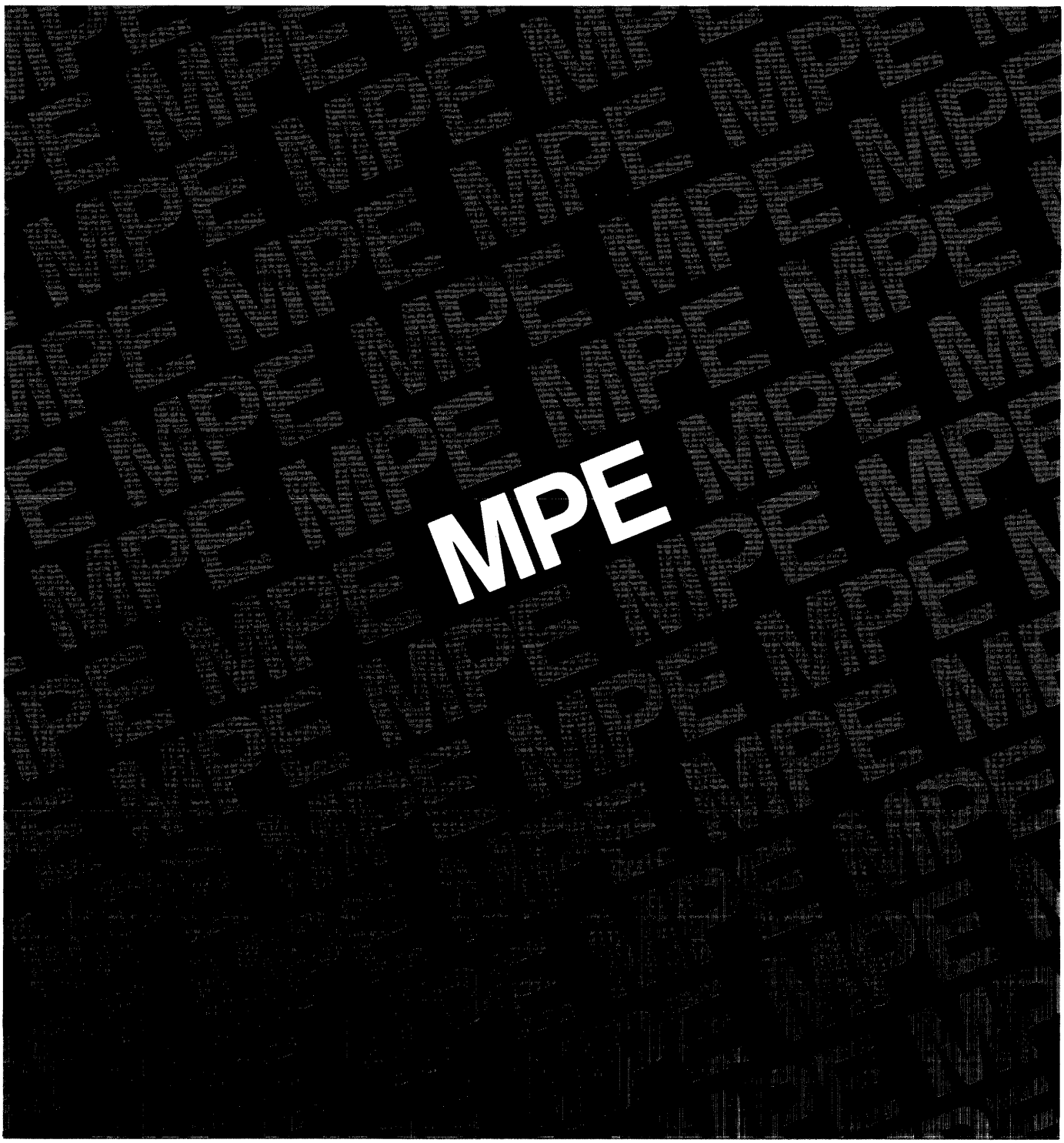
HP 3000 Computer Systems

HEWLETT PACKARD

MPE Commands
reference manual



HP 3000 Computer Systems

MPE Commands

Reference Manual

HEWLETT PACKARD

19447 PRUNERIDGE AVE., CUPERTINO, CALIFORNIA, 95014

Part No. 30000-90009
Product No. 32002B

Printed in U.S.A. 1/81

NOTICE

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 for incidental or consequential damages in connection with the furnishing, performance 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.

PREFACE

This manual documents the commands which control the Multiprogramming Executive Operating System on the HP 3000.

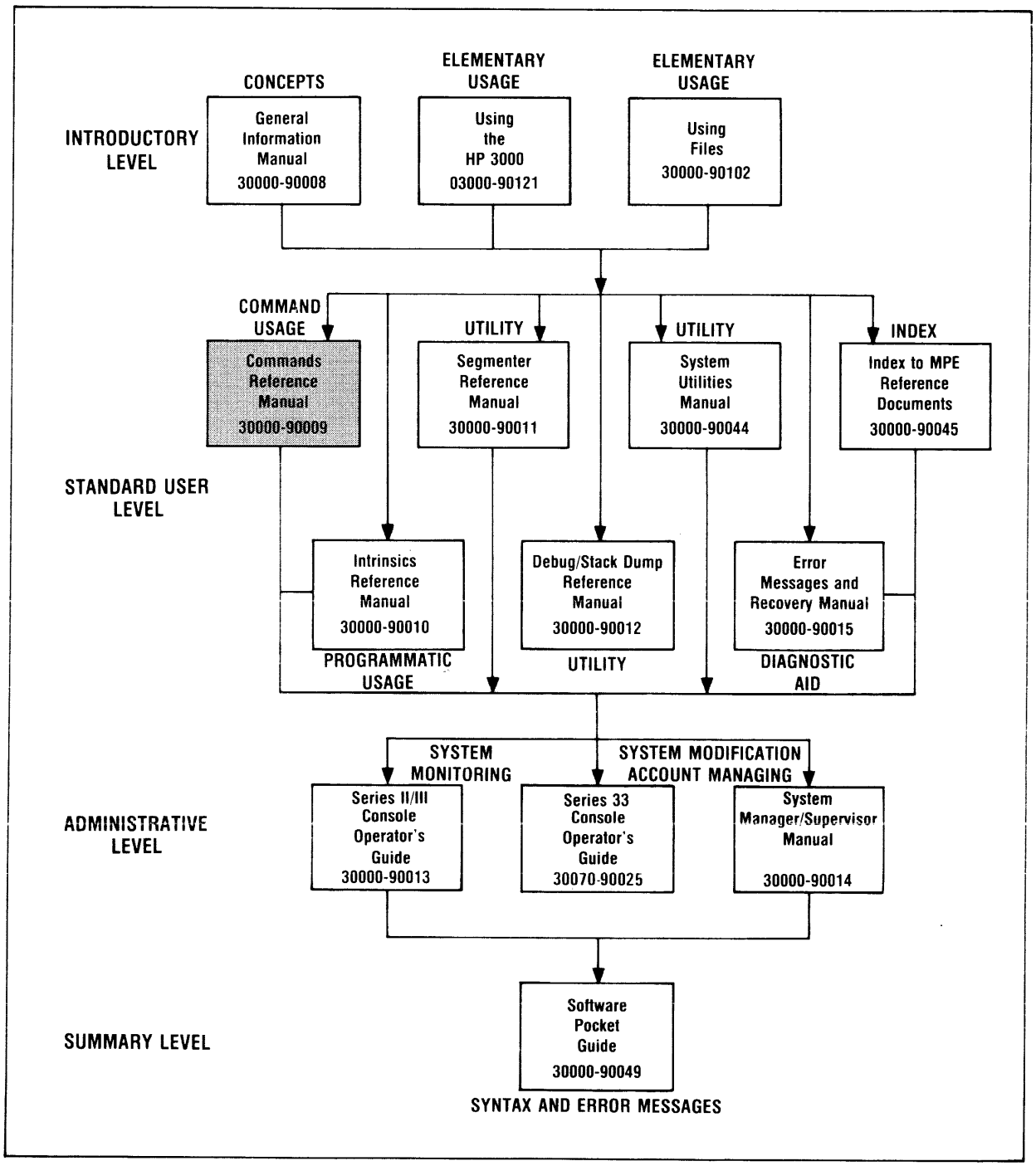
This fourth edition specifies commands and describes the MPE Command Interpreter as compatible with the MPE IV operating system. Readers familiar with previous editions will notice that the manual has undergone internal revisions which result in a more reference type document. Because of this, some of the information you may be used to finding in this manual has been deleted.

It is assumed that you know how to log on to your terminal and have some familiarity with the system. If you need further help or information, refer to the ADDITIONAL DISCUSSION section of each command where additional references may be listed.

As a general guideline, the following supportive documentation will provide any in-depth discussions you may require:

- MPE INTRINSICS REFERENCE MANUAL (30000-90010)
- USING FILES (30000-90102)
- CONSOLE OPERATOR'S GUIDE (30000-90013 & 30070-90025)
- SYSTEM MANAGER/SYSTEM SUPERVISOR MANUAL (30000-90014)

MANUAL PLAN


INTRODUCTORY LEVEL
CONCEPTS
General Information Manual
30000-90008
ELEMENTARY USAGE
Using the HP 3000
03000-90121
ELEMENTARY USAGE
Using Files
30000-90102
COMMAND USAGE
Commands Reference Manual
30000-90009
UTILITY
Segmenter Reference Manual
30000-90011
UTILITY
System Utilities Manual
30000-90044
INDEX
Index to MPE Reference Documents
30000-90045
STANDARD USER LEVEL
Intrinsics Reference Manual
30000-90010
PROGRAMMATIC USAGE
Debug/Stack Dump Reference Manual
30000-90012
UTILITY
Error Messages and Recovery Manual
30000-90015
DIAGNOSTIC AID
SYSTEM MONITORING
SYSTEM MODIFICATION ACCOUNT MANAGING
ADMINISTRATIVE LEVEL
Series II/III Console Operator's Guide
30000-90013
Series 33 Console Operator's Guide
30070-90025
System Manager/Supervisor Manual
30000-90014
SUMMARY LEVEL
Software Pocket Guide
30000-90049
SYNTAX AND ERROR MESSAGES

CONVENTIONS USED IN THIS MANUAL

NOTATION	DESCRIPTION
[ ]	An element inside brackets is *optional*. Several elements stacked inside a pair of brackets means the user may select any one or none of these elements. Example: [A / B] user may select A or B or neither
{ }	When several elements are stacked within braces the user **must** select one of these elements. Example: {A / B / C} user must select A or B or C.
italics	Lowercase italics denote a parameter which must be replaced by a user-supplied variable. Example: CALL *name*
underlining	Dialogue: Where it is necessary to distinguish user input from computer output, the input is underlined. Example: NEW NAME? ALPHA1
superscript C	Control characters are indicated by a superscript C Example: Y^C
return	*return* in italics indicates a carriage return
linefeed	*linefeed* in italics indicates a linefeed
. . .	A horizontal ellipsis indicates that a previous bracketed element may be repeated, or that elements have been omitted.

CONTENTS

CONTENTS (continued)

ILLUSTRATIONS

TABLES

INTRODUCTION TO COMMANDS

SECTION I

MPE commands allow you to initiate, control, and terminate the processing of programs and to request various other system operations. You generally use them for functions external to the source-language programs that you write, although many of these functions may be necessary to support those programs. For example, you use commands to:

- Initiate an interactive session (:HELLO command) or batch job (:JOB command).
- Display status information about sessions and jobs in the system (:SHOWJOB).
- Create, save, and delete files (:BUILD, :SAVE, and :PURGE, respectively); specify and list their characteristics (:FILE and :LISTF); dump them offline and subsequently restore them to the system (:STORE and :RESTORE); and specify security provisions for them (:ALTSEC, :RELEASE, and :SECURE).
- Compile programs (:FORTRAN, :COBOL, :RPG, :SPL), prepare those programs (:PREP), and execute them (:RUN).
- Determine the status of devices (:SHOWDEV and :DSTAT).
- Determine the status of devicefiles, which are disc files originating on or destined for non-sharable devices (:SHOWIN and :SHOWOUT).
- Communicate with other users (:TELL) and with the Console Operator (:TELLOP).
- Define your own commands (:SETCATALOG).
- Access private disc volumes (:MOUNT and :DISMOUNT).
- Obtain assistance in using the Command Interpreter (:HELP).

Many other commands and functions are available as well; complete specifications for all of the commands appear in Section II.

HOW TO USE THIS MANUAL

The reference specifications, appearing in Section II, primarily cover the rules for entering each command. Specifically, they show the command syntax and format; define the parameters and discuss constraints upon them and default values for them; provide an overview of the operation requested by the command; and present examples illustrating proper command entries. Parameter definitions common to more than one command are repeated for each applicable command, to reduce the amount of cross referencing you must do when looking up a definition.

ENTERING COMMANDS

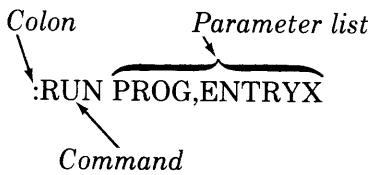
You can enter commands through any standard input device, typically a terminal (for sessions) or a card reader (for jobs). Each command is accepted by the MPE Command Interpreter, which passes it to the appropriate system procedure for execution. Following this execution, control returns to the Command Interpreter, which is now ready for another command.

COMMAND ELEMENTS

Each MPE command consists of:

- A *colon* (required in all cases as an MPE command identifier).
- A *command name* (required in all cases).
- A *parameter list* (used in most cases).

A typical command including all three elements appears as follows:



The *colon* identifies a statement as an MPE command. In an interactive session, MPE prints the colon on the terminal whenever it is ready to accept a command; you respond by entering the remainder of the command after the colon. In a batch job, however, you must enter the colon, placing it in column 1 of the source card (or card image) on which the command is to appear.

The *command name*, which you enter immediately after the colon, requests a specific operation. MPE prohibits embedded blanks within the name, and rejects the command if they appear. MPE interprets the next non-alphanumeric character encountered as the end of the command name; typically this character is a blank. Blanks also may appear between the colon and the command name.

The *parameter list* contains one or more parameters that specify options for the command. It is required in some commands, but is optional or prohibited in others. Parameter lists can include *positional* parameters and/or *keyword* parameter groups (defined below), separated from each other by *delimiters* such as commas, semicolons, equal signs, or other punctuation marks.

Normally, you must separate the parameter list from the command name by one or more blanks. However, when you omit the first optional parameter in a positional list, you can begin the list, starting with a comma or other delimiter that normally follows the first parameter, immediately after the command name. The comma in place of blanks serves as a delimiter, as noted under Positional Parameters, below. Within the parameter list, any delimiter can be surrounded by any number of blanks, permitting a free and flexible command format.

MPE permits both decimal and octal numbers as command parameters. You distinguish between the two by preceding the octal numbers with a percent sign (%).

The end of each command is indicated by the end of the record on which it appears — for example, a *carriage return* for terminal input or the *end of the card* containing the command for card input. But if the last non-blank character of the record is a *continuation character*, the command is continued onto the next record.

NOTE

If you are running programs in batch job mode, bear in mind that MPE scans all 80 columns on each card image, and thus no characters are ignored.

POSITIONAL PARAMETERS

With *positional* parameters, the meaning of a parameter depends upon its position in the parameter list. For example, in the :FORTRAN command, issued to compile a FORTRAN program, the parameter list specifies the input file containing the source program, the output file to which the object program is written, and the output file to which the source listing is transmitted, *always in that order.* In the following :FORTRAN command, for instance, the variable names INP, OUT, and *LST indicate the source, object, and list files, respectively.

```
:FORTRAN INP,OUT,*LST
```

In the above example, the asterisk (*) in *LST is not a delimiter but a special character denoting a back reference to a previously defined file. (See BACK REFERENCE, this section).

Positional parameters are separated (*delimited*) from one another by commas. When you omit an optional positional parameter from within a list, you must still include the delimiter that would normally follow that parameter. Thus, on a listing, two adjacent delimiters indicate a missing optional parameter. When you omit a positional parameter that would otherwise immediately follow a command name, indicate this by entering its delimiter as the first character in the parameter list. When you omit positional parameters from the end of the list, however, you need not include delimiters to signify this — the terminating return or end-of-card is sufficient. The following examples demonstrate how to properly omit parameters from a command:

```
:FORTRAN,USLFL,*LISTFL,MFL,NFL          First parameter omitted.
:FORTRAN *SOURCEFL,,*LISTFL,MFL,NFL     Second parameter omitted.
:FORTRAN *SOURCEFL,USLFL,*LISTFL        Last two parameters omitted.
:FORTRAN                                All parameters omitted.
```

KEYWORD PARAMETERS

When a parameter list is so long that use of positional parameters becomes difficult, MPE provides *keyword* parameter groups. The meaning of such a group is independent of its position in the list — thus, you can enter keyword groups in any order with respect to each other. A keyword group consists of a keyword that denotes the group's meaning, sometimes followed by an equal sign and one or more sub-parameters. Each keyword group is preceded by a semicolon. When more than one sub-parameter appears in a group, they are usually separated from each other by commas. All delimiters can be optionally preceded or followed by blanks.The following example shows a :PREP command containing both positional and keyword parameters. INPT and OUTP are the variable names of the positional parameters. DL and CAP are keywords that designate the keyword parameter groups. PH, DS, and MR are sub-parameters of the keyword group designated by CAP.

```
:PREP INPT,OUTP;DL=500;CAP=PH,DS,MR
```

When both keyword groups and positional parameters form a list, the positional parameters always occur before the keyword groups. When you omit trailing parameters from the positional group in this list, you need not include their delimiters since the occurrence of the first keyword indicates the omission. When you omit optional sub-parameters from a keyword group, simply follow the same rules that apply to positional parameters.

CONTINUATION CHARACTERS

When the length of a command exceeds one record (for instance, one entry-line or source card), you may enter an ampersand (&) as the last non-blank character of the record and continue the command on the next record. This next record must begin with a colon (supplied automatically by MPE in interactive processing, but entered by yourself in batch processing). Optionally, you can embed blanks between the colon that begins the continuation record and the first non-blank character in the continuation record. In the example below, the command contains a continuation character at the end of the first line and an embedded blank at the beginning of the second.

```
:RUN PROGB;NOPRIV;LMAP;STACK=500;PARM=5; &
: DL=600;LIB=G
```

You can continue commands up to 268 characters; prompting colons and continuation ampersands are not counted as part of this total.

When continuing a command onto another line, you must not divide a command name, keyword, positional parameter, or keyword sub-parameter — MPE does not permit any such element to span more than one line.

MPE does not begin interpretation of a command until the last record of the command is read.

COMMAND ERRORS

If you make an error while entering a command in an interactive session, MPE suppresses execution of that command and attempts to determine the cause of the error. If the cause of the error is of a nature that can be pointed out to you easily, MPE prints a caret under the incorrect part of the command, along with an appropriate message. If the command entry is such that MPE cannot print the caret to signify a specific error point, an appropriate error is displayed. In either case, control returns to your terminal.

If you enter an erroneous command in a batch file, and do not precede this command with a :CONTINUE command, MPE suppresses execution of the command. An error message is printed on your standard list device, all subsequent commands in this job are ignored and the job is aborted.

If a command is continued over several lines, and an error is detected, the offending line will be echoed, preceded by the line number on which the error has occurred.

```
:FILE ABC&
:=TAP%&
:;NEW
(1)=TAP%
        ^

UNEXPECTED CHARACTER IN FILE NAME; EXPECTED "." OR "/". IS THE DELIMITER
BETWEEN PARAMETERS CORRECT? (CIERR 582)
```

SEQUENCE NUMBER IN MPE COMMANDS

MPE commands in spooled jobs may have sequence numbers. The rule is that if the first card image (i.e., the one containing the :JOB command) has a positive integer value in the last 8 bytes, MPE assumes that all MPE commands in the job have sequence numbers in the last eight bytes.

The sequence field is checked for each command in such a job. If the value is less than the preceding command's sequence number, a warning is issued. The sequence field may also be all blanks, in which case the sequence field is ignored. If the sequence field is non-numeric and non-blank, a warning is issued. For all cases except the last one, the sequence field is stripped from the command before the command itself is analyzed.

This definition of sequence numbers has certain implications. Ampersands, which indicate continuation lines, must be placed before, not after, sequence fields. Secondly, STREAM files may be numbered or unnumbered. If a file is numbered, MPE will strip the last 8 bytes of each non-MPE record. The sequence of non-MPE records will not be checked, and a program reading these records (data records) from $STDIN will never see the sequence numbers. Finally, input card decks may be numbered or unnumbered. However, all 80 columns of a non-MPE card will always be read, even if the deck is numbered. By convention, when a card deck is sequenced, only the MPE cards should have sequence numbers punched in columns 73/80. If you choose to punch sequence numbers into data cards, your program must be prepared to handle them.

EXECUTING COMMAND LOG ON

Another way to issue a command to MPE is through the command log-on capability. It is used when you want to log on to execute just one MPE command, and you would like to simplify the log-on and log-off procedures. A command log on is performed in an interactive session by entering any MPE command and its parameter list, enclosed in parentheses, followed by the parameter list (user name, account and group names) normally associated with the :HELLO command. The following example shows how someone with the username MYNAME, and acctname MYACCT, can log on to the system, execute a program prepared in a program file called MYPROG, and log off, all in a single MPE command.

```
:(RUN MYPROG) MYNAME.MYACCT
```

EXECUTING COMMANDS PROGRAMMATICALLY

In addition to entering commands directly through your standard input device, MPE allows you to execute many of them from within the programs that you write. You do this by including, within those programs, calls to the COMMAND intrinsic. This intrinsic invokes the Command Interpreter and passes to it command images that MPE will interpret and execute as the corresponding system commands. Complete information on the COMMAND intrinsic appears in the *MPE Intrinsics Reference Manual*.

INTERRUPTING COMMAND EXECUTION

When executing an MPE command, it is sometimes necessary to interrupt and perform another command such as listing your files (:LISTF), creating a new disc file (:BUILD), or determining information about other jobs and sessions (:SHOWJOB). Command interruption is accomplished by pressing the BREAK key on your terminal (sometimes labeled INTERRUPT or ATTENTION). Pressing the BREAK key can result in your command being either suspended or aborted and in some cases, the BREAK will be ignored. The exact effect that using the BREAK key will have on your executing command depends on whether the command is one that executes a subsystem or user program (program command), or whether the command does not execute such a program (non-program command). Table 1-1 indicates which commands are breakable and which ones are not.

INTERRUPTING NON-PROGRAM COMMANDS

Breaking from a breakable non-program command will abort the command and the Command Interpreter will issue a colon prompt for a new MPE command.

Several commands, such as the list commands (:LISTF, :LISTACCT, :LISTGROUP, :LISTUSER, :LISTVS), :STORE, :RESTORE, and :REPORT may require the operator to mount a tape for the output file. If you BREAK from one of these commands while waiting for operator intervention, only three commands will function after the colon prompt is displayed. These are :RECALL, :RESUME, and :REPLY. If you use any other command a warning message is issued and the command is ignored.

If you log on using the :HELLO command and press the BREAK key during output of the log on message, MPE terminates the output, keeps you logged on, and prompts you for your next command.

If you log on using the :( ) command log on and press the BREAK key during output of the log on message, MPE terminates the output and begins executing the command enclosed within the parentheses. If you press the BREAK key after the log-on message is output, MPE breaks the command within the parentheses. If this command is a non-program command, MPE stops its execution and logs you off immediately. If the command is a program command, MPE suspends its execution and prompts you for a new command.

INTERRUPTING PROGRAM COMMANDS

Program commands invoke MPE subsystems or run user programs. All program commands are breakable.

When you press the BREAK key to interrupt one of these program commands, the execution of that command is suspended and the Command Interpreter issues a prompt for a new MPE command. If you then enter a non-program command (other than :HELLO, :BYE, :JOB, or :DATA), the Command Interpreter performs the requested operation and then allows you to re-activate the sus-

pended program by entering :RESUME. But, if you enter another program command (such as :FORTRAN, :EDITOR, or :RUN) or one of the non-program commands :HELLO, :BYE, :JOB, or :DATA, the Command Interpreter prints the following message on your terminal:

ABORT? (YES/NO)

If you respond YES to the ABORT? message, the Command Interpreter aborts the current program and executes the command you entered, in the usual way.

If you had logged on using the :( ) command log on with a program command inside the parentheses, then responding YES to the ABORT? message causes MPE to abort the command and log you off immediately.

If you respond NO to the ABORT? message, the Command Interpreter prints the message NOT ALLOWED IN BREAK and prompts you for another command. If you now enter :RESUME, the suspended program resumes at the point where it was interrupted.

NOTE

User programs are initiated with the BREAK facility enabled. But since you may not wish to let those who run your programs interrupt them under certain circumstances, you may programmatically disable the BREAK facility by using the FCONTROL intrinsic. For example, application programs can restrict user access to the Command Interpreter by disabling the BREAK function.

ABORTING A PROGRAM

When a program contains logical errors, such as an infinite loop, you can abort it by

1. Pressing break. This suspends your program and MPE prompts you for a new command.
2. When you receive a colon prompt, type ABORT. This terminates the program, but in NO WAY disrupts the session.

MPE confirms this termination by displaying

PROGRAM ABORTED PER USER REQUEST (CIERR 989)

then prompts you for a new command.

You can also use the BREAK/:ABORT sequence to abort certain MPE subsystem and command operations. Those operations are indicated in the associated command specifications, under the BREAKABLE? entry. (Some of them abort immediately upon entering BREAK, without requiring the :ABORT command.)

Table 1-1. Non-Program Commands

BREAKABLE	NON-BREAKABLE	
:( ) COMMAND LOG ON	:ABORT	:MOUNT
:BYE	:ALTLOG	:PTAPE
:HELLO	:ALTSEC	:PURGE
:HELP	:ASSOCIATE	:RECALL
:LISTF	:BUILD	:RELEASE
:LISTVS	:COMMENT	:RELLOG
:REDO	:CONSOLE	:REMOTE
:REPORT	:CONTINUE	:RENAME
:RESTORE	:DATA	:RESET
:SHOWCATALOG	:DEBUG	:RESETDUMP
:SHOWDEV	:DISASSOCIATE	:RESUME
:SHOWIN	:DISMOUNT	:SAVE
:SHOWJCW	:DSLINE	:SECURE
:SHOWJOB	:DSTAT	:SETCATALOG
:SHOWME	:ELSE	:SETDUMP
:SHOWOUT	:ENDIF	:SETJCW
:STORE	:EOD	:SETMSG
:STREAM	:EOJ	:SHOWALLOW
	:FILE	:SHOWLOGSTATUS
	:FREERIN	:SHOWTIME
	:GETLOG	:SPEED
	:GETRIN	:TELL
	:IF	:TELLOP
	:JOB	:VSUSER
	:LISTLOG	

Table 1-2. Program Commands (All Breakable)

:APL	:IML
:BASIC	:MRJE
:BASICGO	:PREP
:BASICOMP	:PREPRUN
:BASICPREP	:RJE
:COBOL	:RPG
:COBOLGO	:RPGGO
:COBOLPREP	:RPGPREP
:DSCOPY	:RUN
:EDITOR	:SEGMENTER
:FCOPY	:SPL
:FORTGO	:SPLGO
:FORTPREP	:SPLPREP
:FORTRAN	

BACK REFERENCING

Once you establish a set of specification in a :FILE command, you can apply those specifications to other file references in your job or session simply by using the file's formal designator, preceded by an asterisk (*), in those references. For instance, suppose you use a :FILE command to establish the specifications shown below for the file FILEA, used by program PROGA. You then run PROGA. Now, you wish to apply those same specifications to the file FILEB, used by PROGB, and run that program. Rather than re-specify all these parameters in a second :FILE command, you can simply use :FILE to equate the FILEA specifications to cover FILEB, as follows:

```
:FILE FILEA;DEV=TAPE;REC=-80,4,V;BUF=4   Establishes specifications.
:RUN PROGA                               Runs program A.
:FILE FILEB=*FILEA                       Back references specifications for
                                         FILEA.
:RUN PROGB                               Runs program B.
```

This technique is called *back referencing files*, and the files to which it applies are sometimes known as user *pre-defined files*. Whenever you reference a pre-defined file in an MPE command, you must enter the asterisk before the formal designator if you want the pre-definition to apply. As an example, you would do this when using the Editor in session mode when you wish to transmit the Editor's off-line listings to an off-line device (such as a magnetic tape or line printer). You specify the destination file name as a parameter in the :EDITOR command that invokes the Editor, but you define the file in a previous :FILE command. You then back reference the definition by using the asterisk in the :EDITOR command, as shown below:

```
:FILE L;DEV=LP      Specifies file named L as a line printer.
:EDITOR*L           Runs Editor, directs off-line listing to L, now established as a
                    line printer.
```

When back referencing in a command, you cannot reference any predefined formal file designator of the subsystem you are using. For example, FORTRAN predefines FTNTEXT as its formal file designator, therefore you cannot back reference it in your command: FORTRAN* FTNTEXT.

