Standard (IA, BA)

Notes: Prepare each source language program to be patched differently.

FORTRAN: Use the **MAP**, **LABEL**, and **LOCATION** parameters of **\$CONTROL** command.

COBOL, COBOLII, SPL: Use the **PMAP** parameter of the **PREP** command.

RECOVER5

To Initiate: **RUN RECOVER5.PUB.SYS**

Commands: None.

Capabilities: System manager (SM)

Notes: Once you have performed a **RELOAD** to restore the operating system and user files from a **SYSDUMP** tape, you can use **RECOVER5** to restore any files saved by **SADUTIL**.

SADUTIL

Before you can use the SADUTIL program file in the diagnostic utility system (DUS), you must create a DUS cold-load medium:

1. HELLO FIELD.SUPPORT,HP32231
2. Format and serialize your cartridge tape or flexible disk, using the >FORMAT and >SERIAL commands of the VINIT subsystem.
3. LISTF @DUS.HP32231.SUPPORT|
4. You must have one of the following files to create a copy of (DUS).
 - COPYDUS.HP32231.SUPPORT (Program File)
 - FLOPDUS.HP32231.SUPPORT (Flexible Disc)
 - TAPEDUS.HP32231.SUPPORT (Magnetic Tape)
 - CARTDUS.HP32231.SUPPORT (Cartridge Tape)
5. RUN COPYDUS
6. Mount the installation medium at this time.

Cold Loading the DUS on the Series 37:

1. Mount the medium containing the diagnostic utility system (DUS) on the appropriate drive and place the drive online.
2. At the system console, press **CNTL** B.
3. When the H -> (for Help) prompt is displayed, type LOAD and specify the channel and device on which the medium is mounted. The DUS is now entered into memory.
4. When the message ENTER YOUR PROGRAM NAME appears, type SADUTIL.

Cold Loading the DUS on Series 39/40/42/44/48/52/58:

1. Mount the medium containing the diagnostic utility system on the appropriate drive. Place the drive online.
2. On the system control panel, set the LOAD thumbwheel switches to the channel address and device address of the drive containing the DUS.
3. From the system control panel, press the **HALT** key, then press the **LOAD** key.
4. From the CMP, press **CNTL** B. When the CMP prompt (->) appears on the system console, type HALT. When another prompt is displayed, type LOAD.
5. When the message "ENTER YOUR PROGRAM NAME" appears, type SADUTIL.

Cold Loading the DUS on Series 64/68/70:

1. Mount the tape containing the DUS on the tape drive and place the drive online.
2. If the DCU prompt (C> or M>) is not already present on the system console, press **CNTL** B. When you see the prompt, type HALT. When another prompt appears, type LOAD.
3. If you wish to cold-load from a device other than the pre-set default device, enter the IMB number, channel number and device number, separated by commas, after LOAD:
 - C> (or M>)LOAD imb,channel,device
4. When the message ENTER YOUR PROGRAM NAME appears, type SADUTIL.

SADUTIL

SADUTIL Commands: Set all Cold Load ID's to 1:

CLID

Configure Logical Devices:

CONF [ldev]

Copy disk to disk:

COPY

Edit:

EDIT

The Edit commands are:

Specify Base Sector:

BASE [basesector]

Specify Logical Device:

DISC [ldev]

Modify Sector:

MODIFY [diskaddr] [, wordloc] [, wordcount]

Enter Address:

$$\text{PDSK } firstsect \left\{ \begin{array}{l} , numsect \\ , lastsect \end{array} \right\} \left\{ \begin{array}{l} ; A[0] \\ ; 0[A] \end{array} \right\}$$

Output to Console or printer:

$$\text{OUTM} \left[\begin{array}{c} C \\ P \end{array} \right]$$

Scan Disk for File Labels:

FIND

Description of SADUTIL Commands:

HELP

Print Defective Tracks Table:

PDTT [ldev]

Enter:

PFIL filename [.groupname[.acctname]] [, detail]

Print Volume Label of LDEV:

PVOL [ldev]

Save Files to Serial Disk:

SAVE

Exit SADUTIL:

3-8 Utilities

STOP

SLPATCH

To Initiate: **RUN SLPATCH.PUB.SYS**

Commands: **D** [*segname*,] **D**, *segdisplace* [, *numwords*]
 M [*segname*,] **M**, *segdisplace* [, *numwords*]

Defaults: *numwords*=1

Capabilities: Standard (IA, BA), but privileged mode (PM) is needed to modify the contents of privileged segmented library files.

Notes: Request a PMAP of your segmented library file by using the PREP parameter of the ADDSL SEGMENTER command.

SPOOK5

To Initiate: **RUN SPOOK5.PUB.SYS**

Notes: For the APPEND, COPY, FIND, and LIST commands, range is defined as:
 [ALL]
 or

$$\left[\left[\left\{ \begin{array}{c} \text{renumber1} \\ * \\ \text{FIRST} \\ \text{LAST} \end{array} \right\} \left[\begin{array}{c} + \\ - \end{array} \right] \text{offset} \right] \right] [, \text{count}]$$

$$\left[\left\{ \begin{array}{c} \text{renumber2} \\ * \\ \text{FIRST} \\ \text{LAST} \end{array} \right\} \left[\begin{array}{c} + \\ - \end{array} \right] \text{offset} \right]$$

Commands:

$$A[\text{LTER}] \left\{ \begin{array}{c} * \\ \text{username}[\text{acctname}] \\ \text{dfid}[, \text{dfid}][, \dots] \end{array} \right\} ; \left[\begin{array}{c} P[\text{ri}] = \text{outpri} \\ C[\text{opies}] = \text{copies} \\ D[\text{ev}] = \left\{ \begin{array}{c} \text{ldev} \\ \text{devclass} \end{array} \right\} \end{array} \right]$$

$$\left[\begin{array}{c} , P[\text{ri}] = \text{outpri} \\ , C[\text{opies}] = \text{copies} \\ , D[\text{ev}] = \left\{ \begin{array}{c} \text{ldev} \\ \text{devclass} \end{array} \right\} \end{array} \right] [, \dots]$$

SPOOK5

$$AP[PEND] \left[\begin{array}{c} [range[,filename]] \\ \left[\left\{ \begin{array}{c} username[.acctname] \\ dfid[,dfid][, \dots] \end{array} \right\} ; range[,filename] \right] \\ [END] \end{array} \right]$$

$$C[OPY] \left[\begin{array}{c} [range[,filename]] \\ \left[\left\{ \begin{array}{c} username[.acctname] \\ dfid[,dfid][, \dots] \end{array} \right\} ; range[,filename] \right] \end{array} \right]$$

$$D[EBUG]$$

$$E[XIT]$$

$$F[IND] [\textcircled{0}] ["string"] [,range]$$

$$HELP \left[\begin{array}{c} \left\{ \begin{array}{c} MPE \\ HELP \\ tablecontents \\ command,[keyword] \\ ALL \end{array} \right\} \end{array} \right]$$

$$I[NPUT] \left[\begin{array}{c} username[.acctname] \\ dfid[,dfid] \dots \end{array} \right] ; *tapefile$$

$$KILL$$

$$L[IST] [range]$$

$$M[ODE] \left[W[idth] = \left\{ \begin{array}{c} + \\ - \end{array} \right\} \right] \left[C[ontrols] = \left\{ \begin{array}{c} ON \\ OFF \end{array} \right\} \right]$$

$$O[UTPUT] \left[\begin{array}{c} username[.acctname] \\ dfid[,dfid] \dots \end{array} \right] ; *tapefile [;P[urge]]$$

$$P[URGE] \left\{ \begin{array}{c} dfid \\ * \end{array} \right\} [,dfid][, \dots]$$

$$Q[UIT]$$

$$RUN \left[\begin{array}{c} progfile \\ * \end{array} \right]$$

$$S[HOW] \left[\begin{array}{c} username[.acctname] \\ [dfid[,dfid][, \dots]] \left[; \left\{ \begin{array}{c} \textcircled{0} \\ \left[\textcircled{0} \right] \left\{ \begin{array}{c} I \\ O \end{array} \right\} \end{array} \right\} \right] \end{array} \right]$$

$$T[EXT] \left\{ \begin{array}{c} dfid \\ \textcircled{0} \end{array} \right\}$$

$$X[PLAIN]$$

Capabilities: System managers (SM) or system supervisors (OP) can access any spoolfile.
 Account managers (AM) can access any spoolfile in the logon account.
 Standard users (IA,BA) can access any spoolfile that they created.

VINIT

To Initiate: VINIT [*listdevice*]

Commands: COND *ldev* [;RECOVER]

COPY *fromldev,toldev*[;GEN=[*genindex*]] [;VER[*ify*] [;buf=*size*]

DSTAT $\left[\begin{array}{c} ldev \\ ALL \\ 0 \end{array} \right]$

DTRACK *ldev*

EXIT

FOREIGN *ldev*

FORMAT *ldev*[;IBM[:*valid*]]

HELP

INIT *vname,ldev* [, *vname.groupname.acctname*]
 [;GEN=*genindex*]

PDEFN $\left[\begin{array}{c} *.group.account \\ vname.group.account \end{array} \right]$

PDTRACK *ldev*

PFSPACE $\left\{ \begin{array}{c} ldev [;addr] \\ ALL \end{array} \right\}$

PLABEL *ldev*

SCRATCH *ldev* [;RESET]

SERIAL *ldev*

VERIFY *ldev* [;DTT]

Capabilities: System manager (SM) or system supervisor (OP).

Intrinsics

Intrinsics Function Table

Table 4-1. Intrinsic Functions

Function	Intrinsics		
DATA SEGMENT MANAGEMENT	ALTDSEG DLSIZE DMOVIN	DMOVOUT FREEDSEG GETDSEG	SWITCHDB ZSIZE
DATA TYPES	ASCII BINARY	CTranslate DASCII	DBINARY
DEBUG	DEBUG RESETDUMP	SETDUMP STACKDUMP	
DEVICE CONTROL	FCARD FCONTROL FDEVICECONTROL	FRELATE FSETMODE PTAPE	RESETCONTROL
FILE ACCESS	FCLOSE FLOCK FOPEN	FPARSE FRENAME FSETMODE	FUNLOCK HPACDPUT
FILE INFORMATION	FCHECK FERRMSG FFILEINFO	FGETINFO FLABELINFO FRELATE	PRINTFILEINFO HPACDINFO
INTERPROCESS COMMUNICATION	FINTEXT FINTSTATE	MAIL RECEIVEMAIL	SENDMAIL
I/O	FCARD FCHECK FDELETE FPOINT FREAD FREADBACKWARD FREADIR FREADSEEK	FREADLABEL FRELATE FSETMODE FSPACE FUPDATE FWRITE FWRITEDIR FWRITELABEL	IODONTWAIT IOWAIT PRINT PRINTOP PRINTOREPLY READ READX
LIBRARY PROCEDURES (LOADING & UNLOADING)	LOADPROC	UNLOADPROC	
MESSAGE CATALOG (APPLICATIONS)	CATOPEN CATCLOSE	CATREAD NLAPPEND	
MESSAGE CATALOG (SYSTEM)	GENMESSAGE		

Table 4-1. Intrinsic Functions (continued)

Function	Intrinsics		
NLS	ALMANAC CATCLOSE	NLCONVNUM NLFINDSTR	NLINFO NLKEYCOMPARE
	CATOPEN CATREAD	NLFMTCALENDAR	NLMATCH NLMATCHINIT
	NLAPPEND NLCOLLATE	NLFMTCLOCK	NLNUMSPEC NLREPCCHAR
	NLCONVCLOCK	NLFMTCUSTDATE	NLSCANMOVE NLSUBSTR
	NLCONVCUSTDATE	NLFMTDATE	NLSWITCHBUF
		NLFMTLONGCAL	NLTRANSLATE
		NLFMTNUM NLGETLANG	
PRIVILEGE MODE	GETPRIVMODE	GETUSERMODE	
PROCESS HANDLING	ACTIVATE CAUSEBREAK	GETPROCINFO KILL	QUITPROG RECEIVEMAIL
	CREATE CREATEPROCESS	MAIL PAUSE PROCINFO	SENDMAIL SUSPEND
	FATHER GETPRIORITY	PROCTIME QUIT	TERMINATE
	GETPROCID		
RESOURCE MANAGEMENT	FREELOCRIIN GETLOCRIIN	LOCKGLORIN	UNLOCKGLORIN
	GETORIGIN	LOCKKLOCRIIN	UNLOCKKLOCRIIN
		LOCKRIINOWNER	
SEGMENTER	ADJUSTUSLF CLEANUSL	EXPANDUSLF	INITUSLF
	DUMPPMAP	FINDPMADDR	
		FINDPMAPNAME	
SESSIONS (PROGRAMMATIC CONTROL)	ABORTSESS	STARTSESS	
SYSTEM INFORMATION	CALENDAR CLOCK	FINDJCW FMTCALENDAR	PUTJCW SETJCW TIMER
	COMMAND DATELINE	FMTCLOCK FMTDATE	WHO
SECURITY INFORMATION	HPACDINFO	HPACDPUT	
TOOLS	COMMAND	MYCOMMAND	SEARCH
TRAPS	ARITRAP RESETCONTROL	XARITRAP XCONTRAP	XLIBTRAP XSYSTRAP
USER LOGGING	BEGINLOG CLOSELOG	FLUSHLOG LOGINFO	OPENLOG WRITELOG
	ENDLOG	LOGSTATUS	

4-2 Intrinsics

Intrinsics

ABORTSESS

Enables a program to abort a specified job or session from the system.

```

      IV  DV  IA
ABORTSESS(jsid,jsnum,jstatus);

```

ACTIVATE

Activates a newly created process, or a process suspended with the `SUSPEND` intrinsic. Requires process handling (PH) capability.

```

      O-V  IV  LV
ACTIVATE(pin,allow);

```

ADJUSTUSLF

Adjusts directory space in a USL file by moving the start of the information block forward (or backward) on a user subprogram library (USL) file, thereby increasing (or decreasing) the space available for the file directory block. The overall length of the file does not change. This intrinsic is intended for programmers writing compilers.

```

      I              IV  IV
uslferror:=ADJUSTUSLF(uslfnum,adjustment);

```

ALTDSEG

Reduces the storage required by the extra data segment when moved into main memory and expands storage as required, allowing for a more efficient use of memory. Data segment management (DS) capability is required.

```

      LV  IV  I
ALTDSEG(index,increment,size);

```

ALMANAC

Returns numeric date information.

```
      LV  LA  I      I      I      I
ALMANAC(date,error,yearnum,monthnum,daynum,weekdaynum);
```

ARITRAP

Collectively enables all arithmetic traps (except the IEEE inexact result trap) or disables all arithmetic traps.

```
      LV
ARITRAP(trapstate);
```

ASCII

Converts a binary number to a specified base and represents it as a numeric ASCII string.

```
      I      LV  IV  BA
numchar:=ASCII(binvalue,base,asciieqv);
```

BEGINLOG

Posts a special record to the user logging file to mark the beginning of a logical transaction. When BEGINLOG is called, the logging memory buffer is flushed to ensure that the record gets to the logging file. User logging (LG) or system supervisor (OP) capability is required.

```
      D  LA  I      I      I
BEGINLOG(index,data,length,mode,logstatus)
```

BINARY

Converts a numeric (octal or decimal) ASCII string to a twos complement binary value.

```

      I          BA    IV
bineqv:=BINARY(asciiqv,length);

```

CALENDAR

Returns the calendar date, including the day of year and the year of century.

```

      L
date:=CALENDAR;

```

CATCLOSE

Closes the specified application message catalog facility.

```

      D    LA
CATCLOSE (catindex,error)

```

CATOPEN

Opens the specified application message catalog facility.

```

      D          BA    LA
catindex:=CATOPEN(formaldesignator,error);

```

CATREAD

Reads the specified catalog and returns (or sends) the text as specified.

```

      I          D    IV    IV    LA    BA    IV
msglen:=CATREAD(catindex,setnum,msgnum,error,buffer,buffer,
      BA    BA    BA    BA    BA    BA    IV
parm1,parm2,parm3,parm4,parm5,msgdest);

```

CAUSEBREAK

Interrupts the program (the entire process structure). The CAUSEBREAK intrinsic is the programmatic equivalent to pressing **Break** in a session, and is not applicable in jobs. The program is suspended while in BREAK mode. Execution of the program can resume where the interruption occurred by entering the RESUME command, or be aborted by entering the ABORT command.

```
CAUSEBREAK;
```

CLEANUSL

Deletes all inactive entries from currently managed USL files and returns the file number of the new USL file. Therefore, you must test the condition code immediately upon return from the intrinsic. Unpredictable results occur if an error number is used as a file number.

```
      I      IV      BA  
filenum:=CLEANUSL(uslfnm,formaldesig);
```

CLOCK

Returns the time (hours, minutes, seconds, and tenths of seconds) according to the system timer.

```
      D  
time:=CLOCK;
```

CLOSELOG

Closes access to the user logging facility. User logging (LG) or system supervisor (OP) capabilities are required.

```
      D      I      I  
CLOSELOG(index,mode,logstatus);
```

COMMAND

Executes an MPE command programmatically.

```

           BA      I      I
COMMAND(cmdimage,cmderror,parnum);

```

CREATE

Creates a process as a child of the calling process. Process handling (PH) capability is required.

```

0-V      BA      BA      I  IV      LV
CREATE(formaldesig,entryname,pin,parm,loadflag,
      IV      IV      IV      LV      IV
      stacksize,dlsiz,mazdata,priorityclass,rank);

```

CREATEPROCESS

Creates a process and allows you to assign \$STDIN and \$STDLIST to any file. Process handling (PH) capability is required.

```

0-V      I      I      BA      IA      IA
CREATEPROCESS(createstatus,pin,formaldesig,itemnum,item);

```

CTranslate

Converts a string of characters between EBCDIC and ASCII, or between EBCDIC (HP-specific version of EBCDIC) and KANA8 (8-bit, Japanese International Standard (JIS) version of USASCII code).

```

0-V      IV      CA      CA      IV      CA
CTranslate(transcode,inbuffer,outbuffer,bufferlength,transtable);

```

DASCII

Converts a 32-bit binary number to a specified base and represents it as a numeric ASCII string.

```
      I          IV  IV  BA
numchar:=DASCII(binvalue,base,asciiqv);
```

DATELINE

Returns the current date and time, including the day of week, month, day, year, hours, and minutes.

```
      BA
DATELINE(datebuffer);
```

DBINARY

Converts a numeric ASCII string to a binary value. The numeric ASCII string can be octal, hexadecimal, or decimal.

```
      D          BA  IV
dbineqv:=DBINARY(dasciiqv,length);
```

DEBUG

Invokes the Debug facility from an interactive program and allows object code to be analyzed.

```
DEBUG;
```

DLSIZE

Causes the area between DL and DB in the compatibility mode (CM) stack to be expanded or contracted within the CM stack segment.

```

      I      IV
dldbsize:=DLSIZE(size);

```

DMOVIN

Copies data from an extra data segment into a data area. Data segment management (DS) capability is required.

```

      IV      IV      IV      LA
DMOVIN(index,displacement,number,location);

```

DMOVOUT

Copies data from the data area to an extra data segment. Data segment management (DS) capability is required.

```

      LV      IV      IV      LA
DMOVOUT(index,displacement,number,location);

```

DUMPPMAP

Fills a file with external FPMAP records representing either all of the FPMAP data stored in a program or SL file, or only that for a specified segment. One external record is created as each FPMAP record is encountered.

```

      IV      IV      IV      I      IV      I
DUMPPMAP(progfnum,pmappfnum,xpmapreclen,reccount,segnum,status);

```

ENDLOG

Posts a record to the logging file marking the end of a logical transaction. When the record is posted, **ENDLOG** flushes the user logging memory buffer to ensure that the record gets to the logging file. User logging (LG) or system supervisor (OP) capability is required.

```
      D   LA   I   I   I  
ENDLOG(index,data,length,mode,logstatus);
```

EXPANDUSLF

Changes length of a USL file by creating a USL file with the **increment** length longer or shorter than the USL file specified by **uslfnm**. The old USL file is copied to the new file with the same file name; the old USL file is then deleted.

```
      I           IV   IV  
filenum:=EXPANDUSLF(uslfnm,increment);
```

FATHER

Returns the process identification number (PIN) of the parent calling process. Process handling (PH) capability is required.

```
      I  
pin:=FATHER;
```

FCARD

Drives the HP 7260A optical mark reader (OMR).

```
      I   I   IA   I   I  
FCARD(recode,filenum,bufadr,count,status);
```

FCHECK

Returns specific details about error conditions that occurred when a file system intrinsic returns a condition code indicating an I/O error. FCHECK applies to files on any device.

0-V IV I I D I
FCHECK(*filenum,fserrorcode,translog,blocknum,numrecs*);

FCLOSE

Terminates access to a file on any device.

IV IV IV
FCLOSE(*filenum,disposition,securitycode*);

FCONTROL

Performs various control operations on a file or on the device where the file resides, including:

- Supplying a printer or terminal carriage control directive.
- Verifying I/O.
- Reading the hardware status word for the device where the file resides.
- Setting a terminal's timeout interval.
- Repositioning a file at its beginning.
- Writing an end-of-file marker.
- Skipping forward or backward to a tape mark.

IV IV L
FCONTROL(*filenum,itemnum,item*);

FDELETE

Deactivates a specified logical record in an RIO record.

0-V IV DV
FDELETE(*filenum,recnum*);

FDEVICECONTROL

Provides control operations to a printer, terminal, or a spooled devicefile and is used to:

- Download character sets, forms, and internal or control tables used in printing.
- Control the page size, pen positioning, form and use of character sets, the number of copies to be printed, and all other printing environment characteristics.

```
          IV    LA    IV    IV
FDEVICECONTROL(filenum,buffer,length,controlcode
          LV    LV    I
          parm1,parm2,fserrorcode);
```

FERRMSG

Returns a message corresponding to an FCHECK error number and enables error messages to be displayed from a program.

```
    0-V    I    LA    I
FERRMSG(fserrorcode,msgbuffer,msglength);
```

FFILEINFO

Returns information about a file.

```
    0-V    IV    IV    BA
FFILEINFO(filenum[,itemnum,item] [ ... ]);
```

FGETINFO

Returns access and status information about a file.

```
          IV    BA    L    L
FGETINFO(filenum,formaldesig,foption,aoption
          I    I    L    L    I
          lrecsize,devtype,ldevnum,hdaddr,filecode,
          I    I    I    I    I    I
          lrecptr,eof,filelimit,logcount,physcount,blksize,
          L    I    I    BA    I
          extsize,numextents,userlabels,creatorid,labaddr);
```

FINDJCW

Searches the job control word table for a specified job control word (JCW) and returns its value.

```

      BA      L      I
FINDJCW(jcwname,jcwvalue,jcwstatus);

```

FINDPMAPADDR

Searches the FPMAP in a program or SL file for a specific address in a particular segment. An external FPMAP record corresponding to the nearest entry point at or preceding *address* is returned. The caller must have MULTI-RECORD READ access to open the file.

```

      IV      IV      IV      V
FINDPMAPADDR(progfnun,segnum,address,xpmaprec,
      IV      I
      xpmapreclen,status);

```

FINDPMAPNAME

Searches the FPMAP in a program or SL file for a segment name or an entry point name.

```

      IV      BA      BA      IA
FINDPMAPNAME(progfnun,segname,entname,xpmaprec,
      V      I
      xpmapreclen,status);

```



FINTEXT

Causes the return from your interrupt procedure.

```

      LV
FINTEXT(interruptstate);

```

FINTSTATE

Enables/disables all software interrupts against the calling process.

```
      L          LV
oldstate:=FINTSTATE(interruptstate);
```

FLABELINFO

Returns information from the file label of a disk file.

```
      BA      IV      I
FLABELINFO(formaldesig,mode,fserrorcode,
      IA      REC      IA
itemnum,item,itemerror);
```

FLOCK

Dynamically locks a file. If dynamically locking more than one RIN, multiple RIN (MR) capability is required.

```
      IV      LV
FLOCK(filenum,lockflag);
```

FLUSHLOG

Flushes the contents of the user logging memory buffer to the user logging file. User logging (LG) or system supervisor (OP) capability is required.

```
      D      I
FLUSHLOG(index,logstatus);
```

FMTCALENDAR

Passes any calendar date, in the same format as the **CALENDAR** intrinsic, and returns it in the following format:

```
FRI, JAN 27, 1989
           LV    BA
FMTCALENDAR(date,formatdate);
```

FMTCLOCK

Passes the time of day, in the same format as the **CLOCK** intrinsic, and returns it in the following format:

```
12:39 AM
           DV    BA
FMTCLOCK(time,formattime);
```

FMTDATE

Passes in the calendar date and time of day, in the same format as the **CALENDAR** and **CLOCK** intrinsics, and returns it in the following format:

```
FRI, JAN 27, 1989, 12:39 AM
           LV DV    BA
FMTDATE(date,time,dateandtime);
```

FOPEN

Establishes access to a file and defines the physical characteristics of the file prior to access.

```
      I  O-V    BA    LV    LV    IV
filenum:=FOPEN(formaldesig,foption,aoption,resize,
              BA    BA    IV    IV    IV
              device,formmsg,userlabels,blockfactor,numbuffers,
              IV    IV    IV    I
              filesize,numextents,initialloc,filecode);
```

FPARSE

Parses and validates file designators.