REFERENCE NOTES FOR COMMAND DEFINITIONS

ELEMENTS OF COMMAND FORMAT	
• Leading Colon:	Prompt/command identifier character in interactive sessions. Command identifier character in batch jobs.
• Command Name:	Shown in CAPITAL LETTERS IN REGULAR (ROMAN) TYPE. Contains no blanks, is delimited by a non-alphanumeric character (usually a blank).
• Parameters:	Shown in CAPITAL LETTERS IN REGULAR TYPE when they are literal information that you enter exactly as shown. Shown in *lower-case italics* when they are variable parameters to be replaced by information that you must supply.
• Positional Parameters:	Have significance implied by positional order after command name; use adjacent commas (or semicolons where required) to indicate omitted parameter(s), as follows: COMMANDNAME p1,,p3 (from middle) COMMANDNAME ,p2,p3 (from beginning) COMMANDNAME p1 (from end)
• Keyword Parameters:	Separated by semicolons and can appear in any order.
• Mixed Parameters:	Positional parameters are given first; first keyword indicates end of positional list.
• Optional Parameters:	[A] means A may be included A [ ] means A or B may be B included A { } means A or B must be B included [A] means A and/or B may be [B] included in any order A [ ] underline means A is default. B
USER/SYSTEM DIALOG	
• User input is underlined where necessary for clarity. For example: NEW NAME? ALPHA1	

COMMAND SPECIFICATIONS

SECTION II

The reference specifications for all MPE commands available to standard users appear in this section. For easy reference, commands are presented alphabetically by name; should you prefer to reference them by function, remove these pages and re-arrange them according to the *Functional List of Commands* shown in table 2-1. For each command, the reference specifications show the following information:

- Parameter definitions (including meaning, constraints, and defaults).
- When legal (in jobs or sessions, during break, or programmatically).
- Whether interruptable (with BREAK key).
- Operation or functional description.
- Examples.
- Where described in other manuals.

In the reference specifications, the indication *Available in Break* means that the command can be executed if the current operation is suspended by pressing the BREAK key on the terminal (or calling the CAUSEBREAK intrinsic). The notation *Available Programmatically means that the command is available through the COMMAND intrinsic. CAUSEBREAK and COMMAND are explained in the MPE Intrinsics Reference Manual.*

NOTE

IF YOU HAVE NO PRIOR EXPERIENCE WITH MPE, YOU SHOULD READ THE TEXT DISCUSSION (SECTIONS III THROUGH X) BEFORE ATTEMPTING TO USE THE REFERENCE SPECIFICATIONS IN THIS SECTION.

Table 2-1. Functional List of Commands

FUNCTION	COMMAND
Running Sessions (Section III)	:( ) COMMAND LOG ON :HELLO :BYE :ABORT :RESUME :HELP :EOD :EOF:
Running Jobs (Section IV)	:JOB :EOJ :STREAM :COMMENT :CONTINUE :SETJCW :SHOWJCW :IF :ELSE :ENDIF
Determining Job/Session Status and Resource Usage (Section V)	:SHOWJOB :REPORT :SHOWME
Managing Files (Section VI)	:FILE :RESET :BUILD :LISTF :MOUNT :DISMOUNT :LISTVS :VSUSER :RENAME :PURGE :SAVE :STORE :RESTORE :ALTSEC :RELEASE :SECURE :DATA
Running Subsystems and User Programs (Section VII). Commands marked with asterisk are discussed in pertinent subsystem manuals.	:APL* :BASIC :BASICOMP :BASICPREP :BASICGO :COBOL :COBOLPREP :COBOLGO

Table 2-1. Functional List of Commands (continued)

FUNCTION	COMMAND
Running Subsystems and User Programs Commands marked with asterisk are discussed in pertinent subsystem manuals.	:FORTRAN :FORTPREP :FORTGO :RPG :RPGPREP :RPGGO :SPL :SPLPREP :SPLGO :PREP :PREPRUN :RUN :DEBUG* :DSCOPY* :DSLINE* :EDITOR* :FCOPY* :IML* :RJE* :MRJE* :REMOTE* :REMOTE HELLO* :SEGMENTER* :SETDUMP* :RESETDUMP*
Determining Device and Devicefile Status	:ASSOCIATE :CONSOLE :DISASSOCIATE :DSTAT :RECALL :SHOWALLOW :SHOWDEV :SHOWIN :SHOWOUT
Requesting Utility Operations	:TELL :TELLOP :SETMSG :SPEED :PTAPE :GETRIN :FREERIN :SHOWTIME :SETCATALOG :SHOWCATALOG :SHOWJCW :REDO
Resource Management	:ALTLOG :GETLOG :LISTLOG :RELLOG :SHOWLOGSTATUS

:( ) COMMAND LOG ON

Begins a session, executes the enclosed MPE command, and ends the session upon completion of the command.

SYNTAX

```
:([:]commandname) [sessionname,]username[/userpass]
   .acctname[/acctpass][,groupname[/grouppass]]

   [;TERM=termtype]
   [;TIME=cpusecs]

            BS
            CS
   [;PRI= {    }]
            DS
            ES

    ;INPRI=inputpriority
   [                      ]
    ;HIPRI
```

PARAMETERS

commandname Any MPE command, including its parameter list. The prompting colon may be omitted. (REQUIRED PARAMETER)

sessionname Arbitrary name used in conjunction with username and acctname to form a fully-qualified session identity. Contains from 1 to 8 alphanumeric characters, beginning with a letter. Default is no session name.

username A user name, established by the Account Manager, that allows you to log on under this account. This name is unique within the account. Contains from 1 to 8 alphanumeric characters, beginning with a letter. (REQUIRED PARAMETER)

userpass Your user password, optionally assigned by the Account Manager. Contains from 1 to 8 alphanumeric characters, beginning with a letter.

acctname Name of your account, as established by the System Manager. Contains from 1 to 8 alphanumeric characters, beginning with a letter.

NOTE

Must be preceded by a period as a delimiter. (REQUIRED PARAMETER)

acctpass Account password, optionally assigned by the System Manager. Contains from 1 to 8 alphanumeric characters, beginning with a letter.

groupname Name of file group to be used for local file domain and central processor time charges, as established by the Account Manager. Contains from 1 to 8 alphanumeric characters, beginning with a letter. Default is your home group if one is assigned. (REQUIRED IF HOME GROUP NOT ASSIGNED)

grouppass Group password, optionally assigned by the Account Manager. Contains from 1 to 8 alphanumeric characters, beginning with a letter. Not needed when when you log on under home group, otherwise, required if assigned.

termtype Type of terminal used for input. MPE uses this parameter to determine device-dependent characteristics such as delay factors for carriage returns. Must be a number from 0 to 15. See Appendix A for a list of terminals.

Default: For hardwired terminals, determined by System Supervisor during system configuration. No default for terminals that are not hardwired. Required parameter to insure correct listings if your terminal is not hardwired or if your terminal is not the default *termtype.*

cpusecs Maximum central processor time that your session can use, entered in seconds. When this limit is reached, session is aborted. Must be a value from 1 to 32767. To specify no limit, specify question mark or omit this parameter. Default is no limit.

BS,CS,DS,ES The execution priority class that the Command Interpreter uses for your session, and also the default priority for all programs executed within the session. BS is highest priority, ES is lowest. If you specify a priority that exceeds the highest permitted for your account or user name by the system, MPE assigns the highest priority possible below BS. Default is CS.

NOTE

DS and ES are intended primarily for batch jobs; their use for sessions is generally discouraged.

inputpriority Relative input priority used in checking against access restrictions imposed by *jobfence,* if one exists. Takes effect at log-on time. Must be a value from 1 (lowest priority) to 13 (highest priority). If you supply a value less than or equal to current *jobfence* set by Console Operator, session is denied access. Default is 8.

HIPRI Request for maximum session selection input priority, causing session to be scheduled regardless of current jobfence or execution limit for sessions. If you do not have OP or SM capability, the System will try to log you on with INPRI=13.

NOTE

You can specify this parameter only if you have System Manager or System Supervisor capability

USE

Available	In Session?	NO (initiates a session only)
	In Job?	NO
	In Break?	NO
	Programmatically?	NO
Breakable?		How BREAK functions with :( ) Command Log On depends on when BREAK is pressed. Pressing BREAK during the output of the log-on message suppresses the remainder of the message and causes immediate execution of the command within the parentheses. Pressing BREAK after the log-on message is completed, and during the execution of the command within the parentheses, causes break in the command. Finally, pressing BREAK during the output of the log-off message terminates the message.

OPERATION

This command initiates a session and immediately executes the command contained within the parentheses. Once in contact with the command or subsystem specified inside the parentheses, all input, output, break capabilities, and prompts operate as described in the command or subsystem instructions. After the execution of the command or subsytem is completed, either normally or through an abort, a log off is performed automatically.

NOTE

You must enter the :( ) command log on at a terminal — no other device can be used for this command.

:( ) command log on is not recognized within sessions, as for example, the :HELLO command is.

Commands which would normally cause an end of file, such as :HELLO, :BYE, :JOB, :DATA, :EOF, and :EOJ are not recognized and if used the session is terminated immediately.

Only one MPE command is allowed within the parentheses. Also, the leading and trailing blanks are suppressed within the parentheses.

The log-on and log-off messages can be terminated using BREAK, and the command specified within the parentheses is breakable or not in accordance with its usual break or no break capability. When the command inside the parentheses is aborted, a log off is performed immediately.

If you omit any passwords required in the :( ) command log on, MPE prompts you for them individually.

When MPE initiates the session, it displays the HP 3000 Model Number, and the date and time as follows:

```
HP3000 / MPE IV C.00.00 THU, OCT 9, 1980, 12:15 PM
```

After the log-on message is printed at your terminal, MPE executes the command specified within the parentheses. The information MPE generates as a result of this command is transmitted to your terminal immediately; just as if you had called that command or subsystem during a session. When the command or subsystem is exited, MPE terminates the session at once, and displays the cpu time used (in seconds), the connect time (in minutes), and the date and time, as follows:

```
CPU=1. CONNECT=1. THU, OCT 9, 1980, 12:16 PM
```

EXAMPLE

To log on under username MAC, acctname TECHPUB, and groupname XGROUP, and execute the LISTF command, enter:

```
:(LISTF) MAC.TECHPUB,XGROUP
HP3000 / MPE IV. C.00.00 THU, OCT 9, 1980, 12:15 PM
FILENAME
FIRST       GP005       TRIAL       XPROG <---------- Displayed by MPE
CPU=1. CONNECT=1. THU, OCT 9, 1980, 12:16 PM
```

:ABORT

Aborts current program or operation.

SYNTAX

```
:ABORT
```

PARAMETERS

None

USE

Available	In Session?	Yes (in BREAK only)
	In Job?	NO
	In Break?	YES
	Programmatically?	NO
Breakable?		NO

OPERATION

After you suspend a program or MPE command operation by pressing the BREAK key, the :ABORT command immediately terminates that program or operation. The :ABORT command is available only from a session and only during a break, but in no way disrupts the session. Note that some operations abort immediately upon entering BREAK without requiring the :ABORT command. An :ABORT command results in the Job Control Word (JCW) being set to the SYSTEM 0 state. (For a discussion of Job Control Word, see the Intrinsics Reference Manual.

EXAMPLE

To abort the current operation, enter:

```
:ABORT
```

:ALTLOG

Alters the attributes of an existing logging identifier.

SYNTAX

:ALTLOG *logid*[;LOG=*logfile* {,DISC / ,TAPE}][;PASS=*password*]

PARAMETERS

logid — The logging identifier whose attributes are to be changed. (REQUIRED PARAMETER)

logfile — The new destination file name for the logfile. Must be specified as residing on tape or disc.

password — The new password for the logging identifier.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

Changes the attributes of an existing logging identifier to those specified in the parameter list. Parameters not included in the :ALTLOG command retain their current values. This command can be used only by the creator of the logging identifier.

EXAMPLE

To change the destination logfile of the logging identifier DATALOG to FILEX, and specify that FILEX reside on tape, enter:

```
:ALTLOG DATALOG;LOG=FILEX,TAPE
```

Since the ;PASS parameter was not included, DATALOG retains the password previously specified.

ADDITIONAL DISCUSSION

Console Operator's Guide

:ALTSEC

Permanently changes file's security provisions.

SYNTAX

:ALTSEC *filereference*[;([*fileaccess*[;[*fileaccess*[;. . .]]])]

PARAMETERS

filereference Actual designator of the file whose security provisions are to be altered, written in the format:

filename[/*lockword*][.*groupname*[.*accountname*]]

The *lockword,* if any, must be specified. (REQUIRED PARAMETER)

fileaccess File security specifications, entered as follows:

$$\left\{\begin{matrix}\text{R}\\\text{L}\\\text{A}\\\text{W}\\\text{X}\end{matrix}\right\}:\left\{\begin{matrix}\text{ANY}\\\text{AC}\\\text{GU}\\\text{AL}\\\text{GL}\\\text{CR}\end{matrix}\right\}$$

where R, L, A, W, X specify modes of access by types of users (ANY, AC, GU, AL, GL, CR) as follows:

R = Read
L = Lock (allows opening with dynamic locking option)
A = Append (implicitly specifies L also)
W = Write (implicitly specifies A and L also)
X = Execute

Two or more modes may be specified if they are separated by commas.

The user types are specified as follows:

ANY = Any user
AC = Member of this account only
GU = Member of this group only
AL = Account librarian user only
GL = Group librarian user only
CR = Creating user only

Two or more user types may be specified if they are separated by commas.

Default is R, A, W, L, X: ANY

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

The :ALTSEC command enables you to change the security provisions for any disc file that you have created by permanently deleting all previous provisions specified for this file at the file level and replacing them with those defined in the :ALTSEC parameters. Account level and group level security is not affected, nor is the lockword (if one exists). The command can only be used by the creator of the file.

Note that you can only use this command for a permanent disc file that you have created, and that this command will fail if the group's home volume set is not mounted. When the normal (default) MPE security provisions are in effect, the file must be in your log-on account and must belong to your log-on or home group.

Even though you may be barred by *fileaccess* restrictions from accessing a file that you have created, you still can issue an :ALTSEC command for that file. Thus, you can change the security provisions to allow yourself access to it.

EXAMPLE

You have created a file named FDATA and you wish to change its security provisions to allow write access to yourself only. Enter:

```
:ALTSEC FDATA;(W:CR)
```

To change the security provisions for a program file (FPROG) so that any group user can execute the program but only account and group librarian users can read or write on the file, enter:

```
:ALTSEC FPROG;(X:GU;R,W:AL,GL)
```

ADDITIONAL DISCUSSION

Intrinsics Reference Manual
Using Files
System Manager/System Supervisor Reference Manual

:APL

Accesses APL subsystem.

SYNTAX

```
:APL
```

PARAMETERS

None

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Under control of APL subsystem. See *APL\3000 Reference Manual.*)

OPERATION

Accesses the APL subsystem, which is used to create and execute APL programs.

NOTE

The recommended way of accessing the APL subsystem from terminals equipped with the APL character set is through the command log on. The unusual APL character set and its resultant coding causes unreadable log-on and log-off messages when the :HELLO and :BYE commands are used at APL terminals.

EXAMPLE

```
:(APL) MAC.TECHPUBS
```

ADDITIONAL DISCUSSION

APL 3000 Reference Manual.

:ASSOCIATE

Gives a user operator control of a device.

SYNTAX

:ASSOCIATE *devclass*

PARAMETERS

devclass The name of a logical device class configured during SYSDUMP.

USE

Available	In Session?	YES
	In Job?	NO
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

This command makes a user the controller (operator) of a device class. As such, the user may execute any valid operator command for a device in the device class and will receive the status messages for the devices in that device class.

The operator commands which are made available to users through the :ASSOCIATE command are:

:ABORTIO
:ACCEPT
:ALTSPOOLFILE
:DELETESPOOLFILE
:DOWN
:DOWNLOAD
:DSCONTROL
:FOREIGN
:GIVE
:HEADON
:HEADOFF
:MPLINE
:REFUSE
:REPLY
:RESUMESPOOL
:SHOWCOM
:STARTSPOOL
:STOPSPOOL
:SUSPENDSPOOL
:TAKE
:UP

The system manager must specify in a system file, ASOCIATE.PUB.SYS, which users may associate what devices. This file is created and maintained by the system manager through the system utility, ASOCTABL. A user may :ASSOCIATE a device only after the system manager has made the appropriate entries in ASOCIATE.PUB.SYS. Both the console operator and the user may :DISASSOCIATE a user from a device. In addition, a user implicitly disassociates a device when logging off.

EXAMPLE

```
:ASSOCIATE PRINTER
```

ADDITIONAL DISCUSSION

Console Operator's Guide

:BASIC

Interprets a BASIC program.

SYNTAX

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

PARAMETERS

commandfile Actual designator of source file (device) from which BASIC commands and statements are input. Can be any ASCII input file. Formal designator is BASCOM. Default is $STDINX.

inputfile Actual designator of file containing data input for BASIC program. Can be any ASCII input file. Formal designator is BASIN. Default is $STDINX.

listfile Actual designator of destination file for program listing and output. Can be any ASCII output file. Formal designator is BASLIST. Default is $STDLIST.

NOTE

The formal file designators used in this command (BASCOM, BASIN, BASLIST) cannot be back referenced as actual file designators in the command parameter list. For further information see the OPERATION section of the :FILE command.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Generally used for on-line programming in BASIC, but it can also be used to interpret BASIC programs submitted in batch mode. In batch mode, the BASIC >EOD command is required after any data following the BASIC >RUN command, or after the >RUN command itself if there is no data.

EXAMPLE

To enter commands and data from your standard input device, with program listing and output transmitted to the standard output device, enter:

```
:BASIC
```

To submit BASIC interpreter commands and statements from the command file MYCOMDS, and data from the input file MYDATA, with program listing and output written to the list file MYLIST, enter:

```
:BASIC MYCOMDS,MYDATA,MYLIST
```

(All three files are disc files.)

Note that input files created on the Editor must be kept with the UNN option of the Editor KEEP command.

ADDITIONAL DISCUSSION

Basic Interpreter Manual

:BASICGO

Compiles, prepares, and executes a BASIC program.

SYNTAX

:BASICGO [*commandfile*][,*listfile*]

PARAMETERS

commandfile Actual designator of input file from which BASIC compiler commands are read. Can be any ASCII input file. Formal designator is BSCTEXT. Default is $STDINX.

listfile Actual designator of file on which program listing is written. Can be any ASCII output file. Formal designator is BSCLIST. Default is $STDLIST.

NOTE

The formal file designators used in this command (BSCTEXT,BSCLIST) cannot be back referenced as actual file designators in the command parameter list. See the OPERATION section the :FILE command for further information.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Compiles, prepares, and executes a program from a SAVE FAST file (created by BASIC interpreter). This enables the program to run faster than it would if executed by the interpreter. After the program is written, it can be saved with the BASIC interpreter command SAVE *filename,* FAST. The program then can be compiled, prepared, and executed with the :BASICGO command.

EXAMPLE

To compile, prepare, and execute the BASIC program MYPROG, with the listing directed to the disc file LISTFL, enter:

```
:BASICGO ,LISTFL                 Calls the BASIC compiler
$CONTROL USLINIT,SOURCE          Initializes USL and requests program listing
$COMPILE MYPROG                  Compiles program MYPROG
$EXIT                            Exits from compiler
```

ADDITIONAL DISCUSSION

Basic Compiler Reference Manual

:BASICOMP

Compiles a BASIC program.

SYNTAX

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

PARAMETERS

commandfile Actual designator of input file from which compiler commands are read. Can be any ASCII input file. Formal designator is BSCTEXT. Default is $STDINX.

uslfile Actual designator of user subprogram library (USL) file on which compiled object program is written. Can be any binary output file with filecode of USL (or 1024). Formal designator is BSCUSL. This parameter, if entered, must specify a file created previously in one of four ways:

1. By saving a USL file (with the :SAVE command) created by a previous compilation where the default value was used for the uslfile parameter.
2. By building the USL with the Segmenter command -BUILDUSL (see the *MPE Segmenter Reference Manual*).
3. By creating a new USL file with the MPE :BUILD command and a *filecode* parameter of USL or 1024.
4. By specifying a non-existent *uslfile* parameter, thereby creating a permanent file of the correct size and type.

Default is that $NEWPASS is assigned.

listfile Actual designator of file on which program listing is written. Can be any ASCII output file. Formal designator is BSCLIST. Default is $STDLIST.

NOTE

The formal file designators used in this command (BSCTEXT, BSCUSL, BSCLIST) cannot be back referenced as actual file designators in the command parameter list. For further information, see the OPERATION section of the :FILE command.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Compiles a program from a SAVE FAST file (created by the BASIC interpreter) onto disc in USL format. This enables the program to run faster than it would if left in the form generated by the interpreter. After the program is written, it is stored in a SAVEFAST file with the BASIC interpreter command SAVE *filename*,FAST. The program can then be compiled with the compiler command COMPILE, prepared with the MPE :PREP command, and executed with the MPE :RUN command.

EXAMPLE

To compile the BASIC program MYPROG onto the USL named OBJECT, enter:

:BUILD OBJECT;CODE=USL	*Builds USL file*
:BASICOMP ,OBJECT	*Calls BASIC compiler, specifies USL named OBJECT*
$COMPILE MYPROG	*Compiles SAVE FAST program named MYPROG*
$EXIT	*Exits from compiler*

If you do not choose to build a USL file, the :BASICOMP command compiles your file to $OLDPASS (USL default):

```
.
.
.
```

:BASICOMP	*Runs BASIC compiler, accepting commands from $STDINX, and requesting $NEWPASS/$OLDPASS for RBM output and $STDLIST for listing output.*
$COMPIL MYRUN	*Compiles from SAVE FAST file named MYRUN onto USL named $OLDPASS.*
$EXIT	*Exits from BASIC compiler.*

If you now want to run your program, use the :PREPRUN command:

:PREPRUN $OLDPASS	*Prepares and runs program.*

```
.
.
```

ADDITIONAL DISCUSSION

Basic Compiler Reference Manual

:BASICPREP

Compiles and prepares a BASIC program.

SYNTAX

:BASICPREP [*commandfile*][,[*progfile*][,*listfile*]]

PARAMETERS

commandfile Actual designator of input file from which compiler commands are read. Can be any ASCII file. Formal designator is BSCTEXT. Default is $STDINX.

progfile Actual designator of program file on which prepared program segments are written. Can be any binary output file with filecode of PROG (or 1029). This parameter, if entered, must indicate a file created in one of two ways:

1. With the MPE :BUILD command using the *filecode* parameter PROG or 1029, and a *numextents* parameter value of 1.
2. By specifying a non-existent file in the progfile parameter, in which case a job temporary file of the correct size and type is created.

Default is that $NEWPASS is assigned.

listfile Actual designator of file on which program listing is written. Can be any ASCII output file. Formal designator is BSCLIST. Default is $STDLIST.

NOTE

The formal file designators used in this command (BSCTEXT, BSCPROG, BSCLIST) cannot be back referenced as actual file designators in the command parameter list. For further information, see the OPERATION section of the :FILE command.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Compiles and prepares a program from a SAVE FAST file (created by BASIC interpreter) into a program file on disc. This enables the program to run considerably faster than it would if executed by the interpreter. After the program is written, it can be stored in a SAVE FAST file with the BASIC interpreter command SAVE *filename*,FAST. It then can be compiled/prepared with the MPE :BASICPREP command and executed with the MPE :RUN command.

EXAMPLE

To compile and prepare a program named MYPROG from the BASIC SAVE FAST file named MYCOMDS, with listing directed to the standard list device, enter:

```
:BASICPREP ,MYCOMDS
```

The file MYCOMDS is an ASCII file that contains the following BASIC compiler commands:

```
$CONTROL USLINIT,SOURCE     Initializes USL and lists program
$COMPILE MYPROG             Compiles SAVE FAST program
$EXIT                       Exits from compiler
```

ADDITIONAL DISCUSSION

Basic Compiler Reference Manual.

Creates and immediately allocates new empty file on disc.

SYNTAX

```
                                                          F  BINARY
:BUILD filereference[;REC=[recsize][,[blockfactor][,[U][,          ]]]]
                                                          V  ASCII
      ;CCTL
      [          ]
       ;NOCCTL
      [;TEMP]
      [;DEV=[[dsdevice]#][device]]
      [;CODE=[filecode]]
      [;DISC=[numrec] [,[numextents] [,initalloc]]]
      [;RIO]
      [;NORIO]
      [;MSG]
      [;CIR]
```

PARAMETERS

filereference Actual name of file to be created, in the following format:

filename [/*lockword*] [.*groupname*] [.*acctname*]

Names may contain from 1 to 8 alphanumeric characters, beginning with a letter. If *acctname* is specified, it must be that of your log-on account (you cannot create a file in another account). Default *groupname* and *acctname* are the log-on group and account. (REQUIRED PARAMETER)

recsize Record size. A positive number indicates words while negative indicates bytes. For fixed-length files, this is logical record size. For undefined length, this is the maximum record size. For variable length files this is the maximum logical record size if blockfactor is 1. If not, this is used to calculate the maximum logical record size and physical record size.

Records always begin on word boundaries, therefore the record size is rounded up to the nearest word boundary for block size calculations. For a binary file or a variable length ASCII file, odd byte lengths are rounded up and the extra byte is available for data. However, if an odd byte length record size is specified for a fixed or undefined length record file, the extra byte is not available for data.

For example: a fixed length ASCII file with record size specified as 11 bytes will have only 11 bytes available for data in each logical record. However, to determine actual blocksize, 12 bytes will be used for the record size (blocksize=12 bytes x blockfactor). If the file was specified as a binary file, the 11 bytes would be rounded up to 12 bytes (6 words), all of which are available for each logical record. Default is determined as device configuration.

blockfactor — Number of logical records per physical block. Default is calculated by dividing the specified recsize into the configured blocksize; this value is rounded downward to an integer that is never less than 1. For variable length record files, blockfactor is always set to 1 after using the original value along with recsize to calculate maximum logical record size and physical record size. Blockfactor is ignored for undefined length records.

F, U, V — File contains fixed (F), undefined (U), or variable (V) length records. Default is F for disc files.

BINARY — File contains binary-coded records. Default.

ASCII — File contains ASCII-coded records. Default is BINARY.

CCTL — Indicates carriage-control characters will be supplied with write requests. Default is NOCCTL.

NOCCTL — Indicates carriage-control characters will not be supplied with write requests. Default.

TEMP — File will be created as a job temporary file and will be saved in the job/session temporary file domain when closed. Default is permanent file

device — Specifies the device on which the file is to reside, entered in one of the following forms:

devclass
ldn
*
**vcname*
***volname*

The *devclass* form represents a device class name of up to eight alpha-numeric characters beginning with a letter, as for example, DISC or PVDISC1. If *devclass* is specified, the file is allocated to any available device in that class. If you are opening a file which is to reside on a private volume, you must specify device class DISC; the file then is allocated to any of the home volume set's volumes that fall within that device class. Note that a file cannot span more than one volume set, but can reside on more than one volume within the set.

The logical device number (*ldn*) consists of a three-byte numeric string specifying a particular device. If you are opening a file which is to reside on a private volume, you must specify a disc drive on which one of the volumes in the home volume set resides.

The forms *, **vcname*, and ***volname* are used only if you are opening a file which is to reside on a private volume.

If * is specified, the file is allocated to any of the volumes of the home volume set.

If **vcname* (volume class name) is specified, *vcname* must be a member of the home volume set. The file then is allocated to any of the volumes within the volume class.

If ***volname* (volume name) is specified, *volname* must be a member of the home volume set. The file then is allocated to that volume.

Any of the forms may be used to reference files on a remote computer by preceding the device or volume specification with DSDEVICE #.

Default is device class name DISC.

filecode

Code indicating a specially-formatted file. This code is recorded in the file label and is available to processes accessing the file through the FFILEINFO or FGETINFO intrinsic. For this parameter, any user can specify a positive integer ranging from 0 to 1023. Certain integerse have particular HP-defined meanings, as follows:

Mnemonic	Integer	Meaning
	−400	IMAGE root file.
	−401	IMAGE data set.
	−402	IMAGE file for DS information.
USL	1024	USL file.
BASD	1025	BASIC data file.
BASP	1026	BASIC program file.
BASFP	1027	BASIC fast program file.
RL	1028	RL file.
PROG	1029	Program file.
SL	1031	SL file.
VFORM	1035	VIEW formsfile.
VFAST	1036	VIEW fast forms file.
VREF	1037	VIEW reformat file.
XLSAV	1040	Cross Loader ASCII file (SAVE).
XLBIN	1041	Cross Loader relocated binary file.
XLDSP	1042	Cross Loader ASCII file (DISPLAY).
EDITQ	1050	Edit KEEPQ file (non-COBOL).
EDTCQ	1051	Edit KEEPQ file (COBOL).
EDTCT	1052	Edit TEXT file (COBOL).
TDP	1058	TDP/3000 work file.
RJEPN	1060	RJE punch file.
QPROC	1070	QUERY procedure file.
	1071	QUERY work file.
	1072	QUERY work file.
KSAMK	1080	KSAM key file.
GRAPH	1083	GRAPH specification file.
SD	1084	Self-describing file.
LOG	1090	User Logging logfile.
OPTLF	1130	On-line performance Tool log file.

Using LOG as your designated filecode may not yield the number of records you specify in the DISC = parameter. Most files use the number of records specified in the DISC = parameter as the maximum limit; user logging uses this specified number as a minimum.

Default is 0.

numrec Maximum number of logical records. For fixed and undefined length files the maximum value allowed for this field is 2,147,483,647. However, the maximum sectors per file is 2,097,120 based on the maximum of 65535 sectors per extent, 32 extents maximum. Thus the actual maximum number of records will be limited by blocksize (determined by record size and blockfactor. An approximate practical limit for numrec is 2,097,119 for variable and undefined files, and 267,382,000 for fixed length files. Default is a value of 1023.

NOTE

The file system uses these values to compute other characteristics of the file as well. Therefore, the values (or default values) specified on the :FILE command may be valid within their respective fields, but may cause overflow errors in the computation of internally needed file specifications.

See the Intrinsics Manual for a discussion on calculating file space.

numextents Maximum number of disc extents. This is a value from 1 to 32. Default is 8.

initialloc Number of extents to be initially allocated to the file at the time it is opened. This is a value from 1 to 32. Default is 1.

RIO If RIO is specified, a relative I/O file is created. The record length parameter will implicitly be changed to fixed record length. RIO is a special file access method supported by COBOL II. See the Intrinsics Reference Manual for a discussion of relative I/O.

NORIO A non-relative I/O file is created.

x MSG — A message file is created allowing communication between any set of processes. Acts like a FIFO (first in, first out) queue where records are read from the start of the file and logically deleted and/or appended to the end of file.

y CIR — Acts as normal sequential file until full. When full, the first physical block will be deleted when the next record is written, and remaining blocks will be logically shifted to front of file. Cannot be simultaneously accessed by readers and writers.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

Builds a new file on disc and immediately allocates space for it, initialized to blanks (if an ASCII file) or zeros (if a binary file). Unless the TEMP parameter is specified, the file is permanent. (Note that you must have SAVE access to the group to which the new file is to belong and that you can only build a file belonging to your log-on account.) If the home volume set is not mounted for the group in which the new file is to be built, this command will implicitly cause a volume set mount request to be generated.

The default characteristics of a file created with the :BUILD command are: standard binary disc file, fixed length records of 128 words, blocking factor of 1, 1 buffer, 1023 record limit, and a maximum of 8 extents with 1 extent initially allocated.

EXAMPLE

To create a permanent disc file named WORKFILE, with fixed-length records each 80 bytes long, 3 records per block (*blockfactor*), and in ASCII code, enter:

```
:BUILD WORKFILE;REC=-80,3,F,ASCII;DISC=2000,10,2
```

The file can reside on any disc, has a maximum capacity of 2000 records divided into 10 extents, with 2 extents allocated immediately.

An example of using the :BUILD command to create a new file on a volume set/class is as follows:

```
:BUILD VFILE;DISC=500,10,1;REC=-80;DEV=VCLASS1
                                  Name of existing volume class
```

This file is confined to volumes which reside on devices of that device class.

The following example uses the CODE= parameter. In this particular case you wish to create a logging file called NEWDATA:

```
:BUILD NEWDATA;DISC=3000,1,1;CODE=LOG
```

ADDITIONAL DISCUSSION

Using Files
Intrinsics Reference Manual

:BYE

Ends an interactive session.

SYNTAX

```
:BYE
```

PARAMETERS

None

USE

Available	In Session?	YES
	In Job?	NO
	In Break?	YES
	Programmatically?	NO
Breakable?		YES (Aborts log-off message)

OPERATION

Terminates a session, and displays the cpu time used (in seconds), connect time (in minutes), and the date and time, as follows:

```
CPU=48. CONNECT=35. WED, OCT 12, 1977, 10:56 PM
```

MPE also adds the central-processor time and connect-time, along with the permanent file space used by your session, to the resource-usage counters maintained for your log-on account and group. During your session, you can determine the account and group totals by entering the :REPORT command.

If you hang up the receiver prior to logging off at a terminal with a telephone connection, MPE automatically terminates your session by implicitly issuing a :BYE command.

If you enter a :HELLO command before logging off, MPE terminates your current session and immediately initiates a new one. If you are logged on at a terminal with a telephone connection, MPE does not disconnect the terminal. Instead, MPE maintains the connection, allowing the new session to begin immediately.

EXAMPLE

To terminate a session, enter:

```
:BYE

CPU=1. CONNECT=5. THU, OCT 13, 1977, 12:20 PM
```

Compiles a COBOL program.

SYNTAX

```
:COBOL [textfile][,[uslfile][,[listfile][,[masterfile][,newfile]]]]
```

PARAMETERS

textfile Actual designator of input file from which source program is read. Can be any ASCII input file. Formal designator is COBTEXT. Default is $STDIN.

uslfile Actual designator of user subprogram library (USL) file on which object program is written. Can be any binary output file with *filecode* of USL (or 1024). Formal designator is COBUSL. If entered, this parameter must indicate a file created in one of three ways:

1. By saving a USL file (with the :SAVE comand) created by a previous compilation where the default value was used for the *uslfile* parameter.
2. By building the USL with the Segmenter command -BUILDUSL. (See the *MPE Segmenter Reference Manual.*)
3. By creating a new USL file with the MPE :BUILD command and a *filecode* parameter of USL or 1024.

Default: $NEWPASS is assigned

listfile Actual designator of file on which program listing is written. Can be any ASCII output file. Formal designator is COBLIST. Default is $STDLIST.

masterfile Actual designator of master file which is merged against *testfile* to produce composite source. Can be any ASCII input file. Formal designator is COBMAST. Default is that the master file is not read; input is read from *textfile*.

newfile Actual designator of merged *textfile* and *masterfile*. Can be any ASCII output file. Formal designator is COBNEW. Default is that no file is written.

NOTE

The formal file designators used in this command (COBTEXT, COBUSL, COBLIST, COBMAST, COBNEW) cannot be back referenced as actual file designators in the command parameter list. For further information, see the OPERATION section of the :FILE command.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Compiles COBOL program onto a USL file on disc. If you do not specify a source text file, MPE expects input from your standard input device. If you create the USL prior to compilation, you must specify a *filecode* of USL or 1024. If you do not specify a *listfile,* MPE sends the program listing to the current list device.

EXAMPLE

To compile a COBOL program that you enter from your current input device into an object program in the USL file $NEWPASS, and write the listing to your current list device, enter:

```
:COBOL
```

If the next command is one to prepare an object program, $NEWPASS can be passed to that command by specifying $OLDPASS for the uslfile parameter. A file can only be passed between commands or programs within the same job/session.

To compile a COBOL program residing on the disc file SOURCE into an object program on the USL file OBJECT, with the program listing to be sent to the disc file LISTFL, enter:

```
:BUILD OBJECT;CODE=USL
:COBOL SOURCE,OBJECT,LISTFL
```

ADDITIONAL DISCUSSION

COBOL Reference Manual
COBOL II Reference Manual

:COBOLGO

Compiles, prepares, and executes a COBOL program.

SYNTAX

```
:COBOLGO [textfile][,[listfile][,[masterfile][,newfile]
```

PARAMETERS

textfile Actual designator of input file from which source program is read. Can be any ASCII input file. Formal designator is COBTEXT. Default is $STDIN.

listfile Actual designator of file on which program listing is written. Can be any ASCII output file. Formal designator is COBLIST. Default is $STDLIST.

masterfile Actual designator of master file which is merged against *textfile* to produce composite source. Can be any ASCII input file. Formal designator is COBMAST. Default is that the master file is not read; input is read from *textfile*.

newfile Actual designator of merged *textfile* and *masterfile*. Can be any ASCII output file. Formal designator is COBNEW. Default is that no file is written.

NOTE

The formal file designators used in this command (COBTEXT, COBLIST, COBMAST, COBNEW) cannot be back referenced as actual file designators in the command parameter list. For further information see the OPERATION section of the :FILE command.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Compiles, prepares, and allocates/executes a COBOL program. If you do not specify *textfile,* MPE expects your input from your current input device. If you do not specify *listfile,* MPE writes the listing to your current list device. This command creates a temporary USL file ($NEWPASS) that cannot be accessed, and a temporary program file that can be accessed under the name $OLDPASS.

EXAMPLE

To compile, prepare, and execute a COBOL program entered from your current input device, with the program listing sent to your current list device, enter:

```
:COBOLGO
```

To compile, prepare, and execute a COBOL program from the disc file TEXTEL and send the program listing to the disc file LISTFL, enter:

```
:COBOLGO TEXTFL,LISTFL
```

Text file *List file*

The :COBOLGO command is equivalent to:

Text file *USL file* *List file*

```
:COBOL TEXTFL,$NEWPASS,LISTFL
:PREP $OLDPASS,$NEWPASS
:RUN $OLDPASS
```

Program file *USL file* *Program file*

ADDITIONAL DISCUSSION

Using the HP 3000
COBOL II Reference Manual

:COBOLPREP

Compiles and prepares a COBOL program.

SYNTAX

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

PARAMETERS

textfile Actual designator of input file from which source program is read. Can be any ASCII input file. Formal designator is COBTEXT. Default is $STDIN.

progfile Actual designator of program file on which prepared program segments are written. Can be any binary output file with *filecode* of PROG or 1029. If parameter is specified, it must indicate a file created in one of two ways:

1. With the MPE :BUILD command, specifying a *filecode* of PROG or 1029 and a *numextents* parameter value of 1.
2. By specifying a non-existent file in the *progfile* parameter in which case a job temporary file of the correct size and type is created.

Formal designator is COBPROG.

Default: $NEWPASS is assigned.

listfile Actual designator of file on which program listing is written. Can be any ASCII output file. Formal designator is COBLIST. Default is $STDLIST.

masterfile Actual designator of master file which is merged against *textfile* to produce composite source. Can be any ASCII input file. Formal designator is COBMAST. Default is that the master file is not read; input is read from *textfile*.

newfile Actual designator of merged *textfile* and *masterfile*. Can be any ASCII output file. Formal designator is COBNEW. Default is that no file is written.

NOTE

The formal file designators used in this command (COBTEXT, COBPROG, COBLIST, COBMAST, COBNEW) cannot be back referenced as actual file designators in the command parameter list. For further information see the OPERATION section of the :FILE command.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Compiles and prepares a COBOL program onto a program file on disc. If you do not specify *textfile,* MPE expects your input from your current input device. If you create the program file prior to compilation, you must use the *filecode* parameter of PROG or 1029 and a *numextents* parameter value of 1. If you do not specify *listfile,* MPE sends the listing output to your current list device. The USL file created during compilation is a temporary file passed directly to the preparation mechanism; you can access it under the name $OLDPASS only if the program file is not $NEWPASS.

EXAMPLE

To compile and prepare a COBOL program entered through your current input device, onto the file $NEWPASS, with the listing printed on the current list device, enter:

```
:COBOLPREP
```

If the next command is one to execute the program, the file $NEWPASS is referenced in the execute command under the name $OLDPASS.

To compile and prepare a COBOL source program input from the *textfile* named SFILE into a program file named MYPROG, with the resulting listing sent to the current list device, enter:

```
:COBOLPREP SFILE,MYPROG
```

ADDITIONAL DISCUSSION

Using the HP 3000
COBOL II Reference Manual

Inserts comment into command stream.

SYNTAX

```
:COMMENT [text]
```

PARAMETERS

text Information comprising comment text. If last non-blank character is ampersand (&), comment text is continued onto next line. Default is null comment. (A record containing only :COMMENT is inserted in command stream.)

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

The :COMMENT command allows you to include comments or notes in job listings produced on hard copy devices to create headings or explain the purpose of commands or logic used. After :COMMENT is entered, it can be followed by the message made up of any ASCII characters. This command is used primarily in batch jobs but can be used in sessions as well.

EXAMPLE

The following example employs a comment as a job heading.

```
:JOB MAC.TECHPUBS
PRIORITY = ES; INPRI = 8
PRI= DS; INPRI= 13; TIME= ?
JOB NUMBER = #J8
MON,SEP 8, 1980, 2:21 PM
HP3000 / MPE IV C.00.00

:COMMENT — THIS IS A SAMPLE JOB
:FORTGO
```

:CONSOLE

Displays the location of the operator's console.

SYNTAX

```
:CONSOLE
```

PARAMETERS

None

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

This command allows any user to find the current location of the console. Using this command without any parameters requires no special capabilities and the device number of the system console will be returned.

EXAMPLE

```
:CONSOLE

Console is currently assigned to LDEV 20.
```

ADDITIONAL DISCUSSION

See Console Operator's Guide for a discussion of the operator command :CONSOLE.

:CONTINUE

Overrides job or session error.

SYNTAX

```
:CONTINUE
```

PARAMETERS

None

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		NO

OPERATION

The :CONTINUE command permits a job or session to continue even though the next command results in an error (with accompanying error message). Typically used when anticipating errors that you wish to override and applies only to the command immediately following :CONTINUE.

EXAMPLE

If you anticipate a possible error resulting from the command

```
:RUN MYPROG
```

and wish to override this error and allow a job to continue, enter:

```
:JOB MAC.PUBS
:CONTINUE
:RUN MYPROG
:RUN MYPROG2
:EOJ
```

:DATA

Enters data into system from devicefile. (Cannot be used to enter data from $STDIN.)

SYNTAX

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

PARAMETER

jsname Name of job or session that is to read data. Default is no job/session name.

username Your user name that allows you to access MPE under this account, as established by Account Manager. (REQUIRED PARAMETER)

userpass Your user password, optionally assigned by Account Manager.

acctname Name of account under which job/session is running, as established by System Manager. (REQUIRED PARAMETER)

acctpass Account password, optionally assigned by System Manager.

filename Additional qualifying name for the data that can be used by job or session to access the data. May be used, for instance, to distinguish two separate data decks from different card readers read by the same program. Default is that no distinguishing name is assigned.

Note that the *jsname, username, userpass, acctname, acctpass,* and *filename* parameters all are names that can contain up to 8 alpha-numeric characters, beginning with a letter.

USE

Available	In Session?	YES (but not from $STDIN)
	In Job?	YES (but not from $STDIN)
	In Break?	NO
	Programmatically?	NO
Breakable?		NO

OPERATION

Identifies input to be read from a device file other than your standard job/session input file, for example when reading a file from a batch input device while running an interactive session.

To designate a set of data as an auxiliary file for your job or session, enter the :DATA command, followed by the data and terminated with the :EOD command. The data can only be read by a job or session that has the same identity ([jsname,]username.account), i.e., the same identity that is used on the data command, and can only be submitted via a device configured to accept the :DATA command. If the filename parameter is omitted, data can be read by any access from a job or session with the same identity.

When the data file is placed on the input device and the device is readied, MPE reads the entire file (if input from a spooled device), or the :DATA command only (if input from a non-spooled device). From this point on, your job can access the data, which remains available until it is actually read.

The :DATA command implicitly initiates communication with MPE, and thus is the only command not entered within a formally initiated job or session. Files identified by :DATA may be input on cards, magnetic tape, paper tape, or from a terminal, but not from disc. When entered from magnetic tape, such a file must reside on a single tape volume. The :DATA command may not be used for data embedded within the standard input stream or for data on disc files.

NOTE

> The :DATA command establishes the standard input file of the device on which it is entered as a data file. Therefore, it is not a command to be used on a device which is active within a formally initiated job or session. For example, if you enter the :DATA command from your terminal during a session, it makes the terminal a data file for some unspecified process. Then, until a process issues a request for data from the terminal, it will not be available for any other function.

To designate data for your job or session, enter the :DATA command, followed by the data, followed by the :EOD command. The data can only be read by a job/session that has the same identity ([*jsname*,] *username.acctname*).

If the *filename* parameter is omitted, data can be read by any access from a job/session with the same identity.

When the data file is placed on the input device and the device is readied, MPE reads the entire file (if input from a spooled device), or the :DATA command only (if input from a non-spooled device). From this point on, your job can access the data, which remains available until it is actually read.

The :DATA command implicitly initiates communication with MPE, and thus is the only command not entered within a formally-initiated job or session. Files identified by :DATA may be input on cards, magnetic tape, paper tape, or from a terminal (but not from disc). When entered from magnetic tape, such a file must reside on a single tape volume.

EXAMPLE

As an example, suppose you are running a session identified by the session name SESSA, user name BROWN, and account name ACCT1. You wish to make data on punched cards available to that session, to be used by a program named PROGY. This program references the data under the formal file designator DATAFL. Proceed as follows:

1. Arrange your data deck beginning with the :DATA command and terminating with the :EOD command, as follows:

```
:DATA SESSA, BROWN.ACCT1
      .
      .
      .
   (YOUR DATA)
      .
      .
      .
:EOD
```

2. Load the cards into the card reader and make the reader ready. If the card reader is a spooled device, the data is copied to disc, where it awaits your access. If it is not a spooled device, the :DATA command is read but the subsequent data remains in the card reader until your program accesses it. The :DATA command is used to build an entry in the device directory that identifies the file to the system.

3. Begin your session, making sure that you use precisely the same session identity entered in your :DATA command. (For instance, if you omit the optional session name SESSA, your session will not be able to access the data.) To log on, enter:

```
:HELLO SESSA,BROWN.ACCT1
```

Same identity used in :DATA command

4. Enter a :FILE command equating the formal file designator DATAFL with the card reader. For instance,

```
:FILE DATAFL;DEV=CARD
```

5. Run the program that requires the data; when the program attempts to read the data, it will be available:

```
:RUN PROGY
```

NOTE

If you have not entered the data through the card reader prior to this step, the program transmits a message to the Console Operator requesting this data and waits for him to furnish it. The operator may, at his option, supply the data via the :DATA command or by allocating a device configured not to accept this command; or if he does not have the data, he may send you a message asking you to supply it or he may abort the session.

6. Once the data has been read, it is no longer available to the system. When you run another program requiring this data, you must therefore enter the data again in the above manner.

Invokes the MPE Debug facility. (Privileged Mode capability required)

SYNTAX

```
:DEBUG
```

PARAMETER

None

USE

Available	In Session?	YES
	In Job?	NO
	In Break?	YES
	Programmatically?	NO
Breakable?		NO

OPERATION

(Privileged Mode capability required)
The :DEBUG command is an extension of the MPE Debug facility and is used primarily by systems programmers; its use is explained in the *MPE Debug/Stack Dump Reference Manual.*

IMPORTANT NOTE

The normal checks and limitations that apply to the standard users in MPE are bypassed in privileged mode. It is possible for a privileged mode user to destroy the integrity of the system, including the MPE operating system software itself. Hewlett-Packard will investigate and attempt to resolve problems resulting from the use of privileged mode code. This service, which is not provided under the standard Service Contract, is available on a time and materials billing basis. However, Hewlett-Packard will not support, correct, or attend to any modification of the MPE operating system software.

ADDITIONAL DISCUSSION

MPE Debut/Stack Dump Reference Manual.

:DISASSOCIATE

Removes control of a device from the user.

SYNTAX

:DISASSOCIATE *devclass*

PARAMETERS

devclass The name of a logical device confiured during SYSDUMP.

USE

Available	In Session?	YES
	In Job?	NO
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

This command counteracts a previously issued :ASSOCIATE command in that it removes the control of a device class from a user. The command may be issued by the console operator and by the user. The user implicitly disassociates a device when logging off.

EXAMPLE

```
:DISASSOCIATE PRINTER
```

ADDITIONAL DISCUSSION

Console Operator's Guide.

:DISMOUNT

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

SYNTAX

```
                    *
:DISMOUNT [         |  ][.groupname[.acctname]]
                  vcsname
```

PARAMETERS

*	Specifies the home volume set for the group and account specified, or the log-on group and account if *groupname.acctname* are not specified. Default.
vcsname	Volume set/class name, specifying a previously-defined volume class name or volume set name. Default is *.
groupname.acctname	Specify the group and account which created the volume set. Default is log-on group and account.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	NO
Breakable?		NO

OPERATION

The :DISMOUNT command allows you to dismount a volume set that you previously explicitly mounted via a :MOUNT command. If there are no other users of the volume set, the devices on which it resides are made available to the system. You can only request a dismount for a volume set which you have caused to be mounted; you cannot affect the status of the volume set with respect to other users.

EXAMPLE

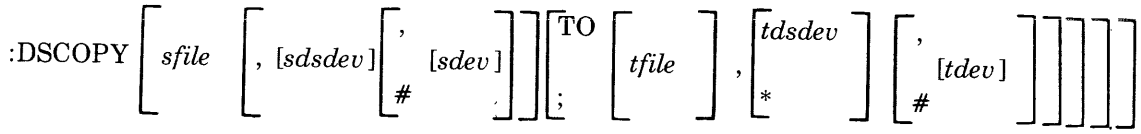
To dismount the volume set MYSET, previously mounted by you via a :MOUNTcommand, enter:

```
:DISMOUNT MYSET
```

:DSCOPY

Enables user to copy disc files locally or between HP 3000's.

SYNTAX



PARAMETERS

sfile (Required Parameter) Identifies the file to be copied. The name can be written in the following format:

sfile [/*lockword*] [.*groupname*] [.*accountname*]

If the source file is in a group.account different from the requestor's logon group.account, the requestor must have read and lock access to the source file.

sdsdev (Optional Parameter) The device classname or logical device number that was used to open the communications link to the remote computer where the source file resides.

Default: The local system (that is, the system where the transfer request is submitted).

sdev (Optional Parameter) The classname or logical device number of the disc where the source file resides.

Default: DISC.

tfile (Optional Parameter) Specifies the file to receive the data. The name can be written in the following format:

tfile [/*lockword*] [.*groupname*] [.*accountname*]

Default: The new file has the same filename, groupname, and accountname as the source file. Security is on for the new file, even though the source file may have been released.

tdsdev (Optional Parameter) The device classname or logical device number that was used to open the communications link to the remote computer where the target file will reside.

Default: DSCOPY copies the sourcefile to the local computer and assigns the same filename as the sourcefile name. If the source computer is the local system, this default causes a file system error (because the file already exists).

Means the target dsdevice (i.e., the target computer) is the same as the source dsdevice (i.e., the source computer).

* (Optional Parameter) The device classname or logical device number of the disc where the new file should reside.

Default: DISC

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO*
Breakable?		NO
*Call the DSCOPY intrinsic rather than use the COMMAND intrinsic.		

OPERATION

This command accesses the Network File Transfer (NFT) program that allows the user to copy disc files from within his own HP 3000, or between HP 3000's, via a DS/3000 communication link. The user can initiate copy operations from sessions, jobs, or programs. DSCOPY can be used to copy users' files and MPE system files as well as data management files such as KSAM/3000 files. It does not copy Image files.

EXAMPLE

To make a local copy of SFILE and name the new file TFILE, use either of the following:

```
:DSCOPY SFILE TO TFILE        or        :DSCOPY SFILE; TFILE
```

To copy a file from the computer connected to dsline SYSA into your logon group (on the local system), enter:

```
:DSCOPY SFILE,SYSA;TFILE
```

To copy a file on the same remote system, enter:

```
:DSCOPY FILEA, SYSX TO FILEB, SYSX
```

To copy a file between two remote systems, enter:

```
:DSCOPY FILEA, SYSX TO FILEB, SYSY
```

ADDITIONAL DISCUSSION

DS/3000 Reference Manual.

:DSLINE

Opens or closes communication lines with DS/3000.

SYNTAX

To open a DS line:

```
:DSLINE dsdevice[;LINEBUF=buffsize]

        [;LOCID=local-id-sequence]
        [;REMID=remote-id-sequence1[,remote-id-sequence2]....]
        [;PHNUM=telephone number]
        [;EXCLUSIVE]
        [;COMP]
        [;NOCOMP]
        [;QUIET]
```

To close a communication line:

```
          dsdevice
:DSLINE  [ds-line-number];CLOSE
          @
```

PARAMETERS

dsdevice — Device class name or logical device number assigned to the DS/3000 communications driver IODS0 during system configuration. This parameter specifies what physical line you wish to use. (REQUIRED PARAMETER)

buffsize — A decimal integer specifying the size (in words) of the DS/3000 line buffer to be used in conjunction with the particular communications line. The integer must be within the range 304 < buffer size > 4095. Default is the buffer size entered in response to the PREFERRED BUFFER SIZE prompt during system configuration.

EXCLUSIVE — Specifies that you want exclusive use of the particular communications line. If the requested line already is open and you have specified the exclusive option, DS/3000 will deny you access to the line. Use of this parameter requires CS capability. Default is that the line is opened non-exclusive.

local-id-sequence — A string of ASCII characters contained within quotation marks or a string of octal numbers separated by commas and contained within parameters. If you wish to use a quotation mark within an ASCII string, use two successive quotation marks. In the case of an octal sequence, each octal number represents one byte and must be within the range 0 - 377. The maximum number of ASCII characters or octal numbers allowed is 16.

The supplied string of ASCII characters or octal numbers defines the ID sequence that will be sent from your HP 3000 to the remote HP 3000 when you attempt to establish the telephone connection. If the remote HP 3000 does not recognize the supplied ID sequence as valid, the telephone connection is terminated. Default is the ASCII or octal string entered in response to the LOCAL SEQUENCE prompt during system configuration.

remote-id-sequence A string of ASCII characters or octal numbers in the same format as described under the *local-id-sequence* parameter.

The supplied strings of ASCII characters or octal numbers define those remote HP 3000 ID sequences that will be considered valid when you attempt to establish the telephone connection. If the remote HP 3000 does not send a valid ID sequence, the telephone connection is terminated. Default is the ASCII and octal strings entered in response to the REMOTE ID SEQUENCE prompt during system configuration.

telephone-number A telephone number consisting of digits and dashes. The maximum length permitted (including both digits and dashes) is 20 characters. Provided that YES was entered in response to the DIAL FACILITY prompt during system configuration, this telephone number will be displayed at the operator's console of your HP 3000 and the operator will then establish the telephone connection by dialing that number at the MODEM. Default is the first one entered in response to the PHONE NUMBER prompt during system configuration.

ds-line-number The DS line number assigned to you by DS/3000 when the particular line was opened.

@ Specifies that you wish to close all lines that you currently have open.

CLOSE Specifies that you wish to close the specified lines.

COMP Overrides the current system default, which was set at configuration time or set by the system operator, and activates data compression. In this way, the mode of operation is set for your subsequent DS activity. This parameter does not affect other users sharing the line.

NOCOMP Deactivates the data compression mode.

QUIET When you issue the DSLINE command with this parameter added, the message identifying the DS line number is suppressed. The messages associated with the subsequent REMOTE HELLO and REMOTE BYE commands will also be suppressed.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	YES
Breakable?		NO

OPERATION

The :DSLINE command opens or closes hardwired or telephone communications lines with DS/3000. To open a line, the specified line must be currently available for your use, that is the console operator must have opened the line.

EXAMPLE

To initiate a local session on System A, obtain access to the hardwired communications line that connects System A to System B, and initiate a remote session on System B, enter:

```
:HELLO USER.X        (Logging on locally)
:DSLINE HDS1         (Accessing DS line)
:REMOTE HELLO USER.X      (Logging on remotely)
```

ADDITIONAL DISCUSSION

DS/3000 Reference Manual

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

SYNTAX

```
                 ldn
:DSTAT [                ]
                 ALL
```

PARAMETERS

ldn	An integer specifying the logical device number of the disc drive whose status is desired.
ALL	Causes the status of all disc drives, system and non-system, to be displayed.