BA IA IA DA

```
FPARSE(formaldesig,result,item,vector);
```

FPOINT

Sets the logical record pointer for a disk file containing fixed-length or undefined-length records to any logical record. When the next FREAD or FWRITE file request is made, this record is read or written to.

IV DV

```
FPOINT(filenum,lrecnum);
```

FREAD

Reads a logical record or portion of a record from a file to the stack.

I IV LA IV

```
transfercount:=FREAD(filenum,buffer,length);
```

FREADBACKWARD

Reads a logical record backward from the current record pointer. Data is presented as if read forward. Can be used to recover tape errors when handling I/O management and data recovery routines.

I IV LA IV

```
transfercount:=FREADBACKWARD(filenum,buffer,length);
```

FREADDIR

Reads a specific logical record or portion of a record from a direct-access disk file to the data stack.

IV LA IV DV
FREADDIR(*filenum,buffer,length,recnum*);

FREADLABEL

Reads a user-defined label from a disk or magnetic tape file.

O-V IV LA IV IV
FREADLABEL(*filenum,buffer,length,labelid*);

FREADSEEK

Moves a record from a disk file to a buffer in anticipation of a FREADDIR intrinsic call.

IV DV
FREADSEEK(*filenum,recnum*);

FREEDSEG

Releases an extra data segment assigned it by the GETDSEG intrinsic. Data segment management (DS) capability is required.

LV LV
FREEDSEG(*index,id*);

FREEOCRIN

Frees all local resource identification numbers (RINs) from allocation to a job/session.

FREEOCRIN;

FRELATE

Determines whether a file pair (on any device) is interactive, duplicative, or both interactive and duplicative.

L IV IV

```
intordup:=FRELATE(infilenum,listfilenum);
```

FRENAME

Renames an open disk file (and its lockword, if applicable). The file being renamed must be either:

- A new file.
- An old file (permanent or temporary), opened for EXCLUSIVE access with the *exclusive* option of the HPFOPEN/FOPEN intrinsics, and with security provisions allowing WRITE access.

IV BA

```
FRENAME(filenum,formaldesign);
```

FSETMODE

Controls the following access modes of files or devices:

- Issuing carriage return and line feed to terminal after a terminal read.
- Reporting tape automatic error recovery.
- Guaranteeing chronological order of user program write requests.
- Blocking program execution until physical completion of write requests.

IV LV

```
FSETMODE(filenum,modeflags);
```

FSPACE

Moves a record pointer forward or backward on a magnetic tape or disk file, spaces physical records on magnetic tape files and logical records on disk files.

IV IV

FSPACE(*filenum,displacement*);

FUNLOCK

Dynamically unlocks a file's global resource identification number (RIN) that was locked with the FLOCK intrinsic.

IV

FUNLOCK(*filenum*);

FUPDATE

Updates (writes) a logical record in a disk file.

IV LA IV

FUPDATE(*filenum,buffer,length*);

FWRITE

Writes a logical or physical record or portion of a record from the stack to a file on any device.

IV LA IV LV

FWRITE(*filenum,buffer,length,controlcode*);

FWRITEDIR

Writes a specific logical record from the stack to a disk file.

IV LA IV DV

FWRITEDIR(*filenum,buffer,length,lrecnum*);

FWRITELABEL

Writes a user-defined label onto a disk file or magnetic tape file that is labeled with an ANSI-standard or IBM-standard label. It also over writes old user labels.

0-V IV LA IV IV
FWRITELABEL(*filenum*,*buffer*,*length*,*labelid*);

GENMESSAGE

Provides access to messages in catalogs that were formatted with the MAKECAT utility.

 I IV IV IV BA IV
msglength:=GENMESSAGE(*filenum*,*setnum*,*msgnum*,*buffer*,*buffersize*,
 IV * * * * *
 paramask,*param1*,*param2*,*param3*,*param4*,*param5*,
 IV I
 msgdestination,*errornum*,

GETDSEG

Creates or acquires an extra data segment for use by the process. Data segment management (DS) capability is required.

 L I LV
GETDSEG(*index*,*length*,*id*);

GETINFO

Returns user-supplied information that was passed to a process when it was created.

0-V BA I I
result:=GETINFO(*infostring*,*infostringlength*,*parm*);

GETJCW

Returns the value of the system-defined job control word (JCW) to the calling process.

```
    L
    jcw:=GETJCW;
```

GETLOCIN

Acquires local resource identification numbers (RINs) for a job/session.

```
    LV
    GETLOCIN(rincount);
```

GETORIGIN

Returns the source of the activation call for the calling process that has been previously suspended and subsequently reactivated. The source of the activation request can be the parent process, a child process, or another source (for example, an interrupt or the timer). Process handling (PH) capability is required.

```
    I
    source:=GETORIGIN;
```

GETPRIORITY

Changes the priority of a process. Process handling (PH) capability is required.

```
    0-V      IV  LV  IV
    GETPRIORITY(pin,priorityclass,rank);
```

GETPRIVMODE

Dynamically enters privileged mode. Privileged mode (PM) capability is required.

```
Q-P
GETPRIVMODE;
```

GETPROCID

Returns the process identification number (PIN) of a child process. Process handling (PH) capability is required.

```
I          IV
pin:=GETPROCID(numchild);
```

GETPROCINFO

Returns status information about the parent or a child process. Process handling (PH) capability is required.

```
D          IV
processinfo:=GETPROCINFO(pin);
```

GETUSERMODE

Dynamically returns a program to nonprivileged mode. Process Handling (PH) capability required.

```
GETUSERMODE;
```

HPACDINFO

Lists security information from the access control definition (ACD) of a specified file or device.

```

      D      IV      *      IV      *      O-V
HPACDINFO(status,itemnum,item[,itemnum,item][, ... ]);

```

HPACDDPUT

Manipulates security information in the access control definition (ACD) of a specified file or device.

```

      D      IV      *      IV      *      O-V
HPACDDPUT(status,itemnum1,item1,itemnum2,item2);

```

INITUSLF

Initializes a USL file to the empty state.

```

      I      IV      IA
uslerror:=INITUSLF(uslfnnum,record);

```

IODONTWAIT

Initiates completion operations for an I/O request.

```

      I      O-V      IV      LA      I      L
fnnum:=IODONTWAIT(filenum,buffer,length,cstation)

```

IOWAIT

Initiates completion operations for an I/O request.

```

      I      O-V      IV      LA      I      L
fnnum:=IOWAIT(filenum,buffer,length,cstation);

```

JOBINFO

Provides access to job and session information.

0-V IV D LA IV LA I
JOBINFO(*jsind,jsnum,jsstatus*[, *itemnum,item,itemerror*] [...]);

KILL

Deletes a child process of the calling process and all of its descendants. Process handling (PH) capability is required.

IV
KILL(*pin*);

LOADPROC

Dynamically loads a compatibility mode (CM) segmented library (SL) procedure and any external procedures it has referenced.

I BA IV I
idnum:=LOADPROC(*procname,library,label*);

LOCKGLORIN

Locks a global resource identification number (RIN). Multiple RIN (MR) capability is required to lock more than one global RIN simultaneously.

IV L BA
LOCKGLORIN(*rinnum,lockflag,rinpassword*);

LOCKLOCRIN

Locks a local resource identification number (RIN).

	IV	L	
	LOCKLOCRIN(<i>rinum</i> , <i>lockflag</i>);		

LOCRINOWNER

Determines process identification number (PIN) of the process that locked a local resource identification number (RIN).

I		IV
<i>pin</i> :=LOCRINOWNER(<i>rinum</i>);		

LOGINFO

Provides information about an opened user logging file (whole file set). User logging (LG) or system supervisor (OP) capability is required.

0-V	DV	I	IV	BA
LOGINFO(<i>index</i> , <i>logstatus</i> [, <i>itemnum</i> , <i>item</i>] [...]);				

LOGSTATUS

Provides information about a currently opened user logging file. User logging (LG) or system supervisor (OP) capability is required.

D	LA	I
LOGSTATUS(<i>index</i> , <i>loginfo</i> , <i>logstatus</i>);		

MAIL

Determines the status of the mailbox used by its parent or child. Process handling (PH) capability is required.

```
      L          IV  I
mailstatus:=MAIL(pin,length);
```

MYCOMMAND

Parses (delineates and defines) parameters for a user-defined command image.

```
      I          BA      BA      IV
entrynum:=MYCOMMAND(cmdimage,delimiters,mazparms,
                    I      DA      BA      032
                    numparms,params,dictionary,definition);
```

NLAPPEND

Appends the appropriate language ID number to the file name.

```
      BA          IV      LA
NLAPPEND(formaldesignator,langnum,error);
```

NLCOLLATE

Compares two character strings in a language-dependent manner.

```
      BA  BA      IV  I      IV  LA  LA
NLCOLLATE(string1,string2,length,result,langnum,error,collseq);
```

NLCONVCLOCK

Checks the validity and converts a string containing a language-dependent time of day to the CALENDAR intrinsic format.

```

      D      BA      IV      IV      LA
time:=NLCONVCLOCK(string,stringlen,langnum,error);

```

NLCONVCUSTDATE

Checks the validity and converts a string containing a numerical date to the CALENDAR intrinsic format.

```

      L      BA      IV      IV      LA
date:=NLCONVCUSTDATE(string,stringlen,langnum,error);

```

NLCONVNUM

Converts native language numbers (with native separators) to ASCII numbers (with ASCII separators or stripped).

```

      IV      BA      IV      BA      I      LA
NLCONVNUM(langnum,instring,inlength,outstring,outlength,error,
      LA      LV      I
numspec,fmtmask,decimals)

```

NLFMTCALENDAR

Formats a date in a language-dependent manner.

```

      LV      BA      IV      LA
NLFMTCALENDAR(date,string,langnum,error);

```

NLFMTCLOCK

Formats time of day in a language-dependent manner.

```
          DV  BA   IV   LA
NLFMTCLOCK(time,string,langnum,error);
```

NLFMTCUSTDATE

Formats a numerical date in a language-dependent manner.

```
          LV  BA   IV   LA
NLFMTCUSTDATE(date,string,langnum,error);
```

NLFMTDATE

Formats a date in a language-dependent manner.

```
          LV  DV  BA   IV   LA
NLFMTDATE(date,time,string,langnum,error);
```

NLFMTLONGCAL

Formats the supplied date according to long calendar format.

```
          LV  BA   IV   LA
NLFMTLONGCAL(date,string,langnum,error)
```

NLFMTNUM

Converts a string containing an ASCII number to a language-specific formatted number using currency name/symbol and native language separators.

```
          IV      BA      IV      BA
NLFMTNUM(langnum,instring,inlength,outstring,
          I      LA      LA      LV      0-V
          outlength,error,numspec,fmtmask,decimals);
```

NLGETLANG

Returns current language information (system, user, data).

```

      I          IV      LA
  langnum:=NLGETLANG(function,error);

```

NLINFO

Returns language-dependent information.

```

      IV      LA      I      LA
  NLINFO(itemnumber,itemvalue,langnum,error);

```

NLKEYCOMPARE

Compares two strings of different length (for use with KSAM).

```

      BA      IV      BA      IV      I      IV      LA      LA
  NLKEYCOMPARE(genkey,length1,key,length2,result,langnum,error,collseq);

```

NLNUMSPEC

Returns the information needed for formatting and converting numbers.

```

      IV      LA      LA
  NLNUMSPEC(langnum,string,error)

```

NLREPCCHAR

Replaces nondisplayable characters of a string.

```

      BA      BA      IV      BV      IV      LA      LA
  NLREPCCHAR(instr,outstr,stringlen,repchar,langnum,error,charset);

```

NLSCANMOVE

Moves and scans character strings according to character attributes.

```
      I          BA BA  LV  IV  IV
numchar:=NLSCANMOVE(instr,outstr,flags,length,langnum,
                    LA  LA  LA
                    error,charset,shift);
```

NLSWITCHBUF

Converts string characters from phonetic order to screen order or vice versa.

```
      IV      BA      BA      IV
NLSWITCHBUF(langnum,instring,outstring,stringlength,
           LV      LA
           lefttoright,error)
```

NLTRANSLATE

Translates a string of characters between ASCII and EBCDIC in a language-dependent manner.

```
      IV BA BA  IV  IV  LA LA
NLTRANSLATE(code,instr,outstr,strlen,langnum,error,table);
```

OPENLOG

Provides access to the user logging facility. User logging (LG) or system supervisor (OP) capability is required.

```
      D  LA  LA  I  I
OPENLOG(index,logid,pass,mode,logstatus);
```

PAUSE

Suspends the calling process for a specified number of seconds.

R
PAUSE(*interval*);

PRINT

Prints character string on job/session listing device.

LA IV IV
PRINT(*message,length,controlcode*);

PRINTFILEINFO

Prints a file information display on the job/session list device.

IV
PRINTFILEINFO(*filenum*);

PRINTOP

Prints a character string on the system console.

LA IV IV
PRINTOP(*message,length,controlcode*);

PRINTOPREPLY

Prints a character string on the system console and solicits a reply.

I LA IV IV LA IV
length:=PRINTOPREPLY(*message,length,zero,reply,mazlength*);

PROCINFO

Provides access to process information.

```
          I      I  IV  IV   BA
PROCINFO(error1,error2,pin[,itemnum,item] [ ... ] );
```

PROCTIME

Returns the accumulated CPU time for a process.

```
  D
time:=PROCTIME;
```

PTAPE

Accepts input from paper tapes that do not contain X-OFF control characters.

```
          IV      IV
PTAPE(filenum1,filenum2);
```

PUTJCW

Assigns the value of a particular job control word (JCW) in the job control word table.

```
          LA      L      I
PUTJCW(jcwname,jcwvalue,jcwstatus);
```

QUIT

Aborts the calling process.

```
          IV
QUIT(num);
```

QUITPROG

Aborts the entire user process structure.

```
          IV  
QUITPROG(num);
```

READ

Reads an ASCII string from \$STDIN into an array.

```
          I          LA          IV  
length:=READ(message,msglength);
```

READX

Reads an ASCII string from \$STDINX into an array.

```
          I          LA          IV  
length:=READX(message,msglength);
```

RECEIVEMAIL

Receives mail from another process. Process handling (PH) capability is required.

```
          L          IV  LA          LV  
mailstatus:=RECEIVEMAIL(pin,location,waitflag);
```

RESETCONTROL

Allows a process to accept another CONTROL-Y signal.

```
RESETCONTROL;
```

RESETDUMP

Disables the abort stack analysis facility. Only the current process is affected.

```
RESETDUMP;
```

SEARCH

Searches a specially-formatted array for a specified entry or name.

```
      I      O-V   LA   IV      BA      BP  
entrynum:=SEARCH(buffer,length,dictionary,definition);
```

SENDMAIL

Sends mail to another process. Process handling (PH) capability is required.

```
      L              IV   IV   LA      LV  
mailstatus:=SENDMAIL(pin,length,location,waitflag);
```

SETDUMP

Arms a call to the system debugger from a process abort.

```
      LV  
SETDUMP(flags);
```

SETJCW

Sets bits in the system job control word (JCW).

```
      LV  
SETJCW(jcword);
```

STACKDUMP

Calls the system debugger to send a stack trace to \$STDLIST or to the file specified in the *formaldesig* parameter. Control then returns to the calling procedure.

```

          LA      IV      IV  DA
STACKDUMP(formaldesig,idnumber,flags,selec);
```

(SPL only):

```

O-V      BA      I      L  DA
STACKDUMP'(formaldesig,idnumber,flags,selec);
```

STARTSESS

Initiates a session on the specified terminal. Programmatic sessions (PS) capability is required.

```

          IV      BA      I  D      IA
STARTSESS(ldev,logonstring,jsid,jsnum,jsstatus);
```

SUSPEND

Suspends a process. Process handling (PH) capability is required.

```

O-V      LV  IV
SUSPEND(allow,rin);
```

SWITCHDB

Switches DB register pointer.

```

          L      O-P  LV
logindex:=SWITCHDB(index);
```

TERMINATE

Releases all resources held by the process and its descendants are released. All remaining files, opened by the process and its descendants, are closed and assigned the same disposition they had when opened.

```
TERMINATE;
```

TIMER

Returns system timer information.

```
D  
count:=TIMER;
```

UNLOADPROC

Dynamically unloads a compatibility mode (CM) segmented library (SL) procedure.

```
IV  
UNLOADPROC(procid);
```

UNLOCKGLORIN

The UNLOCKGLORIN intrinsic unlocks a global resource identification number (RIN) that was locked with the LOCKGLORIN intrinsic.

```
IV  
UNLOCKGLORIN(rinum);
```

UNLOCKLOCIN

The UNLOCKLOCIN intrinsic unlocks a local resource identification number (RIN) that was locked by the LOCKLOCIN intrinsic.

```

      IV
UNLOCKLOCIN(rinum);

```

WHO

Returns the access mode and attributes of the user calling the intrinsic.

```

O-V  D      D      D      BA
WHO(mode, capability, localattr, username,
    BA      BA      BA      L
    groupname, acctname, homename, term);

```

WRITELOG

Writes database and subsystem file records to the user logging file. User logging (LG) or system supervisor (OP) capability is required.

```

      D  LA  I  I  I
WRITELOG(index, data, length, mode, logstatus);

```

XARITRAP

Arms or disarms the user-written arithmetic trap handling procedure.

```

      IV  IV  I  I
XARITRAP(mask, plabel, oldmask, oldplabel);

```

XCONTRAP

Arms or disarms user-written CONTROL-Y trap handling procedure.

```
          IV      I
XCONTRAP(plabel,oldplabel);
```

XLIBTRAP

Enables or disables a user-written software library trap handling procedure.

```
          IV      I
XLIBTRAP(plabel,oldplabel);
```

XSYSTRAP

Enables or disables a user-written system trap handling procedure.

```
          IV      I
XSYSTRAP(plabel,oldplabel);
```

ZSIZE

Alters current DB to Z area of the compatibility (CM) mode stack.

```
          I          IV
newsize:=ZSIZE(size);
```

Segmenter

Initiate SEGMENTER

SEGMENTER [*listfile*]



Commands

Segmenter Command Functions

FUNCTION	COMMANDS		
ADD	ADDRL	ADDSL	
AUXILIARY USL FILE	AUXUSL	COPY	LISTAUX
CHANGES	NEWSEG		
COPY	CLEANSL CLEANUSL	COPY COPYSL	COPYUSL
CREATE	BUILDRL	BUILDSL	BUILDUSL
DELETE	PURGERBM	PURGERL	PURGESL
EXIT	E or EXIT		
GENERAL COMMANDS	HELP	REDO	
LIST	LISTAUX LISTRL	LISTSL LISTUSL	
FPMAP (SYMBOLIC DEBUG)	LISTPMAP	SETFPMAP	SHOW
OPEN	AUXUSL RL	SL USL	
RELOCATABLE BINARY MODULES (RBM)	AUXUSL CLEANUSL COPY	HIDE NEWSEG PREPARE	PURGERBM REVEAL USE
RELOCATABLE LIBRARY FILE (RL)	ADDRL BUILDRL	LISTRL PURGERL	SL
SEGMENTED LIBRARY FILE (SL)	ADDSL BUILDSL CLEANSL	COPYSL LISTSL PURGESL	SL
SYMBOLIC DEBUG	Refer to "FPMAP"		
USER SUBPROGRAM LIBRARY (USL)	AUXUSL BUILDUSL CEASE CLEANUSL	COPY COPYUSL LISTAUX LISTUSL	NEWSEG PREPARE PURGERBM USL

ADDRL

Adds a procedure to the currently managed RL file. You must have READ and WRITE access to the specified RL file.

ADDRL *name[(index)]*

ADDSL

Adds a procedure to an SL file.

ADDSL *name* [;PMAP] [;NOFFPMAP] [;FPMAP] [;NOSYM] [;PATCH=*patchsize*]

AUXUSL

Opens an auxiliary USL file from which RBMs can be transferred to the current USL file.

AUXUSL *filereference*

BUILDRL

Creates a permanent, formatted RL file. You must have SAVE capability in the group to which you are assigning the RL file.

BUILDRL *filereference,records,extents*

BUILDSL

Creates a permanent, formatted SL file. You must have SAVE capability in the group to which you are assigning the SL file.

BUILDSL *SL[.group],records,extents*
BUILDSL *filereference,records,extents*

BUILDUSL

Creates a new job/session temporary USL file. You must use SAVE to keep it as a permanent file.

BUILDUSL *filereference, records, extents*

CEASE

Deactivates one or more entry points in a currently managed USL file.

CEASE $\left[\begin{array}{l} \text{ENTRY,} \\ \text{UNIT,} \\ \text{SEGMENT,} \end{array} \right] \textit{name}[(\textit{index})]$

CLEANSL

Copies the contents of the currently-managed SL file to a new SL file, removing fragments of free space in each segment. File size is not changed.

CLEANSL [*filename*]

CLEANUSL

Copies only the active RBM of the currently managed USL file to a new USL file. File size is not changed.

CLEANUSL [*filename*]

COPY

Copies one or more RBMs from the auxiliary USL file to the currently managed USL file.

COPY $\left[\begin{array}{l} \text{UNIT,} \\ \text{SEGMENT,} \end{array} \right] \textit{name}[(\textit{index})]$

COPYSL

Copies the contents of the currently managed SL file to a new SL file, removing fragments of free space, while allowing the user to expand SL space by a given percentage.

`COPYSL percent[,filename] [;userformat]`

COPYUSL

Copies the entire currently managed USL file to a new USL file. No cleaning is performed. File size can be changed.

`COPYUSL percent[,filename]`

EXIT

Exits from the SEGMENTER, returning control to the MPE command interpreter.

`EXIT`
`E`

HELP

Accesses the HELP subsystem.

`HELP [HELP ] [tablecontents ] [command[,keyword] ]`
`[ALL ] [EXIT ]`

HIDE

Hides the named entry point. Can be used on a primary entry point, RBM or segment.

`HIDE [ENTRY,`
`UNIT,`
`SEGMENT, ] name[ (index) ]`

LISTAUX

Lists the procedures in the currently managed auxiliary USL file.

```
LISTAUX [segname]
```

LISTPMAP

Lists FPMAP information (program, segment, and procedure).

```
LISTPMAP progrname [ ; { segname  
                        procedurename } ]
```

LISTRL

Lists the procedure entry points and external references in the currently managed RL file.

```
LISTRL
```

LISTSL

Lists all procedure entry points and external references in the currently managed SL file.

```
LISTSL [ [ segmentnumber ] [ [ SEGMENT , ] segmentname ]  
        [ ENTRY , procedurename ] ]
```

LISTUSL

Lists the specified segments in the currently managed USL file.

```
LISTUSL [segname]
```

NEWSEG

Changes the segment name associated with an RBM in the currently managed USL file.

```
NEWSEG newsegname,rbmname[(index)]
```

PREPARE

Prepares the active RBMs in the currently managed USL file into a program.

```
PREPARE progfile
```

```
[ ;ZERODB ] [ ;PMAP ] [ ;MAXDATA=segsize ] [ ;STACK=stacksize ] [ ;DL=dlsiz ] [ ;CAP=caplist ]
[ ;RL=filename ] [ { ;FPMAP } ] [ { ;NOFFMAP } ] [ ;NOSYSM ] [ ;PATCH=patchsize ]
```

Note



For parameter definitions, refer to the MPE PREP command in the *MPE V Commands Reference Manual*.

PURGERBM

Deletes one or more RBMs from the currently managed USL file.

```
PURGERB [ UNIT,
          SEGMENT, ] name [(index)]
```

PURGERL

Deletes the entry point of a procedure, or the entire procedure, from the currently managed RL file.

```
PURGERL [ ENTRY,
          UNIT, ] name
```

PURGESL

Deletes a segment entry point, or the entire segment, from the currently managed SL file.

$$\text{PURGESL} \left[\begin{array}{l} \text{ENTRY,} \\ \text{SEGMENT,} \end{array} \right] \textit{name}$$

REDO

Allows you to edit a command entry.

$$\text{REDO} \textit{ [edit request]}$$

REVEAL

Reveals the entry point. Can be used on a primary entry point, RBM, or segment.

$$\text{REVEAL} \left[\begin{array}{l} \text{ENTRY,} \\ \text{UNIT,} \\ \text{SEGMENT,} \end{array} \right] \textit{name} [\textit{(index)}]$$

RL

Opens an RL file for management.

$$\text{RL} \textit{ filereference}$$

SETFPMAP

Sets the system or job/session FPMAP flag.

$$\text{SETFPMAP} \left[\left\{ \begin{array}{l} \text{SYSTEM} \\ \text{SESSION} \end{array} \right\} \left[; \left\{ \begin{array}{l} \text{CONDITIONED} \\ \text{UNCONDITIONED} \end{array} \right\} \right] \right] \left[; \left\{ \begin{array}{l} \text{ON} \\ \text{OFF} \end{array} \right\} \right]$$

SHOW

Displays the conditions of the system and job/session FPMAP flags.

SHOW

SL

Opens an SL file for management.

SL *filereference*

USE

Activates one or more program RBM entry points in the USL currently designed for management.