The default is that if no parameter is included, only the status of non-system discs is displayed.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

The :DSTAT command causes the display of the current status of the disc drives on the system.

EXAMPLE

To display the status of all disc drives on the system enter:

```
:DSTAT ALL

LDEV-TYPE       STATUS          VOLUME (VOLUME SET-GEN)
1-7920          SYSTEM          MH7920U1
2-7925          SYSTEM          MH7925U1
3-7902          DOWNED          *FOREIGN DISC*
4-7920          OFF-LINE
```

To display the status of LDN 7 enter:

```
:DSTAT7
LDEV-TYPE       STATUS          VOLUME (VOLUME SET-GEN)
7-7902          FOREIGN         *UNALLOCATED*
```

:EDITOR

Invokes the Editor.

SYNTAX

:EDITOR [*listfile*]

PARAMETERS

listfile Actual designator of file to receive any output resulting from Editor commands LIST and XPLAIN when the OFFLINE option is specified. Can be any ASCII output file. Formal designator is EDTLIST. Because this file often is a line printer, it usually is defined in an MPE :FILE command and back referenced as follows:

```
:FILE LISTFILE;DEV=LP
:EDITOR*LISTFILE
```

Default: If omitted, default is EDTLIST. If specified with no device parameter, default device is LP.

NOTE

The formal file designator used in this command (EDTLIST) cannot be back referenced as an actual file designator in the command parameter list. For further information, see OPERATION section of :FILE command.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Accesses the Editor subsystem.

EXAMPLE

To access the Editor during a session and specify a line printer (device class LP1) as the list device for offline output, enter:

```
:FILE LISTFILE;DEV=LP1
:EDITOR*LISTFILE
```

ADDITIONAL DISCUSSION

EDIT/3000 Reference Manual.

:ELSE

Provides an alternate execution sequence for an :IF statement.

SYNTAX

```
:ELSE
```

PARAMETERS

None

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	NO
Breakable?		NO

OPERATION

The :ELSE command is used only with the :IF command. The :IF command is used with the :ENDIF command and, optionally, the :ELSE command, to control the execution of a job. The :IF command, :ENDIF command, and optional :ELSE command constitute an IF block. A logical expression is evaluated and, if true, the IF block is executed, if false, the ELSE block (if one exists) is executed.

EXAMPLE

```
:RUN UPDATE1                                        (Run first data base update)
:IF JCW<WARN THEN                                   (Continue if OK so far)
:     SETJCW UPDATE1JCW:=JCW                        (Save that return code)
:     RUN UPDATE2                                   (Run second data base update)
:     IF JCW<WARN THEN                              (Continue if still OK)
:          SETJCW UPDATE2JCW:=JCW                   (Save the second return code)
:          RUN DBSTATUS                             (Report current status of DB)
:          IF (UPDATE1JCW < 50) AND (UPDATE2JCW < 50) THEN
:               RUN DAILYRPT
:          ENDIF
:          IF ((UPDATE1JCW = 1) OR (UPDATE2JCW =1)) THEN
:               RUN WEEKLYRPT
:          ENDIF

:     ELSE                                          (Second update program failed)
:          RUN FIXUP                                (Repair data base and report)
:     ENDIF
:ENDIF
```

Terminates an IF block.

SYNTAX

```
:ENDIF
```

PARAMETERS

None

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	NO
Breakable?		NO

OPERATION

The :ENDIF command is used to terminate an IF block. The :IF command, :ENDIF command, and optional :ELSE commad, constitute an IF block and are used to control the execution of a job. A logical expression is evaluated and, if true, the IF block is executed; if false, the :ELSE block (if one exists) is executed.

EXAMPLE

```
:RUN UPDATE1                                    (Run first data base update)
:IF JCW<WARN THEN                               (Continue if OK so far)
:     SETJCW UPDATE1JCW:=JCW                    (Save that return code)
:     RUN UPDATE2                               (Run second data base update)
:     IF JCW<WARN THEN                          (Continue if still OK)
:          SETJCW UPDATE2JCW:=JCW               (Save the second return code)
:          RUN DBSTATUS                         (Report current status of DB)
:          IF (UPDATE1JCW < 50) AND (UPDATE2JCW < 50) THEN
:               RUN DAILYRPT
:          ENDIF
:          IF ((UPDATE1JCW = 1) OR (UPDATE2JCW =1)) THEN
:               RUN WEEKLYRPT
:          ENDIF

:     ELSE                                      (Second update program failed)
:          RUN FIXUP                            (Repair data base and report)
:     ENDIF
:ENDIF
```

:EOD

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

SYNTAX

```
:EOD
```

PARAMETERS

None

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		NO

OPERATION

The :EOD command is used to signify the end of a data set entered through a standard input device. It is also used to delimit data entered via the :DATA command from data-accepting devices such as paper tape and card readers.

Although in most cases programmers use :EOD for delimiting data, any record beginning with a colon will delimit the data. Using a command other than :EOD for this purpose, however, depends upon whether the standard input file is opened with the file name $STDIN or $STDINX. Refer to Table 2-2 to see what delimiters are permitted with certain types of files.

When using a compiler language that does not provide a convention for terminating compilation (such as END. in SPL), you must enter :EOD after the last record of your source program to ensure proper delimiting of your input. (The :EOD command is not required when using the BASIC interpreter since the subsystem provides different conventions for delimiting data).

PROGRAMMER'S NOTE

The :EOD command causes the READ of the FREAD intrinsic to return the CCG condition code to the calling program. This condition code indicates the end-of-file condition on the terminal.

Table 2-2. END-OF-FILE Indicators

TYPE OF FILE	INDICATORS	
DATA file from Standard input device (for Sessions)	:EOD :EOF	*Terminates $STDIN and $STDINX*
	: followed by any other character	*Terminates $STDIN*
DATA file from Standard input device (for jobs)	:EOJ :JOB :EOD :DATA :EOF:	*Terminates $STDIN and $STDINX*
	: followed by any other character	*Terminates $STDIN. This record is then interpreted by the Command Interpreter as the next command to be executed.*
:DATA files	:EOD :JOB :DATA :EOF:	

EXAMPLE

To terminate a data file entered on cards for a session identified as :DATA SESS1,BLACK.ACCTSP, you would enter:

```
:DATA SESS1,BLACK.ACCTSP
        .
        .
        data
        .
        .
:EOD
```

The following FORTRAN program is an example of how the :EOD command is used to trminate a set of data entered through a standard input device.

```
:FORTRAN

PAGE 0001    HP32102B.01.04   (C)  HEWLETT-PACKARD CO. 1980


>$CONTROL USLINIT
>       PROGRAM MONEY
>       INTEGER QUARTERS,DIMES,PENNIES
>       DISPLAY "INPUT MONEY AMOUNT IN DECIMAL FORM "
>       ACCEPT DECIMALFORM
>       CALL CHANGER(DECIMALFORM,QUARTERS,DIMES,NICKELS,PENNIES)
>       DISPLAY QUARTERS," QUARTERS"
>       DISPLAY DIMES," DIMES"
>       DISPLAY NICKELS," NICKELS"
>       DISPLAY PENNIES," PENNIES"
>       STOP
>       END


PROGRAM UNIT MONEY COMPILED
>       SUBROUTINE CHANGER(DECIMALFORM,QUARTERS,DIMES,NICKELS,PENNIES)
>       INTEGER QUARTERS,DIMES,PENNIES
>       DECIMALFORM = DECIMALFORM*100
>       QUARTERS = DECIMALFORM/25
>       REMAINDER = DECIMALFORM-(QUARTERS*25)
>       DIMES=REMAINDER/10
>       REMAINDER=REMAINDER-(DIMES*10)
>       NICKELS=REMAINDER/5
>       PENNIES=REMAINDER-(NICKELS*5)
>       RETURN


PROGRAM UNIT CHANGER COMPILED
>:EOD


****       GLOBAL STATISTICS        ****
****    NO ERRORS,    NO WARNINGS   ****
TOTAL COMPILATION TIME   0:00:01
TOTAL ELAPSED TIME       0:01:29

END OF COMPILE
```

Simulates hardware end-of-file on input stream from any device.

SYNTAX

:EOF:	(The last colon in this command must be followed by a blank)

PARAMETERS

None

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		NO

OPERATION

Denotes end-of-file read through an input stream. Required to simulate a physical end-of-file on HP 2894A Card Reader/Punch when that device is configured to refuse data submitted via a :DATA command. (This device does not provide a true hardware end-of-file indication.)

EXAMPLE

```
:EOF:          (The last colon in this command must be followed by a blank)
```

:EOJ

Ends a batch job.

SYNTAX

```
:EOJ
```

PARAMETERS

None

USE

Available	In Session?	NO
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		NO

OPERATION

The :EOJ command terminates a batch job and displays the CPU time (in seconds) and the elapsed time since the beginning of the job (rounded upward to the nearest minute). MPE also adds the central processor time and file space used by your job to the resource usage counters maintained for your log-on account and group.

If you omit the :EOJ command from a job, the next :JOB command terminates the current job and starts a new one. (The end of the first job is indicated by the standard end-of-job display, and the beginning of the next job is denoted by the job-initiation display, in the normal way.

EXAMPLE

```
:EOJ

CPU SEC.=4. ELAPSED MIN.=8. TUE, OCT 18, 1977, 4:17 PM
```

Accesses the FCOPY subsystem.

SYNTAX

```
:FCOPY [fcopycommand]
```

PARAMETERS

fcopycommand An FCOPY subsystem command. See the FCOPY Reference Manual for a description of the command syntax.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES

OPERATION

This command accesses the FCOPY subsystem from MPE. If the command is entered with no parameters, FCOPY prompts (>) the user for subsystem commands one at a time until an EXIT command is entered. If the fcopycommand parameter is used, FCOPY executes the FCOPY subsystem command and then returns control to MPE.

EXAMPLE

To access the FCOPY subsystem to execute multiple commands, enter:

```
:FCOPY
HP32212A.3.12 FILE COPIER (C) HEWLETT-PACKARD CO. 1980

>                    (FCOPY subsystem prompt character)
```

To access FCOPY to execute a single command and return control to MPE, enter the command as follows:

```
:FCOPY FROM=UDC.COHEN;TO=;OCTAL;CHAR
                    .
                    .
                    .
:
```

ADDITIONAL DISCUSSION

FCOPY Reference Manual

:FILE

Declares file attributes to be used at file open time. May be used to override programmatic or system default file specifications.

SYNTAX

```
:FILE namespec { devicespec[access][disposition] }
               { access[devicespec][disposition] }
               { disposition[access][devicespec] }
```

PARAMETERS

namespec (NAME) — Consists of the formal name used by the program and may be equated to another file in the system. (REQUIRED PARAMETER)

devicespec (DEVICE) — A list of parameters giving the physical description of the REC, DEV, DISC, CODE, RIO, NORIO, STD, MSG and CIR parameters.

access (ACCESS) — A list of parameters specifying the type of access granted to a file.

disposition (DISP) — Specifies what is to be done with the file after it is closed. This consists of the DEL, SAVE, and TEMP parameters.

SYNTAX FOR NAMESPEC

```
formaldesignator [= *formaldesignator                ]
                 [=$NULL                             ]
                 [=$NEWPASS                          ]
                 [=$OLDPASS                          ]
                 [=$STDIN                            ]
                 [=$STDINX                           ]
                 [=$STDLIST                          ]
                 [=filereference] [,NEW              ]
                                  [,OLD              ]
                                  [,OLDTEMP          ]
```

NAMESPEC PARAMETERS

formaldesignator — Formal name specified by program opening the file. Contains from 1 to 8 alphanumeric characters in each field, beginning with a letter. *formaldesignator* is used as the file name when *filereference* is not specified. The format is:

filename [.*groupname* [.*accountname*]]

**formaldesignator* — Equates the formal designator to a formal designator from a previous :FILE command. When a formal designator is equated to another, no other :FILE command parameters are allowed.

$NULL — System-defined file that is always empty.

$NEWPASS — System-defined job temporary file. Opening $NEWPASS purges any previous temporary file. When $NEWPASS is closed, it is referenced by the name $OLDPASS.

$OLDPASS — System-defined name of the last temporary file closed as $NEWPASS.

$STDIN — System defined name of the standard job input device. A colon as the first character read on this file indicates end-of-data.

$STDINX — Equivalent to $STDIN except colon can be read as the first character and received as data (:JOB, :DATA, :EOJ, and :EOD can not be read as data, however).

$STDLIST — System defined name for standard job/session list device.

filereference — Actual name of file, in the following format:

filename [/*lockword*] [.*groupname* [.*accountname*]]

Each field may contain from 1 to 8 alphanumeric characters, beginning with a letter. Omission of *lockword* causes MPE to ask for it when the file is opened, if one is present. Default *groupname* and *accountname* are the logon group and account.

NEW — Specification that the file is a new file.

OLD — Previously-existing permanent file saved in system or private volume domain. File continues to exist after current session/job terminates.

OLDTEMP — Previously-existing temporary file in session/job temporary file domain. File is deleted at end of current session/job.

NOTE

NEW, OLD, and OLDTEMP specify the file domain indicating where the file exists. If all are omitted, the domain specified in the FOPEN intrinsic takes effect.

SYNTAX FOR DEVICESPEC

```
[;REC = [recsize ] [,[blockfactor ] [,[F ] [,BINARY ]]]]
                                    [U ] [,ASCII ]
                                    [V ]
[;DEV = [[dsdevice ] # ] [device ] [,[outpriority ] [,numcopies ]]]
[;DISC = [numrec ] [,[numextents ] [,initalloc ]]]
[;CODE = [filecode ]]
[;RIO ]
[;NORIO ]
[;STD ]
[;MSG ]
[;CIR ]
```

These parameters can be used for new files only with the exception of REC and DEV. REC can also be specified for system defined files. DEV can be used with OLD files and with files found on DS lines.

:FILE A,OLD;DEV=23 *system looks for data entered on Device 23.*

:FILE A,OLD;DEV=FONZ*DISC *System accessed by DS line*

DEVICESPEC PARAMETERS

recsize

Record size. A positive number indicates words while negative indicates bytes. For fixed-length files, this is logical record size. For undefined length, this is the maximum record size. For variable length files this is the maximum logical record size if blockfactor is 1. If not, this is used to calculate the maximum logical record size and physical record size.

Records always begin on word boundaries, therefore the record size is rounded up to the nearest word boundary for block size calculations. For a binary file or a variable length ASCII file, odd byte lengths are rounded up and the extra byte is available for data. However, if an odd byte length record size is specified for a fixed or undefined length record file, the extra byte is not available for data.

For example: a fixed length ASCII file with record size specified as 11 bytes will have only 11 bytes available for data in each logical record. However, to determine actual blocksize, 12 bytes will be used for the record size (blocksize=12 bytes x blockfactor). if the file was specified as a binary file, the 11 bytes would be rounded up to 12 bytes (6 words), all of which are available for each logical record.

blockfactor

Number of logical records per physical block. Default is calculated by dividing the specified recsize into the configured blocksize; this value is rounded downward to an integer that is never less than 1. For variable length record files, blockfactor is always set to 1 after using the original value along with recsize to calculate maximum logical record size and physical record size. Blockfactor is ignored for undefined length records.

F, U, V — File contains fixed (F), undefined (U) or variable (V) length records.

BINARY — File contains binary-coded records.

ASCII — File contains ASCII-coded records.

device — Contains a string of ASCII characters terminating with any non-alphanumeric character except a slash or period, designating the device on which the file is to reside. This parameter may be specified in one of the following forms:

devclass
ldn
*
**vcname*
***volname*

The *devclass* form represents a device class name of up to eight alphanumeric characters beginning with a letter, as for example, DISC or TAPE. If *devclass* is specified, the file is allocated to any available device in that class. If you are opening a file which is to reside on a private volume, you must specify device class DISC; the file then is allocated to any of the home volume set's volumes that fall within that device class.

The logical device number (*ldn*) consists of a three-byte numeric string specifying a particular device. If you are opening a file which is to reside on a private volume, you must specify a disc drive on which one of the volumes in the home volume set resides.

The forms*,**vcname*, and***volname* are used only if you are opening a file which is to reside on a private volume.

If * is specified, the file is allocated to any of the volumes of the home volume set.

If ***vcname* (volume class name) is specified, *vcname* must be a member of the home volume set. The file then is allocated to any of the volumes within the volume class.

If ***volname* (volume name) is specified, *volname* must be a member of the home volume set. The file then is allocated to that volume.

dsdevice — Any of the forms may be used to reference files on a remote computer by preceding the device or volume specification with *dsdevice #*.

outpriority — The output priority for spooled device files. This is a value between 1 (lowest priority) and 13 (highest priority).

numcopies — Number of copies of file for spooled output device files. Maximum is 127.

numrec

Maximum number of logical records. For fixed and undefined length files the maximum value allowed for this field is 2,147,483,647. However, the maximum sectors per file is 2,097,120 based on the maximum of 65535 sectors per extent, 32 extents maximum. Thus the actual maximum number of records will be limited by blocksize (determined by record size and blockfactor). An approximate practical limit for numrec is 2,097,119 for variable and undefined files, and 267,382,000 for fixed length files.

The file system uses these values to compute other characteristics of the file as well. Therefore, the values (or default values specified on the :FILE command may be valid within their respective fields, but may cause overflow errors in the computation of internally needed file specifications.

See the *Intrinsics Manual* for a discussion on calculating file space.

numextents

Maximum number of disc extents. This is a value from 1 to 32.

initalloc

Number of extents to be initially allocated to the file at the time is it opened. This is a value from 1 to 32.

filecode

Code indicating a specially-formatted file. This code is recorded in the file label and is available to processes accessing the file through the FGETINFO intrinsic. For this parameter, any user can specify a positive integer ranging from 0 to 1023. Certain integers have particular HP-defined meanings, as follows:

Mnemonic	Integer	Meaning
	−400	IMAGE root file.
	−401	IMAGE data set.
	−402	IMAGE file for DS information.
USL	1024	USL file.
BASD	1025	BASIC data file.
BASP	1026	BASIC program file.
BASFP	1027	BASIC fast program file.
RL	1028	RL file.
PROG	1029	Program file.
SL	1031	SL file.
VFORM	1035	VIEW formsfile.
VFAST	1036	VIEW fast forms file.
VREF	1037	VIEW reformat file.
XLSAV	1040	Cross Loader ASCII file (SAVE).
XLBIN	1041	Cross Loader relocated binary file.
XLDSP	1042	Cross Loader ASCII file (DESPLAY).
EDITQ	1050	Edit KEEPQ file (non-COBOL).
EDTCQ	1051	Edit KEEPQ file (COBOL).
EDTCT	1052	Edit TEXT file (COBOL).
RJEPN	1060	RJE punch file.
QPROC	1070	QUERY procedure file.
	1071	QUERY work file.
	1072	QUERY work file.

Mnemonic	Integer	Meaning
KSAMK	1080	KSAM key file.
GRAPH	1083	GRAPH specification file.
SD	1084	Self-Describing file.
LOG	1090	User Logging logfile.
OPTLF	1130	On-line performance tool logfile.

Default is 0.

RIO

If RIO is specified, a relative I/O file is created. The record length parameter will implicitly be changed to fixed record length. RIO is a special file access method supported by COBOL II.

Note that "RIO" and "NORIO" specifications affect only the physical characteristics of the file. If "NOBUF" is specified in the :FILE command, the file will be accessed in non-RIO mode; otherwise RIO access is used with RIO files. Note that "NOBUF" access is provided for special operations on RIO files such as replicating an RIO file. "NOBUF" is not normally used by the RIO user. See the *Intrinsics Manual* for a discussion of relative I/O.

NORIO

A non-relative I/O file is created.

STD

A standard MPE Disc file.

MSG

Specification of a MSG (Message) file allows communication between any set of processes. Acts like a FIFO (first in, first out) queue where records are read from the start of the file and logically deleted and/or are appended to the end of file.

CIR

Acts as normal sequential file until full. When full, the first physical block will be deleted when the next record is written, and remaining blocks will be logically shifted to front of file. Cannot be simultaneously accessed by readers and writers.

SYNTAX FOR ACCESS

```
[;NOCCTL]          [;NOMULTI]
[;CCTL]            [;MULTI]
                   [;GMULTI]

[;ACC=   {IN} ]              [;NOMR]
         {OUT} [             [;MR]
         {UPDATE}
         {OUTKEEP}
         {APPEND}            [;WAIT]
         {INOUT}             [NOWAIT]

[;BUF[= [numbuffers]]]
[;NOBUF]

[;EXC]
[;SHR]
[;EAR]
[;SEMI]

[;NOLABEL]
[;LABEL[=[void][,[type][,[expdate][,seq]]]]]

[;FORMS=formsmsg]

[;NOLOCK]
[;LOCK]
[;NOCOPY]
[;COPY]
```

ACCESS PARAMETERS

NOCCTL	Indicates that carriage control characters are not being specified in writes to the file
CCTL	Indicates that carriage control characters are being supplied in writes to the file.
IN	Read-access only permitted to the file.
OUT	Write-access only permitted to the file.
OUTKEEP	Write-access only permitted to file, except that previous data is not deleted.
APPEND	Append-access only permitted to file. Is invalid for variable length files. (An error of this type will be detected at FOPEN time, FS ERROR 40.)
UPDATE	Any type of access is permitted to the file.

INOUT	File permits input/output access. (Any file intrinsic except FUPDATE can be issued against the file.)
numbuffers	Number of buffers to be allocated for the file. This is an integer from 1 to 16. *numbuffers* is ignored for terminals. The default is 2 buffers.
NOBUF	Specifies that no buffers are allocated for the file.
EXC	Exclusive access. After the file is opened no other accessors are permitted. For message and circular files, EXC means 1 writer and 1 reader.
SHR	Share access. After file is opened other accessors are permitted.
EAR	Exclusive for writer, allows multiple readers. If a message file, changed to SEMI.
SEMI	Same as EAR, i.e., no difference in execution and both set a 2 in exclusive access field. For message files, allows exclusive reader, multiple writers. If not a message file, SEMI acts like EAR (exclusive writer, multiple readers.)
NOMULTI	Prohibits sharing files in MULTI mode.
MULTI	Concurrent accessors of the file may regard the file as if no buffering is taking place. Access-control information is shared by the accessor. Accessors must exist in same job/session.
GMULTI	Same as MULTI except allows accessors to be in different jobs/sessions.
NOMR	Multi-record access to file is not permitted.
MR	Multi-record access to file is permitted.
WAIT	I/O requests to the file must be complete before control is returned to the program.
NOWAIT	Control is returned to the program as soon as I/O requests are queued by MPE. The program does not have to wait for the physical I/O to be complete before resuming execution.
NOLABEL	Specifies that this is not a labeled tape.
LABEL	Specifies that this is a labeled tape.
volid	Six alphanumeric characters identifying a labeled magnetic tape volume. Optional when file opened for output; *volid* for at least the first volume must be specified when file opened for input. Non-printing characters can not be used.
type	Three characters that specify label type information, as follows: ANS ANSI-standard label. IBM IBM-standard label.

expdate — Month/day/year, written in the format mm/dd/yy. This specifies the expiration date of the file, or the date after which information contained in the file is no longer useful. The file can be overwritten after this date.

seg — Up to four numeric characters that specify the position of the file relative to other files on the tape, or one of the following:

- 0 – Causes a search of all volumes until the file is found.
- ADDF – The tape will be positioned so as to add a new file on the end of the volume (or last volume in a multi-volume set). Not to be used for the first file on a volume set.
- NEXT – Tape will be positioned at next file on the tape. If this is the first open for the file, then NEXT will position the file on the tape. If this is not the first open for the file and rewind occurred on the previous close, then the position will remain at the beginning of the previous file.

formsmsg — A message that can be used for such purposes as telling the Console Operator what type of paper to use in the line printer. This message will be displayed on the console and must be verified by the operator before the user can access the file. The message itself is a string of not more than 49 characters terminated with a period.

NOLOCK — Does not allow dynamic locking/unlocking of file through the FLOCK/FUNLOCK intrinsics.

LOCK — Allows dynamic locking/unlocking through FLOCK/FUNLOCK intrinsics.

COPY — For message files only. Allows MSG files to be either copied (logical data record read) or replicated (block read and write completely duplicating file) to another file.

NOCOPY — Default. The MSG file will be accessed in its nature mode, i.e., as a MSG file.

SYNTAX FOR DISPOSITION

```
[;DEL ]
[;TEMP]
[;SAVE]
```

If none of these parameters are supplied, the disposition of the file is as it was when opened, or as specified by the close.

DISPOSITION PARAMETERS

DEL	The file is deleted when closed.
SAVE	The file is saved in the permanent file domain when closed.
TEMP	The file is saved in the job/session temporary domain when closed.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

Allows you to change the specifications for files, including the devices on which they reside, at the time you run your program, overriding specifications supplied to the FOPEN intrinsic. The :FILE command remains in effect for the entire job or session unless revoked by the :RESET command or superseded by another :FILE command.

To use the :FILE command for a file, you must open that file with a *formal file designator,* the name by which your program recognizes the file. In addition, the *formal file designator* provides a way for commands and code outside your program to reference the file.

NOTE

When you are programming in a language other than SPL, and do not yourself write the FOPEN intrinsic calls for files used by your program, the :FILE command is the only way you can control or change the programmatic file specifications.

IMPLICIT :FILE COMMANDS FOR SUBSYSTEMS

When an actual file designator appears as a command parameter, it is automatically equated to a formal file designator used within the subsystem by an implicit :FILE command issued by the command executor. For instance, within the FORTRAN compiler the formal file designator for the *textfile* input is FTNTEXT. When you specify a file named ALSFILE for *textfile* input as shown below,

```
:FORTRAN ALSFILE
```

MPE implicitly issues the following :FILE command, invisible to yourself:

```
:FILE FTNTEXT=ALSFILE
```

You cannot back reference any of the formal file designators associated with the command as actual file designators, i.e., DO NOT use the formal file designators FTNTEXT, FTNUSL, FTNLIST, FTNMAST or FTNNEW as actual file names. Therefore, the use of FTNTEXT as a file name;

```
:FORTRAN*FTNTEXT
```

is invalid because the implicit :FILE command

```
:FILE FTNTEXT=*FTNTEXT
```

issued by the FORTRAN compiler will then back reference itself.

The following is an example of a correct use of the **formaldesignator,* in this case specifying a file on magnetic tape used as a source file during a FORTRAN compilation:

```
:FILE SOURCE=TAPE1,OLD;DEV=TAPE; REC=-80
:FORTRAN*SOURCE
```

Implicitly, the command executor issues the following :FILE command, back referencing your previous :FILE command:

```
:FILE FTNTEXT=*SOURCE
```

Implicit :FILE commands, like explicit :FILE commands, cancel any previous :FILE commands that reference the same formal file designators.

EXAMPLE

A program references two files by the file names (*formaldesignators*) SOURCE and DEST, but you wish to use two existing disc files INX and OUTX as the actual files for the program. This is done by:

```
:FILE SOURCE=INX
:FILE DEST=OUTX
:RUN MYPROG
```

Now you wish to run the program using input from a card reader which has 80-byte records and has the device class name CARD:

```
:FILE SOURCE,OLD;REC=-80;DEV=CARD
```

Output is to be sent to a new file FILEX with 64-word fixed-length records, blocked two records per block in ASCII code. FILEX is limited to 800 records among 10 extents, two of which are to be immediately allocated. The file is to be permanently saved when MYPROG closes it. Enter:

```
:FILE DEST=FILEX,NEW;REC=64,2,F,ASCII;DISC=800,10,2;SAVE
:RUN MYPROG
```

Note that the file equation only modifies those items specified. All other attributes used will come from the FOPEN for DEST or the FOPEN defaults.

ADDITIONAL DISCUSSION

Intrinsics Reference Manual
Using Files Manual

Compiles, prepares, and executes a FORTRAN program.

SYNTAX

```
:FORTGO [textfile][,[listfile][,[masterfile][,[newfile]
```

PARAMETERS

textfile Actual designator of input file from which source program is read. Can be any ASCII input file. Formal designator is FTNTEXT. Default is $STDIN.

listfile Actual designator of file on which program listing is written. Can be any ASCII output file. Formal designator is FTNLIST. Default is $STDLIST.

masterfile Actual designator of master file which is merged against *textfile* to product composite source. Can be any ASCII input file. Formal designator is FTNMAST. Default is for no file to be read; input is read from *textfile.*

newfile Actual designator of merged *textfile* and *masterfile.* Can be any ASCII output file. Formal designator is FTNNEW. Default is for no file to be written.