USE $\left[\begin{array}{l} \text{ENTRY,} \\ \text{UNIT,} \\ \text{SEGMENT,} \end{array} \right] \text{name}[(\text{index})]$

USL

Opens a USL file for management.

USL *filereference*

Intrinsics

ADJUSTUSLF

I IV IV
[errnum:=]ADJUSTUSLF(uslfnum,records);

CLEANUSLF

I IV BA
[filenum:=]CLEANUSLF(uslfnum,filename);

DUMPPMAP

V IV IV I IV I
(progfnnum,pmapfnnum,zpmappreclen,reccount,segnum,status);

EXPANDUSLF

I IV IV
[filenum:=]EXPANDUSLF(uslfnum,records);

FINDPMAPADDR

V IV IV V
(progfnnum,segnum,address,zpmapprec,
IV I
zpmappreclen,status);

FINDPMAPNAME

IV BA BA IA
(*progfnm, segname, entname, xpmaprec,*
V I
xmapreclen, status);

INITUSLF

I IV IA
[*errnum:=*]INITUSLF(*uslfnm, records*);

Debug

Operation

1. Direct call to Debug intrinsic.
2. Reaching breakpoint.
3. Set breakpoint on first executable instruction (DEBUG parameter of PREPRUN or RUN) or set the Debug bit (13:1) of the FLAG parameter of the CREATE intrinsic.
4. Execute Debug. The user must have privileged mode (PM) capability.

Command Operation

1. Bounds checking performed in user mode and when setting breakpoints.
2. Instruction where breakpoint occurred is not executed prior to entering Debug.
3. All numbers, address, and so on default to octal.
4. Logical arithmetic used throughout.
5. In privileged mode (PM), an improper *ldev* (logical device number) parameter supplied in a command may destroy the system.
6. Privileged mode users may access absolute addresses, systems, data, etc.
7. The Debug parameter of the RUN command is ignored if user does not have READ (R) and WRITE (W) access to the program file. Refer to the *MPE V Commands Reference Manual* (32033-90006).
8. All numbers are one-word values, assumed as octal. Double-word integers are not allowed.



Access

Determined by user capability (privileged vs. non-privileged). Not accessible from batch mode calls treated as no operation (NOP).

System Messages

BREAK*PRIV . *segment#* . *location*
DEBUG*PRIV . *segment#* . *location*
TRACE*PRIV . *segment#* . *location*

Note



If you do not have privileged mode (PM) capability, PRIV is not displayed. Only *segment#* and *location* are displayed.

Errors

SYNTAX	Invalid syntax
NO-NO	Invalid information was input
BOUNDS	Bounds violation
FULL	Breakpoint table full
SAME	System/private breakpoint conflict existing
CHECK	New breakpoint conflicts with existing breakpoint Byte index within command string is in error.

Breakpoint Commands

Note



When setting group and account breakpoints with a breakpoint command, use the prefix G (for group) or P (for account) before the segment identifier.

Set Group and Account Breakpoint:

```
[ G ] [ segment. ] offset ...
[ P ]
```

*AP

Allow Private User Breakpoints:

```
*AP
```

*AS

Allow system breakpoints:

```
*AS
```

B

Set Temporary Breakpoint (first time only):

```
B[segment.]offset[, ... ]
```

Set temporary breakpoint, break once on the *n*th execution (do not use on branch):

```
B[segment.]offset:n[, ... ]
```

Set temporary breakpoint, break once when *cond* is true:

```
B[segment.]offset:cond[, ... ]
```

Set temporary breakpoint, break always (do not use on branch):

```
B[segment.]offset:0[, ... ]
```

Set permanent breakpoint, break on every *n*th execution (do not use on branch):

```
B[segment.]offset:0n[, ... ]
```

Set permanent breakpoint, break every time *cond* is true (do not use on branch):

```
B[segment.]offset:0cond[, ... ]
```

BT

Set trace pointer - when trace pointer is reached, the register values are displayed, and the program continues to run:

$B[T][segment.]offset$

B@

List all breakpoints:

$B@$

*B

List all breakpoints in specified process:

$B[pin]@$

Operating system listing format for each breakpoint:

$LCST = \begin{bmatrix} P \\ G \end{bmatrix} lsn, P=pc, CST=asn, \begin{bmatrix} B \\ T \end{bmatrix} [@] t/u$

Where:

LCST	Logical code segment index
P	Account public segment library (SL)
G	Group segment library (SL)
lsn	Logical code segment table (CST) number
pc	Program counter
asn	Actual code segment table (CST) number
B	Breakpoint
T	Tracepoint
@	Permanent breakpoint. (No '@' indicates temporary breakpoint.)
t	Total number of executions allowed by conditional breakpoint.
u	Total number of times breakpoint actually executed.

***BA**

Set breakpoint absolute code segment table number (CST):

 $BA[segment.]offset[, \dots]$

***BAX**

Set breakpoint in absolute expanded code segment table number (CSTX):

 $BAX[segment.]offset[, \dots]$ Debug another process, use *pin* numbers:
$$B[T] \left[\left[pin. \right] \begin{bmatrix} A[X] \\ G \\ P \\ S \end{bmatrix} segment. \right] offset \left[: \left[\bullet \right] \begin{bmatrix} cond \\ count \end{bmatrix} \right] [, \dots]$$

***BS**

Debug system SL (segment library):

 $BS[segment.]offset[, \dots]$

C

Clear all user group (G) or account (P) breakpoints:

$$C \left[\begin{bmatrix} G \\ P \end{bmatrix} segment. \right] offset [, \dots]$$

C@

Clear all user breakpoints or tracepoints:

C@

***C**

Clear breakpoints in another process, use *pin* number:

$$C \left[\begin{bmatrix} S \\ A \end{bmatrix} \begin{bmatrix} X \end{bmatrix} \right] segment. \left[offset [, \dots] \right]$$

Where:

S Indicates breakpoints in the system Segment Library (SL).

A Indicates breakpoints in absolute code segment table (CST).

AX Indicates breakpoints in absolute expanded code segment table (CSTX).

$$C \left[\begin{bmatrix} pin \end{bmatrix} \begin{bmatrix} A \\ G \\ P \\ S \end{bmatrix} \begin{bmatrix} X \end{bmatrix} \right] segment. \left[offset [, \dots] \right]$$

***C@**

Clear all system-owned breakpoints if operating in AS mode or all user-owned breakpoints if operating in AP mode:

C@

***C**

Clears all breakpoints in the specified process:

C[*pin*.]@

E@

Terminate program:

E@

E

Execute EXIT *n* when *n* is **expr**:

E[expr]

R

Resume execution and optionally set another breakpoint:

$$R \left[\left[\text{segment.} \right] \text{offset} \left[: \left[\text{Q} \right] \text{cond} \right] \right]$$

To resume and set a group or account segmented library breakpoint, use prefix G (for group) or P (for account) before the segment parameter:

$$R \left[\left[\left\{ \begin{array}{c} G \\ P \end{array} \right\} \text{segment.} \right] \text{offset} \left[: \left[\text{Q} \right] \text{count} \right] \right]$$

To resume and set a breakpoint for system segmented library with prefix S before the segment parameter:

$$R \left[\left[S \text{ segment.} \right] \text{offset} \left[: \left[\text{Q} \right] \text{count} \right] \right]$$

To resume and set a breakpoint in another process, use *pin* number:

$$R \left[\left[\left[\begin{array}{c} A \left[x \right] \\ G \\ P \\ S \end{array} \right] \text{pin.} \right] \left[\text{segment} \right] \text{offset} \left[\left[\left[\text{Q} \right] \left[\begin{array}{c} \text{cond} \\ \text{count} \end{array} \right] \right] \left[, \dots \right] \right] \right]$$

Display Commands

Note



The bank number in EA mode follows EA; for example:

D EA2+10 Displays one word location 10 of bank 2.

D EA2.10 Displays one word location 10 of bank 2.

For CO and DA, the offset immediately follows the mnemonic (CO or DA) unless it is an expression involving a calculation when it is enclosed in parentheses, for example:

D DA22+6,6 Displays 6 words starting at location 6 data segment 22.

D CO(4+6),3 Displays first 3 words of segment 12 (octal).

D

Display memory:

D[*dispbse*] [*offset*] [, *count*] [, *mode*]

Where:

dispbse Stack (data) relative display bases (DB, DL, Q, S) DB is the default.
Code relative display bases (PB, P, PL).

offset Location at which display begins. The colon (:) specifies preceding *expr* is an indirect address, as in D Q+5: ,#8 ,A.

count Number of locations to display.

mode

O Octal (default)

I Decimal integer

H Hexadecimal

C Code

A ASCII

***D A**

Absolute relative (base = location 0):

D A

***D CO**

Code segment relative (base = base of segment):

D CO

***D COX**

Absolute expanded code segment (CSTX):

D COX

***D DA**

Data segment relative (base = base of segment):

D DA

***D DX**

Current absolute DB relative (base = absolute DB):

D DX

***D EA**

Extended absolute address (base = bank specified):

D EA

***D SY**

System global relative (base = system base):

D SY

DR

Displays registers:

DR[,register][, ...]

register = DL, MDST, MF, Q, S, Z, X, ST, P, 1, 2, 3, 4

***DR**

Displaying all registers (register parameter omitted), also includes the following values:

DR

PCBProcess control block index.

CSTAbsolute code segment index.

CSTXAbsolute expanded code segment index.

STAXStack segment index.

DSTExtra data segment index.

DX Current value of DB register, if in absolute mode.

EA Current bank number, if in absolute mode.

In addition, the segment number displayed as LCST is preceded by S for a system library segment.

***DV**

Display virtual memory:

`DV[ldev] +startsector[,count][,mode]`

Where:

ldev Logical device number

count Number of sectors displayed.

mode O Octal (default)
 I Decimal integer
 A ASCII

Startsector Signifies the starting sector address to be displayed. If the sector address requires more than 16 bits, it must be entered as:

low-order bits: high-order bits

Listing Commands

L

Direct list output to *fileref* file. If no *fileref*, switch back to user's terminal:

`L[fileref]`

L0

Closes an open file and switches back to terminal:

`L0`

*L

Direct list output to *ldev* device. If no *ldev*, switch back to user's terminal:

L[*ldev*]

Memory/Register Modification Commands

M

Modify memory:

M[[*modbase*] [*offset*] [, *count*] [*mode*]]

Where:

<i>modbase</i>	DB, DL, Q, S	
<i>offset</i>	Location at which modification begins.	
<i>count</i>	Number of locations to modify.	
<i>mode</i>	O	Octal
	I	Decimal integer
	A	ASCII
	C	Code
	Return	Leaves value unchanged and terminates command.

In privileged mode the following values are allowed for *modbase*:

A	Absolute 0
SY	System global
DA (dst)	Data segment number DST
DX	Absolute DB
EA	Extended absolute address

MR

Modify *register*:

MR[, *register*][, ...]

register DL, Q, S, Z, X, ST, P, 1, 2, 3, 4

(Return) Leaves value unchanged and terminates command.

ST.(2:7) may be changed. In privileged mode, all ST bits can be changed.

DL<=0<=Q<=S<=Z

DL and Z expand and contract only in blocks of %200 words, limited by MAXDATA, DL, and STACKSIZE parameters in the PREP, PREPRUN, or RUN commands.

If register omitted, all registers between DL and Z are modified.

\$register

Modify single register value:

\$register: = *expr*

Where:

expr Signifies new value.

register Can be ST, X, DL, Q, S, Z, P, 1, 2, 3, 4

Calculation Display Command

=expr

Calculate and display *expr* value:

=expr[, *mode*]

<i>mode</i>	O	Octal
	A	ASCII
	I	Decimal integer
	H	Hexadecimal
	C	Code

Example: **=#4+(55*#12)-"A",I**

=expr

Trace Command

T

Trace stack markers:

T[MF=*n*]

Where:

MF=1 The segment is physically mapped.

MF=0 The segment is logically mapped.

Displays Q-displacement, LCST, P (relative) for markers, not including initial stack marker.

For privileged mode (PM) T also displays absolute CST.

Listing format for each marker is:

$$Q-dq, LCST = \begin{bmatrix} P \\ G \\ S \end{bmatrix} \quad lsn, P=pc, CST=asn, \left[NF = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right]$$

Where:

LCST Logical code segment index

dq Displacement from current Q

P Account public segment library (SL)

G Group segment library (SL)

S System segment library (SL)

lsn Logical code segment(CST)

pc P (relative) address

asn Absolute (actual) code segment (CST)

Segment Freeze Commands

Caution



Attempting to freeze a segment that is already frozen or to unfreeze a segment that has already been unfrozen will cause a system failure. (Be especially careful when freezing segments; some may have already been frozen by the system.)

*F

Freeze code or data segment in memory:

$$F \left\{ \begin{array}{l} CO \\ DA \end{array} \right\} [X] \text{ } segment$$

Where:

CO	Code segment
COX	Absolute CSTX code segment
DA	DA segment

*U

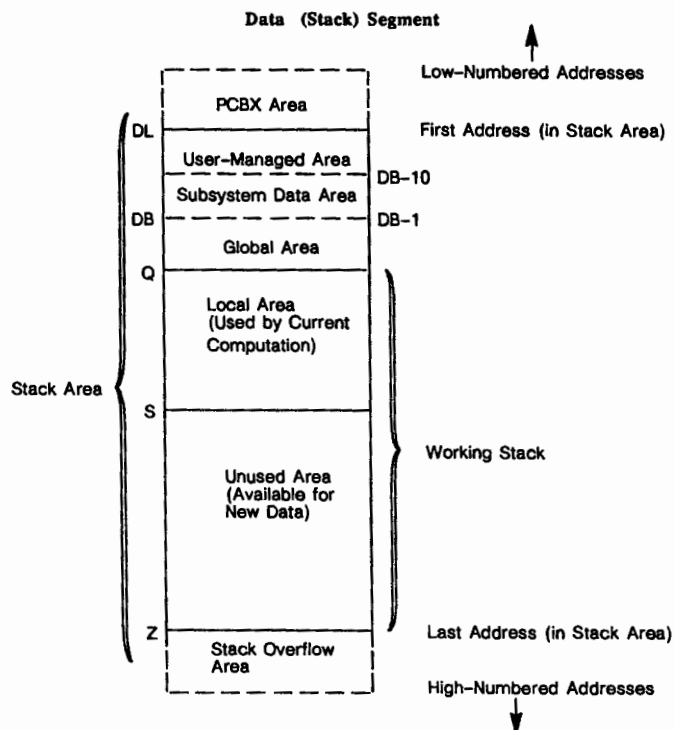
Unfreeze frozen code or data segment in memory:

$$U \left\{ \begin{array}{l} CO \\ DA \end{array} \right\} [X] \text{ } segment$$

Where:

CO	Code segment
COX	Absolute CSTX code segment
DA	DA segment

Segment and Register Contents



- DL - Data Limit
- DB - Data Base (pointer to Global Area)
- Q - Stack Marker
- S - TOS
- Z - Last Stack Address

LG200028_029

Stack Marker Format

Stack Marker

Q-3	X (Index Register)
Q-2	Prel (P+1-PB)
Q-1	Status
Q-0	ΔQ

Q-3 = Current contents of X-register.

Q-2 = Return address of code segment (P+1, PB rel.)

Bit 0: 0=normal
1=control-Y or trace bit

Bit 1: 0=logically mapped
1=physically mapped

Q-1 = Current contents of status register.

Q-0 = ΔQ : the number of words between new and previous Q.

Note: *Parm* parameter of :RUN command found in Q-4 of outer block or main program.

LG200028_050

Status Register

STATUS REGISTER

			J			K									
0	1	-	3	4	-	6	7	-	9	10	-	12	13	-	15
M I T R O C						CC		SEGMENT NUMBER							

NUMERIC

G OP1>OP2
OP>0

Alpha

E OP1=OP2
OP=0

Special

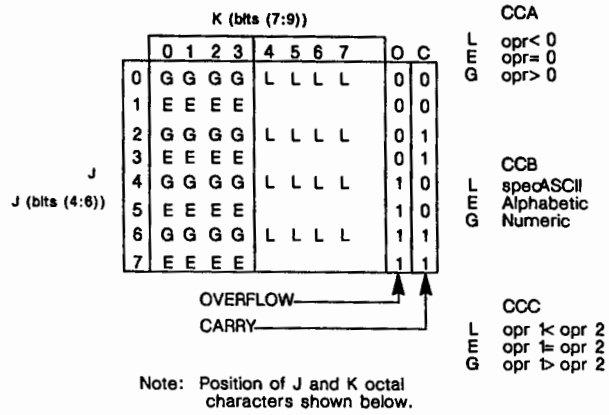
L OP1>OP2
OP>0

6-7 CC field

0	> G
1	< L
2	= E
3	Undef.

LG200028_025a

Condition Codes



CCA

L opr < 0
E opr = 0
G opr > 0

CCB

L specASCII
E Alphabetic
G Numeric

CCC

L opr 1 < opr 2
E opr 1 = opr 2
G opr 1 > opr 2

CCD

L Non-responding device controller.
E Responding device controller, or device ready.
G Device not ready (busy).

LG200028_026

PMAP Format

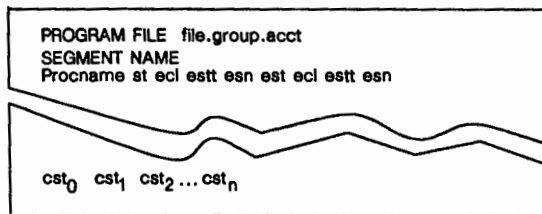
PROGRAM FILE file.group.acct				
Segname Isn				
NAME	STT	CODE	ENTRY	SEG
ep/pname	en	begloc	eploc	Isn-ep
SEGMENT LENGTH		seglng		
PRIMARY DB pdb INITIAL STACK is CAPABILITY cap SECONDARY DB sdb INITIAL DL idl TOTAL CODE tc TOTAL DB tdb MAXIMUM DATA mdl TOTAL REC tr ELAPSED TIME: et PROCESSOR TIME: cpu				

file.group.acct = Program file name.
 segname = Segment name.
 Isn = Logical segment no.
 ep/pname = Name of program unit entry-point procedure.
 en = Assigned entry no. in Segment Transfer Table.
 begloc = Beginning location of procedure code in segment.
 eploc = Location of entry-point in segment.
 Isn-ep = Logical segment number of segment containing this external procedure.
 seglng = Segment length (words).
 pdb = Primary DB area size.
 sdb = Secondary DB area size.
 tdb = Total DB area size.
 et = Preparation time elapsed (minutes).
 is = Initial stack size.
 idl = Initial DL size.
 mdl = Maximum area available for data (Z-DL).
 cap = Program file capability.
 tc = Total code in file.
 tr = Total records in file.
 cpu = CPU time used for preparation.

Note: All numbers in octal.

LG200028_027

LMAP Format



file.group.acct = Program file name.
 Procname = External procedure name.
 st = Type of segment referencing external procedure:

PROG = Program
 GSL = Group Segment Library
 PSL = Public Segment Library

ecl = External parameter checking level.
 estt = External segment transfer table (STT) no.
 esn = External logical segment no.
 est = Entry point segment type:

GSL: Group Segment Library
 PSL: Public Segment Library
 SSL: System Segment Library

ecl = Entry-point parameter checking level.
 estt = Entry-point segment transfer table (STT) no.
 esn = Entry-point logical segment no.

cst₀...cst_n = List of code segment table nos. to which
 program file segments were assigned, ordered by logical
 segment no. (0,1,2,...n, reading left-to-right.)

Note: All numbers in octal.

LQ200028_028

FCOPY

Initiate FCOPY

RUN FCOPY.PUB.SYS

$$\left[; \text{INFO} = \text{"FROM"} \left[= \left\{ \begin{array}{c} \text{fname} \\ * \\ [\text{empty}] \end{array} \right\} \right] \text{TO} \left[= \left\{ \begin{array}{c} (dfile, kfile) \\ \text{fname} \\ * \\ [\text{empty}] \end{array} \right\} \right] \left[; \text{optlist} \right] \right]$$

or

$$\text{FCOPY} \left[\text{FROM} \left[= \left\{ \begin{array}{c} \text{fname} \\ * \\ [\text{empty}] \end{array} \right\} \right] ; \text{TO} \left[= \left\{ \begin{array}{c} dfile, kfile \\ \text{fname} \\ * \\ [\text{empty}] \end{array} \right\} \right] \left[; \text{optlist} \right] \right]$$

Commands

$$\text{FROM} \left[= \left\{ \begin{array}{c} \text{fname} \\ * \\ [\text{empty}] \\ \$\text{CTUL} \\ \$\text{CTUR} \end{array} \right\} \right] ; \text{TO} \left[= \left\{ \begin{array}{c} (dfile, kfile) \\ \text{fname} \\ * \\ [\text{empty}] \\ \$\text{CTUL} \\ \$\text{CTUR} \\ \$\text{NULL} \\ \$\text{HARD} \end{array} \right\} \right]$$

Keywords

Specify Carriage Control:

$$\left[\begin{array}{l} ;\text{CCTL} \\ ;\text{NOCCTL} \end{array} \right]$$

Deblock Blocked Records:

$$[; \text{DEBLOCK} = \text{logical-record-length}]$$

Copy Multiple Tape Files:

$$\left[; \text{FILES} = \left\{ \begin{array}{l} \text{number of files} \\ \text{ALL} \end{array} \right\} \right]$$

Translate Code:

$$\left[\begin{array}{l} ;\text{EBDICIN} \\ ;\text{EBDICOUT} \\ ;\text{BCDICIN} \\ ;\text{BCDICOUT} \\ ;\text{EBCDIKIN} \\ ;\text{EBCDIKOUT} \end{array} \right] \left[= \left\{ \begin{array}{l} \text{field} \\ (\text{field} [, \dots]) \end{array} \right\} [, \text{EXCLUDE}] \right]$$

Select Record Subsets:

$$\left[; \text{SUBSET} \left[\begin{array}{l} = \text{"characterstring"} [, [\text{column}] [, \text{EXCLUDE}]] \\ = \# \text{patternlist\#} [, [\text{column}] [, \text{EXCLUDE}]] \\ = (\text{range} [; \text{range}] [; \dots]) \end{array} \right] \right]$$

Where *range* is either:

$$\begin{array}{c} [\text{starting-record-number}] [, \text{number-of-records}] \\ \text{or} \\ [\text{starting-record-number}] [: \text{last-record-number}] \end{array}$$

Shift:

$$[; \text{UPSHIFT}]$$

Skip EOF:

$$\left[; \text{SKIPEOF} = \left[\begin{array}{l} \left\{ \begin{array}{l} + \\ - \end{array} \right\} \text{from-eofs} \\ \text{from-filenum} \end{array} \right] \left[, \left\{ \begin{array}{l} \left\{ \begin{array}{l} + \\ - \end{array} \right\} \text{to-eofs} \\ \text{to-filenum} \end{array} \right\} \right] \right]$$

Ignore Errors:

$$[; \text{IGNERR} = [\text{number-of-errors}]]$$

Create New File:

$$[; \text{NEW}]$$

Verify Copy:

$$[; \text{VERIFY} = [\text{number-of-errors}]]$$

7-2 FCOPY

Compare File to File:

[;COMPARE[=*number-of-errors*]]

Display Numeric Codes:

$$\left[; \left\{ \begin{array}{c} \text{OCTAL} \\ \text{HEX} \end{array} \right\} \left[; \left\{ \begin{array}{c} \text{CHAR} \\ \text{CLEAR} \\ \text{KANA} \end{array} \right\} \right] \left[; \text{NORECNUM} \right] \left[; \text{TITLE} = "title" \right] \right]$$

Display Characters:

$$\left[; \left\{ \begin{array}{c} \text{CHAR} \\ \text{CLEAR} \\ \text{KANA} \end{array} \right\} \left[; \left\{ \begin{array}{c} \text{HEX} \\ \text{OCTAL} \end{array} \right\} \right] \left[; \text{NORECNUM} \right] \left[; \text{TITLE} = "title" \right] \right]$$

Determine Copy Sequence for KSAM File:

[;KEY=*character location*]

Copy Data From KSAM File Only:

[;NOKSAM]

Do Not Copy User Labels:

[;NOUSERLABELS]

Native Language (used only with UPSHIFT, EBCDICIN, or EBCDICOUT function):

$$\left[; \text{LANG} = \left\{ \begin{array}{c} \text{langname} \\ \text{langnum} \end{array} \right\} \right]$$

Examples

To copy three disk files to tape and require three replies:

```
file t;dev=tape
run fcopy.pub.sys
from=file1;to=*t
from=file2;to=*;skipeof=,2
from=file3;to=*;skipeof=,3
```

To copy three files from tape to disk with the same record size:

```
run fcopy.pub.sys
from = *t;to=filea;new
from = *;to=fileb;skipeof=+1;new
from = *;to=filec;skipeof=+1;new
exit
```


SORT/MERGE

Initiate SORT

RUN SORT.PUB.SYS



SORT Commands

A

A[ltseq] *modspec1* [, *modspec2*] [, ...]

$$\text{modspec} = \begin{bmatrix} \text{EACH} \\ \text{MERGE} \end{bmatrix} \left\{ \begin{array}{l} \text{string spec} \\ \text{numbyte spec} \\ \text{range string spec} \end{array} \right\} \begin{bmatrix} = \\ \text{WITH} \end{bmatrix} \left\{ \begin{array}{l} \text{string spec} \\ \text{numbyte spec} \\ \text{range string spec} \end{array} \right\}$$

DATA

DATA [IS] $\left\{ \begin{array}{l} \text{ASCII} \\ \text{EBCDIC} \end{array} \right\}$ [,] SEQ[uence] [IS] $\left\{ \begin{array}{l} \text{ASCII} \\ \text{EBCDIC} \end{array} \right\}$

E

E[nd]

EX

EX[it]

I
$$I[\text{input}] \left\{ \begin{array}{l} * \\ \$STDIN[X] \\ fname \\ (fname1[,fname2][, \dots]) \end{array} \right\} [, \#records][, recsize]$$

K

K[ey] *keyspec1* [, *keyspec2*] [, ...]