NOTE

The formal file designators used in this command (FTNTEXT, FTNLIST, FTNMAST, FTNNEW) cannot be back referenced as actual file designators in the command parameter list. For further information see OPERATION section of :FILE command.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Compiles, prepares, and allocates/executes a FORTRAN program. If you do not specify a source file, MPE expects your input from your standard job/session input device. If you do not specify *listfile,* MPE writes your listing to your standard job/session list device. This command creates a temporary user subprogram library (USL) file ($NEWPASS) that is passed directly from the compiler to the Segmenter and cannot be accessed. A temporary program file is created during preparation and can be accessed under the file name $OLDPASS.

EXAMPLE

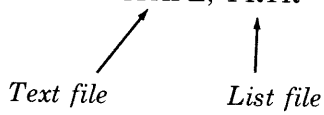
To compile, prepare, and execute a FORTRAN program entered from the disc file SOURCE and transmit the resulting program listing to the disc file LISTFL, enter:

```
:FORTGO SOURCE,LISTFL
```

To enter your source input from a device other than your standard input device, and/or direct the listing to a device other than your standard listing device, simply name the input and listing files as command parameters:

```
:FILE MTAPE;DEV=TAPE
:FILE PRTR;DEV=FASTLP

:FORTGO *MTAPE,*PRTR
```

Identifies MTAPE as a magnetic tape file.
Identifies PRTR as a special line printer file (FASTLP class name).
Compiles from MTAPE, writes listing to PRTR, prepares, and executes.



ADDITIONAL DISCUSSION

FORTRAN Reference Manual.

Compiles and prepares a FORTRAN program.

SYNTAX

:FORTPREP [*textfile*][,[*progfile*][,[*listfile*][,[*masterfile*][,*newfile*]]]]

PARAMETERS

textfile Actual designator of input file from which source program is read. Can be any ASCII input file. Formal designator is FTNTEXT. Default is $STDIN.

progfile Actual designator of program file on which prepared program segments are written. Can be any binary output file with *filecode* of PROG or 1029. If entered, this parameter must indicate a file created in one of two ways:

1. With the MPE:BUILD command using a *filecode* parameter of PROG or 1029.

NOTE

A program file is limited to only one extent. Thus, the *numextents* parameter of the :BUILD command must be 1.

2. By specifying a non-existent file in the *progfile* parameter, in which case a job/session temporary file of the correct size and type is created.

Default: $NEWPASS is assigned.

listfile Actual designator of file on which program listing is written. Can be any ASCII output file. Formal designator is FTNLIST. Default is $STDLIST.

masterfile Actual designator of master file which is merged against *textfile* to produce composite source. Can be any ASCII input file. Formal designator is FTNMAST. Default is that the master file is not read; input is read from *textfile*.

newfile Actual designator of merged *textfile* and *masterfile*. Can be any ASCII output file. Formal designator is FTNNEW. Default is that no file is written.

NOTE

The formal file designators used in this command (FTNTEXT, FTNPROG, FTNLIST, FTNMAST, FTNNEW) cannot be back referenced as actual file designators in the command parameter list. For further information, See OPERATION section of :FILE command.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Compiles and prepares a FORTRAN program onto a program file on disc. If you do not specify a source file, MPE expects your input from your current job/session input device. If you create the program file prior to compilation, you must assign a *filecode* of PROG or 1029. If you do not specify *listfile*, MPE sends the output to your current list device. The user subprogram library (USL) file created during compilation is a temporary file passed directly to the preparation mechanism; you can access it under the name $OLDPASS only if the program file is not prepared into the file named $NEWPASS. The program file is also opened as a temporary file.

EXAMPLE

To compile and prepare a FORTRAN program entered through the current input device, onto the standard default file $NEWPASS, with the listing printed on the current list device, enter:

```
:FORTPREP
```

If the next command is one to execute the program, the file $NEWPASS is referenced in the execute command under the name $OLDPASS.

To compile and prepare a FORTRAN source program from a text file named TEXTX into a program file named PROGX, with the reulting listing sent to the list file LISTX, enter:

```
:FORTPREP TEXTX,PROGX,LISTX
```

Text file *Program file* *List file*

The :FORTPREP command combines the compilation and preparation steps. It is equivalent to:

Text file *USL file* *List file*

```
:FORTRAN TEXTX,$NEWPASS,LISTX
:PREP $OLDPASS,PROGX
```

USL file *Program file*

ADDITIONAL DISCUSSION

FORTRAN Reference Manual.

Compiles a FORTRAN program.

SYNTAX

```
:FORTRAN [textfile][,[uslfile][,[listfile][,[masterfile][,newfile]]]]
```

PARAMETERS

textfile Actual designator of input file from which source program is read. Can be any ASCII input file. Formal designator is FTNTEXT. Default is $STDIN.

uslfile Actual designator of user subprogram library (USL) file on which object program is written. Can be any binary output file with *filecode* of USL or 1024. Formal designator is FTNUSL. If entered, this parameter must indicate a file created previously in one of four ways:

1. By saving a USL file (with the MPE:SAVE command) created by a previous compilation where the default value was used for the *uslfile* parameter.
2. By building the USL with the Segmenter –BUILDUSL command. (See the *MPE Segmenter Reference Manual.*)
3. By creating a new USL file with the MPE:BUILD command with a *filecode* of USL or 1024.
4. By specifying a non-existent *uslfile* parameter, thereby creating a permanent file of the correct size and type.

Default: $NEWPASS is assigned.

listfile Actual designator of file on which program listing is written. Can be any ASCII output file. Formal designator is FTNLIST. Default is $STDLIST.

masterfile Actual designator of master file which is merged against *textfile* to produce composite source. Can be any ASCII input file. Formal designator is FTNMAST. Default is that the master file is not read; input is read from the *textfile*.

newfile Actual designator of merged *textfile* and *masterfile*. Can be any ASCII output file. Formal designator is FTNNEW. Default is that no file is written.

NOTE

The formal file designators used in this command (FTNTEXT, FTNUSL, FTNLIST, FTNMAST, FTNNEW) cannot be back referenced as actual file designators in the command parameter list. For further information see OPERATION section of :FILE command.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Compiles FORTRAN program onto a USL file on disc. If you do not specify *textfile,* MPE expects your input from your current input device. If you create the USL prior to compilation, you must assign a *filecode* of USL or 1024. If you do not specify *listfile,* MPE sends the listing to the current list device.

EXAMPLE

To compile a FORTRAN program entered from your current input device into an object program in the USL file $NEWPASS, and write the listing to your current list device, enter:

```
:FORTRAN
```

If the next command is one to prepare an object program, $NEWPASS can be passed to that command as an input file named $OLDPASS.

The following example compiles a program and creates a USL file.

```
FORTRAN MYSOURCE,MYUSL,MYLIST        Compiles program, creates USL file.
```

Text file *USL file* *List file*

To compile a FORTRAN program, creating the USL file with the :BUILD command, enter:

```
:BUILD OBJECT:CODE=USL
:FORTRAN SOURCE,OBJECT,LISTFL
```

(You must specify the CODE parameter as USL if you choose to create a USL file with the :BUILD command.)

Text file *USL file* *List file*

ADDITIONAL DISCUSSION

FORTRAN Reference Manual.

Releases a global Resource Identification Number (RIN).

SYNTAX

:FREERIN *rin*

PARAMETERS

rin The RIN to be released. Must be a number from 1 to the configured maximum. (REQUIRED PARAMETER).

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		NO

OPERATION

A Resource Identification Number is used to manage a resource shared between two or more jobs or sessions so that only one job or session at a time can access that resource.

The user acquires a RIN from the system by entering the :GETRIN command. When all users are finished with the RIN, the users who acquired it returns it to the system by entering the :FREERIN command. To free a RIN, you must be the original owner of that RIN, i.e., the user who actually issued the GETRIN command that allocated the RIN and assigned it a password.

EXAMPLE

To release RIN 1, enter:

```
:FREERIN 1
```

ADDITIONAL DISCUSSION

Intrinsics Reference Manual.

:GETLOG

Establishes a logging identifier on the system.

SYNTAX

:GETLOG *logid*; LOG=*logfile* {,DISC / ,TAPE} [;PASS=*password*]

PARAMETERS

logid — The logging identifier to be established. Contains from 1 to 8 alphanumeric characters, beginning with a letter.
(REQUIRED PARAMETER)

logfile — The name of the file to receive data from the logging procedure. Must be specified as residing on DISC or TAPE. Contains from 1 to 8 alphanumeric characters beginning with a letter.
(REQUIRED PARAMETER)

password — Logging identifier password, optionally assigned by the creator for protection against illegal use of his identifier. Contains from 1 to 8 alphanumeric characters, beginning with a letter.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

:GETLOG specifies the logging identifier and an optional password by entering the logid and other information specified in the command into the directory of logging identifiers.

The creator of the logging identifier can allow other users access to his logging identifier by notifying them of the identifier and password. Note that if a password is specified it will be required whenever the logging process is accessed. Users accessing the logging system with this identifier must supply the identifier and passwork in the OPENLOG intrinsic.

There cannot be two logids with the same name on the system at the same time. The LISTLOG command can be used to find out what logids currently exist.

EXAMPLE

To create the logging identifier FINANCE and associate it with the tape logfile NEWDATA, enter:

```
:GETLOG FINANCE;LOG=NEWDATA, TAPE
```

ADDITIONAL DISCUSSION

Console Operator's Guide.

:GETRIN

Acquires global Resource Identification Number (RIN) and assigns password to it.

SYNTAX

:GETRIN *rinpassword*

PARAMETERS

rinpassword Password required in the intrinsic that locks the RIN. This is a string of up to eight alphanumeric characters, beginning with a letter. (REQUIRED PARAMETER)

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

The :GETRIN command acquires a global RIN from the MPE RIN pool typically during a session. You must also assign an arbitrary password for the RIN that aids in restricting its use to proper users only. You can then give this RIN and the associated password to cooperating users so that it can be locked and unlocked by them. For instructions on how to lock and unlock the RIN, and how to pass a RIN and its password as intrinsic parameters, see the MPE *Intrinsics Reference Manual.*

Users who know the RIN and its password can use it in their programs (in jobs or sessions) until the user who acquired the RIN releases it via the :FREERIN command. The RIN acquired is always a positive integer unique within MPE. The total number of RINs MPE can allocate is specified when the system is configured and cannot exceed 1024. If all RINs currently available are acquired by other users when you request one, MPE rejects the request and issues the message:

RIN TABLE FULL

In this case, you must wait until one of the RINs becomes available or see your system manager about raising the maximum number of RINs that can be assigned.

EXAMPLE

To acquire a global RIN and assign to it the password MYRIN, enter:

```
:GETRIN MYRIN
```

MPE responds with the RIN number assigned, as for example:

```
RIN: 1
```

ADDITIONAL DISCUSSION

MPE *Intrinsics Reference Manual.*

:HELLO

Initiates an interactive session.

SYNTAX

```
:HELLO  [sessionname,]username[/userpass].acctname[/acctpass]
        [,groupname[/grouppass]]

        [;TERM=termtype]
        [;TIME=cpusecs]

                BS
                CS
        [;PRI={    }]
                DS
                ES

        ;INPRI=inputpriority
        [                     ]
        ;HIPRI
```

PARAMETERS

sessionname — Arbitrary name used in conjunction with *username* and *acctname* parameters to form a session identity. Contains from 1 to 8 alphanumeric characters, beginning with a letter. Default is that no session name is assigned.

username — A user name, established by Account Manager, that allows you to log on under this account. Contains from 1 to 8 alphanumeric characters, beginning with a letter. (REQUIRED PARAMETER)

userpass — User password, optionally assigned by Account Manager. Contains from 1 to 8 alphanumeric characters, beginning with a letter.

acctname — name of account, as established by Account Manager. Contains from 1 to 8 alphanumeric characters, beginning with a letter. The *acctname* parameter must be preceded by a period. (REQUIRED PARAMETER)

acctpass — Account password, optionally assigned by System Manager. Contains from 1 to 8 alphanumeric characters, beginning with a letter.

groupname — Name of group to be used for local file domain and CPU time charges. Established by Account Manager. Contains from 1 to 8 alphanumeric characters, beginning with a letter. Default is your home group if you are assigned one by Account Manager. (Optional if you have a home group, required if home group not assigned.)

grouppass — (Required if assigned and you are logging on under other than home group. Not required if logging on under home group.) Group password, optionally assigned by Account Manager. Contains from 1 to 8 alphanumeric characters, beginning with a letter.

termtype Type of terminal used for input. MPE uses this parameter to determine device-dependent characteristics such as delay factors for carriage returns. Must be a number from 0 to 16. See Appendix A for a list of terminals.

Default: For hardwired terminals, determined by System Supervisor during system configuration. No default for terminals that are not hardwired. Required parameter to insure correct listings if your terminal is not hardwired or if your terminal is not the default *termtype*.

cpusecs Maximum CPU time that session can use, entered in seconds. When limit is reached, session is aborted. Must be a value from 1 to 32767. To specify no limit, enter question mark ("?") or "UNLIM", or omit parameter. Default is no limit.

BS, CS, DS, ES Execution priority class. BS is highest priority, ES is lowest. If you specify a priority that exceeds the highest permitted for your account or user name by the system, MPE assigns the highest priority possible below BS. Default is CS.

NOTE

DS and ES are used primarily for batch jobs; their use for sessions is discouraged.

inputpriority Relative input priority used in checking against access restrictions imposed by the *jobfence,* if one exists. Takes effect at log-on time. Must be a value from 1 (lowest priority) to 13 (highest priority). If a value is specified that is less than or equal to current *jobfence* set by Console Operator, session is denied access. Default is 8.

HIPRI Request for maximum session-selection input priority, causing session to be scheduled regardless of current *jobfence* or execution limit for sessions. This parameter can be specified only by users with System Manager or System Supervisor capability. (If not, the system tries to log you on with INPRI=13.) Default is the current *jobfence* and execution limit.

USE

Available	In Session?	YES
	In Job?	NO
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Log-on message aborted)

OPERATION

The :HELLO command initiates an interactive session and must be entered from a terminal; no other device can be used for this command. You must supply both a valid user name and account name in your log-on commands. Otherwise, MPE rejects your log-on attempt and prints an error message to that effect. If your log-on attempt is accepted, however, MPE verifies this by printing specific log-on information and prompting you for your next MPE command. Such information appears in the following example, where a user has logged on under the user name MAC and the account name TECHPUBS:

```
:HELLO MAC.TECHPUBS
HP3000 / MPE IV C.00.00 TUE, OCT 7, 1980, 2:09 PM
:  <------  MPE prompts for next command
```

Sometimes a welcome message from the Console Operator will appear following MPEs verification of your log-on.

The session number assigned by MPE always uniquely identifies your session to MPE and to other users. MPE assigns such numbers to sessions in sequential order as they are logged on.

In certain instances, you may be required to furnish information in addition to the user and account names in your :HELLO command or :( ) command log on. This information includes:

- Group Name.
- One or more passwords.
- Terminal type code.

GROUP NAME

The group you select at log on for your local file domain is known as your *log-on group.* If your account manager has associated a home group with your user name, and if you desire this group as a log-on group, you need not specify this — MPE automatically assigns the home group as your log-on group when you log on. But if you desire to use some other group as log-on group, you must specify that group's name in your log-on command in this way:

```
:HELLO MAC.TECHPUBS,YGROUP
                    |
                Group name
```

NOTE

If your user name is not related to a home group, you must enter a group name in your :HELLO command or :( ) command log on — otherwise, your log-on attempt is rejected.

Once you log on, if the normal (default) file security provisions of MPE are in force, you have unlimited access to all files in your log-on and home groups. Furthermore, you can read files and execute programs stored in the public group of your account and the public group of the System account. You cannot, however, access any other files in any way. Further information about files and file security can be found in the MPE *Files Reference Manual.*

PASSWORDS

To enhance the security of an account, and to prevent unauthorized accumulation of charges against the account, the System Manager may assign a *password* to the account. Similarly, an Account Manager may associate passwords with the user names and groups belonging to his account. If you are using an account, user name, or group (other than your home group) that has a password, you must furnish that password when you log on. Include the password after the name of the protected entity, separated from that name by a slash mark (/). (In MPE, slash marks denote security.) For instance, if the group XGROUP requires a password and if you wish this group as your log-on group, you could enter the password in this fashion:

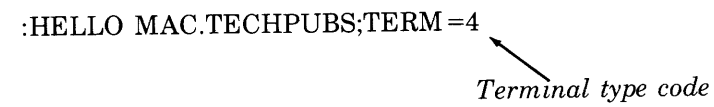
NOTE

When specifying your home group as your log-on group, you need not enter a password even if that group has such a password.

Sometimes, when logging on to the system, it is more convenient to have MPE prompt you for any required passwords. You do this by omitting the passwords from the log-on command. Then, if you log on at a hard-copy terminal, the command is printed in the normal way and MPE prompts you for the password, then turns echo off. Echo is turned on after all passwords are read.

TERMINAL TYPES

MPE must be able to determine, for its own use, certain characteristics about the terminal in order to conduct the session, such as input and output speed. If you log on using a different type of hardwired terminal (or one that is not hardwired) than the type the System Manager has configured, you must specify your terminal type when you log on. For instance, if you are logging on at an HP 2600A, but the default type at your site is a HP 2640A you would enter:



EXAMPLE

To start a session named ALPHA, with the username MAC, accountname TECHPUBS, group XGROUP, grouppass XPASS, enter:

```
:HELLO ALPHA,MAC.TECHPUBS,XBROUP,XPASS
HP3000 / MPE IV C.000.00 THU, OCT 9, 1980, 12:15 PM
:          (MPE displays colon to prompt for command)
```

NOTE

When the :HELLO command is issued from a session it will not start another session if:

1. Spaces precede the command.
2. The command is executed using a :REDO command.
3. The command is issued in a user defined command (UDC).

Accesses the Help subsystem.

SYNTAX

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

PARAMETERS

HELP — Displays information for the HELP command. Treated the same as entering :HELP with any other command (because HELP is a command).

tablecontents — Any of several items for which information may be obtained. These items are:

SESSIONS
JOBS
PROGRAMS
FILES
MANAGE
OPERATOR
SPOOLER
UTILITY

Thus, if you want information on running sessions, you would enter:

:HELP SESSIONS

command — Any MPE command. MPE displays the command name and syntax. In addition, a list of keywords for that command is displayed. Keywords for all commands except the :FILE command are:

PARMS
OPERATION
EXAMPLE

where

PARMS	lists all parameters of the specified command.
OPERATION	describes the use of the specified command.
EXAMPLE	displays an example showing usage of the specified command.

The :FILE command has the additional keywords:

NAME
DEVICE
ACCESS
DISP

Entering :HELP command,ALL causes MPE to display all information for that command (syntax, parameters, operation, and example).

Keyword

One of the keywords described under the *command* parameter. The *keyword* parameter can be entered with a command, as in :HELP HELLO,PARMS. In this case, PARMS is a *keyword* and must be separated from the command with a comma. The *keyword* parameter also can be entered alone if you are running the Help subsystem in subsystem mode and have accessed a command, then received the prompt (>) from Help.

For example,

:HELP DEBUG

:DEBUG (Privileged Mode capability required)

Invokes the Debug facility.

SYNTAX

:DEBUG

KEYWORDS: PARMS, OPERATION

>PARMS (*keyword* entered by itself in response to prompt)

ALL

Displays entire table of contents for :HELP command. If entered with a command, as in

:HELP SPL,ALL

displays all information for that command.

EXIT

Exits Help subsystem.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		YES (Aborts)

OPERATION

The :HELP command accesses the Help subsystem. If entered with no parameters, as in

:HELP

Help enters the subsystem mode, displays a table of contents, a greater than (>) prompt, and awaits your input. Entering any table of contents item such as SESSIONS produces a listing of all the commands used in running sessions. Entering any command name produces the syntax for that command and a list of keywords. Entering a keyword, such as PARMS produces a listing of all items for that keyword (all parameters in this case). Entering EXIT or pressing the BREAK key terminates the Help subsystem. Entering Control Y stops the display. Entering Control S stops the display and entering Control Q resumes the display (useful if the screen is full as when using a CRT terminal).

Entering carriage return in subsystem mode causes MPE to display information up to the next keyword or command. For example, after entering STORE, MPE displays STORE syntax, the keyword list (PARMS, OPERATION, EXAMPLE) and prompts (>). Entering carriage return again causes MPE to display all Parms information for the :STORE command. Entering carriage return once more (after the prompt) causes MPE to display all OPERATION information for the :STORE command, and so on. (This is similar to page turning through a manual.)

Entering :HELP with a parameter causes Help to enter the immediate mode. Information pertaining to that parameter is displayed immediately. For example,

:HELP DEBUG

causes Help to display:

:DEBUG (Privileged Mode capability required)

Invokes the MPE Debug facility.

SYNTAX

:DEBUG

KEYWORDS: PARMS, OPERATION

EXAMPLE

To obtain information concerning MPE utility functions, enter:

```
:HELP UTILITY
```

MPE displays:

Utility functions. Following are the commands used.

PTAPE	SHOWTIME
SETMSG	TELL
SPEED	TELLOP

You can use any command name as a keyword.

Used to control the execution sequence of a job.

SYNTAX

```
:IF     [(]logexpr[)] THEN
          .
          .
          .
ELSE
          .
          .
          .
ENDIF
```

PARAMETERS

logexpr Logical expression, consisting of relational operators or combinations of relational operators. The relational operators allowed are:

>	greater than
<	less than
>=	greater than or equal
<=	less than or equal
=	equal
<>	not equal

Compound logical expressions can be formed using the AND, OR, and NOT logical operators and nesting with parentheses.

The allowed operands are: JCW or CIERROR; any Job Control Word values saved in a name with the :SETJCW command; OK, WARN, FATAL, or SYSTEM (as defined under the :SETJCW command); or integer constants.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	NO
Breakable?		NO

OPERATION

The :IF command begins an IF block, and is used with an :ENDIF command and, optionally, an :ELSE command to control the execution of a job. An IF block consists of all the commands after the IF up to, but not including, the next :ELSE or :ENDIF statement that has the same nesting level as the IF statement. Another similar block can follow the :ELSE statement. Nesting of blocks is allowed to 15 levels. The :ENDIF statement ends the IF block.

The logical expression is evaluated and, if true, the IF block is executed; if false, the ELSE block (if one exists) is executed.

EXAMPLE

```
:RUN UPDATE1                                    (Run first data base update)
:IF JCW<WARN THEN                               (Continue if OK so far)
:     SETJCW UPDATE1JCW:=JCW                    (Save that return code)
:     RUN UPDATE2                               (Run second data base update)
:     IF JCW<WARN THEN                          (Continue if still OK)
:          SETJCW UPDATE2JCW:=JCW               (Save the second return code)
:          RUN DBSTATUS                         (Report current status of DB)
:          IF (UPDATE1JCW < 50) AND (UPDATE2JCW < 50) THEN
:               RUN DAILYRPT
:          ENDIF
:          IF ((UPDATE1JCW = 1) OR (UPDATE2JCW =1)) THEN
:               RUN WEEKLYRPT
:          ENDIF
:     ELSE                                      (Second update program failed)
:          RUN FIXUP                            (Repair data base and report)
:     ENDIF
:ENDIF
```

Enables use of the Inquiry and Development Facility (IDF) portion of IML/3000 if authorized in the IML configuration file.

SYNTAX

IML [ENHANCE = { 0 | 1 | 2 | 3 }][;BLANKS]

PARAMETERS

ENHANCE

Controls the display of characters. If this parameter is not used, the HP 264x terminal will display 3270 normal intensity characters as 264x half-bright characters, and 3270 enhanced characters as 264x full-bright (normal) characters. Using option "0" will give the same result as not using ENHANCE at all.

The following table shows the display enhancements provided by options 0, 1, 2, 3.

Option	3270 Normal Intensity	3270 High Intensity
0	264x half bright	264x normal
1	264x normal	264x underline
2	264x normal	264x inverse video
3	264x inverse video	264x normal

BLANKS

With use of this parameter the IDF will not convert leading blanks into "nulls," thus transmitting the field to the host with leading blanks. If this parameter is not used, IDF will convert leading blank input characters in an unprotected field into "nulls" and thus not transmit the leading blanks to the host. This parameter may be used to right-justify numeric input data.

USE

Available	In Session?	YES
	In Job?	NO
	In Break?	NO
	Programmatically?	NO
Breakable?		NO

OPERATION

A terminal user or a device which the IML manager has allowed access to in the configuration file can use this command to start the IML Inquiry and Development Facility (IDF). The user responds to IML prompts to name a configuration file and a specific IBM device number for emulation.

EXAMPLE

If the user wishes to have 3270 High Intensity displayed as underline and does not want to convert leading blanks to zeros, (s)he would enter:

```
:IML ENHANCE=1; BLANKS
HP32229A.00.00 IML/3000 (c) HEWLETT-PACKARD 1979
CONFIGURATION FILE NAME? REMOTE1.PUB.SYS
DEVICE NUMBER? 26
```

REMOTE1 names a configuration file that defines a 3270 control unit with attached terminals and printers. DEVICE NUMBER 26 specifies which of the devices attached to the 3270 control unit the user's terminal should emulate.

ADDITIONAL DISCUSSION

IML Manual.

Initiates a batch job.

SYNTAX

```
:JOB [jobname,]username[/userpass].acctname[/acctpass]
     [,groupname[/grouppass]]

     [;TIME=cpusecs]

              BS
              CS
     [;PRI={     }]
              DS
              ES

      ;INPRI=inputpriority
     [                      ]
      ;HIPRI

     [;RESTART]
     [;OUTCLASS=[device][,outputpriority][,numcopies]]
```

PARAMETERS

jobname — Arbitrary name used with *username* and *acctname* parameters to form a job identity. Contains from 1 to 8 alphanumeric characters, beginning with a letter. Default is no job name is assigned.

username — User name, established by the Account Manager. Contains from 1 to 8 alphanumeric characters, beginning with a letter. (REQUIRED PARAMETER)

userpass — User password, optionally assigned by Account Manager. Contains from 1 to 8 alphanumeric characters, beginning with a letter.

acctname — Account name, established by System Manager. Contains from 1 to 8 alphanumeric characters, beginning with a letter. The *acctname* parameter must be preceded by a period. (REQUIRED PARAMETER)

acctpass — Account password, optionally assigned by System Manager. Contains from 1 to 8 alphanumeric characters, beginning with a letter.

groupname — (Optional if home group assigned, required if not.) Name of group used for local file domain and for CPU time charges, as established by Account Manager. Contains from 1 to 8 alphanumeric characters, beginning with a letter. Default is home group if one is assigned.

grouppass — (Optional parameter if accessing MPE under home group; required parameter if assigned and accessing MPE under other than home group) Group password, optionally assigned by Account Manager. Contains from 1 to 8 alphanumeric characters, beginning with a letter.

cpusecs — Maximum CPU time allowed job, in seconds. When this limit is reached, job is aborted. Must be value from 1 to 32767. To specify no limit, enter question mark or omit this limit. Default is system configured job limit.

BS,CS,DS,ES — Execution priority class. BS is highest priority, ES is lowest. If a priority is specified that exceeds the highest permitted for the account or user name by the system, MPE assigns the highest priority possible below BS. Default is DS, unless CS is specified by System Supervisor with the :JOBPRI command.

inputpriority — Relative input priority used in checking against access restrictions imposed by *jobfence* (if one exists). Takes effect when job is entered. Must be value from 1 (lowest priority) to 13 (highest priority). If value is less than or equal to *jobfence* set by Console Operator, job is deferred. Default is 8.

HIPRI — Request for maximum input priority, causing job to be scheduled regardless of current *jobfence* or execution limit for jobs. Over rides *inputpriority*. You can specify this parameter only if you have System Manager or System Supervisor capability. If not, the system tries to log you on with INPRI=13. Default is 8 or *inputpriority* if this parameter is specified.

RESTART — Request or restart a spooled job interrupted by system termination/restart. Takes effect automatically when system is subsequently restarted with the WARMSTART option.

This parameter applies only to jobs initiated on spooled input devices, and is ignored for other jobs. Default is that spooled jobs are not restarted after system termination/restart.

device — Class name or logical device number of device to receive listing output. Cannot specify a magnetic tape unit. Default is $STDLIST.

NOTE

Non-sharable device (ND) file access attribute is required in order to use this parameter.

outputpriority — Output priority for job list file, if destined for spooled line printer or card punch. Used to select next spooled devicefile (on disc) for output, among all those contending for a specific printer or punch. Must be a value from 1 (lowest priority) to 13 (highest priority). Note that when *outputpriority* is 1, output always is deferred. Thus, you should use an *outputpriority* of 2 or greater if you wish output written from disc.

This parameter applies only to output destined for spooled output devices, and is ignored for other output. Default is 8, if logging enabled; 13 if disabled.

numcopies — Number of copies of job listing to be produced. This parameter applies only when listing is directed to a spooled device, and is ignored in other cases. Must be a value from 1 to 255. Default is 1.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		NO

OPERATION

Initiates a batch job by establishing contact with MPE. When MPE begins the job, it displays the following information on the list device:

PRIORITY – (BS, CS, DS, or ES)

INPRI JOB NUMBER – as assigned by MPE to identify job.

Date and time.

HP 3000 and the MPE Version Level, in the form HP 3000/MPE III.*v.uu.ff* where *v* is the version level (a letter), *uu* is the update level (a number), and *ff* is the fix level (a number).

In the :JOB command as in the :HELLO command, you must always include your user name and account name, which you obtain from your Account Manager. If you omit either of these names or enter them incorrectly, MPE rejects your job and prints an error message on the standard listing device. If your job is accepted, however, MPE begins job processing. If the job is entered through a spooled input device, the job is copies to a disc file and initiated from that file rather than the originating device.

If the standard listing file is a line printer, MPE prints a special header page prior to listing the :JOB command, as shown below. (The Console Operator can disable the printing of this page via the console command :HEADOFF.)

The job number assigned by MPE always uniquely identifies your job to MPE and other users. MPE assigns such numbers in sequential order as jobs are accepted.

Sometimes, the job-acceptance information includes a message from the Console Operator following the standard display. When present, this is the same message output in the log-on information for sessions.

The user and account name combination :JOB constitutes a fully qualified job identity — the minimum information required for job initiation. Under certain circumstances, however, you may need to furnish the following additional information in your :JOB command:

- File Group Name.
- User, Account, and/or Group Passwords.

The cases in which this information is required and the rules for supplying it are the same as for the :HELLO command for sessions, except that:

1. If you omit a required password from the :JOB command, MPE rejects your access attempt without prompting you for the password.

2. When you enter the :JOB command through a device other than a terminal and the standard input device is different than the standard listing device, MPE never echoes passwords entered.

3. When the standard listing device is a line printer, and you do not specify a file group name, central-processor time limit, execution priority, and/or input priority in the :JOB command, the default values assigned by MPE for the omitted parameters appear on the job listing.

The following :JOB command, entered from a card reader, illustrates how to specify the group name MYGROUP and the group password MYPASS:

```
:JOB MAC.TECHPUBS,MYGROUP/MYPASS
```

EXAMPLE

To start a job named BETA, *username* MAC, *accountname* TECHPUBS, *groupname* XGROUP, and *grouppass* XPASS, enter:

```
:JOB BETA,MAC.TECHPUBS,XGROUP/XPASS

PRIORITY=ES; INPRI=8
JOB NUMBER = #J1
TUE, OCT 7, 1980, 2:01 PM
HP3000 / MPE IV.C.00.00
```

Note that subsequent MPE commands must begin with a colon (:), entered in column 1 of the input record.

ADDITIONAL DISCUSSION

Using Files Manual.
Intrinsics Reference Manual.

Lists descriptions of one or more permanent disc files.

SYNTAX

```
                    0
                    1
:LISTF [fileset][,  ][;listfile]
                    2
                   -1
```

PARAMETERS

fileset Specifies the set of files to be listed. Default is @. This parameter is of the form:

filedesignator[.*groupdesignator*[.*acctdesignator*]]

fileset can be entered in any of the following formats and may use wild card characters, in any order, as replacements.

file.group.account	List file named in specified group and account.
file.group	List file named in specified group.
file	List file named.
@.group.account	List all files in specified group and account.
@.@.account	List all files in all groups in specified account.
@.@.@	List all files in system.
@.@	List all files in all groups in log-on account.
@	List all files in log-on group. Default.
@.group	List all files in specified group.
file.@.account	List file named in any group of specified account.

NOTE

The characters @, #, and ? can be used as wild card characters in the *fileset* parameter. These wild card characters have the following meanings:

@— specifies zero or more alphanumeric characters.

— specifies one numeric character.

? — specifies one alphanumeric character.

The characters can be used as follows:

n @	List all files starting with the character *n*.
@*n*	List all files ending with the character *n*.
n @*x*	List all files starting with the character *n* and ending with the character *x*.
n ##..#	List all files starting with the character *n* followed by up to seven digits (useful for listing all EDIT/3000 temporary files).
?*n* @	List all files whose second character is *n*.
n?	List all two-character files starting with the character *n*.
?*n*	List all two-character files ending with the character *n*.

0 Display file name only. Default.

1 Display file name, plus file code, record size (W indicates words, B indicates bytes), record format (F, U, or V), whether ASCII (A) or binary (B) records, whether carriage-control option is taken (C if so), current end-of-file location, and maximum number of records allowed in file. Default is 0.

2 Display file name, file code, record size (W indicates words, B indicates bytes), record format (F, U, or V), whether ASCII (A) or binary (B) records, whether carriage-control option is taken (C if so), current end-of-file location, maximum number of records allowed in file, blocking factor, number of disc sectors in use (including those in use for file label and user headers), number of extents currently allocated, and maximum number of extents allowed. Default is 0.

-1 Display octal listing of file label. This option can be requested only by users with System Manager capability (all files) or Account Manager capability (all files in own account). Default is 0. The first line of the listing is the directory entry for the file being listed. The remainder of the listing is the file label.

listfile Name of output file on which descriptions are written. Automatically specified as new ASCII file with variable-length records closed in temporary domain, user-supplied carriage-control characters (CCTL), OUT access mode, and EXC (exclusive access) option. All other characteristics are same as :FILE command default specifications. Default is $STDLIST.

NOTE

If you specify the -1 option and direct listing output to a non-spooled device (such as a magnetic tape unit) that enters the NOT READY state, the System Directory will be locked down and the system may hang up. Avoid this combination of specifications if possible.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		YES (Aborts)

OPERATION

Lists descriptions of one or more disc files at level of detail you select. You need not have access to a file to list a description of it, however, a file description will not be listed unless the file's home volume set is mounted. A standard user may list level 0, 1, and 2 information for any file in the system. A user with Account Manager capability may list level -1 data for files in his own account. A user with System Manager capability may list -1 data for any file in the system. Doing a :LISTF to the line printer or magnetic tape will print the date and time at the top.

Note that this command applies only to permanent disc files in the system file domain.

The output may appear in the following formats:

Level 0 (Default)

```
:LISTF

FILENAME

APPB        APPC        APPD        APPE        APPF        CONMSG
CONOP1      CONOP4      CONOPS      CONOPEX     CONSG       EXAMPLES
GIMAGE      GIMCOM      GIMDOC      GIMDS       GIMOP       INDEX
K1030955    LINDA       LIST        LOGON       MEMLOGAN    MTS
MTS3000     OPFRONT     PS          REPLY       SCHED       VINIT
```

Level 1

```
:LISTF,1
ACCOUNT=  LEWIS          GROUP=  PUB
FILENAME  CODE  ------------LOGICAL RECORD--------
                      SIZE  TYP         EOF       LIMIT
CONOP1                 80B  FA           10          10
DOALL                  80B  FA            9           9
EXAMPLES               80B  FA           20          20
LIST                   88B  FA           15          15
PROSE                  80B  FAR           5           5
PS          PROG      128W  FB            7           7
REPLY                  80B  FA           30          30
```

See :BUILD or :FILE command for a list of mnemonic codes.

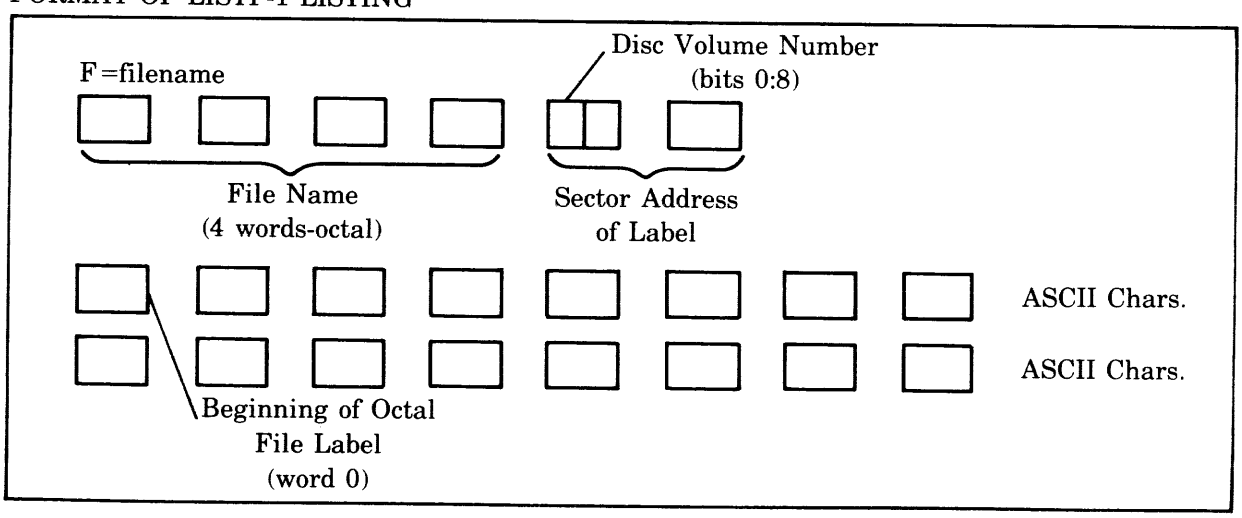
Level 2

```
:LISTF,2
ACCOUNT=  USERS          GROUP=  BARBARA
FILENAME  CODE  -------------LOGICAL RECORD-----------  ----SPACE----
                    SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX
APPB                 88B  FA          414        414  16      162 14 14
APPC                 80B  FA          131        131  16       50  1  1
APPD                 80B  FA          719        719  16      230 16 16
APPE                 80B  FA          118        118  16       45  1  1
APPF                 80B  FA          224        224  16       75 15 15
APPM                 80B  VAM         119        240  16       12  8  8
APPC                 80B  FAO         127        208  16       70  8  8
APPK      KSAM       80B  FA           16        200  16       20  2  7
CONMSG    BASD      308W  FB          348        408   1     1092  7  8
CONOP1    BASD      308W  FB           68        108   1      210  5  8
```

Level-1

```
:LISTF EXAMPLES,-1
F  =  EXAMPLES
042530 040515 050114 042523 001000 033261                    EXAMPLES..6.
042530 040515 050114 042523 050125 041040 020040 020040 EXAMPLESPUB.....
046105 053511 051440 020040 041101 051102 020040 020040 LEWIS...BARB....
020040 020040 020040 020040 020202 004040 000001 116221 ................
116221 116221 000000 000000 002400 000000 000000 000024 ................
000000 000000 147462 016063 002005 177660 001200 002400      2.3........
000017 000017 000000 000024 001000 033261 000000 000000 ..........6.....
000000 000000 000000 000000 000000 000000 000000 000000 ................
000000 000000 000000 000000 000000 000000 000000 000000 ................
000000 000000 000000 000000 000000 000000 000000 000000 ................
000000 000000 000000 000000 000000 000000 000000 000000 ................
000000 000000 000000 000000 000000 000000 000000 000000 ................
```

FORMAT OF LISTF-1 LISTING



KEY FOR FILE TYPE

```
:listf,2
ACCOUNT=  USERS          GROUP=  RUTH
FILENAME  CODE  -------------LOGICAL RECORD----------  ----SPACE----
                   SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX
                         ---
                         123
```

COLUMN 1 = RECORD FORMAT

This can be
- *Fixed* length
- *Variable* length

COLUMN 2 = ASCII/BINARY Option

Indicates whether file is
- *ASCII*
- *BINARY*

COLUMN 3 = KIND of file

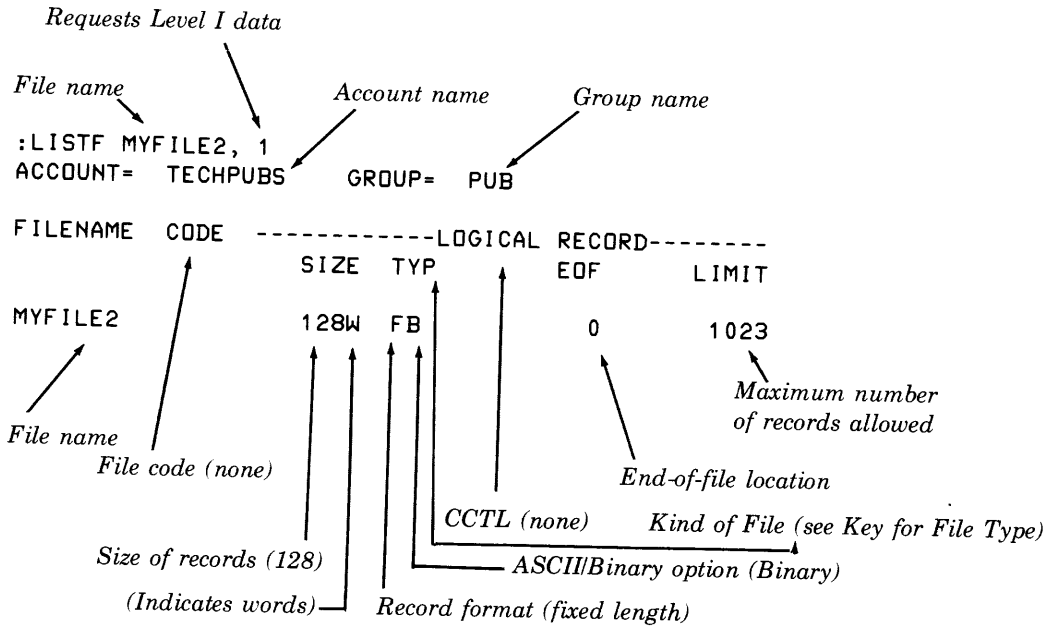
File can be
- Blank
  indicates *standard* file unless *KSAM* appears in code field
- O
  indicates Circular file
- M
  indicates a Message file
- R
  indicates a Relative I/O file

EXAMPLE

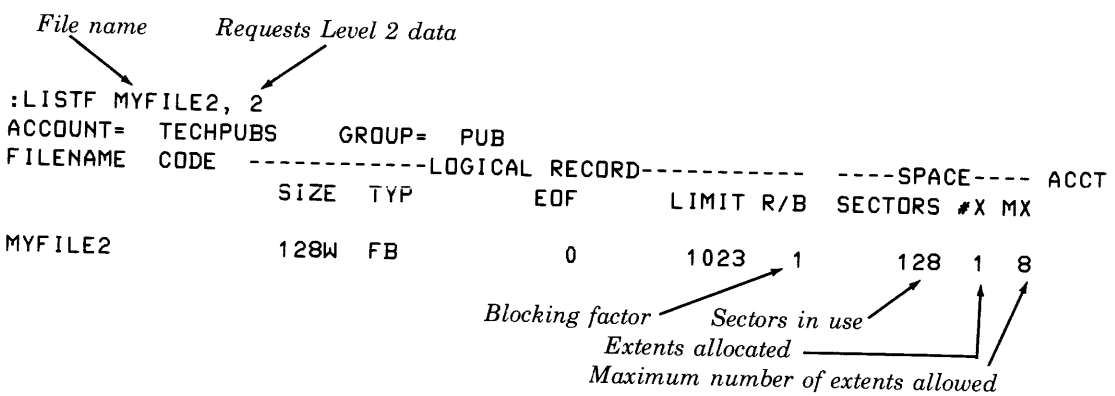
To list the names of all files in your log-on group, enter:

```
:LISTF
```

To display the file code, record size and format, ASCII/binary and carriage-control options, current end-of-file location, and maximum number of records allowed in a file, request level 1 information with the :LISTF command:



To list a greater level of detail, including blocking factor, number of disc sectors in use, number of extents currently allocated and maximum number of extents allowed; use level 2 information option:



To request that the file information be displayed on devices other than the standard listing device, name the desired device in the command:

```
:FILE PRTR;DEV=LP   Equates name PRTR with device class name LP.
:LISTF @.@,2;*PRTR  Directs Level 2 description of all files in all groups of log-on account to
                    PRTR.
```

To list the name, file code, record format, ASCII vs. binary code information, carriage-control option, end-of-file location, and maximum number of records for the file named BASE in the group USERGP in your log-on account, enter:

```
:LISTF BASE.USERGP,1
```

ADDITIONAL DISCUSSION

Using Files Manual.

:LISTLOG

Lists currently active logging identifiers on the system.

SYNTAX

:LISTLOG [*logid*[;PASS]]

PARAMETERS

logid — The specific logging identifier to be verified. Default is to list all currently active logging identifiers on the system.

PASS — Causes the password associated with the logging identifier to be displayed. Can be used only by the creator of the logging identifier.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

Lists the logging identifier specified with its associated creator and logfile. If the logid parameter is not entered, all logging identifiers on the system are displayed with their creators and logfiles. The pass parameter, which can be used only by the creator of the logging identifier specified, causes the password associated with the logging identifier to be listed.

EXAMPLE

To list all logging identifiers on the system, enter:

```
:LISTLOG

LOGID      CREATOR              LOGFILE

FINANCE    DATABASE.FINANCE     NEWDATA
IN LOG     TOM.LAB              TEMP.TOM.LAB     (disc file)
ERRORS     TOM.LAB              OTHERLG          (tape file)
```

ADDITIONAL DISCUSSION

For a discussion of user logging, see the Console Operator's Guide.

:LISTVS

Produces a formatted listing of volume set definition information.

SYNTAX

```
                    0
:LISTVS [vslist][,1][;listfile]
                    2
```

PARAMETERS

vslist Specifies the volume set definitions to be listed. Default is @. This parameter is of the form:

vsdesignator[.*groupdesignator*[.*acctdesignator*]]

vslist can be entered in any of the following formats and may use wild card characters, in any order, as replacements.

vsname.group.account List information for volume set vsname in the specified group and account.

@.group.account List all home volume sets in the specified group and account.

@.@.account List all home volume sets for all groups in the specified account.

@ List all home volume sets in log-on group and account.

@.@ List all home volume sets for all groups in log- on account.

@.@.@ List all home volume sets in system.

vsname.@.account List information for volume set *vsname* for all groups in the specified account.

NOTE

The characters @, #, and ? can be used as wild card characters in the *vslist* parameter. These wild card characters have the following meanings:

@ — specifies zero or more alphanumeric characters.

— specifies one numeric character.

? — specifies one alphanumeric character.

The characters can be used as follows:

n@ List all volume set definitions starting with the character *n*.

@n List all volume set definitions ending with the character *n*.

n@x List all volume set definitions starting with the character *n* and ending with the character *x*.

n##..# List all volume set definitions starting with the character *n* followed by up to seven digits.

?n@ List all volume set definitions whose second character is *n*.

n? List all two-character volume set definitions starting with the character *n*.

?n List all two-character volume set definitions ending with the character *n*.

0 Display the following information:

Volume set name and volume set definition creator.

1 Display the information shown for 0 plus:

Volume names of volume set members and the types of devices on which the volumes reside.

Volume class names and member volume names.

2 Display the information shown for 0 and 1, plus:

Logical device numbers (*ldn's*) and use counts of the devices on which the volume set is mounted. This information is not displayed if the volume set is not mounted; instead, information is displayed indicating if there are sufficient AVAILABLE *ldn's* of the right type to permit the volume set to be mounted. Default is 0.

listfile Actual designator of output file to receive listings. This file must be named in a :FILE command and back referenced as in the following example:

```
:FILE LIST;DEV=LP
:LISTVS @,1;*LIST
```

Default is $STDLIST.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		YES

OPERATION

The :LISTVS command allows you to determine the configuration of a volume set and produce a formatted listing of volume set definition information.

1. Volume set names and volume class names.

2. Volume set member volume names and types.

3. Volume class member volume names and types.

4. Logical device numbers (*ldn's*) and status of the devices on which the volume set is mounted. This information is omitted for any volume set not currently mounted.

The *detail* parameter of the :LISTVS command determines the amount of information displayed. If *detail* is omitted or if a 0 is specified, item 1 above is displayed. If *detail* is 1, items 1 through 3 are displayed. If *detail* 2 is specified, items 1 through 4 are displayed.

To display volume set definition information for your home volume set, enter:

```
:LISTVS,1
```

EXAMPLE

To display volume set definition information for volume sets in your log-on group and account, with the listing sent to a line printer, enter:

```
:FILE LIST;DEV=LP
:LISTVS @,1;*LIST
```

ADDITIONAL DISCUSSION

System Manager/System Supervisor Reference Manual.

:MOUNT

Requests the Console Operator to mount a volume set.

SYNTAX

```
                  *
:MOUNT [          ][.groupname[.acctname]]
        vcsname

        [;GEN=[genindex]]
```

PARAMETERS

*	Specifies the home volume set for the group and account specified, or the log-on group and account if *groupname.acctname* are not specified. Default.
vcsname	Volume class/set name, specifying a previously- defined volume class name or volume set name. Default is *.
groupname.acctname	Specify the group and account under which the volume set was created. Default is log-on group and account.
genindex	A value from −1 to 32767 specifying which generation of the home volume set is to be mounted. A value of −1 indicates that any generation is permitted. If omitted, the system will ignore the generation when attempting to satisfy the :MOUNT request.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	NO
Breakable?		NO

OPERATION

The :MOUNT command allows you to become attached to a specific volume set. In addition, this command allows you to specify that the volume set is to be treated as a home volume set and therefore it is to be bound to the system's accounting structure with respect to a particular group. The :MOUNT command is rejected if there is an operator-initiated :LDISMOUNT command pending for the specified volume set. The requesting job is suspended until the mount is completed or rejected.

In addition, various user commands which cause access to your log-on group's home volume set will initiate mount requests if the volume set is not mounted already. An example of one of these commands (:BUILD) is as follows:

```
:BUILD VFILE;DISC=500,10,1;REC=-80;DEV=VCLASS1
```

PROGRAMMATIC MOUNT REQUEST. You may issue an FOPEN call referencing a file residing on an unmounted volume set; this causes an implicit user-initiated mount request.

An FOPEN mount remains in effect until a corresponding FCLOSE intrinsic call is issued. The programmatic request is used when a single job/session step requires a certain volume set. See the *MPE Intrinsics Reference Manual* for a description of programmatic mounting.

EXAMPLE

To request the Console Operator to mount volume set MYSET, generation index 43, enter:

```
:MOUNT MYSET;GEN=43
```

:MRJE

Initiates execution of the Multileaving Remote Job Entry (MRJE) facility.

SYNTAX

```
:MRJE
```

PARAMETERS

None

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

The :MRJE command is used to initiate execution of the MRJE facility for submitting jobs to, and retrieving output from, a host computer. The host computer may be using either a HASP Version 3.1, HASP-II Version 4.0, or JES2 job entry subsystem.

EXAMPLE

```
:MRJE
```

ADDITIONAL DISCUSSION

MRJE/3000 Reference Manual.

Prepares program from a User Subprogram Library (USL) file onto a program file.

SYNTAX

```
:PREP uslfile,progfile
      [;ZERODB]
      [;PMAP]
      [;MAXDATA=segsize]
      [;STACK=stacksize]
      [;DL=dlsize]
      [;CAP=caplist]
      [;RL=filename]
      [;PATCH=patchsize]
```

PARAMETERS

uslfile — Actual designator of USL file on which program has been compiled. (REQUIRED PARAMETER)

progfile — Actual designator of program file on which prepared program segments are written. Can be any binary output file. This program file must be created in one of two ways:

1. By creating a new file with the MPE :BUILD command with a *filecode* of PROG or 1029, and a *numextents* parameter value of 1.

2. By specifying a non-existent file in the *progfile* parameter, in which case a file of the correct size and type is created. This file is a job temporary file. (REQUIRED PARAMETER)

ZERODB — Request to initialize to zero the initially-defined, user-managed (DL-DB) area, and uninitialized portions of the DB-Q (initial). Default is that these areas are not affected.

PMAP — Request to produce a descriptive listing of the prepared program on file whose formal designator is SEGLIST. If no :FILE command is found referencing SEGLIST, listing is produced on $STDLIST. Default is no listing. See Appendix D for a listing of a prepared program.

segsize — Maximum stack area (Z-DL) size permitted, in words. This parameter is included when it is expected that the size of DL-DB or Z-DB areas will be changed during program execution. Default is that MPE assumes that these areas will not change.

stacksize — Size of initial local data area (Z-Q initial) in stack, in words. (This value, if specified, must be between 511 and 32767 words. This parameter over rides *stacksize* estimated by Segmenter. Default is estimated by MPE Segmenter.

dlsize — DL-DB area to be initially assigned to stack. This area is of interest mainly in programmatic applications. In all cases, the DL-DB area is rounded upward so that the distance from the beginning of the stack data segment to the DB-address is a multiple of 128 words. Default is estimated by the MPE Segmenter.

caplist Capability-class attributes associated with program, specified as two-character mnemonics. If more than one mnemonic is specified, each must be separated from its neighbor by a comma.

The mnemonics are:

IA	Interactive access
BA	Local batch access
PH	Process handling
DS	Data segment management
MR	Multiple resource management
PM	Privileged mode

Note that you can specify only those capabilities that you possess (through assignment by the Account Manager or System Manager). Default is IA, BA (if you possess these capabilities).

filename Actual designator of RL file to be searched to satisfy external references during preparation. This can be any permanent binary file of type RL. It need not belong to log-on group, nor have a reserved, local name. This file yields a single segment that is incorporated into the segments of the program file. See the *MPE Segmenter Reference Manual* for a discussion of RL files. Default is that no library is searched.

patchsize Specifies size of patch area. This size will apply to all segments within the program file.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

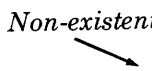
OPERATION

The :PREP command prepares a compiled source program for execution. Unless you prepare the program onto a previously created program file, this command will create a program file of the appropriate format for you in the job/session temporary file domain. In fact, it is recommended that you specify a non-existent program file in the :PREP command. This allows MPE to create a file of optimum size and characteristics. (See example 1)

A compiled program is prepared by searching a relocatable library (RL) to satisfy references to external procedures required by the program. When the program is prepared, such procedures are linked to the program in the resulting program file. To use a relocatable library (RL) via the :PREP command, the user requires read and lock capability.

EXAMPLE

To prepare a program from the USL file USLX to the program file PROGX, enter:




```
:PREP USLX,PROGX      Prepares program into program file PROGX.
:SAVE PROGX           Saves program file.
```

You can create a program file in the permanent file domain by using the :BUILD command. When you do this, you must be sure to specify a *filecode* of PROG (or 1029) for this file, and to limit the file to one extent; program files are not allowed to span more than one extent. The following :BUILD command creates such a file, which is then used by the :PREP command:

```
:BUILD PFILE;CODE=PROG;DISC=,1
:PREP UFILE,PFILE
```

To prepare a program from the USL file named USEFILE and store it in a program file named PROGFILE, list the prepared program, assign a *stacksize* of 511 words, and assign batch-access capability only for the program, enter:

```
:PREP USEFILE,PROGFILE;PMAP;STACK=511;CAP=BA
```

ADDITIONAL DISCUSSION

Using Files Manual.

:PREPRUN

Prepares and executes a compiled program.

SYNTAX

```
:PREPRUN uslfile[,entrypoint]
         [;NOPRIV]
         [;PMAP]
         [;DEBUG]
         [;LMAP]
         [;ZERODB]
         [;MAXDATA=segsize]
         [;PARM=parameternum]
         [;STACK=stacksize]
         [;DL=dlsize]

                G
         [;LIB= {P}]
                S

         [;CAP=caplist]
         [;RL=filename]
         [;NOCB]
```

PARAMETERS

uslfile — Actual designator of USL file on which program has been compiled. (REQUIRED PARAMETER)

entrypoint — Program entry point where execution is to begin. May be primary entry point of program, or any secondary entry point in program's outer block. Default is primary entry point.