Where:

keyspec *position,length[,type][,DESC]*

type BYTE,INT,DOUBLE,REAL,LONG,PACKED,PACKED*,DISPLAY-
TRAILING-SIGN, DISPLAY-LEADING-SIGN, DISPLAY-TRAILING-
SIGN-SEPARATE, DISPLAY-LEADING-SIGN-SEPARATE, or
CHARACTER.

L
$$L[\text{language}] \quad [IS] \quad \left\{ \begin{array}{l} langid \\ langname \end{array} \right\}$$

O
$$O[\text{output}] \left\{ \begin{array}{l} filename \\ * \\ \$STDLIST \end{array} \right\} [, NUM][, KEY]$$

8-2 SORT/MERGE

RESET

RESET

SH

SH[OW] $\left\{ \begin{array}{l} \text{SEQUENCE[,OFFLINE]} \\ \text{TABLE[,OFFLINE]} \\ \text{NOSEQUENCE} \\ \text{NOTABLE} \end{array} \right\}$

V

V[erify]

:

: [MPE command]

SORT Formal File Designators

Table 8-1. SORT Formal File Designators

Filename	Description
LIST	List File
TEXT	Text File
DISPLOUT	Display File
INPUT	Input File
OUTPUT	Output File

:

SORT Intrinsic

SORTINIT

```
          IA          IA          IV          IV          DV
SORTINIT([inputfiles] [, outputfiles] [, outuptoption] [, reclen] [, numrecs]
          IV          IA          IA          LP          P          IA
[, numkeys] [, keys] [, altseq] [, keycompare] [, errorproc] [, statistics]
          L          I          I          IA          I
[, failure] [, errorparm] [, spaceallocation] [, charseq] [, parm2]);
```

SORTINPUT

```
          BA          IV
SORTINPUT([record] [, length]);
```

SORTOUTPUT

```
          BA          IV
SORTOUTPUT([record] [, length]);
```

SORTEND

```
SORTEND;
```

SORTSTAT

```
          IA
SORTSTAT([statistics]);
```

SORTTITLE

```
SORTTITLE;
```

SORTERRORMESS

```
          IV          BA          I
SORTERRORMESS([errorcode] [, message] [, length]);
```

SORTINITIALF

```
          IV          IV          IV          IV          DV
SORTINITIALF([inputfile] [, outputfile] [, outuptoption] [, reclen] [, numrecs]
          IV          IA          P          LP          IA          L
[, numkeys] [, keys] [, errorsproc] [, keycompare] [, statistics] [, failure]);
```

8-4 SORT/MERGE

SORTINITIAL

IV IV IV IV DV
SORTINITIAL([inputfile] [, outputfile] [, outputoption] [, reclen] [, numrecs] ,
IV IA P LP IA L
[, numkeys] [, keys] [, errorsproc] [, keycompare] [, statistics] [, failure]);

Initiate MERGE

RUN MERGE.PUB.SYS

MERGE Commands

A

A[ltseq] modspec1[, modspec2] [, ...]

modspec= $\begin{Bmatrix} \text{EACH} \\ \text{MERGE} \end{Bmatrix} \begin{Bmatrix} \text{string spec} \\ \text{numbyte spec} \\ \text{range string spec} \end{Bmatrix} \begin{Bmatrix} = \\ \text{WITH} \end{Bmatrix} \begin{Bmatrix} \text{string spec} \\ \text{numbyte spec} \\ \text{range string spec} \end{Bmatrix}$

DATA

DATA [IS] $\begin{Bmatrix} \text{ASCII} \\ \text{EBCDIC} \end{Bmatrix}$ [,] SEQ[uence] [IS] $\begin{Bmatrix} \text{ASCII} \\ \text{EBCDIC} \end{Bmatrix}$

E

E[nd]

EX

EX[it]

I

I[nput] $\begin{Bmatrix} * \\ \$\text{STDIN}[X] \\ \text{fname} \\ \text{fname1}, \text{fname2} \end{Bmatrix} [, \text{fname3}] [, \dots]$

K

$K[ey] \text{ } keyspec1[;keyspec2] [; \dots]$

Where:

$keyspec$ $position, length[, type][, DESC]$

$type$ BYTE, INT, DOUBLE, REAL, LONG, PACKED, PACKED*,
DISPLAY-TRAILING-SIGN, DISPLAY-LEADING SIGN, DISPLAY-
TRAILING-SIGN-SEPARATE, DISPLAY-LEADING-SIGN-SEPARATE, or
CHARACTER.

L

$L[anguage] [IS] \left\{ \begin{array}{l} langid \\ langname \end{array} \right\}$

O

$O[utput] \left\{ \begin{array}{l} filename \\ * \\ \$STDLIST \end{array} \right\} [, num \ records] [, KEY]$

RESET

RESET

SH

$SH[OW] \left\{ \begin{array}{l} RSEQUENCE[, OFFLINE] \\ GTABLE[, OFFLINE] \\ TNOSEQUENCE \\ FNOTABLE \end{array} \right\}$

V

$V[erify]$

:

: [MPE command]

MERGE Formal File Designators

Table 8-2. MERGE Formal File Designators

Filename	Description
LIST	List File
TEXT	Text File
DISPLOUT	Display File
INPUT	Input File
OUTPUT	Output File

MERGE Intrinsic

MERGEINIT

```

      IA      P      IA      P
MERGEINIT([inputfiles] [,preprocessor] [,outputfiles] [,postprocessor]
      LV      IV      IA      IA      LP      P      IA
[,keyonly] [,numkeys] [,keys] [,allseq] [,keycompare] [,errorproc] [,statistics]
      L      I      I      IA      I
[,failure] [,errorparm] [,spaceallocation] [,charseq] [,parm#]);

```

MERGEOUTPUT

```

      BA      I
MERGEOUTPUT([record] [,length]);

```

MERGEEND

```

MERGEEND;

```

MERGESTAT

```

      IA
MERGESTAT([statistics]);

```

MERGETITLE

```

MERGETITLE;

```

MERGEERRORMESS

```
IV      BA      I  
MERGEERRORMESS([errorcode][,message][,length]);
```

MERGE

```
IV      IA      IV      IV      IV  
MERGE([numinputfiles][,inputfile][,outputfile][,keyonly][,numkeys],  
IA      P      P      P      LP  
[,keys][,preprocessor][,postprocessor][,errorproc][,keycompare]  
IA      L  
[,statistics][,failure]);
```

Edit

Initiate EDITOR

EDITOR

or

FILE *name*;DEV= $\left\{ \begin{array}{l} ldev \\ devclass \end{array} \right\}$

EDITOR **name*

or

FILE EDTTEXT=*name*
RUN EDITOR.PUB.SYS,BASICENTRY

Formal file designator is EDTLIST.



Commands

A

A[*dd*] [*Q*] [*linenumber*] [,HOLD[*Q*] [,NOW]]

B

B[egin] [*Q*]

C

$C[hange][Q] \left\{ \begin{array}{l} col [/col] \\ string \end{array} \right\} TO string [IN rangelist]$

Where:

$rangelist = range, range, \dots$

CO

$CO[py][Q] range TO linenumber [BY increment]$

D

$D[delete][Q][rangelist]$

Where:

$rangelist = range, range, \dots$

E

$\left\{ \begin{array}{l} E[nd] \\ E[xit] \end{array} \right\}$

F

$F[ind][Q] range$

Second position in *range* is upper bound.

G

$$G[ather][Q]\left\{\begin{array}{l} range\ TO\ linenumber \\ ALL[TO\ linenumber] \end{array}\right\}[BY\ increment]$$

Renumbers the textfile:

$$G[ather]\ ALL$$

H

$$H[old][Q][range[,APPEND]]$$

Null range implies clear hold file.

I

$$I[nsert][Q]position[BY\ increment][,HOLD[Q][,NOW]]$$

J

$$J[oin][Q]\ filename\ \left[\left\langle\left\{\begin{array}{l} flinenumber[flinenumber] \\ filerecnum[filerecnum] \end{array}\right\}\right\rangle\right]$$

$$[TO\ linenumber]\ [BY\ increment]\ [[,]\ UNN[umbered]\]$$

Where: *flinenumber* and *filerecnum* apply to filename; *linenumber* applies to workfile.

K

$$\left\{\begin{array}{l} K[eeP]\ [filename]\ [range]\ [,UNN[umbered]] \\ K[eeP]\ [Q]\ filename \end{array}\right\}$$

The Q option means quick, not quiet, and does not allow *range* and UNN[umbered].

L

$L[ist][Q][range][,UNN[umbered]][,OFFLINE][,TRANSLATE][,NOTEXT]$

M

$M[odify][Q][rangelist]$

Where:

$rangelist = range, range, \dots$

N

$N[ot]$

O

$O[r]$

P

$P[rocedure] \left[procedurename, \begin{bmatrix} S \\ P \\ G \end{bmatrix} [,rangelist] \right]$

Where:

$rangelist = range, range, \dots$

Q

Q [string]
Z::

R

R[replace][Q][rangelist][,HOLD[Q][,NOW]]''

Where:

rangelist=range,range, ...

S

S[et] optionlist

Options are separated by commas:

optionlist: FROM=linenumber DELTA=increment
DEPTH=integer LENGTH=colnum
LEFT=colnum RIGHT=colnum
TIMES[S]=integer SIZE=integer
LINES=mazlines TABCHAR=[string]

[LONG] [REAR] [DISPLAY] [FIXED] [POLL]
[SHORT] [FRONT] [QUIET] [VARIABLE] [BATCH]

[FORMAT={ DEFAULT }] [TABS={ colnum[,colnum] ... }]
[COBOL] [NOTABS]

T

T[ext] [file [{ linenum/linenum
#recnum/#recnum/#recnum }] [,UNN[umbered]]]

U

U[se][file]

V

V[erify] *optionlist*

Where: *optionlist* includes all SET command keywords, plus TOTAL, FILES, and ALL.

W

W[hile](FLAG)

To find all locations of a string in a text file and allow the user to modify lines in textfile, enter:

```
/WHILE  
/FIND "string"  
/MODIFY*
```

X

X[plain] $\left[\begin{array}{l} \textit{command} \\ \text{ALL} \end{array} \right]$ [,OFFLINE]

Y

Y[es]

:

Z

`Z::=`

Use to set up a command parameter string. Then use `Z::` embedded within a command to invoke this string.

:

`: [ MPE command ]`

MPE commands may be passed in this way only if they are programmatically executable.

TurboIMAGE

Initiate DBSCHEMA

```
RUN DBSCHEMA.PUB.SYS[;PARM=n]
```

Where:

- $n=1$ If an actual file designator has been equated to DBSTEXT.
 $n=2$ If an actual file designator has been equated to DBSLIST.
 $n=3$ If actual file designators have been equated to both DBSTEXT and DBSLIST.

File	Use	Formal File Designator	Default Actual File Designator
textfile	Schema and Schema Processor commands	DBSTEXT	\$STDINX
listfile	output listing	DBSLIST	\$STDLIST

```
file dbstext=orderssc
file dbslst;dev=lp
run dbschema.pub.sys;parm=3
```



Commands

```
$PAGE ["character-string"] [ , ... ]

$page "orders data base schema", "version2"

$CONTROL [ LIST
          NOLIST ] [ ,ERRORS=nnn ] [ ,LINES=nnnnn ]

[ ,ROOT
  ,NOROOT ] [ ,BLOCKMAX=nnnn ] [ ,TABLE
  ,NOTABLE ]

$control list,errors=50,lines=60,root,blockmax=512,table

$TITLE ["character-string"] [ , ... ]

$title "orders schema-preliminary"
```

Schema Structure

```
BEGIN DATA BASE data base name [,LANGUAGE: language];
PASSWORDS: password part
ITEMS: item part
SETS: set part
END.
```

The form of the *password part* is:

```
user class number [password];
```

```
.
```

```
user class number [password];
```

The form of the *item part* is:

```
item name, [sub-item count] type designator [sub-item length]  
[(read/write class list)];
```

The form of the *set part* for master data sets is:

```
{ NAME: } set name, { MANUAL }  
{ N: } [(read/write class list)] [, device class];  
{ M }  
{ AUTOMATIC }  
{ A }
```

```
{ ENTRY: } item name [(path count)],  
{ E: }
```

```
.
```

```
item name [(path count)];
```

```
{ CAPACITY } maximum entry count;  
{ C: }
```

The form of the *set part* for detail data sets is:

```
{ NAME: } set name, { DETAIL }  
{ N: } [(read/write class list)] [, device class];  
{ D }
```

```
{ ENTRY } item name [ ( [ ! ] master set name [ ( sort item name ) ] ) ],  
{ E: }
```

```
.
```

item name [(*!*)*master set name* [(*sort item name*)]]);

{ CAPACITY: } *mazimum entry count*;
 { C: }

Initiate DBLOAD

[FILE DBLOAD[=*file name*] [;*DEV=device*]]

RUN DBLOAD.PUB.SYS

WHICH DATA BASE? *data base name* [/maintenance word]

DATA SET *n*: *z* ENTRIES

.

.

.

END OF VOLUME *m*, *y* READ ERRORS RECOVERED

DBLOAD OPERATION COMPLETED

END OF PROGRAM

Where:

- n* The number of the data set loaded from the backup volume.
- z* The number of entries loaded into the data set.
- m* The number of the volume.
- y* The number of READ errors from which DBLOAD has recovered.

Note: *z* will equal zero if the data set is an automatic master.

Initiate DBRECOV

RUN DBRECOV.PUB.SYS [,OPTION]

Options:

[	RESTART	]
	ABORT	
	PURGE	]

Commands

CONTROL *param* [,*param* ...]

Param may be:

ABORTS, EOF=*nnnn*, ERRORS=*nnnn*, MODEX, MODE4, NOABORTS, NOSTAMP, NOSTATS
NOSTORE, STAMP, STATS, STOPTIME=*mm/dd/yy hh:mm*, or STORE

>control nostore,stats

EXIT

FILE *fileref*, *userref* [,*rmode*,*fmode*]

file part,mary.mktg,2,2

PRINT { DBTABLE
 FILETABLE }

RECOVER *data base name* [/maintenance word] [.group[.account]]
 [,*data base 2* ...]

recover orders/sell.image.datamgmt

ROLLBACK *data base name* [/maintenance word] [.group[.account]]
 [,*data base2* ...]

RUN

Initiate DBRESTOR

```
[FILE DBRESTOR [=filename] [ ;DEV=device] [ ;BUF  
;NOBUF ]  
  
RUN DBRESTOR.PUB.SYS  
WHICH DATA BASE? data base name [/maintenance word]  
DATA BASE RESTORED  
END OF PROGRAM  
  
file dbrestor=r;dev=tape;buf  
run dbrestor.pub.sys  
WHICH DATA BASE? orders/sell
```

Initiate DBSTORE

```
[ FILE DBSTORE [=filename] [ ;DEV=device] [ ;BUF  
;NOBUF ] ]  
  
RUN DBSTORE.PUB.SYS  
WHICH DATA BASE? data base name [/maintenance word]  
DATA BASE STORED  
END OF PROGRAM
```

Initiate DBUNLOAD

[FILE DBLOAD[=*filename*] [;DEV=*device*]

RUN DBUNLOAD.PUB.SYS [;CHAINED
[;SERIAL]

CHAINED is the default.

WHICH DATA BASE? *data base name* [/maintenance word]

DATA SET *n*: *x* ENTRIES EXPECTED, *x* ENTRIES PROCESSED

.

.

END OF VOLUME *m*, *y*, WRITE ERRORS RECOVERED

DATA BASE UNLOADED

END OF PROGRAM

Where:

- | | |
|----------|--|
| <i>n</i> | The number of data sets in the data base. |
| <i>x</i> | The number of entries expected and the number of entries copied. |
| <i>m</i> | The number of the volume. |
| <i>y</i> | The number of write errors from which DBNUNLOAD has recovered. |

Intitiate DBUTIL

RUN DBUTIL.PUB.SYS

Commands

ACT[ivate] *data-base-access file name*

>>act orddb

CRE[ate] *data base name [/maintenance word]*

DEA[ctivate] *data-base-access filename*

DIS[able] <i>data base name [/maintenance word]</i> FOR	{ LOGGING RECOVERY ACCESS DUMPING ILR AUTODEFER ROLLBACK }
---	--

>>disable orders/sell for access

EN[able] <i>data base name [/maintenance word]</i> FOR	{ LOGGING RECOVERY ACCESS DUMPING ILR AUTODEFER ROLLBACK }
--	--

ERA[se] *data base name [/maintenance word]*

EXI[t]

HEL[p] *[command name]*

>>help create

MOV[e] *TurboIMAGE file name to device*

>>move orders05 to disc2

PUR[ge] *data base name [/maintenance word]*

REL[ease] *data base name*

Commands

SEC[ure] *data base name*

SET *data base name* [/maintenance word]

$\left\{ \begin{array}{l} \text{MAINT} = \text{maintenance word} \\ \text{BUFFSPECS} = \text{numbuffers (from-users/to-users)} \\ \quad [, \text{numbuffers (from-users/to-users)}] [, \dots] \\ \text{LOGID} = \text{log identifier} \\ \text{PASSWORD classnum} = [\text{password}] \\ \text{SUBSYSTEMS} = \text{none/read/rw} \\ \text{LANGUAGE} = \text{languageid} \end{array} \right\}$

>>set orders maint=sale

SHO[w] *database name [/maint word]*

{	MAINT	}	
{	ALL	}	
{	BUFFSPECS	}	
{	LOCKS	}	
{	USERS	}	
{	LOGID	}	[OFFLINE]
{	FLAGS	}	
{	PASSWORDS	}	
{	SUBSYSTEMS	}	
{	LANGUAGE	}	
{	DEVICE	}	

>>show orders/sale all offline

TRA { SYSON, *traceword*, *tracefilename* }
 SYSOFF }

VER[ify] *data-base-access file name*

>>verify orddba

Intrinsics

Note



The parameter *status* represents a communication array. The success or failure of a procedure determines its contents. The *TurboImage Reference Manual (32215-90003)* lists status array contents for each procedure.

Calling IMAGE Procedures:

COBOL

CALL "name" USING *parameter,parameter, ... ,parameter*

FORTRAN

CALL *name* (*parameter,parameter, ... ,parameter*)

PASCAL

name (*parameter,parameter, ... ,parameter*)

SPL

name (*parameter,parameter, ... ,parameter*)

BASIC

linenumber CALL *name* (*parameter,parameter, ... ,parameter*)

Note



All parameters are required. In BASIC, all procedure names begin with an X.

DBBEGIN

A A I A I
DBBEGIN(*base,text,mode,status,txtlen*);

DBCLOSE

A IorA I A
DBCLOSE(*base, dset, mode, status*);

DBCONTROL

A A I A
DBCONTROL(*base, qualifier, mode, status*);

DBDELETE

A IorA I A
DBDELETE(*base, dset, mode, status*);

DBEND

A A I A I
DBEND(*base, text, mode, status, textlen*);

DBERROR

A A I
DBERROR(*status, buffer, length*);

DBEXPLAIN

A
DBEXPLAIN(*status*);

DBFIND

A IorA I A IorA A
DBFIND(*base, dset, mode, status, item, argument*);

DBGET

A IorA I A A A A
DBGET(*base, dset, mode, status, list, buffer, argument*);

MODE	READING METHODS
1	Reread
2	Serial read
3	Backward serial read
4	Directed read
5	Chained read
6	Backward chained read
7	Calculated read
8	Primary calculated read

DBINFO

A IorA I A A
DBINFO(*base, qualifier, mode, status, buffer*);

DBLOCK

A IorA I A
DBLOCK(*base, qualifier, mode, status*)

Locking modes:

Mode 1	Database, unconditional
Mode 2	Database, conditional
Mode 3	Data set, unconditional
Mode 4	Data set, conditional
Mode 5	Data entries, unconditional
Mode 6	Data entries, conditional

DBMEMO

A A I A I
DBMEMO(*base, text, mode, status, textlen*);

DBOPEN

A A I A
DBOPEN(*base, password, mode, status*);

Access Mode	Associated Capabilities	Concurrent Modes Allowed
1	Modify with enforced locking. Allow concurrent modify.	1,5
2	Update, allow concurrent update.	2,6
3	Modify exclusive.	None
4	Modify, allow concurrent read.	6
5	Read, allow concurrent modify.	1,5
6	Read, allow concurrent modify.	6 and either 2, 4, or 8.
7	Read, exclusive	None
8	Read, allow concurrent read.	6,8

DBPUT

A I or A I A A A
DBPUT(*base, dset, mode, status, list, buffer*);

DBUNLOCK

A IorA I A
DBUNLOCK(*base, dset, mode, status*);

DBUPDATE

A IorA I A A A
DBUPDATE(*base, dset, mode, status, list, buffer*);

IMAGE Error Messages

Table 10-1. Intrinsic Exceptional Conditions

Condition Code	Condition	Returned by:
10	Beginning of file	DBGET
11	End of file	DBGET
12	Directed beginning of file	DBGET
13	Directed end of file	DBGET
14	Beginning of chain	DBGET
15	End of chain	DBGET
16	Data set full	DBPUT
17	No master entry	DBFIND
17	No entry	DBGET, DBUPDATE, DBDELETE
18	Broken chain	DBGET
20	Data base locked or contains locks	DBLOCK, modes 2,4,6
22	Data set locked by another process.	DBLOCK, modes 4,6
23	Entries locked within set.	DBLOCK, mode 4
24	Item conflicts with current locks.	DBLOCK mode 6
25	Entries already locked.	DBLOCK mode 6
41	Critical item	DBUPDATE
42	Read only item	DBUPDATE
43	Duplicate search item value	DBPUT
44	Chain head	DBDELETE
50	Buffer too small	DBGET

IMAGE Error Messages

Table 10-2.

Condition Code	Condition	Returned by:
51	Insufficient stack for BIMAGE temporary buffer	XDBGGET, XDBPUT
52	Invalid number of parameters	XDBUPDATE, XDBINFO
53	Invalid parameter	XDBGGET, XDBPUT
54	Status array too small	XDBUPDATE, XDBINFO
60	Data base access disabled	DBOPEN
61	This data base opened more than 63 times by same process	DBOPEN
62	DBG	DBGGET, DBPUT, DBUPDATE, DBLOCK
63	Bad DBG	all intrinsics
64	PCBXdata segment area full	DBOPEN
66	The current DBG for the data base does not appear correct (IMAGE) internal error)	
71	Logging is disabled	DBMEMO
1xx	Missing chain head	DBPUT
2xx	Full chain	
3xx	Full master	
65	Cannot allocate requested number of buffers	DBOPEN
67	Bad DBU	all intrinsics
68	Bad DBB	all intrinsics

QUERY

Initiate QUERY

```
RUN QUERY.PUB.SYS
```

Commands

A

A[*lter*] *procedure name*

QUERY prompts for **Insert**, **Replace**, **Delete**, or **End** statements by printing >>. Where *m* is the line number of the first line to be replaced or deleted, and *n* is the line number of the last line to be replaced or deleted. If *n* is specified, the range of lines from *m* to *n* is replaced or deleted.

```
Insert          >>/I, m
statement:      >> insertion line(s)

Replace         >>/R, m[, n]
statement:      >> replacement line(s)

Delete          >>/D, m[, n]
statement:

End             >>/E
statement:
```



Note



Terminating the ALTER command with a **CTRL** **Y** causes cancellation of the entire command; the procedure remains in its original state.

AD

$$\left\{ \begin{array}{l} \text{AD}[\text{d}] \\ \text{U}[\text{pdate}] \text{ADD,} \end{array} \right\} [\text{database name:}] \text{ data set name}$$

AS

$$AS[sign] \ [database\ name:] \ LOCKOPTION = \begin{Bmatrix} ON \\ OFF \end{Bmatrix}$$
B

$$[-]B[ase] = database\ name$$
C

$$C[reate] \ procedure\ name, \begin{Bmatrix} filename \\ command\ END \end{Bmatrix}$$

Where: *command* is one of the following QUERY commands:

```

FIND
FIND ALL
FIND CHAIN
JOIN
MULTIFIND
MULTIFIND ALL
REPORT
REPORT ALL
SUBSET
UPDATE ADD
UPDATE DELETE
UPDATE REPLACE
C[reate] SPACE

```

CL

$$CL[ose] \ database\ name$$
CO

$$CO[mment] \ text$$
D

$$D[isplay] \ procedure\ name \begin{Bmatrix} ,m[,n] \\ ,filename \end{Bmatrix}$$

Where: *m* and *n* are the line numbers of the first and last lines to be displayed.

$$D[isplay] \ LIST$$
11-2 QUERY

DBLIST

DBLIST = [database name [, ...]]