NOPRIV — Declaration that program segments will be placed in non-privileged (user) mode. This parameter is intended for programs prepared with privileged mode capability. Normally, program segments containing privileged instructions are executed in privileged mode only if program was prepared with privileged mode (PM) capability class. (A program containing legally compiled privileged code, placed in non-privileged mode, may abort when an attempt is made to execute it.) If NOPRIV is specified, all segments are placed in non-privileged mode. (Library segments are not affected because their mode is determined independently.) Default is segments of privileged mode program will remain in privileged mode.

PMAP — Request to produce a descriptive listing of the prepared program on file whose formal designator is SEGLIST. If no :FILE command referencing SEGLIST is found, listing is produced on current list device. Default is no listing. See Appendix D for listing of prepared and loaded program.

DEBUG — A request to set a breakpoint on the first executable instruction of the program. Default is no breakpoint is set.

LMAP — Request to produce a descriptive listing of the allocated (loaded) program on file whose formal designator is LOADLIST. If no :FILE command referencing LOADLIST is found, listing is produced on $STDLIST. Default is no listing.

ZERODB — Request to initialize to zero initially-defined, user-managed (DL-DB) area, and unitialized portions of the DB-Q (initial) area. Default is these areas are not affected.

segsize — Maximum stack area (Z-DL) area permitted, in words. This parameter is included if it is expected that sizes of DL-DB or Z-DB areas will be changed during program execution. Default is MPE assumes that these areas will not be changed.

parameternum — Value that can be passed to program for control or other purposes. When program is executed, this value can be retrieved from address Q(initial)-4, where Q(initial) is Q- address for outer block of program. Value can be octal number or signed or unsigned decimal number. Default is Q(initial)0-4 address is filled with zeros.

stacksize — Size of local data area (Z-Q(initial)) in stack, in words. If specified, this value must exceed 511 words. This parameter overrides *stacksize* estimated by MPE Segmenter. Default is estimated by MPE Segmenter.

dlsize — DL-DB area to be initially assigned to stack. This area is of interest mainly to programmatic applications. In all cases, the DL-DB area is rounded upward so that the distance from the beginning of the stack data segment to the DB- address is a multiple of 128 words. Default is estimated by MPE Segmenter.

G — Search segmented procedure libraries of the program file's group and account in the following order: group library, account library, system library. Default is S.

P — Search segmented procedure libraries of the program file's group and account in the following order: account library, system library. Default is S.

S — Search system library for external references to segmented procedures. Default.

caplist — Capability-class attributes associated with the program, specified as two-character mnemonics. If more than one mnemonic is specified, each must be separated from its neighbor by a comma.

The mnemonics are:

IA	Interactive access
BA	Local batch access
PH	Process handling
DS	Data segment management
MR	Multiple resource management
PM	Privileged mode

You can only specify those attributes that you possess through assignment by the Account Manager or the System Manager. Default is IA and BA (if you possess these capabilities).

filename Actual designator of RL file to be searched to satisfy external references during preparation of the program. This can be any permanent file of type RL. It need not belong to log-on group nor does it require a reserved, local name. This file yields a single segment that is incorporated into the segments of the program file. See the *MPE Segmenter Reference Manual* for a description of RL files. Default is no library is searched.

NOCB Request that file system not use stack segment (PCBX) for its control blocks, even if sufficient space is available. This will permit you to expand your stack (via the DLSIZE or ZSIZE intrinsics) to the maximum possible limit at a later time, but will cause the File Management System to operate more slowly for this program.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Prepares and executes a program compiled in a User Subprogram Library (USL). Command prepares a temporary program file and then executes the program in that file. This command permits searching both relocatable (RL) and segmented (SL) libraries to satisfy external references.

EXAMPLE

To prepare and execute a program on the USL file USEF, with no special parameters declared, enter:

```
:PREPRUN USEF
```

To obtain a descriptive listing of the prepared program, and a listing of the allocated (loaded) program, enter:

```
        Requests prepared program listing
                    |
                    v
:PREPRUN XUSL;PMAP;LMAP
                      \
                       v
        Requests loaded program listing
```

To prepare and execute a program on the USL file UBASE, beginning execution at the entry point RESTART, declaring a *stacksize* of 800 words, and specifying that the library LIBA will be searched to satisfy external references, enter:

```
:PREPRUN UBASE,RESTART:STACK=800;RL=LIBA
```

ADDITIONAL DISCUSSION

Using Files Manual.

Reads paper tape without X-OFF control.

SYNTAX

:PTAPE *filename*

PARAMETERS

filename Name of existing ASCII file on disc, to which input data is to be written from a paper tape. Normally, this is a file with variable-length records; the record size specified must be large enough to contain the longest paper tape record. (REQUIRED PARAMETER)

USE

Available	In Session?	YES
	In Job?	NO
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

Permits programs to read input originating from a terminal coupled to a paper tape reader/punch, when the tape does not include the X-OFF control character punched on the tape. (The X-OFF character, when encountered on a tape, turns the tape-reading mechanism off. This character normally appears following the carriage-return/line-feed character combination used to delimit each record.) The :PTAPE command transfers all data from the tape to an ASCII file (named in the *filename* parameter) on disc. This operation stores all records onto the disc file in the order read from the tape, but deletes all carriage-return and line-feed characters. (If the tape contains X-OFF characters, the operation also deletes these. The X-OFF characters are not ignored by the tape reader, however, and so you must restart the reader after each X-OFF is read.) The input terminates when the CONTROL Y character is encountered on the tape or entered from the terminal, returning control to the keyboard. The input data now can be read from the disc file by any program.

Note that the :PTAPE command is required ONLY for input from HP tape readers associated with terminal keyboards.

You cannot use the :PTAPE command to copy tapes containing more than 32767 bytes of information. If this is attempted, MPE reads only the first 32767 bytes on the tape and then terminates the operation.

EXAMPLE

To copy input from a paper tape onto a disc file named TAPEFILE, create an appropriate file (with the MPE :BUILD command) and then enter the following :PTAPE command:

```
:BUILD TAPEFILE;REC=-80,3,V,ASCII
:PTAPE TAPEFILE
```

Deletes file from system.

SYNTAX

:PURGE *filereference*[,TEMP]

PARAMETERS

filereference Actual file designator of file to be deleted, in this format:

filename [/*lockword*] [.*groupname* [.*acctname*]]

To delete the file, you must have write (W) access to it. (REQUIRED PARAMETER)

TEMP Indicates that the file is a temporary file in the job/session temporary file domain. You must enter this parameter to delete a temporary file, otherwise the appropriate domain is not searched for the file. Default is that a permanent file is assumed.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

Deletes a disc file from the system. Applies only to files on disc. If the volume set on which the file resides is not mounted, this command will generate an implicit mount request.

If the file does not exist in the domain specified (temporary or permanent (permanent is default)), the following message is displayed:

FILE *filename* NOT FOUND

NOTE

To purge both a KSAM data file and its associated key file, you must enter the :PURGE command twice, once for each file.

EXAMPLE

To delete the permanent file PFILE, enter:

```
:PURGE PFILE
```

To delete the temporary file TFILE, enter:

```
:PURGE TFILE,TEMP
```

ADDITIONAL DISCUSSION

Using Files Manual.

Displays all pending console: REPLY messages.

SYNTAX

```
:RECALL
```

PARAMETERS

None

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		YES (Aborts)

EXAMPLES

To display all pending console messages which require a :REPLY response, enter:

```
:RECALL
REPLY(S) PENDING:
010:05/ #J19/15/LDEV # FOR "L00576" ON TAPE 1600 (NUM)?
```

If there are REPLYs pending, the test of the request(s) will appear at your terminal, as shown above. If there are *no* REPLYs pending the following message will appear at the console:

```
:RECALL
NO REPLIES PENDING
```

ADDITIONAL DISCUSSION

Console Operator's Guide

:REDO

Allows you to edit a command entry.

SYNTAX

```
:REDO
```

PARAMETERS

None

USE

Available	In Session?	YES
	In Job?	NO
	In Break?	YES
	Programmatically?	NO
Breakable?		YES (Aborts)

OPERATION

The :REDO command allows you to correct certain kinds of errors in an incorrect command entry or to change a correct command entry eliminating the need for re-entering the command in its entirety. The :REDO command only applies to the last command entered. When the :REDO command is entered, MPE enters a mode similar to the Editor and displays the command to be modified.

To modify the command output by MPE, position the cursor (using the space bar on the terminal) under the character(s) to be modified, then enter one of the following sub-commands:

D - Delete. Deletes the character above the cursor. If D is repeated, each character above each D is deleted.

I - Insert. Inserts one or more characters immediately preceding the character above the cursor. The D and I sub-commands can be used in conjunction to delete characters, then insert new characters.

R - Replace. Replaces the characters above the cursor with new characters. If one character is entered, the character above the cursor is replaced; if two characters are entered, two characters, (the character above the cursor and the character to the right) are replaced; and so forth for additional characters. R is the default sub-command.

U - Undo. Cancels the effect of the previous D, I, or R sub-command. Entering a U, carriage return, then another U cancels all previous sub-commands for this :REDO command and restores the line being corrected to its original form.

EXAMPLE

```
:FILE A;REC=..G;DISC=10000,16,1;SAVE
             ↑
:ERR nnn EXPECTED F, V, OR U

:REDO                                (Request to enter command string)
:FILE A;REC=,,G;DISC=10000,16,1;SAVE  (MPE displays command)
             RF                      (Replace G with F)
:FILE A;REC=,,F;DISC=10000,16,1;SAVE  (Corrected command displayed)
```

Note that the letter G can be replaced by F without entering the R sub-command (R is the default sub-command). For example,

```
:FILE A;REC=,,G;DISC=10000,16,1;SAVE  (MPE displays command)
             F                       (Replace G with F)
```

Temporarily suspends all security provisions for a file.

SYNTAX

:RELEASE *filereference*

PARAMETERS

filereference Actual designator of file whose security provisions are to be suspended, written in this format:

filename [/*lockword*] [.*groupname* [.*acctname*]]

If the file has a lockword, it must be specified. (REQUIRED PARAMETER)

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

Temporarily suspends security provisions for file at all (file, group, and account) levels (except for privileged files) allowing any user in system unlimited access to the file. Suspension remains in effect until a :SECURE command is entered for this file to restore its previous security provisions. Suspension remains valid after job termination, or system failure followed by coldload or reload. (It can be negated, however, if system is reloaded from a :SYSDUMP tape created before :RELEASE command was entered.) Note that this command can be used only for permanent files on disc whose labels identify you as the creator of the file. In addition, the :RELEASE command will fail if the group's home volume set is not mounted. When the normal (default) MPE security provisions are in effect, the file must be in your log-on account and must belong to your log-on or home group.

This command will not cause a volume set to be mounted. If the file's home volume set is not mounted at the time this command is issued, the file will not be found and thus cannot be released.

The :RELEASE command does not affect the file's lockword, if any. It also does not modify the file security settings recorded in the system; it merely bypasses them temporarily.

EXAMPLE

To release all security provisions for the file named FILE1, enter:

```
:RELEASE FILE1
```

Removes a logging identifier from the system.

SYNTAX

:RELLOG *logid*

PARAMETERS

logid The logging identifier to be removed from the system. (REQUIRED PARAMETER)

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

The :RELLOG command removes a logging identifier from the system by deleting it from the directory of logging identifiers. This command may be issued only by the user who created the logging identifier specified in the logic parameter. After :RELLOG is issued, programs containing the removed logging identifier will not be allowed to access the logging system.

EXAMPLE

```
:RELLOG DATALOG
```

ADDITIONAL DISCUSSION

Console Operator's Guide.

:REMOTE

Used for transmitting commands between a local HP computer and a remote HP computer in a DS/3000 Network environment.

SYNTAX

:REMOTE [*dslinenumber*][*mpecommand*]

PARAMETERS

dslinenumber The *dslinenumber* that was returned by DS/3000 when the communications line was opened. (A *dslinenumber* is returned in response to (1) a :DSLINE command, or (2) a :REMOTE HELLO command that includes the DSLINE= parameter.) A *dslinenumber* is required only if more than one communications line is open simultaneously. If *dslinenumber* is omitted, the line which you most recently opened is referenced.

The form of the line number returned by DS/3000 is:

DS LINE NUMBER = #L3 (where 3 is the *dslinenumber)*

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		Not breakable if *MPE command* is not specified. Is breakable if *MPE command* you specify is breakable.

OPERATION

To use the :REMOTE command, you must first establish a local session. Also, the communications line must have been opened with the :DSCONTROL console operator command.

After the line is open, you can gain logical access to the line with the :DSLINE command. Next, you should establish a remote session with the :REMOTE HELLO command, or the sequence of commands, :REMOTE (to transfer control to the remote MPE command Interpreter) followed by the standard MPE #HELLO command. Note that the remote system prompts with a pound sign (#).

Once a remote session is established, the :REMOTE command can be used to execute any valid MPE command on the remote computer system. For multiple commands, the :REMOTE command (without the command parameter) allows the input command stream to be executed by the Command Interpreter on the remote system. (See EXAMPLE.) To return to the local system, respond to the pound sign (#) prompt with a colon (:).

Variations of the :REMOTE command may be used in multi-node DS/3000 networks with three or more nodes.

EXAMPLE

```
:HELLO MIKE.OSE
ENTER USER PASSWORD:

HP3000 / MPE IV C.00.00.  FRI, AUG 22, 1980,  3:07 PM
               PIRANHA
FILE MIKE;DEV=TAPE
FILE LP;DEV=LP;CCTL
FILE FLP;DEV=FASTLP;CCTL
FILE SLP;DEV=SLOWLP;CCTL
FILE TAPE;DEV=TAPE
FILE BACKUP;DEV=TAPE;REC=-80,16,F,ASCII;ACC=APPEND
COMMENT  &DBDON'T FORGET TO RUN PNOTE!
:DSLINE RADAR
DS LINE NUMBER = #L3.
:REMOTE HELLO MIKE.OSE
USER PASSWORD (PASS)?

HP3000 / MPE IV C.00.00.  FRI, AUG 22, 1980,  3:10 PM
               RADAR
   STROLD DONE LAST NIGHT 8/21/80....
:REMOTE LISTF
NO FILES FOUND IN FILE-SET (CIWARN 431)
:LISTF

FILENAME

CHEC002C  DOCVSTR   F07F3450  F09FMISC  F51F3427  F51FPTCT
F59FALOC  F59FMISC  F64F3450  F76F3450  F76FNEWP  F85FPTCT
FCAF3450  FHEF002C  J00JXXXX  J51JXXXX  JONUDC    JXXJXXXX
JYYJXXXX  MASKKPR   MASKMEMO  MASKRESP  NOTEDATA  NOTEFILE
PHEP002C  SRBNOTES  TGENMP    TGENMS    UDC       WORKKPR
WORKMEMO  WORKNOTE  WORKQUAL  WORKRESP  WORKVSTR  XBJACK
XCAP      XOP

:DSLINE MY
DS LINE NUMBER = #L5.
:REMOTE HELLO MIKE.OSE
ENTER USER PASSWORD:

HP3000 / MPE IV C.00.00.  AUG 22, 1980,  3:11 PM
              &DBMELLO YELLO

       SYSTEM WAS RELOADED 8/19/80
:REMOTE LISTF

FILENAME

IOTEST      PNOTE       TEST        XCAP        XOP
```

```
:REMOTE

#SHOWME
USER: #S158,MIKE.OSE,PAIVINEN      (NOT IN BREAK)
MPE VERSION: HP32002X.08.02
CURRENT: FRI, AUG 22, 1980,  3:11 PM
LOGON:   FRI, AUG 22, 1980,  3:11 PM
CPU SECONDS: 1          CONNECT MINUTES: 1
$STDIN LDEV: 72          $STDLIST LDEV: 72
                     &DBMELLO YELLO

       SYSTEM WAS RELOADED 8/19/80...
#LISTF

FILENAME

IOTEST       PNOTE        TEST         XCAP        XOP

#:

:REMOTE 3SHOWME
USER: #S286,MIKE.OSE,PAIVINEN      (NOT IN BREAK)
MPE VERSION: HP32002C.00.00
CURRENT: FRI, AUG 22, 1980,  3:11 PM
LOGON:   FRI, AUG 22, 1980,  3:10 PM
CPU SECONDS: 1          CONNECT MINUTES: 2
$STDIN LDEV: 81          $STDLIST LDEV: 81
                    RADAR
   STROLD DONE LAST NIGHT 8/21/80....
:REMOTE 3 LISTF
NO FILES FOUND IN FILE-SET-(CIWARN 431)
:REMOTE SHOWME
USER: #S158,MIKE.OSE,PAIVINEN      (NOT IN BREAK)
MPE VERSION: HP3200C.00.00
CURRENT: FRI, AUG 22, 1980,  3:12PM
LOGON:   FRI, AUG 22, 1980,  3:11 PM
CPU SECONDS: 1          CONNECT MINUTES: 2
$STDIN LDEV: 72          $STDLIST LDEV: 72
                     &DBMELLO YELLO

       SYSTEM WAS RELOADED 8/19/80...
:DSLINE @;CLOSE
ABORT THE REMOTE SESSION ON #L5?YES
SESSION ABORTED BY SYSTEM MANAGEMENT
CPU=1. CONNECT=2. FRI, AUG 22, 1980,  3:12 PM
ABORT THE REMOTE SESSION ON #L3?YES
SESSION ABORTED BY SYSTEM MANAGEMENT
CPU=1. CONNECT=3. FRI, AUG 22, 1980,  3:12 PM
2 DS LINES WERE CLOSED.
```

ADDITIONAL DISCUSSION

DS/3000 Reference Manual

:REMOTE HELLO

Establishes communication between a local computer and a remote computer, and initiates a session on the remote system.

SYNTAX

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

PARAMETERS

sessionname — Arbitrary name used in conjunction with *username* and *acctname* parameters to form a session identity. Contains from 1 to 8 alphanumeric characters, beginning with a letter. Default is no session name is assigned.

username — A user name, established by an Account Manager on a remote HP 3000, that allows you to log on the remote system under this account. Contains from 1 to 8 alphanumeric characters, beginning with a letter. (REQUIRED PARAMETER)

userpass — User password, optionally assigned by an Account Manager on a remote HP 3000. Contains from 1 to 8 alphanumeric characters, beginning with a letter.

acctname — Name of account, as established by System Manager of the remote system. Contains from 1 to 8 alphanumeric characters, beginning with a letter. The *acctname* parameter must be preceded by a period. (REQUIRED PARAMETER)

acctpass — Account password, optionally assigned by the System Manager of the remote system. Contains from 1 to 8 alphanumeric characters, beginning with a letter.

groupname — Name of a remote system group to be used for file domain and CPU time charges. Established by Account Manager. Contains from 1 to 8 alphanumeric characters, beginning with a letter. Default is your remote system home group if you are assigned one by Account Manager. (Optional if you have a home group, required if home group not assigned.)

grouppass — (Required if assigned and you are logging on under other than home group. Not required if logging on under home group.) Group password, optionally assigned by the Account Manager on a remote system. Contains from 1 to 8 alphanumeric characters, beginning with a letter.

termtype Termtype, if present, has no meaning and is ignored. The term type for your local session implicitly defines your log-on terminal type for a remote session.

cpusecs Maximum remote CPU time that your remote session can use, entered in seconds. When limit is reached, session is aborted. Must be a value from 1 to 32767. To specify no limit, enter question mark (?) or UNLIM, or omit parameter. Default is no limit.

BS, CS, DS, ES Execution priority class. BS is highest priority, ES is lowest. If you specify a priority that exceeds the highest permitted for your account or user name by the system, MPE assigns the highest priority possible below BS. Default is CS.

NOTE

DS and ES are used primarily for batch jobs; their use for sessions is discouraged.

inputpriority Relative input priority used in checking against access restrictions imposed by the remote system's *jobfence*, if one exists. Takes effect at log-on time. Must be a value from 1 (lowest priority) to 13 (highest priority). If a value is specified that is less than or equal to current *jobfence* set by the remote Console Operator, session is denied access. Default is 8.

HIPRI Request for maximum session-selection input priority, causing your remote session to be scheduled regardless of current *jobfence* or execution limit for sessions. This parameter can be specified only by users with System Manager or System Supervisor capability on the remote system. If the user does not have OP or SM capability, the system tries to log the user on with INPRI=13. Default is the current *jobfence* and execution limit.

dsdevice The device class name or logical device number assigned to the DS/3000 communications driver IODS0 during system configuration. This parameter specifies what physical line you wish to use. Optional parameter if a line already is open; otherwise it is required.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		YES (Log-on message aborted)

OPERATION

The :REMOTE HELLO command is used to obtain access to a physical communications line and establish a session on a remote HP 3000.

The line to the remote computer must have been opened by the Console Operator before a :REMOTE HELLO command can be completed successfully. In addition, the line must be opened from your local session. This can be done by including the optional *dsdevice* parameter in the :REMOTE HELLO command or by first issuing a :DSLINE *xxx* command and then omitting the *dsdevice* parameter (refer to EXAMPLE).

When you open a line by including the *dsdevice* parameter, the line remains open only for the duration of your remote session. When the remote session terminates, the line is automatically closed.

After a session exists on a remote HP 3000, you can issue :REMOTE commands that include any valid MPE command. It is also possible to issue a :REMOTE command (without parameters) to transfer control to the remote Command Interpreter. See the :REMOTE command.

EXAMPLE

To establish the communications line FONZ and log on to System B from System A, you could enter either of the following examples:

```
:HELLO NANCY.USERS
USER PASSWORD (PASS)?
HP3000 / MPE IV C.00.00. WED, SEP 3, 1980, 1:51 PM

:REMOTE HELLO NANCY.SAYLOR;DSLINE=FONZ
HP3000 / MPE IV C.00.00. WED, SEP 3, 1980, 1:52 PM

DS LINE NUMBER = #L5.
:


:HELLO NANCY.USERS
USER PASSWORD (PASS)?
HP3000 / MPE IV C.00.00. WED, SEP 3, 1980, 1:55 PM

:DSLINE FONZ
DS LINE NUMBER = #L3.
:REMOTE HELLO NANCY.SAYLOR
HP3000 / MPE IV C.00.00 WED, SEP 3, 1980, 1:56 PM
:
```

ADDITIONAL DISCUSSION

DS/3000 Reference Manual.

:RENAME

Changes identity (file name, lockword, and/or group name) of disc file.

SYNTAX

:RENAME *oldfilereference,newfilereference*[,TEMP]

PARAMETERS

oldfilereference Current name of file, written in the following format:

filename [/*lockword*] [.*groupname* [.*acctname*]]

If *acctname* is specified, it must be that of your log-on account. (REQUIRED PARAMETER)

newfilereference New name of file, in the same format as *oldfilereference*. If *acctname* is specified, it must be that of your log-on account. If *groupname* is specified, it must be one to which you have SAVE access. If *acctname* and/or *groupname* are omitted, the log-on account and/or group are assumed. (REQUIRED PARAMETER)

TEMP Indicates that old file was, and new file will be, in temporary job/session file domain. Default is permanent file is assumed.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

Changes system file identification for permanent or temporary disc file. Effectively removes file with old name from system and creates another file with identical contents and new name. This command can be used to change the name of a file, to move a file from one group to another (by specifying a different *groupname* in the *newfilereference* parameter), or to change the lockword. You cannot rename files across accounts.

The volume set must be mounted for the group in which this file is to be renamed or this command will fail. In addition, renaming is confined to a volume set (i.e., a file cannot be renamed across volume sets).

Note that in order to rename a file you must be the creator of the file and the system must be able to open the file exclusively. (Exclusive implies lock access).

EXAMPLE

To change the name of a temporary file from OLDFILE to NEWFILE, moving it from your log-on group to the group named NEWG, enter:

```
:RENAME OLDFILE,NEWFILE.NEWG,TEMP
```

To change the lockword of the permanent file FILE2 from LOCKA to LOCKB, enter:

```
:RENAME FILE2/LOCKA,FILE2/LOCK
```

To transfer a file from one group to another within the same account, use the :RENAME command, simply naming the new group in the second parameter. (You must be the creator of the file to use this command.) For example,

```
:RENAME MYFILE.GROUP1,MYFILE.GROUP2
```

Old group *New group*

NOTE

> To use :RENAME in this way, you must have SAVE access to the group named in the second parameter (GROUP2 in the previous example). In addition, both groups must be in the system domain or must both reside on the same volume set (renaming of files across volume sets is not allowed).

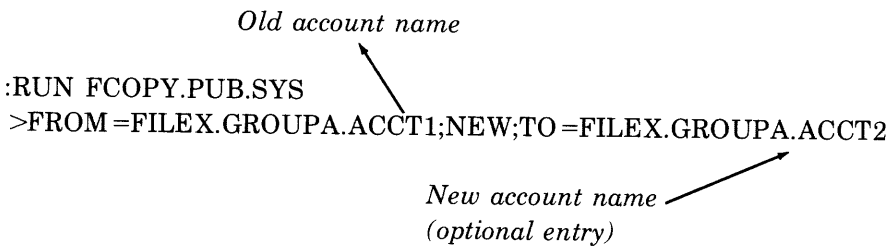
To transfer a file from one account to another, proceed as follows:

1. Log on to the computer under the account presently containing the file.
2. Enter the :RELEASE command to temporarily suspend any MPE security provisions covering the file. For example:

   ```
   :RELEASE FILEX          File name
   ```

   You can only enter this command if you are the creator of the file.
3. Log off from this account, and log on under the account to which the file is to be transferred.

4. Run the File Copier Subsystem (FCOPY) to copy the file from the old account into this account. For example,



NOTE

The renaming of files across volume sets is not allowed (this would require that the operation physically transfer the file between different volume sets).

A copy of FILEX now exists under GROUPA of ACCT2; the original FILEX still exists under GROUPA of ACCT1. (ACCT2 must be your log-on account.)

ADDITIONAL DISCUSSION

Using Files Manual.

:REPORT

Displays accounting information for logon account and group.

SYNTAX

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

PARAMETERS

groupset Specifies the accounts and groups for which information is to be listed. The permissible entries and the capability required (shown in parentheses) are as follows: Account Manager is shown as AM; System Manager as SM.

groupname Reports on the specified group in the log-on account. Standard user can only specify his log-on group.

@ Reports on all groups in the log-on account (AM or SM).

groupname.acctname Reports on the specified group in the specified account (SM).

@.acctname Reports on all groups in the specified account (SM).

@.@ Reports on all groups in all accounts (SM).

groupname.@ Reports on specified group in any account.

Default: For standard user: his log-on group.
For Account Manager: All groups in his own account.
For System Manager: All groups in all accounts.

NOTE

The characters @, #, and ? can be used as wild card characters in any order in the *groupset* parameter. These wild card characters have the following meanings:

@ — specifies zero or more alphanumeric characters.

— specifies one numeric character.

? — specifies one alphanumeric character.

The characters can be used as follows:

n@ Report on all groups starting with the character *n*.

@n Report on all groups ending with the character *n*.

n@x Report on all groups starting with the character *n* and ending with the character *x*.

	n##..#	Report on all groups starting with the character *n* followed by up to seven digits.
	?*n*@	Report on all groups whose second character is *n*.
	n?	Report on all two-character groups starting with *n*.
	?*n*	Report on all two-character groups ending with *n*.
listfile	Actual file designator of output file to which information is to be written. Output may be re-directed with a :FILE back reference as follows: :FILE LIST1,DEV =LP :REPORT ,*LIST1 Default is $STDLIST.	
volset	Directs the reporting of accounting information from the specified volume set.	

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		YES (Aborts)

OPERATION

Displays the total resource usage logged against accounts and groups, and the limits on those resources. For the standard user, this information covers his own group only; an Account Manager may specify all groups in his account; and the System Manager may specify all groups in all accounts.

The information includes usage counts and limits for permanent file space (in sectors), cpu time (in seconds), and session connect time (in minutes). The file space usage count reflects file space used as of the present time, but the cpu time and connect time usage reflects these counts as they were immediately prior to the start of the current job.

The type of output written to *listfile* depends on the type of file (ASCII or binary) specified or implied. If *listfile* is an ASCII file, a standard ASCII listing is produced; on this listing, an unlimited quantity is denoted by a double asterisk (**). If *listfile* is a binary file (typically used to help in automatic processing of the report data), a 17-word record is written for each account/group.

On both ASCII and binary *listfiles*, the entry for each account is followed immediately by the entries for all of its groups.