DE

DE[stroy] procedure name

DEF

DEF[ine]

DEL

{ DEL[ete]
U[pdate] DELETE }

E

E[xit]

F

F[ind] [#LIMIT = i;] { relation
item identifier M[at]hing "pattern" }

{ { AND } { relation
OR } { item identifier M[at]hing "pattern" } } ... [END]

Where:

item [database name:] [data set name.]
identifier= data item name [(subscript)]

pattern a series of special characters and operators described below. It must be enclosed in quotes.

Pattern Character	Represents
a	upper or lower case alphabetic characters (A-Z, a-z)
b	blank
d	a digit (0-9)
l	lower case alphabetic character (a-z)
u	upper case alphabetic character (A-Z)
?	any character

F

	Pattern Operator	Description
	!	transparency
	,	choice
	:	range
	{ }	grouping
	[]	optional
	+	repetition - 1 or more
	*	repetition - 0 or more
<i>relation</i>	[<i>database name</i> :][<i>data set name</i> .] <i>data item name</i> [(<i>subscript</i>)] <i>relop</i> " <i>value</i> " [, ...]	

Note



Value must be surrounded by quotes if it is not to be upshifted or if it contains special characters.

Where: *relop* is one of the following relational operators: Is equal to; multiple values separated by commas may be used:

$\left\{ \begin{array}{l} = \\ IS \\ IE \\ EQ \end{array} \right\}$

Is not equal to; multiple values separated by commas may be used:

$\left\{ \begin{array}{l} \# \\ ISNOT \\ INE \\ NE \\ <> \end{array} \right\}$

Is less than:

$\left\{ \begin{array}{l} ILT \\ LT \\ < \end{array} \right\}$

Is not less than (is greater than or equal to):

$\left\{ \begin{array}{l} INLT \\ GE \\ >= \end{array} \right\}$

Is greater than:

11-4 QUERY

$$\left\{ \begin{array}{l} \text{IGT} \\ \text{GT} \\ 5 \end{array} \right\}$$

Is not greater than (is less than or equal to):

$$\left\{ \begin{array}{l} \text{INGT} \\ \text{LE} \\ \leq \end{array} \right\}$$

$\text{IB } value_1, value_2$ Is between and including $value_1$ and $value_2$.

find employee.salary igt 1000

Where:

data set name employee

data item salary

name

relop igt (is greater than)

value 1000

F[ind] ALL [#LIMIT = i;] [database name:]
[data set name.] data-item-name

find all labor.badge#

F

$$F[ind] \text{ CHAIN } \{ \#LIMIT = i; \} \text{ item identifier } \left\{ \begin{array}{l} IS \\ IE \\ EQ \\ = \end{array} \right\} "value"$$
$$\left[\left\{ \begin{array}{l} AND \\ OR \end{array} \right\} \text{ item identifier } \left\{ \begin{array}{l} IS \\ IE \\ EQ \\ = \end{array} \right\} "value" \dots [END] \right]$$

Where:

item identifier *master set name.master search item,*
 detail set name. detail search item

value as defined for FIND

find chain employee.badge#,labor.badge# ie "1234"

F[ind] *procedure name* [,character]

Where: *character* is any printable ASCII character and causes the procedure to be listed.

FO

$$FO[rmat] \left[\begin{array}{l} database\ name[:] \\ ALL \end{array} \right] \left[\begin{array}{l} data\ set\ name \\ data\ item\ name \\ SETS \\ ITEMS \\ PATHS \end{array} \right]$$

H

H[elp] [Command name [FU[nction]] [FO[rmat]] [PA[rameters]]]

J

J[oin] *data item equivalence* [, ...]
 [; *data set equivalence*] [, ...] [END]

Where:

data item [database name:]
equivalence *data set name*. *data item name*
 [(*subscript*)] [0] TO [0]
 [database name:]
data set name. *data item name*
 [(*subscript*)]
data set *dummy data set name*=*data set name*
equivalence

J[oin] *procedure name* [, *character*]

Where: *character* is any printable ASCII character and causes the procedure to be listed.

LANGUAGE=

LA[*language*] = { *langname* }
 { *langnum* }

L

L[ist] [*database name* :] { *data set name* }
 { *data item name* }

[FOR *relation* [{ AND } *relation*] ...] [END]

Where:

data item list *data item name* [(*subscript*)]
 [, *data item name* [(*subscript*)]] ...
relation [*data set name* .] *data item name* [(*subscript*)]
relop "value" [, "value"] [, ...]
relop as defined for FIND
value as defined for FIND
list labor for badge# is 09.18'' and f-name is joe

LISTREDO

LISTR[edo] [m] [,n] $\begin{bmatrix} ;ABS \\ ;REL \\ ;UNN \end{bmatrix}$

M

M[ode]=mode number

MULTID

MULTID[b] database name

MU

MU[litfind] [#LIMIT = i;] $\left\{ \begin{array}{l} \text{relation} \\ \text{item identifier M[atting] "pattern"} \end{array} \right\}$
 $\left[\left\{ \begin{array}{l} \text{AND} \\ \text{OR} \end{array} \right\} \left\{ \begin{array}{l} \text{relation} \\ \text{item identifier M[atting] "pattern"} \end{array} \right\} \right] \dots [\text{END}]$

Where:

relation $\begin{bmatrix} \text{database name:} \\ \text{data set name.} \\ \text{dummy data set name.} \end{bmatrix}$
data item name [(subscript)] relop
 $\left\{ \begin{array}{l} \$\text{MISSING} \\ \text{"value" [, ...]} \end{array} \right\}$

item identifier $\begin{bmatrix} \text{database name:} \\ \text{data set name.} \\ \text{dummy data set name.} \end{bmatrix}$
data item name [(subscript)]

pattern As defined for FIND

relop As defined for FIND

value As defined for FIND

$$\text{MU}[\text{ltifind}] \text{ ALL } [\text{\#LIMIT} = i;] \left[\begin{array}{l} \text{database name;} \\ \text{data set name.} \\ \text{dummy data set name.} \end{array} \right] \text{data item name}$$

join sales.stock# to inventory.stock#
multifind all sales.stock#

MU[ltifind] procedure name [,character]

Where: *character* is any printable ASCII character and causes the procedure to be listed.

Note MULTIFIND, MULTIFIND ALL, and MULTIFIND *procedure name* only can be used after a JOIN command.



O

$$O[\text{utput}] = \left\{ \begin{array}{l} \text{TERM} \\ \text{LP} \end{array} \right\}$$

PA

PA[ssword]=password

P

P[roc-file]=filename [,n]

Where: *n* is a number from 5 to 400 specifying the number of records to build a new Proc-file.

REDO

$$\text{RED}[o] \left[\begin{array}{l} m \\ -n \\ \text{string} \end{array} \right]$$

REL

REL[ease]

REN

REN[ame] *old procedure name, new procedure name*

REN

REPL

```
{ REPL[ace],  
  U[ppdate] REPLACE, } [ data item name: ] [ (subscript) ]="value";  
  
[ data item name [ (subscript) ]="value"; ] ... END
```

R

R[eport] *report statement* [, ...]END

Note

If only R[eport] is entered, QUERY will prompt for **report statements** by displaying >> until END is entered.



**Report
Statements**

	Syntax
Detail statement	>>D[detail number], <i>print element</i> , <i>print position</i> [,SPACE A[number]] [,SPACE B[number]] [,SKIP{ $\begin{matrix} A \\ B \end{matrix}$ }] [,E{ $\begin{matrix} number \\ Z \end{matrix}$ }] >>D[etail number], <i>user procedure call</i>
Edit statement	>>E[number, "edit mask"]
Group statement	>>G[level, <i>print element</i> , <i>print position</i> [,SPACE A[number]] [,SPACE B[number]] [,SKIP{ $\begin{matrix} A \\ B \end{matrix}$ }] [,E{ $\begin{matrix} number \\ Z \end{matrix}$ }] >>G[level, <i>user procedure call</i>
Header statement	>>H[header number, <i>print element</i> , <i>print position</i> : [,SPACE A[number]] [,SPACE B[number]] [,E{ $\begin{matrix} number \\ Z \end{matrix}$ }] or >>H[header number, <i>user procedure call</i>

11-10 QUERY

Output control statement

```
>>[LINES = integer]
>>|[NOPAGE]
>>|[[OUT=] LP
>>|[PAUSE]
>>|[UNIFYDETAIL]
>>|[NOPUNCH]
```

Register statement

```
>>|Rnumber,  $\left\{ \begin{array}{l} L[\text{oad}] \\ A[\text{dd}] \\ S[\text{ubtract}] \\ M[\text{ultiply}] \\ D[\text{ivide}] \end{array} \right\}, \text{data element}[, \text{NOREPEAT}]$ 
```

Sort statement

```
>>|S[level], data item name  $\left[ , \left\{ \begin{array}{l} \text{ASC} \\ \text{DES} \end{array} \right\} \right]$ 
```

Total statement

```
>>Tlevel, print element, print position

[,SPACE A[number]] [,SPACE B[number]]  $\left[ ,\text{SKIP} \left\{ \begin{array}{l} A \\ B \end{array} \right\} \right]$ 

 $\left[ ,E \left\{ \begin{array}{l} \text{number} \\ Z \end{array} \right\} \right]$ 

 $\left[ , \left\{ \begin{array}{l} \text{ADD} \\ \text{AVERAGE} \\ \text{COUNT} \end{array} \right\} [, \text{NOREPEAT}] \right]$ 
```

or

```
>>Tlevel,  $\left\{ \begin{array}{l} Rn \\ \text{user procedure call} \end{array} \right\}$ 
```

Where:

```
print element  $\left[ (\text{database name}): \left[ \begin{array}{l} \text{data set name.} \\ \text{dummy data set name.} \end{array} \right] \right]$ 

data item name  $[(\text{subscript})]$ 
```

or

Rnumber

or a series of characters enclosed in quotes ("").

R

data element $\left[\text{data base name:} \right] \left[\begin{array}{l} \text{data set name.} \\ \text{dummy data set name.} \end{array} \right]$

data item name $\left[(\text{subscript}) \right]$

or

Rnumber

or a number enclosed in quotes (").

print position The rightmost *print position* (column number) for the *print element*. For character data, this is the rightmost character; for numeric data, it is the position of the least significant digit.

SPACE
A[*number*]= The number of lines (from 1 to 5) to space after printing the report line. If *number* is omitted, one line is spaced.

SPACE
B[*number*]= The number of lines (from 1 to 5) to space before printing the report line. If *number* is omitted, one line is spaced.

Indicates skip to the top of the next report page after printing the report line (SKIP A) or before printing the report line (SKIP B):

$\text{SKIP} \left\{ \begin{array}{l} \text{A} \\ \text{B} \end{array} \right\}$

Provides editing specifications. *Enumber* specifies that an edit defined in the identically numbered statement is to be used to format a value. EZ specifies that leading zeros are to be suppressed.

$\text{E} \left\{ \begin{array}{l} \text{number} \\ \text{Z} \end{array} \right\}$

user procedure call *procedure name* [(*user parameter*)],/

$\left[, \text{SPACE A}[\text{number}] \right] \left[, \text{SPACE B}[\text{number}] \right] \left[, \text{SKIP} \left\{ \begin{array}{l} \text{A} \\ \text{B} \end{array} \right\} \right]$

$\text{R}[\text{eport}] \left[\text{LINES} = \text{integer} \right] \left[\text{NOPAGE} \right] \left[\left[\text{OUT} = \right] \text{LP} \right] \left[\text{PAUSE} \right]; \text{ALL}$

$\left[, \text{character} \right]$

$\left[, - \right]$

Where: *character* is any printable ASCII character and causes *data item names* to be omitted from the report.

$\text{R}[\text{eport}] \left[\text{LINES} = \text{integer} \right] \left[\text{NOPAGE} \right] \left[\left[\text{OUT} = \right] \text{LP} \right] \left[\text{PAUSE} \right];$

procedure name [, *character*]

Where: *character* is any printable ASCII character and causes the procedure to be listed.

S

`[-]S[ets] = [data set name [, ... ]]`

SA

`SA[ve] filename [/lockword]`

SET

`SET[locks]`

SH

$$\text{SH}[\text{ow}] \left\{ \begin{array}{l} \text{DBLIST [data base name]} \\ \text{JOIN [data base name]} \\ \text{LOCKOPTION [data base name]} \\ \text{LANG [uage]} \end{array} \right\}$$
SUB

$$\text{SUB}[\text{set}] [\text{\#LIMIT} = i;] \left\{ \begin{array}{l} \text{relation} \\ \text{item identifier M[atting] "pattern"} \end{array} \right\}$$

$$\left[\left\{ \begin{array}{l} \text{AND} \\ \text{OR} \end{array} \right\} \left\{ \begin{array}{l} \text{relation} \\ \text{item identifier M[atting] "pattern"} \end{array} \right\} \right] \dots [\text{END}]$$

Where:

relation = as defined for FIND or MULTIFIND

item identifier as defined for FIND or MULTIFIND

=

pattern = as defined for FIND

`SUB[set] procedure name [,character]`

Where: *character* is any printable ASCII character, and causes the procedure to be listed.

T

`T[ransbegin] [data base name:] ["text"]`

TRANSE

`TRANSE[nd] [data base name:] ["text"]`

TRANSM

TRANSM[emo] [data base name:] ["text"]

UN

UN[do]

U ADD

$\left\{ \begin{array}{l} \text{U[pdate]} \text{ ADD,} \\ \text{AD[d]} \end{array} \right\} \text{ [data base name:] data set name}$

U DELETE

$\left\{ \begin{array}{l} \text{U[pdate]} \text{ DELETE} \\ \text{DEL[ete]} \end{array} \right\}$

Note UPDATE DELETE can only be used following the FIND command.



U REPLACE

$\left\{ \begin{array}{l} \text{U[pdate]} \text{ REPLACE,} \\ \text{REPL[ace]}, \end{array} \right\} \text{ data item name [(subscript)]="value";}$
[data item name [(subscript)]="value";] ... END

U

U[pdate] procedure name [,character]

Where: *character* is any printable ASCII characters and causes the procedure to be listed.

V

V[ersion]

X

X[eq] filename [,NODATA]

AMSTER

KSAM**Commands****BUILD / B**

```

B[ uild] filereference1

;KEYFILE=filereference2

;KEY=keytype,keylocation,keysize [ , [ keyblocking ] ] [ , DUP[ licate ] ]
[ , RDUP[ licate ] ]

[ ;REC=[ recsize ] [ , [ blockfactor ] [ , [ F ] [ , BINARY ] ] ]
[ , V ] [ , ASCII ] ] ]

[ ;TEMP ] [ ;DEV=device ] [ ;CODE=filecode ]

[ ;DISC=[ numrec ] [ , [ numextents ] [ , initialloc ] ] ]

[ ;LANG={ langnum }
           { langname } ]

[ ;KEY=keytype,keylocation,keysize [ , [ keyblocking ] ] [ , DUP[ licate ] ] ]
[ , RDUP[ licate ] ] ]

[ ;KEYENTRIES=numentries ] [ ;LABELS=numlabels ] [ ;KEYDEV=device ]

[ ;FIRSTREC=recnum ]

[ ;NODULOCKS ]

```

ERASE

ERASE *filereference*

EXIT / E

E[*xit*]

HELP / H

H[*elp*]

KEYDUMP / KD

K[*ey*]D[*ump*] [*filereference*] [;SEQ=*keysequence*]

[;SUBSET={ { - }*position* } [, *number*]]
 "*string*"

[;FILE=*formaldesignator*] [;SORT] [;OFFLINE]

KEYINFO / KI

K[*ey*]I[*nfo*] [*filereference*] [;OFFLINE] [;RECOVER]

KEYSEQ / KS

K[ey]S[eq] [filereference] [;SEQ=keysequence] [;OFFLINE] [;NOLIST]

PURGE

PURGE filereference[,TEMP]

RENAME / R

R[ename] oldfilereference,newfilereference[,TEMP]

SAVE / S

S[ave] [filereference]

VERIFY / V

V[erify] [filereference] [;OFFLINE] [;NOCHECK]

Intrinsics

FCHECK

IV I I D I
FCHECK(*filenum*, [*errorcode*], [*tllog*], [*blknum*], [*numrecs*]);

FCLOSE

IV IV IV
FCLOSE(*filenum*, *disposition*, *seccode*);

FCONTROL

IV IV L
FCONTROL(*filenum*, *controlcode*, *param*);

FERRMSG

I LA I
FERRMSG(*errorcode*, *msgbuf*, *msglngh*);

*FFINDBYKEY

IV BA IV IV IV
FFINDBYKEY(*filenum*, *keyvalue*, *keylocation*, *keylength*, *relop*);

***FFINDN**

IV DV IV
 FFINDN(*filenum*,*number*,*keylocation*);

FGETINFO

IV BA L L I I
 FGETINFO(*filenum*[,*filename*][,*foptions*][,*aoptions*][,*recsize*][,*devtype*]
 L L I D D D D D
 [, *ldnum*][,*hdaddr*][,*filecode*][,*recptr*][,*eof*][,*flimit*][,*logcount*][,*physcount*]
 I L I I BA D
 [, *blksize*][,*extsize*][,*numextents*][,*userlabels*][,*creatorid*][,*labaddr*]);

***FGETKEYINFO**

IV A A
 FGETKEYINFO(*filenum*,*ksamparam*,*ksamcontrol*);

FLOCK

IV LV
 FLOCK(*filenum*,*lockcond*);

FOPEN

I BA LV LV IV BA
filenum:=FOPEN(*formaldesignator*[,*foptions*][,*aoptions*][,*recsize*][,*device*]
 BA IV IV IV DV
 [, *ksamparam*][,*userlabels*][,*blockfactor*][,*numbuffers*][,*filesize*]
 IV IV IV
 [, *numextents*][,*initalloc*][,*filecode*]);

FPOINT

IV DV
FPOINT(*filenum*,*recnum*);

FREAD

I IV LA IV
lgth:=FREAD(*filenum*,*target*,*tcount*);

*FREADBYKEY

I IV LA IV BA IV
lgth:=FREADBYKEY(*filenum*,*target*,*tcount*,*keyvalue*,*keylocation*);

*FREADC

I IV LA IV
lgth:=FREADC(*filenum*,*target*,*tcount*);

FREADDIR

IV LA IV DV
FREADDIR(*filenum*,*target*,*tcount*,*recnum*);

FREADLABEL

IV LA IV IV
FREADLABEL(*filenum*,*target* [, *tcount*] [, *labelid*]);

***FREMOVE**

IV
FREMOVE(*filenum*) ;

FSETMODE

IV LV
FSETMODE(*filenum,modeflags*) ;

FSPACE

IV IV
FSPACE(*filenum,displacement*) ;

FUNLOCK

IV
FUNLOCK(*filenum*) ;

FUPDATE

IV LA IV
FUPDATE(*filenum,target,tcount*) ;

FWRITE

IV LA IV LV
FWRITE(*filenum,target,tcount,control*) ;

FWRITELABEL

IV LA IV IV
FWRITELABEL(*filenum*,*target* [, *tcount*] [, *labelid*]);

HP32208

D
version:=HP32208

Table A-2. FOPTIONs for Use with FOPEN

BITS	(0:2)	(2:3)	(5:1)	(6:1)	(7:1)	(8:2)	(10:3)	(13:1)	(14:2)
FIELD	Reserved	File Type	Disallow :FILE	MPE Tape Labels	Carriage Control	Record Format	Default Designator	ASCII/ Binary	Domain
MEANING		00 0=STD 00 1=KSAM 01 0=PIO 10 0=CR 11 0= MSG	0=Allow :FILE 1=No :FILE	0=NON LAELED TAPE 1=LAELED TAPE	0=NOCCTL 1= CCTL	00=Fixed 01=Variable 10=Undefined	000=Filename 001=\$STDLS 010=\$NEWPASS 011=\$OLDPASS 100=\$STDIN 101=\$STDINX 110=\$NULL	0=Binary 1=ASCII	00=New File 01=Old Permanent File 10= Old Temporary File 11=Old Perm. or Temp. File

Figure 12-1. Foptions

Table A-3. AOPTIONS for Use with FOPEN

BITS	(0:3)	(3:1)	(4:1)	(5:2)	(7:1)	(8:2)	(10:1)	(11:1)	(12:4)
FIELD	Reserved	File Copy	No-Wait I/O	Multi Access	Inhibit Buffering	Exclusive Access	Dynamic Locking	Multi- record Access	Access Type
MEANING		0=Access in file's native mode 1=Access as standard sequential file	1=No Wait 2=Non No- Wait	00=Non-multi- access 01=Only intra- job multi- access 10=inter-job multi- access allowed	0=BUF 1=NOBUF	00=Default 01=Exclusive 10=Exclusive Access Read 11=Share	0=No FLOCK Allowed 1=FLOCK Allowed	0=No Multi- Record 1=Multi- Record	0 000=Read Only 0 001=Write only 0 010=Write (save) only 0 011=Append only 0 100=Read/write 0 101=Update 0 110=Execute

Figure 12-2. Aoptions

BASIC Procedures

Format: *statement label CALL procedurename(parameters)*

Example: 250 CALL BKCLOSE(filenum,stat)

BKCLOSE	<i>filenum,status</i>
BKDELETE	<i>filenum,status</i>
BKERROR	<i>status,message</i>
BKLOCK	<i>filenum,status[,condition]</i>
BKOPEN	<i>filenum,status,fname[,access][,dynamiclock][,exclusive][,sequence]</i>
BKREAD	<i>filenum,status[,parameterlist]</i>
BKREADBYKEY	<i>filenum,status,keyvalue,keylocation,parameterlist</i>
BKREWRITE	<i>filenum,status,parameterlist</i>
BKSTART	<i>filenum,status[,keyvalue][,keylocation][,relation]</i>
BKUNLOCK	<i>filenum,status</i>
BKVERSION	<i>status,message</i>
BKWRITE	<i>filenum,status,parameterlist</i>

Note	<i>Parameterlist</i> is a list of variables into which data is read (BKREAD, BKREADBYKEY) or a list of variables or constants containing data to be written (BKREWRITE,BKWRITE).
-------------	--



Status Parameter Descriptions

First Character	Remaining Characters
0 successful completion	0 no further information 2 duplicate key value
1 at end or beginning of file	0 no further information
2 invalid key	1 sequence error 2 duplicate key error 3 no record found 4 boundary violation
7 request denied	1 file is already locked
8 invalid call	1 invalid number of parameters 2 invalid parameter 3 insufficient space for data in <i>parameterlist</i>
9 (file system error)	0 through 255 corresponding to file system error codes

COBOL Procedures

Format: CALL "procedurename" USING *parameterlist*

Example: CALL "CKREAD" USING *filetbl,stat,rec,recsize*

CKCLOSE	<i>filetable status</i>
CKDELETE	<i>filetable status</i>
CKERROR	<i>status result</i>
CKLOCK	<i>filetable status lockcond</i>
CKOPEN	<i>filetable status</i>
CKOPENSRR	<i>filetable status</i>
CKREAD	<i>filetable status record recordsize</i>
CKREADBYKEY	<i>filetable status record key keyloc recordsize</i>
CKREWRITE	<i>filetable status record recordsize</i>
CKSTART	<i>filetable status relop key keyloc keylength</i>
CKUNLOCK	<i>filetable status</i>
CKWRITE	<i>filetable status record recordsize</i>

Note All parameters are required and must be separated by at least one space.



COBOL Procedures

filetable parameter:

```
Word +-----+
  1 | filenumber |
  |-----|
  2 | |
  |-----|
  3 | filename (8 characters) |
  |-----|
  4 | |
  |-----|
  5 | |
  |-----|
  6 | input-output type |
  |-----|
  7 | access mode |
  |-----|
  8 | lock/unlock | previous operation |
  +-----+
```

status parameter:

```
/---left character ---\---right character---\
+-----+
| status-key-1 | status-key-2 | <---Status Word
+-----+
```

If left character of <i>status</i> (<i>status-key-1</i>) equals:	Then right character of <i>status</i> (<i>status-key-2</i>) equals:
0 (successful)	0 (no further information) 2 (duplicate key)
1 (at end)	0 (no further information)
2 invalid key	1 (sequence error) 2 (duplicate key) 3 (no record found) 4 (boundary violation)
3 (request denied)	0 (lock denied) 1 (unlock denied)
9 (file system error)	<i>n</i> (where <i>n</i> is the MPE file system error code)

FORTTRAN Procedures

Format: `CALL procedurename (parameters)`

Example: `CALL FCLOSE(fileno,disp,code)`
or
`CKCLOSE(filetab,stat)`

Note FORTRAN may call either the SPL intrinsic or the COBOL procedures.



SPL Intrinsic

Format: `intrinsicname(parameterlist);`

or

`return:=intrinsicname(parameterlist);`

Example: `FCLOSE(filenum,disp,seccode);`

or

`length:=FREAD(filenum,target,count);`

VPLUS

Initiate FORMSPEC

RUN FORMSPEC.PUB.SYS



FORMSPEC Function Keys

FORMSPEC

+-----+-----+-----+-----+-----+-----+-----+-----+-----+								
	PREV	NEXT	FIELD				MAIN/	
	FORM	FORM	TOGGLE	REFRESH	PREV	NEXT	RESUME	EXIT
	f1	f2	f3	f4	f5	f6	f7	f8
+-----+-----+-----+-----+-----+-----+-----+-----+-----+								

Where:

Forms File: *filename[/lockword] [.groupname[.accountname]]*

Form Name: 1-15 character unique filename, beginning with an alphabetic character; may include other alphabetic characters, digits, and underlines (.). The name is upshifted.

Field Tag: Begins with alphabetic character and may include uppercase and lowercase alphabets, digits, and underlines; must be unique to the form; entered with field delimiters on form layout.

Field Name: Same rules as form name; alternate name assigned to a field on Field Menu.

Menu Sequence

```
RUN FORMSPEC.PUB.SYS
|
|
FORMS FILE MENU
|
| * define/open forms file
|
MAIN MENU
|
| * add a form
| * list forms or entire file
| * delete save field or form
| * copy a form
| * compile forms file
| * relate one form to another
|
|-> GLOBALS MENU
|
| * specify forms file global values
|
|-> TERMINAL/LANGUAGE MENU
|
| * specify terminals to use with forms file
| * specify FORMSPEC native language characteristics
| * specify forms file characteristics pertaining
|   to native language support
|
|-> SAVE FIELD MENU
|
| * specify save fields
|
|-> ARB MENU
|
|-> FORM MENU
|
| ^
| | * specify form characteristics
| |
| | FORM LAYOUT MENU
| | ^
| | | * design layout of form
| | |
| |
|
|-> FIELD MENU

* override default characteristics for each field
* specify processing specifications
```

Global Definitions	Head form:	Any form name; default = first form in file in ASCII collating sequence
	Enhancement:	I = Inverse Video U = Underline B = Blinking H = Half Bright S = Security 1-8 = Color Pair Number NONE = No enhancements
	Default	Field = IH (Inverse Video, Half Bright)
	Enhancements:	Error = IU (Inverse Video, Underline) Window = IH (Inverse Video, Half Bright)
	Window Line:	Any screen line number from 1-24 or zero for no window; default = line 24 (bottom line)
	Define Function Key Labels:	Y = Obtain the Global Function Key Labels Menu to change the function key labels. N = Delete previously defined labels. To retain the default global labels, leave this field blank.
Global Function Key Labels	Function Key:	For each function key, enter the first line of the label in the first field and the second line of the label in the second field. Each field may contain up to eight characters.
Form Definition	Form Name:	Any form name unique to forms file
	Next Form Name:	Any form name in file, or \$END , \$HEAD , \$RETURN , \$REFRESH
	Repeat Option:	N = Do not repeat current form A = Repeat current form, appending to previous form R = Repeat current form, overlaying previous form
	Next Form Option:	C = Clear current form before displaying next form A = Append next form to current form F = Freeze current form, then append next form
	Reproduced From:	If you are generating a child form, enter the parent form here. (Optional.)
	Comments:	Enter any comment up to 50 characters long.

Local Form Function	Key Labels:	Y = Obtain the Form Functions Key Labels Menu and change function key labels. ■ = Delete previously defined labels. To retain local default labels, leave this field blank.
	Form Function Key Labels	For each function key, enter the first line of the label in the first field and the second line of the label in the second field. Each field may contain up to eight characters.
Screen Design	Field Delimiters:	Printing: start field =[(open bracket) stop field =] (close bracket) Non-printing: start field = CNTL F2 or ESC stop field = CNTL F3 or ESC
	Field Tag:	Begins with alphabetic character and may include upper and lowercase alphabets, digits and underlines; must be unique to the form; entered with field delimiters on screen design.
	Field Length:	Number of characters within field delimiters.
	Unprotected Field Enhancement:	Determined by default or global specifications, may be changed for individual fields on Field Menu .
Field Definition	Field Number:	Number assigned to each field in order of initial field definition, starting with 1. May only be changed with the RENUMBER batch mode command.
	Field Name:	By default, upshifted field tag; may be changed to valid field name unique to form. Format is the same as form name.
	Field Type:	O = Optional field; skip edit checks if blank (default) R = Required field; error if blank D = Display-only field; protected from user entry P = Process field; perform edit checks even if blank

Data Type: CHAR = Any ASCII characters allowed (default)
 NUM = Floating-point signed number, commas allowed
 NUMn = Fixed-point signed number, maximum *n* decimal places, commas allowed
 IMPn = Implied decimal point signed number, *n* decimal positions, commas allowed
 DIG = Digits only allowed
 MDY = Date in order: *month day year*
 DMY = Date in order: *day month year*
 YMD = Date in order: *year month day*
 month = 1 or 2 digits in range 1-12, 3-letter abbreviation, or full name of month, correctly spelled.
 day = 1 or 2 digits in range 1-31
 year = 2 or 4 digits of legitimate year
 separator = Slash (/), dash (-), comma (,) or blank (if blank, *month* and *day* must be 2 digits)

Field Enhancement: I = Inverse video
 B = Blinking
 U = Underline
 E = Half bright inverse video
 S = Security
 1-8 = Color Pair Number
 NONE = No enhancements

The enhancements may be specified in any combination. The default may be set on the **Global Menu**.

Initial Value: Any value that matches field's data type and length

Save Field Definition

Name: Valid name unique to forms file. Format is the same as form name.

Data Types: Any legal type; CHAR, NUM, NUMn, IMPn, DIG, MDY, DMY or YMD

Length: Maximum number of characters in field

Initial Value: Any value of field's data type and within field length

**Terminal &
Language
Selection**

Enter **X** in the fields by the terminals which will be used with the forms file. Default: the forms file will run on the 264X family and the 262X family, but local edits and security display enhancements on the 264X family will not be supported.

If using native language support, set the language ID number. Default is 0 for Native-3000. Use -1 for international or specify the language number of any language configured on your system.

Note



The security display enhancement feature is terminal-dependent, available only on 2393A, 2394A, 2624B, 2625A, 2626A, 2628A, 700/92, 700/94, and HP150's.

**3075/3076
3081A
Globals**

Split Message Pause:	Integer; number of seconds to retain a line of text on the screen.
Wait for user to press ENTER:	Enter an X in this field to retain a line of text on the screen until user presses ENTER .
Error light:	Character; indicates which key's light will be turned on when errors are detected. May be any alphabetic character. (Default: E .)
HOLES/ MARKS:	Tell whether multiple function reader will be reading cards with holes or marks. (Default: holes)
Corner Cut Required:	Enter YES or NO . (Default: YES)
Clock On/ After/ NONE:	Whether clocking marks occur at the same time as data, after the data, or do not occur at all. (Default: NONE)
Barcode Reader Format:	Enter UCP , EAN , I39 , I25 , ILV , or MAT . (Default: UPC)
Disable Check Digit:	Enter Y or N . (Default: N)

Reserved Words

ALL	DISPLAY	139	LT	OF	UPC
APPEND	EAN	IF	MAGSTRIPE	PRINTER	UPSHIFT
BARCODE	ELSE	ILV	MARKS	RELAY	\$EMPTY
CAD	EQ	IN	MAT	REPEAT	\$END
CDIGIT	FAIL	INIT	MATCH	RIGHT	\$HEAD
CENTER	FIELD	JUSTIFY	MFR	SET	\$LENGTH
CFORM	FILL	KEYBOARD	MINLEN	STDCHAR	\$REFRESH
CHANGE	FINISH	LARGECHAR	NE	STOP	\$RETURN
CLEAR	FREEZE	LE	NFORM	STRIP	\$STATE
COD	GE	LEADING	NIN	THEN	\$TODAY
CONFIG	GT	LEFT	NOCUT	TO	
CUT	HOLES	LIGHT	NONE	TRAILING	
DEVICE	125	LOCALEDITS	NOREPEAT	TYPEV	

Operands	Field Name:	Name of existing field in current form
	Save Field Name:	Name of save field in current forms file
	Constant:	Character, numeric, or data type value; or system defined constant.
	Character Type:	Any ASCII characters within single or double quotes (" or ')
	Numeric Type:	Digit string with optional leading sign and optional decimal point
	Date Type:	Any legal date format in order MDY, enclosed within exclamation points (!)
	System Defined:	\$EMPTY (any data type) = Null value \$LENGTH (numeric type) = Current field length \$STATE (character type) = 2-character U.S. state abbreviation \$TODAY (date type) = Today's date \$END (form name) = Terminate forms \$HEAD (form name) = First form \$REFRESH (form name) = Clear form \$RETURN (form name) = Previous form
	Arithmetic Expression:	Any numeric constants, fields, save fields, or index retrieve operands in parentheses combined by the following operators: + (add), - (subtract), * (multiply), / (divide), % (percent of)

Result of evaluating operand in form:

where *index* = numeric field, save field, or arithmetic expression

Enclosed within parentheses in *edit statements* or
as operand in *arithmetic expression*

Configuration Statements:

```
[ JUSTIFY
  MUST_FILL ... [ DEC DIGITS n ]
  REQUIRED ]
```

[DEC_TYPE_EUR
DEC_TYPE_US] [TRANSMIT ONLY]
$$\text{LIGHT} \left\{ \begin{array}{c} \text{C} \\ \text{A} \\ \text{B} \\ \dots \\ \text{N} \\ \text{O} \\ \text{P} \end{array} \right\}$$

Note: More than one character may be selected; separate characters with commas.

DEVICE	{	PRINTER	}			
		KEYBOARD				
		MAGSTRIPE				
		TYPEV				
		MFR		[	CAD MARKS	]
				NONE HOLES		
				CAD HOLES	[	
		NOCUT				
		DISPLAY		[	LARGECHAR	]
				STDCHAR		
BARCODE	[	UPC	]			
	I25					
	I39					
	ILV					
	MAT					
	EAN					

Note: More than one option may be selected;
separate options with commas.

Edit

Statements:

CDIGIT { 10 } { "message" }

{ GT }
 { LT }
 { GE }
 { LE } operand ["message"]
 { EQ }
 { NE }

{ IN }
 { MIN }
 { operand [, operand] ... }
 { lowoperand:highoperand [, lowoperand:highoperand] ... }
 ["message"]

MATCH pattern ["message"]

Pattern	Each character in <i>pattern</i> is generic or actual;
Characters:	generic characters determine type of character to enter in that position; actual characters indicate exact character to enter in that position.
Generic Characters:	a= Upper or lowercase letter (A-Z, a-z) u= Uppercase letter (A-Z) l= Lowercase letter (a-z) b= Blank (space) d= Digit (0-9) ?= Any character
Pattern Operators:	!= Transparency - Required Example: !d means enter the letter d, not a digit , = Choice - Required Example: a,d allows any letter or digit : = Range - Required Example: 1:5 means enter one of the digits 1,2,3,4,5 { }= Grouping - Required Example: {a,d} allows 1 letter or 1 digit []= Grouping - Optional Example: [a,d] allows 1 letter, 1 digit or nothing + = Repetition - Required Example: d+ means enter as many digits as desired * = Repetition - Optional Example: d* means enter as many digits as desired or nothing Note: <i>Pattern</i> must not require leading or trailing blanks in data since such blanks are stripped before the match is executed.
MINLEN:	<i>operand</i> [" <i>message</i> "]

Form Sequence Statements:	$\text{CHANGE CFORM TO } \left\{ \begin{array}{l} \text{NOREPEAT} \\ \text{REPEAT} \\ \text{REPEAT APPEND} \end{array} \right\}$ $\text{CHANGE NFORM TO } \left\{ \begin{array}{l} \left[\begin{array}{l} \text{CLEAR} \\ \text{APPEND} \\ \text{FREEZE APPEND} \end{array} \right] \left[\begin{array}{l} \text{"formname"} \\ \text{field name} \\ \text{index retrieve operand} \\ \$\text{RETURN} \\ \$\text{HEAD} \\ \$\text{END} \\ \$\text{REFRESH} \end{array} \right] \end{array} \right\}$
Control Statements:	$\text{FAIL } [\text{"message"}]$ STOP
Conditional Statements	$\text{IF } [\text{operand}] \text{ editstatement THEN } [\text{statement}]$ $\dots \text{ ELSE } [\text{statement}] \dots [\text{statement}]$ $[\text{statement}] \dots$
Data Movement Statements:	$\text{SET } \left\{ \begin{array}{l} \text{destination TO source} \\ \text{destination} \\ \text{TO source} \\ \text{fieldname} \end{array} \right\}$ $\{ \text{source=} \} \left\{ \begin{array}{l} \text{field name} \\ \text{save field name} \\ \text{constant} \\ \text{arithmetic expression} \\ \text{index retrieve operand} \end{array} \right\}$ $\{ \text{destination=} \} \left\{ \begin{array}{l} \text{field name} \\ \text{save field name} \end{array} \right\}$

Data Formatting Statements	CONFIG	Performs the specified configuration during form presentation. Refer to the Phase Specifications Table below.
	FIELD	Performs the specified <i>statements</i> during the field edit phase. Refer to the Phase Specification Table below.
	FILL	$\left\{ \begin{array}{l} \text{TRAILING} \\ \text{LEADING} \end{array} \right\} \text{"character"}$
	FINISH	Performs the specified <i>statements</i> during the finish phase. Refer to the Phase Specifications Table below.
	INIT	Performs the specified <i>statements</i> during initialization. Refer to the Phase Specifications Table below.
	JUSTIFY	$\left\{ \begin{array}{l} \text{LEFT} \\ \text{RIGHT} \\ \text{CENTER} \end{array} \right\}$
	SET TO	<i>thisfield</i> Format entered data to default format according to data type. (Refer to the Conversion Table below.)
	STRIP	$\left\{ \begin{array}{l} \text{TRAILING} \\ \text{LEADING} \\ \text{ALL} \end{array} \right\} \text{"characters"}$
	UPSHIFT	

Conversion Between Data Types

Source Type	Destination Type					
	CHAR	NUM	NUM _n	IMP _n	DIG	DATE
CHAR	truncate or pad with blanks on right	Illegal	Illegal	Illegal	Illegal	Illegal
Any Numeric Type	truncate or pad with blanks on right	right justify, replace leading zeros with blanks; pad trailing blanks; strip any commas				Illegal
		try to fit 9 decimal places; round or truncate, insert decimal as needed	round or truncate fractions, insert decimal point as needed; fill trailing zeros	remove any decimal point; may strip leading fractional zeros	value must be positive, round to integer	
Any Date Type	truncate or pad with blanks on right	Illegal	Illegal	Illegal	Illegal	left justify; pad trailing blanks; convert to dd/dd/dd in destination order

Phase Specifications

Configure						
initialize			INIT	INIT	[INIT]	[INIT]
			statements	statements	statements	statements
Field		[FIELD]			FIELD	FIELD
Edit		statements	statements		statements	statements
Finish	FINISH		FINISH	FINISH		FINISH
	statements		statements	statements		statements

Initiate REFSPEC

RUN REFSPEC.PUB.SYS

REFSPEC Function Keys							
PREV	NEXT				MAIN/		
REFORMAT	REFORMAT	REFRESH	PREV	NEXT	RESUME	EXIT	
f1	f2	f3	f4	f5	f6	f7	f8

Identifiers	Reformat	filename[/lockword][.groupname[.accountname]]
	File:	
	Forms	Name of forms file created through FORMSPEC
	File:	
	Reformat	first form name in each input forms sequence
	Id:	

MENU SEQUENCE

```

RUN REFSPEC.PUB.SYS
|
|
FORM FILE MENU
|
| * specify forms file for reformatting
|
MAIN MENU
|
| * add, list delete a reformat
|
| * list entire reformat file
|
| * compile reformat file
|
| * go to existing reformat or output field
|
| * go to FORMS FILE MENU
|-> GLOBALS MENU
|
| * specify globals
|-> INPUT FORMS MENU 1
|
| * define first reformat input sequence
|
OUTPUT RECORD MENU 1
|
| * specify output records for first reformat
|-> OUTPUT FIELD MENU 1
|
| .
| .
| .
| OUTPUT FIELD MENU n
|
| * further define each field in
|   output record
|-> INPUT FORMS MENU 2
OUTPUT RECORD MENU 2
OUTPUT FIELD MENU 1
|
| .
| .
| .
OUTPUT FIELD Menu n

```


Initiate REFSPEC

- * define second reformat in the same way

INPUT FORMS Menu n

- * define subsequent reforms

- .

- .

- .

Global Specifications	Output Record Format:	F = Fixed-length records (default) V = Variable-length records U = Undefined-length records
	Record Length:	Specify total number of characters in output record; default = 80 characters
	Upshift:	Y = Shift letters to uppercase N = Do not shift letters (default)
	Convert to EBCDIC:	Y = Write output record in EBCDIC N = Leave data in ASCII code (default)
Reformat Specifications	Record Terminator:	Specify terminator for each output record as 1 or more constants: * Quoted string of ASCII characters * Numeric equivalent of ASCII code preceded by \$ * System constant: \$LF = Line feed \$CR = Carriage return \$GS = Group separator \$US = Unit separator \$RS = Record separator
	Field Separator:	Separate fields in output record with any of the constants listed above under Record Terminator .
	Input Forms Sequence:	At least one and up to 10 form names; first name (format id) must be unique. Only data from forms in same order as forms in input forms sequence is reformatted.
	Output Record:	Name each field, portion of field, or constant to be written to output record; input fields must be in a form listed in associated input forms sequence. Input Field: name of field in form in input forms sequence (required) Substring Start: first character (offset) in input field (optional) Input Length: number of characters in input field counting from column 1 of field or Substring Start (optional) Form Name: name of form containing input field (optional)

Initiate REFSPEC

Output Field: unique name for output field (optional)

Starting Column: column number in output record where output field starts (optional)

Length: number of characters in output field (optional)

Start of Record: character to indicate that this is first field, or constant in output record (optional)

Constant: constant to be written to output record; may be *constant* listed under **Record Terminator** (optional)

Output Field: Each field in output record may be formatted according to:

Data Type Specify type to which data in input field is to be converted; default = **CHAR**. Allowed conversions are:

Any data type to **CHAR**
Any numeric type to any numeric type
Any date type to any date type

REFSPEC Statements:	CHECKDIGIT	$\left\{ \begin{array}{l} 10 \\ 11 \end{array} \right\}$	
	FILL	$\left\{ \begin{array}{l} \text{ALL} \\ \text{LEADING} \\ \text{TRAILING} \end{array} \right\}$	character
	JUSTIFY	$\left\{ \begin{array}{l} \text{RIGHT} \\ \text{LEFT} \\ \text{CENTER} \end{array} \right\}$	
	SIGN	$\left\{ \begin{array}{l} \text{LEFT} \\ \text{RIGHT} \\ \text{ZONE} \\ \text{FLOAT} \\ \text{NONE} \end{array} \right\}$	PLUS (Y or N)
	STRIP	$\left\{ \begin{array}{l} \text{ALL} \\ \text{LEADING} \\ \text{TRAILING} \end{array} \right\}$	"characters"

Initiate REFORMAT

Specify name of file containing reformat specifications entered by REFSPEC:

FILE REFFILE=*reformatfile*

Specify name of file containing data to be reformatted:

FILE BATCH=*batchfile*

Specify name of file to which reformatted data is written:

FILE OUTFILE=*outputfile*

List error messages on line printer, not at terminal (optional):

FILE REFLIST;DEV=LP

List reformatted data (optional):

FILE TESTLIST;DEV=LP

Run program REFORMAT:

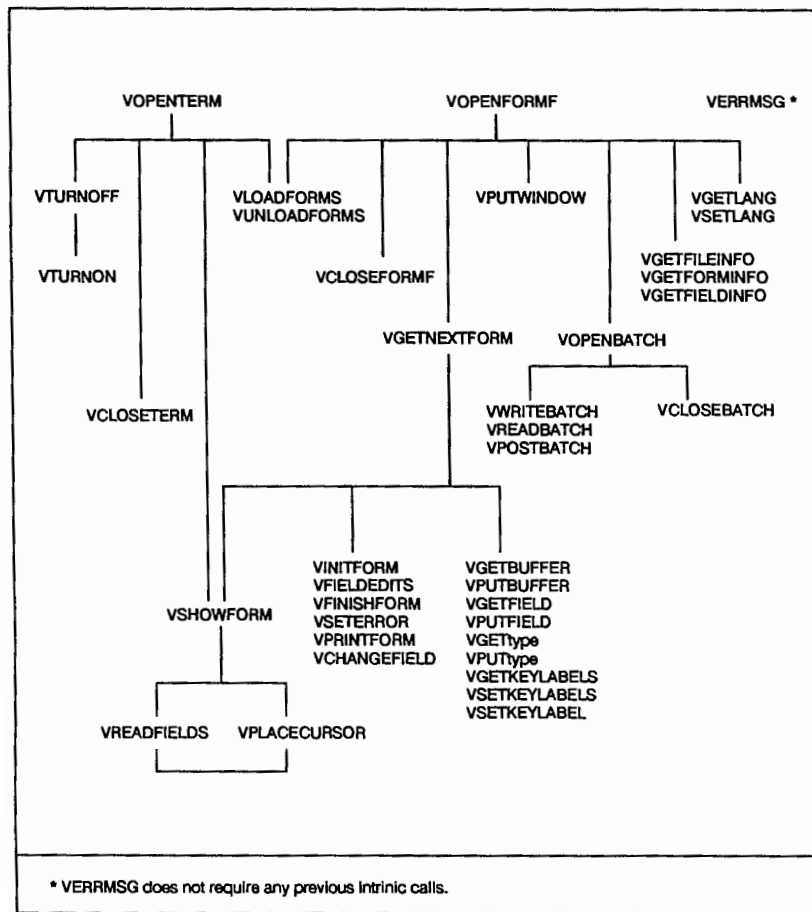
RUN REFORMAT.PUB.SYS

Intrinsics

Formats	COBOL:	CALL "intrinsicname" USING parameterlist
	FORTRAN:	CALL intrinsicname (parameterlist)
	SPL:	intrinsicname (parameterlist);
	BASIC:	statement label CALL intrinsicname (parameterlist)
	PASCAL	intrinsicname (parameter/[parameter2] ...);
Procedures	VCHANGEFIELD	comarea,specbuffer,numentries
	VCLOSEBATCH	comarea
	VCLOSEFORMF	comarea
	VCLOSETERM	comarea
	VERRMSG	comarea,buffer,buflen,actualen
	VFIELDDEITS	comarea
	VFINISHFORM	comarea
	VGETBUFFER	comarea,buffer,buflen
	VGETFIELD	comarea,fieldnum,fieldbuf, buflen,actualen,nextfldnum
	VGETFIELDINFO	comarea,infobuf,infobuflen
	VGETFILEINFO	comarea,infobuf,infobuflen
	VGETFORMINFO	comarea,infobuf,infobuflen
	VGETKEYLABELS	comarea,formorglob,numoflabels,labels formorglob = 0 retrieves global labels formorglob = 1 retrieves current form labels
	VGETLANG	comarea,langnum
	VGETNEXTFORM	comarea
	VGETtype	comarea,fieldnum,variable Note: type may be INT, DINT, REAL, or LONG
	VINITFORM	comarea
	VLOADFORMS	comarea,numofforms,formsloaded,forms numofforms = -1 workspace configuration under user control numofforms = 0 no local form storage numofforms = 1 to 4 One to four forms can be stored locally.

VOPENBATCH	<i>comarea, batchfile</i>
VOPENFORMF	<i>comarea, formfile</i>
VOPENTERM	<i>comarea, termfile</i>
VPLACECURSOR	<i>comarea, fieldnum</i>
VPOSTBATCH	<i>comarea</i>
VPRINTFORM	<i>comarea, printcntl, pagecntl</i>
VPUTBUFFER	<i>comarea, buffer, buflen</i>
VPUTFIELD	<i>comarea, fieldnum, fieldbuf, buflen, actualen, nextfldnum</i>
VPUTtype	<i>comarea, fieldnum, variable</i>
VPUTWINDOW	<i>comarea, message, msglen</i>
VREADBATCH	<i>comarea</i>
VREADFIELDS	<i>comarea</i>
VSETERROR	<i>comarea, fieldnum, message, msglen</i>
VSETKEYLABEL	<i>comarea, formorglob, numoflabels, labels</i> <i>formorglob = 0 set current form labels</i> <i>formorglob = 1 replaces current form labels</i>
VSETKEYLABELS	<i>comarea, formorglob, numoflabels, labels</i> <i>formorglob = 0 sets global labels</i> <i>formorglob = 1 replaces current form labels</i>
VSETLANG	<i>comarea, langnum, error</i>
VSHOWFORM	<i>comarea</i>
VUNLOADFORM	<i>comarea, whichform</i>
VWRITEBATCH	<i>comarea</i>

Intrinsics



LG200140_015

COMAREA

Used by all VPLUS procedures, this area must be initialized to zero and COMAREALEN specified (currently 60 words; 85 words if using 307x or 3081 terminals); a word is a 2-byte quantity before first procedure call; if applicable, set LANGUAGE and USRBUFLN (BASIC).

Outline of COMAREA Contents

Type	Offset		Name	Content
	SPL	Other		
Two byte integer	0	1	CSTATUS	Status or error code, 0=successful
	1	2	LANGUAGE	0=COBOL;1=BASIC;2=FORTRAN 3=SPL;5=PASCAL, FORTRAN 77, HP BUSINESS BASIC
	2	3	COMAREALEN	Total length of COMAREA (in two-byte words)
	3	4	USRBUFLN	BASIC only; length of COMAREA extension in two-byte words
	4	5	CMODE	0=Collect; 1=Browse/Modify
	5	6	LASTKEY	Code of last key pressed at terminal 0=ENTER 1=f1 2=f2 3=f3 4=f4 5=f5 6=f6 7=f7 8=f8 <0=fieldnumber of field touched on HP150 -999=area that does not contain a field was touched
	6	7	NUMERRS	Number of edit errors found in current form
	7	8	WINDOWENH	ASCII code for window enhancement; initialized for FORMSPEC value, can be changed to particular code (see codes under screen design)
	8	9	MULTIUSAGE	Next form flag 1=Child or sibling to previous form 0=Not a member of the family

COMAREA

Outline of COMAREA Contents (continued)

Type	Offset		Name	Content
	SPL	Other		
Two-byte integer	9	10	LABELOPTION	0=No labels are used 1=Labels are to be used
Character Array (16-byte)	10-17	11-18	CFNAME	Name of current form (15 characters)
	18-25	19-26	NFNAME	Name of next form (15 characters)
Two-byte integer	26	27	REPEATAPP	Repeat/Append flag for current form; 0=Normal sequence, no repeat/append 1=Repeat current form 2=Repeat and append current form
	27	28	FREEZEAPP	Freeze/Append flag for next form; 0=Clear current form,display next 1=Append next form to current form 2=Freeze current form,append next
	28	29	CFNUMLINES	Number of lines in current form
	29	30	DBUFLEN	Total number of bytes needed in data buffer for all concatenated fields (including display-only) in current form
Two-byte integer	30	31	reserved - initialize to zero.	
Two-byte unsigned integer	31	32	LOOKAHEAD	Form preload indicator. 0- ON - Preload the forms 1- OFF - Do not preload the forms
	32	33	DELETEFLAG	Flag to delete current batch record: FALSE (all zeros) - do not delete True (all 1's) - delete record

Outline of COMAREA Contents (continued)

Type	Offset		Name	Content
	SPL	Other		
Two-byte unsigned integer	33	34	SHOWCONTROL	Override VSHOWFORM optimizations; bit 15=1 Force form to be written 14=1 Force data to be written 13=1 Force window to be written 12=1 Stop display after form without fields 11=1 Add closing bracket 10=1 Do not enable keyboard 9=1 Enable form pre-loading 8=1 Enable Aids/Models/Userkeys 0=1 Enable Touch Support
Two-byte integer	34	35	Reserved - initialize to zero	
Two-byte integer	35	36	PRINTFILNUM	MPE file # VPRINTFORM list file
	36	37	FILERRNUM	MPE file error # returned by procedures
	37	38	ERRFILENUM	MPE file # of VPLUS error message file
	38	39	FORMSTORE-SIZE	Number of forms in form storage buffer.-1 = Workspace configuration under user control; 0 = No local form storage.
integer (6 bytes)	39-41	40-42	Reserved - initialize to zero	
4-Byte integer	42-43	43-44	NUMRECS	Number of non-deleted records in batch file
	44-45	45-46	RECNUM	Record number of current batch record; records are counted from zero

COMAREA**Outline of COMAREA Contents (continued)**

Type	Offset		Name	Content
	SPL	Other		
integer	46-47	47-48	Reserved - initialize to zero	
Logical (2-byte)	48	49	FILEN	MPE file number of terminal
Integer	49-53	50-54	Reserved - initialize to zero	
Logical (2-byte)	54	55	RETRIES*	Maximum number of retries. value=0 - use default (4 retries) value>0 - use this value as maximum value< - no retries
	55	56	TERM OPTIONS*	Terminal control options: bits: 0-8 reserved for system use: 9-0 - 01 enables ENTER/FCN key timeout 11 or 00 disables ENTER/FCN key timeout 13-14 - 01 enables AUTOREAD 11 or 00 disables AUTOREAD 15-0 display mode message 15-1 suppress mode message; disable VSHOWFORM bell
	56	57	ENVIRON	First byte = logical device number of terminal. Second byte = reserved
	57	58	USERTIME*	User defined time-out length in seconds
	58	59	IDENTIFIER	Terminal type (see supported terminals)

* Not supported on the HP 3075, HP 3076, and HP 3081A terminals.

Outline of COMAREA Contents (continued)

Type	Offset		Name	Content
	SPL	Other		
Logical (2 bytes)	59	60	LABINFO	First byte = number of labels the terminal supports. Second byte = length of labels (characters).
Integer (2 bytes)	60-63	61-64	Reserved for system use	
	64	65	BUFFERCONTROL	control ARB processing
	65	66	BUFFERSTATUS	indicate successful data conversion
	66-69	67-70	Reserved for system use	
THIS AREA REFERENCED ONLY WHEN USING HP 3075, HP 3076 AND HP 3081 TERMINALS				
Logical	70	71	NUMFLDS	Number of fields on current form.
Two-byte integer	71	72	SPLITPAUSE	Length of time in seconds to pause between lines of text. Default: 3 seconds -1= wait for user to hit key 0= do not pause >0= wait specified number of seconds
	72	73	LEFTMODULE	Which, if any, module is present: 0= no module 1= printer 2= multifunction reader 3= RS232 interface 4= typev badge reader 5= magstrip reader 6= barcode reader 7= HP-IB interface
	73	74	RIGHTMODULE	Which, if any, module is present: 0= no module 1= printer 2= multifunction reader 3= RS232 interface 4= typev badge reader 5= magstrip reader 6= barcode reader 7= HP-IB interface

COMAREA

Outline of COMAREA Contents (continued)

Type	Offset		Name	Content
	SPL	Other		
Two-byte integer	74	75	KEYBOARD	Keyboard type: 0= standard keyboard (12 function keys) 1= alphanumeric keyboard (28 function keys)
	75	76	DISPLAY	Terminal display type: 0= numeric display. 1= alphanumeric display. 2= mini CRT display.
	76	77	KEYBOARD-OVER	Whether to override keyboard input. -1= Override and enable the keyboard without regard to forms design 0= Do not override (Default)
Character (2-Byte)	77	78	ERRORLIGHT	First byte which light will be turned on when an error is detected. (Default:E.) Second byte is reserved.
Logical (2-Byte)	78-79	79-80	USERLIGHTSON	Lights on/off indicator. Indicates whether additional lights will be turned on during run time. 0= OFF; 1= ON. (Default:OFF.) Word 1 (Bytes 1 and 2): 1 in bit position 0 = "@" light ON 1 in bit position 1 = "A" light ON 1 in bit position 2 = "B" light ON 1 in bit position 15= "0" light ON Word 2 (Bytes 3 and 4): 1 in bit position 0 = "P" light ON The remaining bits are reserved.
Integer (2-Bytes)	80-84	81-85	Reserved - initialize to zero	

Data Types Allowed for Languages

Data Type	Language				
	COBOL	FORTRAN	BASIC	PASCAL	SPL
Character	Display PIC X(n)	CHARACTER	STRING	PACKED ARRAY OF CHAR	BYTE ARRAY
Two-byte integer	COMP PIC S9 thru PIC S9(4)	INTEGER*2	INTEGER	SUBRANGE -32768..32767	INTEGER
Unsigned two-byte integer	COMP PIC 9 thru PIC 9(4)	LOGICAL	INTEGER (with value <32767)	SUBRANGE 0..65535	LOGICAL
Four-byte integer	COMP PIC S9(5) thru PIC S9(9)	INTEGER*4	INTEGER INTEGER*	INTEGER	DOUBLE
Four-byte Real	—	REAL	REAL	REAL	REAL
Eight-byte Long	—	DOUBLE PRECISION	LONG	LONGREAL	LONG
* In BASIC, a double integer can be represented by two consecutive integers; the first contains the high-order digits of values above 32767 or is zero, the second contains the low order digits of values above 32767 or the entire value up to 32767.					

Initiate ENTRY

RUN ENTRY.PUB.SYS
ENTER FORMS FILE NAME AND PRESS RETURN:
ENTER BATCH FILE NAME AND PRESS RETURN:
ENTER LANGUAGE ID NUMBER AND PRESS RETURN:*

Note: FORMS FILE is an existing file containing form definitions
BATCH FILE is new or existing MPE file name; may be fully qualified

*This prompt appears only when a language other than NATIVE-3000 is installed on your system.

Collect Mode							
	HEAD				NEXT		
	FORM	PRINT	REFRESH		FORM	BROWSE	EXIT
	f1	f2	f3	f4	f5	f6	f7 f8
Browse Mode							
	FIRST	DELETE			PREV	NEXT	
	REC	REC	PRINT	REFRESH	REC	REC	COLLECT EXIT
	f1	f2	f3	f4	f5	f6	f7 f8

ID		MODEL	GRAPH	COLOR	TOUCH	LFS	MDT	LOCAL EDITS	SEC ENH	X.25 PAD
0	✓	✓✓								
1		2640B								
2		2644A								
3		2645A								
4		2641A								
5		2648A	□							
6		2647A	□							
7	✓	2621A								
8		2626A				□			□	
9		2624A					□	□	□	
10		2642A								
11		2622A								•
12		2623A	□							•
13		2624B				□	□	□	□	•
14		2382A								
15		3075A								
15		3081A								
16		3076A								
17	+	3077A								
18	✓	2621B								
31		2647F	□							
51		2703A	□							
52		2627A	□	□						□
55		2622E								
56		2392A							□	□
57		2394A				□	□	□	□	□
61		2625A					□	□	□	□
62		2628A					□	□	□	□
63		2625A	□				□	□	□	□
64		2628A	□				□	□	□	□
65		2393A	□				□	□	□	□
66		2397A	□	□			□	□	□	□
67		2393A	□		□		□	□	□	□
68		2397A	□	□	□		□	□	□	□
70		HP150	□		□		□	□	□	□
72		700/92						□	□	□
74		700/94				□	□	□	□	□
80		IBM3270					□	□	□	

✓ Not a supported terminal

✓✓ Unknown

+ Can only be opened and closed by VPLUS/V

• Check ROMS for auto keyboard lock

LG200028_052

1

—

Special Terminal Keys

Key	Meaning
<code>CTRL X</code>	Delete current line
<code>CTRL H</code>	Delete last character
<code>CTRL Y</code>	Default subsystem break character
<code>CTRL J</code>	Turn on linefeed
<code>CTRL M</code>	Produces a <code>RETURN</code> (carriage return and linefeed)
<code>BREAK</code>	MPE break
<code>ESCAPE ;</code>	Stop echoing
<code>ESCAPE :</code>	Resume echoing
<code>CTRL S</code>	Stop output
<code>CTRL Q</code>	Resume output
<code>␣</code>	<code>ESCAPE</code>

029 Card-Punch Transliterations

ASCII	[= 029
ASCII	\ = 029 0-8-2
ASCII	] = 029 !
ASCII	L = 029]
ASCII	! = 029



ASCII Instruction Set



Byte Position

CHAR	Left	Right	Decimal	CHAR	Left	Right	Decimal
NUL	000000	000000	0	-	026400	000055	45
SOH	000400	000001	1	.	027000	000056	46
STX	001000	000002	2	/	027400	000057	47
ETX	001400	000003	3	0	030000	000060	48
EOT	002000	000004	4	1	030400	000061	49
ENQ	002400	000005	5	2	031000	000062	50
ACK	003000	000006	6	3	031400	000063	51
BEL	003400	000007	7	4	032000	000064	52
BS	004000	000010	8	5	032400	000065	53
HT	004400	000011	9	6	033000	000066	54
LF	005000	000012	10	7	033400	000067	55
VT	005400	000013	11	8	034000	000070	56
FF	006000	000014	12	9	034400	000071	57
CR	006400	000015	13	:	035000	000072	58
SO	007000	000016	14	;	035400	000073	59
SI	007400	000017	15	<	036000	000074	60
DLE	010000	000020	16	=	036400	000075	61
DC1	010400	000021	17	>	037000	000076	62
DC2	011000	000022	18	?	037400	000077	63
DC3	011400	000023	19	@	040000	000100	64
DC4	012000	000024	20	A	040400	000101	65
NAK	012400	000025	21	B	041000	000102	66
SYN	013000	000026	22	C	041400	000103	67
ETB	013400	000027	23	D	042000	000104	68
CAN	014000	000030	24	E	042400	000105	69
EM	014400	000031	25	F	043000	000106	70
SUB	015000	000032	26	G	043400	000107	71
ESC	015400	000033	27	H	044000	000110	72
FS	016000	000034	28	I	044400	000111	73
GS	016400	000035	29	J	045000	000112	74
RS	017000	000036	30	K	045400	000113	75
US	017400	000037	31	L	046000	000114	76
SPACE	020000	000040	32	M	046400	000115	77
!	020400	000041	33	N	047000	000116	78
","	021000	000042	34	O	047400	000117	79
#	021400	000043	35	P	050000	000120	80
\$	022000	000044	36	Q	050400	000121	81
%	022400	000045	37	R	051000	000122	82
&	023000	000046	38	S	051400	000123	83
'	023400	000047	39	T	052000	000124	84
(	024000	000050	40	U	052400	000125	85
)	024400	000051	41	V	053000	000126	86
*	025000	000052	42	W	053400	000127	87
+	025400	000053	43	X	054000	000130	88
,	026000	000054	44	Y	054400	000131	89
				Z	055000	000132	90

LG200028_001

CHAR	Left	Right	Decimal
[	055400	000133	91
\	056000	000134	92
]	056400	000135	93
^	057000	000136	94
_	057400	000137	95
`	060000	000140	96
a	060400	000141	97
b	061000	000142	98
c	061400	000143	99
d	062000	000144	100
e	062400	000145	101
f	063000	000146	102
g	063400	000147	103
h	064000	000150	104
i	064400	000151	105
j	065000	000152	106
k	065400	000153	107
l	066000	000154	108
m	066400	000155	109
n	067000	000156	110
o	067400	000157	111
p	070000	000160	112
q	070400	000161	113
r	071000	000162	114
s	071400	000163	115
t	072000	000164	116
u	072400	000165	117
v	073000	000166	118
w	073400	000167	119
x	074000	000170	120
y	074400	000171	121
z	075000	000172	122
{	075400	000173	123
:	076000	000174	124
}	076400	000175	125
-	077000	000176	126
DEL	077400	000177	127

LG200028_002

Index of OP Code Groups

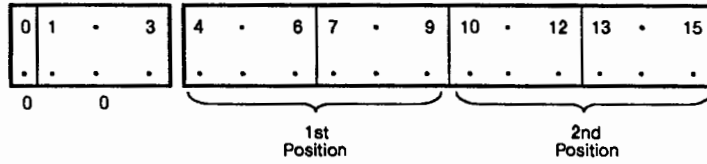
ADAX	0	CPRB	1	DZRO	0
ADBX	0	CSL	1	EADD	2
ADD	0	CSR	1	ECMP	2
ADDD	2	CVAD	2	EDIV	2
ADDI	2	CVBD	2	EMPY	2
ADDM	4	CVDA	2	ENEG	2
ADDS	3	CVDB	2	ESUB	2
ADXA	0	DABZ	1	EXF	2
ADXB	0	DADD	0	EXIT	3
ADXI	2	DASL	1	FADD	0
AND	0	DASR	1	FCMP	0
ANDI	3	DCMP	0	FDIV	0
ASL	1	DCSL	1	FIXR	0
ASR	1	DCSR	1	FIXT	0
		DDEL	0	FLT	0
BCY	1	DDIV	2	FMPY	0
BE	4	DDUP	0	FNEG	0
BG	4	DECA	0	FSUB	0
BGE	4	DECB	0	HALT	3
BL	4	DECM	4	IABZ	1
BLE	4	DECX	0	INCA	0
BNE	4	DEL	0	INCB	0
BNCY	1	DELB	0	INCM	4
BNOV	1	DFLT	0	INCX	0
BOV	1	DISP	3	IXBZ	1
BR	4	DIV	0	IXIT	2
BRE	1	DIVI	2	LADD	0
BRO	1	DIVL	0	LCMP	0
BTST	0	DLSL	1	LDB	4
CAB	0	DLSR	1	LDD	4
CIO	3	DMPY	2	LDEA	2
CMD	3	DNEG	0	LDI	2
CMP	0	DPF	2	LDIV	0
CMPB	8 0	DSUB	0	LDNI	2
CMPD	2	DTST	0	LDPN	3
CMPI	2	DUP	0	LDPP	3
CMPM	4	DXBZ	1	LDX	4
CMPN	2	DXCH	0	LDXA	0

LG200028_003

LDXB	0	NSLD	2	STAX	0
LDXI	2	OR	0	STB	4
LDXN	2	ORI	3	STBX	0
LLBL	3	PAUS	3	STD	4
LLSH	2	PCAL	3	STOR	4
LMPY	0	PCN	2	SUB	0
LOAD	4	PLDA	2	SUBD	2
LOCK	2	PSDB	3	SUBI	2
LRA	4	PSEB	3	SUBM	4
LSEA	2	PSHR	2	SUBS	3
LSL	1	PSTA	2	SXIT	3
LSR	1	QASL	2	TASL	1
LST	3	QASR	2	TASR	1
LSUB	0	RCLK	3	TBA	4
MABS	2	RIO	3	TBC	1
MDS	2	RMSK	3	TBX	4
MFDS	2	RSW	2	TCBC	1
MOVE	2	SBXI	3	TEST	0
MPY	0	SCAL	3	TIO	3
MPYD	2	SCAN	1	TNSL	1
MPYI	2	SCLK	3	TRBC	1
MPYL	0	SCU	2	TSBC	1
MPYM	4	SCW	2	UNLK	2
MTBA	4	SDEA	2	WIO	3
MTBX	4	SED	3	XAX	0
MTDS	3	SETR	2	XBX	0
MVB	2	SIN	3	XCH	0
MVBL	2	DIO	3	XCHD	3
MVBW	2	SLD	2	XEQ	3
MVLB	2	SMSK	3	XOR	0
NEG	0	SRD	2	XORI	3
NOP	0	SSEA	2	ZERO	0
NOT	0	SST	3	ZROB	0
				ZROX	0

LG200028_004

00 Stack OPs



CC = condition, O = overflow, C = carry, Cl = clears bit

C	O	C		C	O	C	
C				C			
			00 NOP				40 DEL
			01 DELB				41 ZROB
			02 DDEL	A			42 LDXB
A	X	X	03 ZROX	A			43 STAX
A	X	X	04 INCX	A			44 LDXA
A	X	X	05 DECX	A			45 DUP
			06 ZERO	A			46 DDUP
			07 DZRO	A			47 FLT
C			10 DCMP	C			50 FCMP
A	X	X	11 DADD	A	X		51 FADD
A	X	X	12 DSUB	A	X		52 FSUB
A	Cl	X	13 MPYL	A	X		53 FMPY
A	X		14 DIVL	A	X		54 FDIV
A	X		15 DNEG	A			55 FNEG
A			16 DXCH	A			56 CAB
C			17 CMP	C			57 LCMP
A	X	X	20 ADD	A		X	60 LADD
A	X	X	21 SUB	A		X	61 LSUB
A	X		22 MPY	A		X	62 LMPY
A	X		23 DIV	A	X		63 LDIV
A	X	X	24 NEG	A			64 NOT
A			25 TEST	A			65 OR
A			26 STBX	A			66 XOR
A		X	27 DTST	A			67 AND
A			30 DFLT	A	X	X	70 FIXR
B			31 BTST	A	X	X	71 FIXT
A			32 XCH				72 Reserved
A	X	X	33 INCA	A	X	X	73 INCB
A	X	X	34 DECA	A	X	X	74 DECB
A			35 XAX				75 XBX
A	X	X	36 ADAX	A	X	X	76 ADBX
A	X	X	37 ADXA	A	X	X	77 ADXB

L0200028_005

15-6 ASCII Instruction Set

01 Shifts/Branches

0	1	•	3	4	•	6	7	•	9	10	•	12	13	•	15
•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
0	1														
A				X	0	0	0			← SC →					ASL
A	O	C		X	0	0	1			← SC →					ASR
A				X	0	0	2			← SC →					LSL
A				X	0	0	3			← SC →					LSR
A				X	0	0	4			← SC →					CSL
A				X	0	0	5			← SC →					CSR
A				X	0	0	6		0		0				SCAN
A	X	X		1	0	0	7		+/-	← P branch →					IABZ
A				X	0	1	0			← SC →					TASL
A				X	0	1	1			← SC →					TASR
A	X	X		1	0	1	2		+/-	← P branch →					IXBZ
A	X	X		1	0	1	3		+/-	← P branch →					DXBZ
		CI		1	0	1	4		+/-	← P branch →					BCY
		CI		1	0	1	5		+/-	← P branch →					BNCY
A				X	0	1	6		0		0				TNSL
A					0	1	7			← SC →					QASL X
A										← SC →					QASR X
A				X	1	0	0			← SC →					DASL
A				X	1	0	1			← SC →					DASR
A				X	1	0	2			← SC →					DLSL
A				X	1	0	3			← SC →					DLSR
A				X	1	0	4			← SC →					DCSL
A				X	1	0	5			← SC →					DCSR
•				1	1	0	6		+/-	← P branch →					CPRB§
A	X	X		1	1	0	7		+/-	← P branch →					DABZ
	CI			1	1	1	0		+/-	← P branch →					BOV
	CI			1	1	1	1		+/-	← P branch →					BNQV
••				X	1	1	2			bit position					TBC
••				X	1	1	3			bit position					TRBC
••				X	1	1	4			bit position					TSBC
••				X	1	1	5			bit position					TCBC
				1	1	1	6		+/-	← P branch →					BRO
				1	1	1	7		+/-	← P branch →					BRE
C	O	C													
C															

CI = clears bit

§ = uses Index Reg

•• bit = 0 CCE, bit = 1 CCG or CCL

SC = shift count (0-63)

P branch signed magnitude (0-31)

bit position (0-63) [MOD 16]

CPRB X>(S) CCG X<(S-1) CCL

(S-1) <= X < (S) CCE

LG200026_006

02 Moves/Immediates

02 MOVES/IMMEDIATES

0	1	•	3	4	•	6	7	•	9	10	•	12	13	•	15
•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
0	2														
				0	0	0	PB/DB	0		SDEC	MOVE				
				0	0	1	PB/DB	0		SDEC	MVB				
				0	1	0				SDEC	MVBL				
B				0	1	1				SDEC	MABS				
				0	1	2				SDEC	SCW				
				0	1	3				SDEC	MTDS				
				0	1	4				SDEC	MVLB				
				0	1	5				SDEC	MDS				
				0	1	6				SDEC	SCU				
				0	1	7				SDEC	MFDS				
B				0	2	0	N/A		U	SDEC	MVBW				
C				0	2	1	PB/DB	0		SDEC	CMPB				
A				0	3	0					RSW				
A				0	3	0			1		LLSH\$				
A				0	3	2			0		PLDA\$				
				0	3	2			1		PSTA\$				
A				0	3	4			0		LSEA				
				0	3	4			1		SSEA				
A				0	3	4			2		LDEA				
				0	3	4			3		SDEA				
				0	3	6			0		IXIT				
				0	3	6			1		Reserved				
				0	3	6			2		PCN				
				0	3	6			3		Reserved				
A	X			0	4	1			0		EADD				
A	X			0	4	1			1		ESUB				
A	X			0	4	1			2		EMPTY				
A	X			0	4	1			3		EDIV				
A				0	4	1			4		ENEG				
C				0	4	1			5		ECMP				
				0	5	7			0		DMUL				
				0	5	7			1		DDIV				
A		X		0	6	0			1		DMPY				

LG200028_007

0	1	.	3	4	.	6	7	.	9	10	.	12	13	.	15
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

A	X			0	6	0	2	CVAD
A	X			0	6	0	3	CVDA
A	X			0	6	0	4	CVBD
A	X		0 2	0	6	0	5	CVDB
A	X	X		0	6	0	6	SLD
A	X	X		0	6	0	7	NSLD
A	X			0	6	1	0	SRD
A	X			0	6	1	1	ADDD
C	X			0	6	1	2	CMPD
A	X			0	6	1	3	SUBD
A	X			0	6	1	4	MPYD
A				1	0	← Imm Opr	→	LDI
A				1	1	← Imm Opr	→	LDXI
C				2	0	← Imm Opr	→	CMPI
A	X	X		2	1	← Imm Opr	→	ADDI
A	X	X		3	0	← Imm Opr	→	SUBI
A	X			3	1	← Imm Opr	→	MPYI
A				4	0	← Imm Opr	→	DIVI
				4	1	SBK DB DL Z STA X Q S		PSHR
A				5	0	← Imm Opr	→	LDNI
A				5	1	← Imm Opr	→	LDXN
C				6	0	← Imm Opr	→	CMPN
A				6	1	J J J J J K K K K		EXF
A				7	0	J J J J J K K K K		DPF
				7	1	SBK DB DL Z STA X Q S		SETR
C	O	C						

§ = uses Index Register
 SDEC pop stack (0-3)
 Imm Opr Immediate operand (0-255)
 JJJJ Beginning bit position (0-15)
 KKKK field length (0-15)
 * SCW CARRY → Terminating Character
 * SCU CARRY → Terminating Character

LG200028_008

03 I/O Linkage Control

0	1	•	3	4	•	6	7	•	9	10	•	12	13	•	15
•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•

0				0	0	0	0	K	K	K	K	LST
				0	0	0	1	0	0	0	0	PAUS
				0	0	1	0	0	0	0	D/E	SED
3				0	0	0	6		0			XCHD
				0	0	6			1			PSDB
				0	0	6			2			DISP
3				0	0	6			3			PSEB
				0	1	0			0			SMSK
				0	1	0			1			SCLK
				0	1	2			0			RMSK
				0	1	2			1			RCLK
•				0	1	1	0	K	K	K	K	XEQ
D				0	1	1	1	K	K	K	K	SIO
D				0	2	0	0	K	K	K	K	RIO
D				0	2	0	1	K	K	K	K	WIO
1				0	2	1	0	K	K	K	K	TIO
1				0	2	1	1	K	K	K	K	CIO
				0	3	0	0	K	K	K	K	CMD
1				0	3	0	1	K	K	K	K	SST
				0	3	1	0	K	K	K	K	SIN
				0	3	1	1	K	K	K	K	HALT
				0	1	←	STT	→				SCAL
				1	0	←	STT	→				PCAL
				1	1	←	SDEC + (4)	→				EXIT
				2	0	←	SDEC + (1)	→				SXIT
A				2	1	←	Imm Opr	→				ADX1
A				3	0	←	Imm Opr	→				SBXI
				3	1	←	PL - Disp	→				LLBL
				4	0	←	P - Disp	→				LDPP
				4	1	←	P - Disp	→				LDPN
				5	0	←	Imm Opr	→				ADD5
				5	1	←	Imm Opr	→				SUB5
				6	0		0		0			Reserved
A				6	1	←	Imm Opr	→				ORI
A				7	0	←	Imm Opr	→				XORI
A				7	1	←	Imm Opr	→				ANDI
C	O	C										

KKKK Stack displacement (0-15)
STT entry position (0-255)

LG200028_009

Imm Opr Immediate operand (0-255)
1. normally CCE, non-responding CCL
D norm CCE, non-respond CCLL, not ready CCG
3. norm CCE, IF error CCL
* XEQ depends on instruction executed.



Memory Reference

0	1	•	3	4	•	6	7	•	9	10	•	12	13	•	15
•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
0			3	0	4										
A				X	1	0									
A				X	1	1									
				0	5										
						0									
						2									
						4									
						6									
						X	1	1							
C				0	6	X	1	0							
C						X	1	1							
A	X	X		0	7	X	1	0							
A	X	X				X	1	1							
A	X	X	1	0		X	1	0							
A	X	X				X	1	1							
A	X	X	1	1		X	1	0							
A	X	X				X	1	1							
A	X	X	1	2		X	1	0							
A	X	X				X	1	1							
A			1	3		X	1	0							
A						X	1	1							
				1	4	X	1	0							
						X	1	1							
						1	0	1							
B				1	5	X	1	0							
A						X	1	1							
				1	6	X	1	0							
						X	1	1							
				1	7	X	1	0							
						X	1	1							
C	C	O	C												

7 8 9 10 - 12 13 - 15
P { P+ 0 • • • • • • • • 0:377
P- 1 • • • • • • • • 0:377

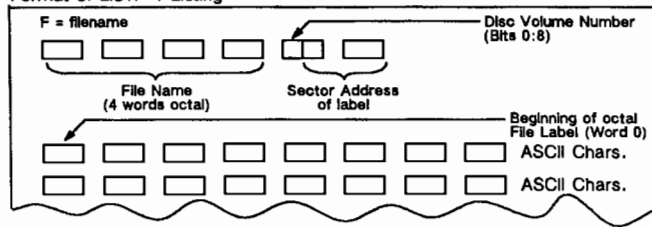
DQS { DB+ 0 • • • • • • • • 0:377
Q+ 1 0 • • • • • • • • 0:177
Q- 1 1 0 • • • • • • • • 0:77
S- 1 1 1 • • • • • • • • 0:77

GEL { 1. Less than (BL)
2. Equal (BE)
3. Less than or equal (BLE)
4. Greater than (BG)
5. Not equal (BNE)
6. Greater or equal (BGE)

LQ200028_010

File Labels

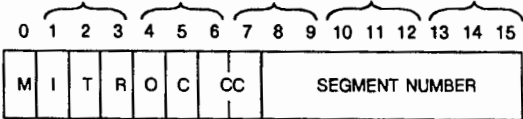
Format of LISTF-1 Listing



Words Dec.	Octal	Contents
0-3	0-3	Local file name.
4-7	4-7	Group name.
8-11	10-13	Account name.
12-15	14-17	Identity of file creator.
16-19	20-23	File lockword.
20-21	24-25	File security matrix.
22	26	(Bits 0:15) Not used.
		(Bit 15:1) File secure bit:
		If 1, file secured.
		If 0, file released.
23	27	File creation date.
24	30	Last access date.
25	31	Last modification date.
26	32	File code.
27	33	File control block vector.
28	34	(Bit 0:1) Store Bit. (If on, :STORE or :RESTORE in progress.)
		(Bit 1:1) Restore Bit. (If on, :RESTORE in progress.)
		(Bit 2:1) Load Bit. (If on, program file is loaded.)
		(Bit 3:1) Exclusive Bit. (If on, file is opened with exclusive access.)
		(Bits 4:4) Device sub-type.
		(Bits 8:6) Device type.
		(Bit 14:1) File is open for write.
		(Bit 15:1) File is open for read.
29	35	(Bits 0:8) Number of user labels written.
		(Bits 8:8) Number of user labels.
30-31	36-37	Maximum number of logical records.
32-33	40-41	Private Volume Information.
34	42	Checksum.
35	43	Cold-load identity.
36	44	F options specifications.
37	45	Logical record size (in negative bytes).
38	46	Block size (in words).
39	47	(Bits 0:8) Sector offset to data.
		(Bits 8:3) Not used.
		(Bits 11:5) Number of extents minus 1.
40	50	Logical size of last block.
41	51	Extent size.
42-43	52-53	Number of logical records in file.
See Appendix B in Intrinsic Manual		
44-107	54-153	Two-word addresses of up to 32 disc extents, beginning with address of first extent (words 44-45).
		Device Class.
125-128	175-200	

LG200028_011

Status Register



- M Mode User Privileged
- I Ext Interrupts Enabled
- T User Traps Enabled
- R Right Stack Op Pending
- O Overflow
- C Carry
- CC Condition Code
 - CCL = 1
 - CCE = 2
 - CCG = 0

CONDITION CODES

CCA CCL Operand < 0
 CCE Operand = 0
 CCG Operand > 0

CCB CCL Special ASCII Char.
 CCE Alphabetic
 CCG Numeric

CCC CCL Operand 1 < OPR 2
 CCE Operand 1 = OPR 2
 CCG Operand 1 > OPR 2

CCD CCL Non-responding device controller
 CCE Responding device controller, or device ready
 CCG Device not ready (busy)

CC FIELD

6-7	Condition
0	> G
1	< L
2	= E
3	Reserved

STACK MARKER (normal)

X	INDEX
Relative P	P + 1 - PB
STATUS	
Q,S	ΔQ

SPECIAL STACK MARKER (resulting from ICS interrupt)

X	INDEX
Relative P	P + 1 - PB
STATUS	
Q	ΔQ
	DB BANK
S	DB

LG200028_012

Code Segment Table

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	M	R	T	LENGTH											
RESERVED															
													PB BANK		
ADDRESS															

A Absent bit (=1 if segment is absent)

M Mode bit (=1 if privileged mode)

R Reference bit (for statistical use by operating system, set to 1 when accessed)

T Trace bit (=1 to call Trace routine)

LENGTH This value times 4 (max = 16,380)

ADDRESS Absolute memory address (for PB)
or low order 16 bits of absolute disc address
if absent.

PB BANK Bank number if present of High Order
Disc address if absent.

SEGMENT TRANSFER TABLE Words

STT Length

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	U	0	0	0	0	0	0	LENGTH							

U Uncallable bit for STT#=0 (enter at PB + 0)

LENGTH Maximum = 255 (Calls from external
segments may reference only the first 128
entries, PL thru PL-127.)

Local Program Label

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	U	ADDRESS													

U Uncallable bit

ADDRESS PB relative, + only

External Program Label

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	STT#								SEG#						

STT # = STT = entry number in target segment,
maximum = 127

SEG # = Target segment

LG200028_013

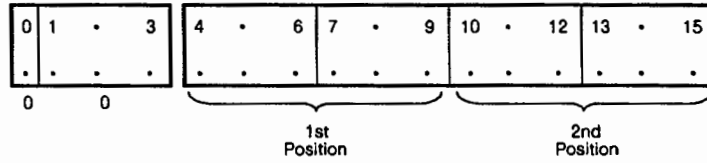
Interrupts/Traps

STT Entry Number	Interrupt	Parameter
1	EXTERNAL INTERRUPT	DEV #
2	BOUNDS VIOLATION	100401
3	ILLEGAL MEMORY ADDRESS	101001
4	NON-RESPONDING MODULE	101401
5	SYSTEM PARITY ERROR	102001
6	ADDRESS PARITY ERROR	102401
7	DATA PARITY ERROR	103001
	MODULE INTERRUPT	MODULE #
11	POWER FAIL	104401
20	UNIMPLEMENTED INSTRUCTION	110001
21	STT VIOLATION	110401
22	CST VIOLATION	111001
23	DST VIOLATION	111401
24	STACK UNDERFLOW	112001
25	PRIVILEGED MODE VIOLATION	112401
30	STACK OVERFLOW	114001
31	USER TRAPS	
	INTEGER OVERFLOW	1
	FLOATING-POINT OVERFLOW	2
	FLOATING-POINT UNDERFLOW	3
	INTEGER DIVIDE BY 0	4
	FLOATING-POINT DIVIDE BY 0	5
	EXT. PRECISION OVERFLOW	10
	EXT. PRECISION UNDERFLOW	11
	EXT. PRECISION DIVIDE BY 0	12
	DECIMAL OVERFLOW	13
	INVALID ASCII DIGIT	14
	INVALID DECIMAL DIGIT	15
	INVALID SOURCE WORD COUNT	16
	RESULT WORD COUNT OVERFLOW	17
	DECIMAL DIVIDE BY 0	20
37	ABSENT CODE SEGMENT	
	PCAL	P-LABEL
	EXIT	N
	IXIT	0
40	TRACE	
	PCAL	P-LABEL
	EXIT	N
	IXIT	0
41	STT ENTRY UNCALLABLE	P-LABEL
42	ABSENT DATA SEGMENT	DST #
43	POWER ON	121401
44	COLD LOAD	0

Note: If parameter not shown, parameter is external program label.

LG200026_014

00 Stack Ops



CC = condition, O = overflow, C = carry, CI = clears bit

C					C				
C	O	C			C	O	C		
			00	NOP				40	DEL
			01	DELB				41	ZROB
			02	DDEL	A			42	LDXB
A	X	X	03	ZROX	A			43	STAX
A	X	X	04	INCX	A			44	LDXA
A	X	X	05	DECX	A			45	DUP
			06	ZERO	A			46	DDUP
			07	DZRO	A			47	FLT
C			10	DCMP	C			50	FCMP
A	X	X	11	DADD	A	X		51	FADD
A	X	X	12	DSUB	A	X		52	FSUB
A	CI	X	13	MPYL	A	X		53	FMPY
A	X		14	DIVL	A	X		54	FDIV
A	X		15	DNEG	A			55	FNEG
A			16	DXCH	A			56	CAB
C			17	CMP	C			57	LCMP
A	X	X	20	ADD	A		X	60	LADD
A	X	X	21	SUB	A		X	61	LSUB
A	X		22	MPY	A		X	62	LMPY
A	X		23	DIV	A	X		63	LDIV
A	X	X	24	NEG	A			64	NOT
A			25	TEST	A			65	OR
A			26	STBX	A			66	XOR
A		X	27	DTST	A			67	AND
A			30	DFLT	A	X	X	70	FIXR
B			31	BTST	A	X	X	71	FIXT
A			32	XCH				72	Spare
A	X	X	33	INCA	A	X	X	73	INCB
A	X	X	34	DECA	A	X	X	74	DECB
A			35	XAX				75	XBX
A	X	X	36	ADAX	A	X	X	76	ADBX
A	X	X	37	ADXA	A	X	X	77	ADXB

LQ200028_015

01 Shifts/Branches

0	1	•	3	4	•	6	7	•	9	10	•	12	13	•	15
•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
0	1			0	1	X	0	0	0	← SC →					
A						X	0	0	1	← SC →					ASL
A						X	0	0	2	← SC →					ASR
A						X	0	0	3	← SC →					LSL
A						X	0	0	4	← SC →					LSR
A						X	0	0	5	← SC →					CSL
A						X	0	0	6	← SC →					CSR
A						X	0	0	7	← SC →					SCAN
A	X	X				X	0	0	0	+/- ← P branch →					IABZ
A						X	0	1	0	← SC →					TASL
A						X	0	1	1	← SC →					TASR
A	X	X				1	0	1	2	+/- ← P branch →					IXBZ
A	X	X				1	0	1	3	+/- ← P branch →					DXBZ
A			CI			1	0	1	4	+/- ← P branch →					BCY
A			CI			1	0	1	5	+/- ← P branch →					BNCY
						X	0	1	6	← SC →					TNSL
						1			7	← SC →					Spare
A						X	1	0	0	← SC →					DASL
A						X	1	0	1	← SC →					DASR
A						X	1	0	2	← SC →					DLSL
A						X	1	0	3	← SC →					DLSR
A						X	1	0	4	← SC →					DCSL
A						X	1	0	5	← SC →					DCSR
*						1	1	0	6	+/- ← P branch →					CPRB§
A	X	X				1	1	0	7	+/- ← P branch →					DABZ
			CI			1	1	1	0	+/- ← P branch →					BOV
			CI			1	1	1	1	+/- ← P branch →					BNOV
**						X	1	1	2	bit position →					TBC
**						X	1	1	3	bit position →					TRBC
**						X	1	1	4	bit position →					TSBC
**						X	1	1	5	bit position →					TCBC
						1	1	1	6	+/- ← P branch →					BRO
						1	1	1	7	+/- ← P branch →					BRE
A	O	C				0	0	1	7	← SC →					QASR
A						1	0	1	7	← SC →					QASL

CI = clears bit

§ = uses Index Reg

** bit = 0 CCE, bit = 1 CCG or CCL

SC = shift count (0-63)

P branch signed magnitude (0-31)

bit position (0-63) [MOD 16]

CPRB X<(S-1) CCL, X>(S) CCG
(S-1) ≤ X ≤ (S) CCE

LG200028_016

02 Moves/Immediates

0	1	•	3	4	•	6	7	•	9	10	•	12	13	•	15
•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
			0	2			0	0	0 PB/DB 0				SDEC		MOVE
							0	0	1 PB/DB 0				SDEC		MVB
							0	1	0				SDEC		MVBL
							0	1	0			1			two-
															two-word "hardware" instructions. See page 5.
							0	1	1				SDEC		MABS
B		*					0	1	2				SDEC		SCW
							0	1	3				SDEC		MTDS
							0	1	4				SDEC		MVLB
		*					0	1	5				SDEC		MDS
							0	1	6				SDEC		SCU
							0	1	7				SDEC		MFDS
B							0	2	0 N/A				SDEC		MVBW
C							0	2	1 PB/DB 0				SDEC		CMPB
							0	3	0			0			RSW
A							0	3	0			1			LLSH§
			0	2			0	3	0			2			two-
															two-word I/O instruction. See page 5.
A							0	3	2			0			PLDA§
							0	3	2			1			PSTA§
A							0	3	4			0			LSEA
							0	3	4			1			SSEA
A							0	3	4			2			LDEA
							0	3	4			3			SDEA
							0	3	6			0			IXIT
							0	3	6			1			NOP
							0	3	6			2			PCN
							0	3	6			3			NOP
A	X						0	4	1			0			EADD
A	X						0	4	1			1			ESUB
A	X						0	4	1			2			EMPTY
A	X						0	4	1			3			EDIV
A							0	4	1			4			ENEG
C							0	4	1			5			ECMP
							0	5	7			0			DMUL
							0	5	7			1			DDIV
A		X					0	6	0			1			DMPY
C	O	C													
C															

LG200028_017

0	1	.	3	4	.	6	7	.	9	10	.	12	13	.	15
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

A	X			0	6	S	0		2	CVAD
A	X			0	6	S	0		3	CVDA
A	X			0	6	S	0		4	CVBD
A	X		0 2	0	6	S	0		5	CVDB
A	X	X		0	6	S	S	0	6	SLD
A	X	X		0	6	S	S	0	7	NSLD
A	X			0	6	S	S	1	0	SRD
A	X			0	6	S	S	1	1	ADDD
C	X			0	6	S	S	1	2	CMPD
A	X			0	6	S	S	1	3	SUBD
A	X			0	6	S		1	4	MPYD
A				1	0			Imm Opr		LDI
				1	1			Imm Opr		LDXI
C				2	0			Imm Opr		CMPI
A	X	X		2	1			Imm Opr		ADDI
A	X	X		3	0			Imm Opr		SUBI
A	X			3	1			Imm Opr		MPYI
A				4	0			Imm Opr		DIVI
				4	1	SBK	DB	DL	Z STA X Q S	PSHR
A				5	0			Imm Opr		LDNI
				5	1			Imm Opr		LDXN
C				6	0			Imm Opr		CMPN
A				6	1	J J		J J K	K K K	EXF
A				7	0	J J		J J K	K K K	DPF
				7	1	SBK	DB	DL	Z STA X Q S	SETR
C	O	C								
C										

S } SDEC
SS }

S = uses Index Reg

SDEC pop stack (0-3)

Imm Opr Immediate operand (0-255)

JJJJ Beginning bit position (0-15)

KKKK field length (0-15)

* SCW CARRY → Terminating Character

* SCU CARRY → Terminating Character

LG200028_018

Two-Word “Hardware” Instructions

First Word	Second Word	MNE
020303	000000	SIOP
	1	HIOP
	2	RIOC
	3	WIOC
	6	INIT
	7	MCS
	10	SEML
	11	STRT
	12	DUMP

Instruction Codes 0–6 Set CC

Note: The SEML instruction is not available on the Series 44.

LG200028_019a

Two-Word I/O Instructions

First Word	Second Word	MNE
020104	000000	RCCR
	000001	SCLR
	000002	TOFF
	000003	TON
	000004	SBM
	000007	Reserved
	000010	SINC

No Condition Codes Set

Note: The SBM instruction is not available on the Series 30/33.

LG200028_051

Memory Reference

0	1	.	3	4	.	6	7	.	9	10	.	12	13	.	15
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

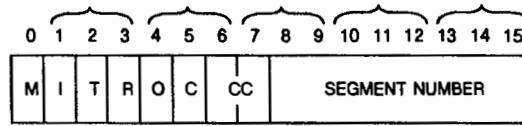
A				0	4	X	1	0	← P →	LOAD
A						X	1	1	← DQS →	
				0	5		0	+/-	← P rel branch →	TBA
							2	+/-	← P rel branch →	MTBA
							4	+/-	← P rel branch →	TBX
							6	+/-	← P rel branch →	MTBX
						X	1	1	← DQS →	STOR
C				0	6	X	1	0	← P →	CMPM
C						X	1	1	← DQS →	
A	X	X		0	7	X	1	0	← P →	ADDM
A	X	X				X	1	1	← DQS →	
A	X	X	X	1	0	X	1	0	← P →	SUBM
A	X	X	X			X	1	1	← DQS →	
A	X	X	X	1	1	X	1	0	← P →	MPYM
A	X	X	X			X	1	1	← DQS →	
A	X	X	X	1	2	X	1	0	← DQS →	INCM
A	X	X	X			X	1	1	← DQS →	DECM
A				1	3	X	1	0	← P →	LDX
A						X	1	1	← DQS →	
				1	4	X	1	0	+/- ← P rel branch →	BR
						X	1	1	← DQS indirect →	BR
						1	0	1	GEL +/- ← P branch →	BCC
B				1	5	X	1	0	← DQS →	LDB
A						X	1	1	← DQS →	LDD
				1	6	X	1	0	← DQS →	STB
						X	1	1	← DQS →	STD
				1	7	X	1	0	← P →	LRA
						X	1	1	← DQS →	
C										
C	O		C							

		7	8	9
P	{	P+	0	0:377
		P-	1	0:377
DQS	{	DB+	0	0:377
		Q+	1 0	0:177
		Q-	1 1 0	0:77
		S-	1 1 1	0:77

GEL	{	1. Less than (BL)	4. Greater than (BG)
		2. Equal (BE)	5. Not equal (BNE)
		3. Less than or equal (BLE)	6. Greater or equal (BGE)

LG200028_021

Status Register



M Mode User Privileged
 I Ext Interrupts Enabled
 T User Traps Enabled
 R Right Stack Op Pending
 O Overflow
 C Carry
 CC Condition Code
 CCL = 1
 CCE = 2
 CCG = 0

CONDITION CODES

CCA CCL Operand < 0
 CCE Operand = 0
 CCG Operand > 0

 CCB CCL Special ASCII Char.
 CCE Alphabetic
 CCG Numeric

 CCC CCL Operand 1 < OPR 2
 CCE Operand 1 = OPR 2
 CCG Operand 1 > OPR 2

CC FIELD

6-7	Condition
0	> G
1	< L
2	= E
3	Undefined

STACK MARKER (normal)

X	INDEX
	Relative P P + 1 - PB
	STATUS
Q.S	ΔQ

SPECIAL STACK MARKER (resulting from ICS interrupt)

X	INDEX
	Relative P P + 1 - PB
	STATUS
Q	ΔQ
	DB BANK
S	DB

L.G200028_022

Code Segment Table

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	M	R	T	LENGTH/4											
RESERVED															
RESERVED												PB BANK			
ADDRESS															

A Absent bit (=1 if segment is absent)

M Mode bit (=1 if privileged mode)

R Reference bit (for statistical use by operating system, set to 1 when accessed)

T Trace bit (=1 to call Trace routine)

LENGTH This value times 4 (max = 16,380) = segment length

ADDRESS Absolute memory address (for PB) or low order 16 bits of absolute disc address if absent.

PB BANK Bank number if present of High Order Disc Address if absent.

SEGMENT TRANSFER TABLE Words

STT Length

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	U	0	0	0	0	0	0	LENGTH							

U Uncallable bit

LENGTH Maximum = 255 (Calls from external segments may reference only the first 128 entries, PL thru PL-127.)

Local Program Label

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	U	ADDRESS													

U Uncallable bit

ADDRESS PB relative, + only

External Program Label

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	STT#								SEG#						

STT # = STT = entry number in target segment, maximum = 127

SEG # = Target segment

LG200028_023

Interrupts/Traps

STT Entry Number	Interrupt	Parameter
	EXTERNAL INTERRUPT	DEV #
1	BOUNDS VIOLATION	100401
3	NON-RESPONDING MODULE	101401
6	MEMORY DATA PARITY ERROR	103001
11	POWER FAIL	104401
12	SYSTEM CLOCK	(CR-LC)
20	UNIMPLEMENTED INSTRUCTION	110001
21	STT VIOLATION	110401
22	CST VIOLATION	111001
23	DST VIOLATION	111401
24	STACK UNDERFLOW	112001
25	PRIVILEGED MODE VIOLATION	112401
30	STACK OVERFLOW	114001
31	USER TRAPS	114401
	INTEGER OVERFLOW	1
	FLOATING-POINT OVERFLOW	2
	FLOATING-POINT UNDERFLOW	3
	INTEGER DIVIDE BY 0	4
	FLOATING-POINT DIVIDE BY 0	5
	EXT. PRECISION OVERFLOW	10
	EXT. PRECISION UNDERFLOW	11
	EXT. PRECISION DIVIDE BY 0	12
	DECIMAL OVERFLOW	13
	INVALID ASCII DIGIT	14
	INVALID DECIMAL DIGIT	15
	INVALID SOURCE WORD COUNT	16
	RESULT WORD COUNT OVERFLOW	17
	DECIMAL DIVIDE BY 0	20
37	ABSENT CODE SEGMENT	
	PCAL	P-LABEL
	EXIT	N
	IXIT	0
40	TRACE	
	PCAL	P-LABEL
	EXIT	N
	IXIT	0
41	STT ENTRY UNCALLABLE	P-LABEL
42	ABSENT DATA SEGMENT	DST #
43	POWER ON	121401
44	COLD LOAD	CHAN-DEV#

Note: If parameter not shown, parameter is external program label.

LQ200028_024

Memory Conversion Table

128	words/PAGE or SECTOR
512	pages/BANK
65536	words/BANK
524288	words/megabyte
8	BANKS/megabyte
4096	pages/megabyte

LG200028_032

Part No. 32033-90023
Printed in U.S.A. 1989
E1089