EXAMPLE

To obtain accounting information for your group, enter the :REPORT command. The accounting information follows the listing of this command:

```
:REPORT @.SUPPORT
ACCOUNT          FILESPACE-SECTORS      CPU-SECOND      CONNECT-MINUTES
   /GROUP          COUNT     LIMIT    COUNT     LIMIT   COUNT     LIMIT
SUPPORT            23116        **   624700        **    98880        **
   /BUGGROUP        7465        **     9233        **    45451        **
   /CREATOR            0        **        0        **        0        **
   /GAMES              0        **      241        **      133        **
   /HP30126            0        **        1        **        4        **
   /HP32000            0        **       31        **       71        **
   /HP32002        14933        **    15482        **     3137        **
   /HP32190            0        **        2        **        1        **
   /HP32192            0        **       43        **       17        **
```

Double asterisk signifies no limit

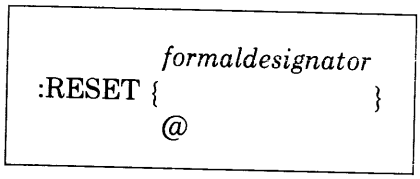
ADDITIONAL DISCUSSION

System Manager/System Supervisor Reference Manual.

:RESET

Cancels :FILE command.

SYNTAX



PARAMETERS

formaldesignator Formal file name of file on which :FILE command will be cancelled.

@ Directive to reset all formal file designators noted in all :FILE commands previously encountered in this job session.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

Resets the formal designator for a file to its original meaning, negating the effect of any :FILE command that has referenced this formal designator up to this point in the job. The :RESET command applies to files on any device.

EXAMPLE

To cancel the effects of a previous :FILE command that specified characteristics for a file programmatically referred to as ALPHA, enter:

```
:RESET ALPHA
```

:RESETDUMP

Disables Stackdump facility.

SYNTAX

```
:RESETDUMP
```

PARAMETERS

None

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

Disarms MPE Stackdump facility (armed by :SETDUMP command), described in the *MPE Debug/Stackdump Reference Manual.* When entered in BREAK, does not modify state of processes already created. If Stackdump facility is not enabled, :RESETDUMP command has no effect.

EXAMPLE

To disable the Stackdump facility, enter:

```
:RESETDUMP
```

ADDITIONAL DISCUSSION

MPE Debug/Stack Dump Reference Manual.

:RESTORE

Returns files (stored on magnetic tape or serial disc) to system.

SYNTAX

```
:RESTORE restorefile[;[fileset][,. . .]]
        [;KEEP][;DEV=device][;SHOW][;FILES=maxfiles][;OLDDATE]
```

PARAMETERS

restorefile Name of magnetic tape or serial disc file on which files to be restored exist. This file must be referenced in the back-reference format by using an asterisk. (This format references a previous :FILE command that has defined the file as a magnetic tape or serial disc file.)

A message is output on the system console requesting the Console Operator to mount the tape or serial disc platter identified by the *restorefile* parameter, and allocate the device. (REQUIRED PARAMETER)

fileset Set of files to be restored. Default is @. This parameter is of the form:

filedesignator[.*groupdesignator*[.*acctdesignator*]]

fileset can be entered in any of the following formats and may use wild card characters, in any order, as replacements.

file.group.account Restore the specified file in the specified group and account.

file.group Restore specified file in specified group.

file Restores the specified file.

@.group.account Restore all files in specified group and account.

@.group Restore all files in specified group.

@.@.account Restore all files in all groups in specified account.

@ Restore all files in logon group. Default for users.

@.@ Restore all files in all groups on logon account. Default for account manager.

@.@.@ Restore all files in system. Default for system manager and system supervisor.

file.@.account Restore file named in any group of specified account.

NOTE

The characters @, #, and ? can be used as wild card characters in the *fileset* parameter. These wild card characters have the following meanings:

@ — specifies zero or more alphanumeric characters.

— specifies one numeric character.

? — specifies one alphanumeric character.

The characters can be used as follows:

n @	Restore all files starting with the character *n*.
@*n*	Restore all files ending with the character *n*.
n @*x*	Restore all files starting with the character *n* and ending with the character *x*.
n #..#	Restore all files starting with the character *n* followed by up to seven digits.
?*n* @	Restore all files whose second character is *n*.
n?	Restore all two-character files starting with *n*.
?*n*	Restore all two-character files ending with *n*.

NOTE

More than one *fileset* may be used with a single :RESTORE command. The number of *filesets* that may be specified with one :RESTORE command is limited as follows: up to 10 by account name; up to 15 by group name and account name; and up to 20 by file name, group name, and account name. The *fileset* is a positional parameter.

KEEP

Specifies that if a file referenced in the :RESTORE command currently exists on disc, the file on disc is kept and the corresponding tape or serial disc file is not copied into the system. Default: If an identically-named file exists in the system, that file is replaced with the one on tape or serial disc. If the identically-named file on disc is busy, however, the disc file is kept and the tape or serial disc file is not restored.

device

Specifies the device on which the file is to reside, entered in one of the following forms:

devclass
ldn

The device class (*devclass*) specifies the type of device. If *devclass* is specified, the file is allocated to any of the home volume set's volumes that fall within that device class.

The logical device number (ldn) specifies a specific device. If *ldn* is specified, the file will be allocated to that device only if one of the volumes in the home volume set currently occupies the device.

Default: MPE attempts to restore the file on a logical device compatible with the type/sub-type specified in the file's file label and the type/ sub-type of the mounted home volume set. If this fails, an attempt is made to restore the file on the same device class as specified in the file's file label and that of the mounted home volume set. If this fails, an attempt is made to restore the file on any member of the home volume set. If this fails, the file is not restored.

SHOW — Request to list names of restored files. Default: Only total number of files restored, list of files not restored, (and the reason each was not restored), and the count of files not restored are listed. The listing is sent to $STDLIST (formal designator SYSLIST) unless a :FILE command is entered to send the listing to some other device. For example,

```
:FILE SYSLIST;DEV=LP
```

entered before the :RESTORE command would send the listing to a line printer.

maxfiles — Maximum number of files that may be restored. Default is 4000.

OLDDATE — Keeps the *accdate* and *moddate* on files being restored unchanged. Default is to alter these dates.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		YES (Aborts)

OPERATION

Reads back into system, on disc, a file or files stored offline by :STORE or :SYSDUMP commands. If you have System Manager or System Supervisor capability, you can restore any file from a :STORE tape or serial disc, assuming the account and group to which the file belongs and the user who created the file are defined in the system. If you have Account Manager capability, you can restore any file in your account (but cannot restore those with negative file codes unless you also have Privileged Mode capability. If you have standard user capability only, you can restore any file in your log-on account if you have SAVE access to the group to which the file belongs; you cannot, however, restore those with negative file codes unless you have Privileged Mode capability. If the file to be restored is protected by a lockword, you must supply the lockword via the :RESTORE command unless you have System Manager, System Supervisor, or Account Manager and you are restoring within your own account) capability. If you are logged on in an interactive session, MPE prompts you for omitted lockwords. However, if you are a System Manager, System Supervisor, or Account Manager (restoring within your own account), you are not required to provide lockwords. Any file belonging to a group whose home volume set has not been pre-mounted will not be restored.

The listing output by :RESTORE is sent to a file whose formal designator is SYSLIST. If not specified, this file is equated by default to the current list device ($STDLIST).

Before entering :RESTORE, you must identify *restorefile* as a magnetic tape or serial disc file with a :FILE command, as follows:

:FILE *formaldesignator*[=*filereference*];DEV=*device*[;REC=*recsize*]

The device parameter must indicate the device class name or logical unit number of a magnetic tape or serial disc unit. The *recsize* parameter must be entered only if the record size of files being :RESTORED is different than 4096. All other parameters for restorefile are supplied by the :RESTORE command. If you attempt to supply any of these parameters, MPE rejects the :RESTORE command.

EXAMPLE

To restore all files belonging to your log-on group from the *restorefile* named BACKUP, enter:

```
:FILE BACKUP;DEV=TAPE
:RESTORE *BACKUP;@;KEEP;SHOW
```

In response, the Console Operator receives a request to mount the tape identified as BACKUP. If a serial device file satisfying the @ specification already exists in the system, it is not restored because the KEEP parameter was specified.

To have the list of restored files printed on a line printer, enter:

```
:FILE SYSLIST;DEV=LP
:FILE BACKUP;DEV=TAPE
:RESTORE*BACKUP;@;KEEP;DEV=MHDISC;SHOW
```

If a file satisfying the @ specification already exists in the system, it is not restored.

ADDITIONAL DISCUSSION

System Manager/System Supervisor Reference Manual.

:RESUME

Resumes execution of a suspended operation.

SYNTAX

```
:RESUME
```

PARAMETERS

None

USE

Available	In Session?	YES (While in BREAK mode only)
	In Job?	NO
	In Break?	YES
	Programmatically?	NO
Breakable?		NO

OPERATION

After a program or MPE command operation is suspended by pressing the BREAK key or by using the CAUSEBREAK intrinsic, the :RESUME command resumes execution of the operation at the point where the execution was suspended.

Note that the :RESUME command is legitimate only during a BREAK. Many MPE commands are aborted rather than suspended by a BREAK, and thus cannot be resumed.

If, instead of :RESUME, you enter another program command (such as :EDITOR, :FORTRAN or :RUN) or one of the non-program commands, :HELLO, :BYE, :JOB or :DATA, the Command Interpreter prints the following message on your terminal:

ABORT? (YES/NO)

If you respond YES to the ABORT? message, the Command Interpreter aborts the current program and executes the command you entered, in the usual way.

If you had logged on using the :( ) command log on with a program command inside the parentheses, then responding YES to the ABORT? message causes MPE to abort the command and log you off immediately.

If you respond NO to the ABORT? message, the Command Interpreter prints the message NOT ALLOWED IN BREAK and prompts you for another command. If you now enter :RESUME, the suspended program resumes at the point where it was interrupted.

EXAMPLE

```
:RESUME
```

:RJE

Calls RJE/3000.

SYNTAX

:RJE [*commandfile*][,[*inputfile*][,[*listfile*][,*punchfile*]]]

PARAMETERS

commandfile Actual designator of file from which RJE/3000 reads its directives. Can be any ASCII input file. Formal designator is RJECOM. Default is $STDIN.

inputfile Actual designator of file from which RJE/3000 reads input data to be transmitted to the remote computer. Can be any ASCII input file. Formal designator is RJEIN. Default is $STDIN.

listfile Actual designator of file to receive listed output obtained from remote computer. Can be any ASCII output file. Formal designator is RJELIST. Because this file usually is a line printer, it is defined in a :FILE command and back referenced as follows:

```
:FILE LP;DEV=LP
:RJE CFILE,INFILE,*LP
```

Default is $STDLIST.

punchfile Actual designator of file to receive punched output obtained from remote computer. Can be any output file. Formal designator is RJEPUNCH. Because this file usually is a non-disc file, it is defined in a :FILE command and back referenced as follows:

```
:FILE PUNFILE;DEV=PTPUNCH
:RJE CFILE,INFILE,,*PUNFILE
```

Default is $OLDPASS (if $OLDPASS exists) or $NEWPASS if $OLDPASS does not exist.

NOTE

The formal file designators used in this command (RJECOM, RJEIN, RJELIST, RJEPUNCH) cannot be back referenced as actual file designators in the command parameter list. For further information, see the OPERATION section of the :FILE command.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Invokes the HP 2780/3780 Emulator. The Emulator permits transfer of data between an HP 3000 Computer System and a variety of remote processors in a full multiprogramming environment. The Emulator makes the HP 3000 Computer System appear to a remote processor as either an IBM 2780 or 3780 Data Transmission Terminal.

EXAMPLE

To invoke the Emulator in a session, enter directives to it via your current input device, and input data via the disc file MYDATA, with listing output directed to a line printer (LISTFL) and punched output to a card punch (PUNCHFL), enter:

```
:FILE LISTFL;DEV=LP
:FILE PUNCHFL;DEV=PTPUNCH
:RJE ,MYDATA,*LISTFL,*PUNCHFL
```

ADDITIONAL DISCUSSION

RJE/3000 Reference Manual (HP 2780/3780 Emulator Reference Manual).

:RPG

Compiles an RPG program.

SYNTAX

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

PARAMETERS

textfile Actual designator of input file from which source program is read. Can be any ASCII input file. Formal designator is RPGTEXT. Default is $STDIN.

uslfile Actual designator of USL file on which object program is written. Can be any binary input file with a *filecode* of USL or 1024. Formal designator is RPGUSL. If this parameter is specified, it must indicate a file previously created in one of four ways:

1. By saving a USL file, with the MPE :SAVE command, created by a previous compilation where the default value was used for the *uslfile* parameter.
2. By building the USL with the MPE Segmenter-BUILDUSL command. (See the *MPE Segmenter Reference Manual.*)
3. By creating a new USL file with the MPE :BUILD command with a *filecode* of USL or 1024.
4. By specifying a non-existent *uslfile* parameter, thereby creating a permanent file of the correct size and type.

Default: $NEWPASS is assigned.

listfile Actual designator of file on which program listing is written. Can be any ASCII output file. Formal designator is RPGLIST. Default is $STDLIST.

masterfile Actual designator of master file to be merged against *textfile* to produce composite source. Can be any ASCII input file. Formal designator is RPGMAST. Default is that the master file is not read, input is read from *textfile*.

newfile Actual designator of merged *textfile* and *masterfile*. Can be any ASCII output file. Formal designator is RPGNEW. Default is that no file is written.

NOTE

The formal file designators used in this command (RPGTEXT, RPGUSL, RPGLIST, RPGMAST, RPGNEW) cannot be back referenced as actual file designators in the command parameter list. For further information, see OPERATION section of the :FILE command.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Compiles RPG program onto a User Subprogram Library (USL) file on disc. If you do not specify *textfile,* MPE expects input from your current job input device. If the USL file is created prior to compilation, it must have a *filecode* of USL or 1024.

EXAMPLE

To compile an RPG program entered from the current input device into an object program in the USL file $NEWPASS, and write the listing to the current list device, enter:

```
:RPG
```

If the next command is one to prepare an object program, $NEWPASS can be passed to that command by referencing $OLDPASS. Note that a file can only be passed between commands or programs within the same job or session.

To compile an RPG program residing on the disc file SOURCE into an object program on the USL file OBJECT, with the program listing sent to the disc file LISTFL, enter:

```
:BUILD OBJECT;CODE=USL
:RPG SOURCE,OBJECT,LISTFL
```

You must specify the CODE parameter as USL if you choose to create a USL file with the :BUILD command.

To compile an RPG program creating the USL file with the *uslfile* parameter, enter:

```
:RPG SOURCE,OBJECT,LISTFL
```

Textfile *USL file* *List file*

ADDITIONAL DISCUSSION

RPG/3000 Compiler Reference Manual.

:RPGGO

Compiles, prepares, and executes an RPG program.

SYNTAX

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

PARAMETERS

textfile Actual designator of input file from which source program is read. Can be any ASCII input file. Formal designator is RPGTEXT. Default is $STDIN.

listfile Actual designator of file on which program listing is written. Can be any ASCII output file. Formal designator is RPGLIST. Default is $STDLIST.

masterfile Actual designator of master file which is merged against *textfile* to produce composite source. Can be any ASCII input file. Formal designator is RPGMAST. If masterfile is not designated, input is read from textfile.

newfile Actual designator for merged textfile and *masterfile*. Can be any ASCII output file. Formal designator is RPGNEW. Default is that no file is written.

NOTE

The formal file designators used in this command (RPGTEXT, RPGLIST, RPGMAST, RPGNEW) cannot be back referenced as actual file designators in the command parameter list. For further information, see OPERATION section of :FILE command.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Compiles, prepares, and executes an RPG program. If you do not specify a source file, MPE expects input from your current input device. This command creates a temporary User Subprogram Library (USL) file ($NEWPASS) that you cannot access, and a temporary program file that you can access under the name $OLDPASS.

EXAMPLE

To compile, prepare, and execute an RPG program entered from the current input device, with the program listing sent to your current list device, enter:

```
:RPGGO
```

To compile, prepare, and execute an RPG program read from the disc file SOURCE and send the resulting program listing to the disc file LISTFL, enter:

```
:RPGGO SOURCE,LISTFL
```

ADDITIONAL DISCUSSION

RPG/3000 Compiler Reference Manual.

:RPGPREP

Compiles and prepares an RPG program.

SYNTAX

:RPGPREP [*textfile*][,[*progfile*][,[*listfile*][,*masterfile*][,*newfile*]]]

PARAMETERS

textfile Actual designator of input file from which source program is read. Can be any ASCII input file. Formal designator is RPGTEXT. Default is $STDIN.

progfile Actual designator of program file on which prepared program segments are written. Can be any binary input file with *filecode* of PROG or 1029. If entered, this parameter must indicate a file created in one of two ways:

1. By creating a new program file with the MPE :BUILD command with a *filecode* of PROG or 1029 and a *numextents* parameter value of 1.

2. By specifying a non-existent file in the *progfile* parameter, in wich case a job session temporary file of the correct size and type is created. Default is that $NEWPASS is assigned.

listfile Actual designator of file on which program listing is written. Can be any ASCII output file. Formal designator is RPGLIST. Default is $STDLIST.

masterfile Actual designator of master file which is merged against *textfile* to produce composite source. Can be any ASCII input file. Formal designator is RPGMAST. Default is that master file is not read; input is read from *textfile*.

newfile Actual designator of merged *textfile* and *masterfile*. Can be any ASCII output file. Formal designator is RPGNEW. Default is that no file is written

NOTE

The formal file designators used in this command (RPGTEXT, RPGPROG, RPGLIST, RPGMAST, RPGNEW) cannot be back referenced as actual file designators in the command parameter list. For further information, see OPERATION section of :FILE command.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Compiles and prepares an RPG program onto a program file on disc. If you do not specify a source file, MPE expects input from your current input device. The User Subprogram Library (USL) file created during compilation is a temporary file passed to the preparation mechanism; you can access it with the name $OLDPASS only if the program file is not $NEWPASS.

EXAMPLE

To compile and prepare an RPG program entered through the current input device, onto the file $NEWPASS, with the listing printed on the current list device, enter:

```
:RPGPREP
```

If the next command is one to execute the program, the file $NEWPASS is referenced in the execute command under the name $OLDPASS.

The USL file created during compilation is a temporary file passed directly to the preparation mechanism; you can access it, but only under the name $OLDPASS, and only if the program was not prepared into the file named $NEWPASS. Therefore, if you want to save the compiled USL, you must specify some other file as the second positional parameter in this command, and then use a :SAVE command. For instance:

Note the first positional parameter omitted, *textfile* read from standard input device.

```
:RPGPREP ,COMFL          Compiles into $NEWPASS, PREPARED INTO COMFL.
:SAVE $OLDPASS,NUSL      Saves USL, changing file name from $OLDPASS
                         (formerly $NEWPASS) to NUSL.
```

The program file is also created as a temporary file (unless you prepare the program onto a previously-created file). Therefore, if you want to save a new program file, you may use the :SAVE command as follows:

```
:RPGPREP ,COMFL          Compiles into $NEWPASS, prepares into COMFL.
:SAVE $OLDPASS,NUSL      Saves USL under name NUSL.
:SAVE COMFL              Saves program file named COMFL.
```

Alternatively, if you want to create a new permanent file into which to prepare your program, you may use the :BUILD command. When you do this, however, you must specify a filecode of PROG (or 1029) to specify program file format, and limit this file to one extent.

Specifies program file format *Specifies 1 extent*

```
:BUILD PROGFL;CODE=PROG;DISC=,1    Creates permanent program file.
:RPGPREP ,PROGFL                   Compiles into $NEWPASS, prepares
                                   into PROGFL.
```

If you are receiving your input file from a device other than the standard input device, for instance, from a disc file created via the Editor, and wish to transmit your listing to a device other than the standard listing device — for instance, to another line printer, enter:

```
:FILE LINEA;DEV=12                          Identifies line printer (logical device 12).
:RPGPREP EDTDISC,COMFL,*LINEA               Compiles and prepares program.
```

Text file — *Program file* — *List file*

ADDITIONAL DISCUSSION

RPG/3000 Compiler Reference Manual.

Executes a prepared program.

SYNTAX

```
:RUN progfile[,entrypoint]

      [;NOPRIV]
      [;LMAP]
      [;DEBUG]
      [;MAXDATA=segsize]
      [;PARM=parameternum]
      [;STACK=stacksize]
      [;DL=dlsize]


              G
      [;LIB={ P }]
              S

      [;NOCB]
      [;INFO=string]
      [;STDIN = [*formaldesig ]] [;STDLIST = [*formaldesig      ]]
                [fileref      ]]             [fileref [,NEW]  ]]
                [$NULL        ]]             [$NULL           ]]
```

PARAMETERS

progfile — Actual designator of program file that contains prepared program. (REQUIRED PARAMETER)

entrypoint — Program entry point where execution is to begin. May be primary entry point of program or any secondary entry point in program's outer block. Default is primary entry point.

NOPRIV — Declaration that program segments will be placed in non-privileged (user) mode. This parameter is intended for programs prepared with privileged mode capability. Normally, programs containing privileged instructions are executed in privileged mode only if program was prepared with privileged mode (PM) capability class. (A program containing legally compiled privileged code, placed in non-privileged mode, may abort when an attempt is made to execute it.) If NOPRIV is specified in the :RUN command, all program segments are placed in non-privileged mode. (Library segments are not affected because their mode is determined independently.)

Note that NOPRIV produces the same effect as omitting the $OPTION PRIVILEGED and OPTION PRIVILEGED entries in SPL source input. Default is that privileged mode programs will remain in privileged mode.

LMAP	Request to produce a descriptive listing of the allocated (loaded) program on file whose formal designator is LOADLIST. If no :FILE command is found that references LOADLIST, listing is sent to $STDLIST. Default is no listing. See Appendix D for a loaded program listing.
DEBUG	Request to issue a Debug call before the first executable instruction of the program. This parameter is ignored when a non-privileged user runs a program having privileged mode capability. This parameter also is ignored if user does not have read and write access to program file. Default is Debug call is not issued.
segsize	Maximum stack area (Z-DL) size permitted, in words. This parameter is included if it is expected that size of DL-DB or Z-DB areas will be changed during program execution. Default is MPE assumes areas will not be changed.
parameternum	Value that can be passed to program as a general parameter for control or other purposes. When program is executed, this value can be retrieved from address Q(initial)-4, where Q(initial) is Q address for outer block of program. Value can be octal number or signed or unsigned decimal number. Default is Q(initial)-4 address is filled with zeros.
stacksize	Size of initial local data area (Z-Q(initial)) in stack. This value must exceed 511 words, and overrides *stacksize* estimated by MPE Segmenter. Default is estimated by Segmenter.
dlsize	DL-DB area to be initially assigned to stack. This area is of interest mainly in programmatic applications. In all cases, the DL-DB area is rounded upward so that the distance from the beginning of the stack data segment to the DB address is a multiple of 128 words. Default is estimated by Segmenter.
G	Search segmented procedure libraries of the program file's group and account to satisfy external references during allocation in the following order: group library, account public library, system library. Default is S.
P	Search segmented procedure libraries of the program file's group and account to satisfy external references during allocation in the following order; account public library, system library. Default is S.
S	Search system library only to satisfy external references during allocation. Default.
NOCB	Request that file system not use stack segment (PCBX) for its control blocks, even if sufficient space is available. This permits expansion of the stack (with the DLSIZE and ZSIZE intrinsics) to the maximum possible limit at a later time, but causes the File Management System to operate more slowly for this program.
STDIN	Use of this parameter allows the user to specify the file to be used by the program being executed.
STDLIST	Use of this parameter allows the user to specify the file to be used by the program being executed.

formaldesig The formal file designator for a file previously specified in a file equation.

fileref The name of an old parameter disc file or, if the NEW option is specified, the name to be assigned to a job/session temporary disc file created with system defaults.

The system defaults for the NEW file are fixed length ASCII 132 byte records with a maximum file size of 1023 records.

If nothing is specified on the right-hand side of the " =" sign or if "STDIN =" and "STDLIST =" are not specified on the :RUN command, the job/session main $STDIN and $STDLIST files are assumed to serve as $STDIN and $STDLIST for the program. In session mode, $STDIN and $STDLIST are the interactive session terminal. In job mode, $STDIN is usually the spooled input device file that introduced the job, and $STDLIST is usually the spooled line printer.

INFO A string of ASCII characters delimited by a pair of 's or "s. Maximum length of the string is 255 characters, including the delimiting characters.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Executes a program prepared in a program file. This command permits searching of segmented libraries (SL's) but not relocatable libraries (RL's) to satisfy external references.

If the volume set containing the file to be run is not mounted, this command implicitly causes that volume set to be mounted.

EXAMPLE

To run a program stored in the program file PROG4, beginning at the entry point SECLAB, enter:

```
:RUN PROG4,SECLAB
```

To get a listing of the loaded program, enter:

```
:RUN XLAB;LMAP
```

The following example runs a program with $STDIN set to an old disc file named INPUT and $STDLIST set to the line printer:

```
:FILE LPFILE; DEV=LP
:RUN TESTPROG; MAXDATA=10000; STDIN=INPT; STDLIST=*LPFILE
```

The next example runs a program using the STDIN parameter, setting $STDIN to an old disc file named INPT, this time referenced through a file equation, and $STDLIST set to a temporary disc file named RESULTS that is automatically created by the :RUN command.

```
:FILE INFILE=INPT,OLD;
:RUN TESTPROG; DEBUG; STDIN=*INFILE; STDLIST=RESULTS,NEW
```

The following example of the :RUN command uses the INFO parameter to pass a string to the program:

```
:RUN MYPROG; MAXDATA=2000; INFO= 'A test with "and" characters'
```

Note that if the delimiting character is desired within the string, it can be doubled. In the above example, the string passed to the program is:

```
A TEST WITH "AND" CHARACTERS
```

Saves file in system file domain.

SYNTAX

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

PARAMETERS

$OLDPASS — File currently being passed. After this file is saved, no file in the job/session temporary file domain can be referenced by the name $OLDPASS.

newfilereference — New actual file designator to be assigned $OLDPASS when it is made permanent, written in the format:

filename [/*lockword*][.*groupname* [.*acctname*]]

If *acctname* is used, this must indicate the log-on account. If *groupname* is used, this must indicate a group to which you have SAVE access, as defined by your Account Manager. If *groupname* is omitted, log-on group is assigned. *newfilereference* is required as a parameter if $OLDPASS is used.

tempfilereference — Actual designator of temporary file to be made permanent under the same designator. The file is deleted from job/session temporary file domain and entered in system file domain. This parameter is written in the format:

filename [/*lockword*][.*groupname* [.*acctname*]]

If *acctname* is used, this must indicate the log-on account. If *groupname* is used, this must indicate a group to which you have SAVE access, as defined by your Account Manager. If *groupname* is omitted, log-on group is assigned. *tempfilereference* is required as a parameter if $OLDPASS/*newfilereference* is not used.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

The :SAVE command saves a temporary file currently in a temporary file domain by converting it to a permanent file in the system file domain. This command is necessary when the subsystem or program that created your file does not allow you the option to save it while the program is executing. You must specify a new file name for $OLDPASS because MPE does not allow use of $OLDPASS as a permanent file name. In addition, the volume set for the group in which the file is to be saved must be mounted because files cannot be saved across volume sets. Note that this command applies only to temporary files on disc. It is analogous to opening a file (FOPEN CALL) and then closing it (FCLOSE CALL) with permanent file disposition.

EXAMPLE

To save the temporary file $OLDPASS, containing an object program, onto the program file PROGFILE, enter:

```
:SAVE$OLDPASS,PROGFILE
```

To save the temporary file TEMPFL as a permanent file of the same name, enter:

```
:SAVE TEMPFL
```

To save temporary file DATAFILE under the GROUPX group, enter:

```
:SAVE DATAFILE.GROUPX
```

Note: You must have SAVE access to the GROUP group

To save a temporary file whose present name is not $OLDPASS and change its name, you must also use the :RENAME command:

```
:SAVE DATAFILE.GROUPX                          saves file DATAFILE
:RENAME DATAFILE.GROUPX,DATABASE.GROUP         changes name to DATABASE
```

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

SYNTAX

:SECURE *filereference*

PARAMETERS

filereference Actual designator of the disc file whose security provisions are to be restored, written in the format:

filename [/*lockword*][.*groupname* [.*acctname*]]

(REQUIRED PARAMETER)

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

Restores all security provisions for a file that were suspended previously by a :RELEASE command in this or another job.

This command can be used only for a permanent disc file whose file label identifies the user as the creator of that file. In addition, the :SECURE command will fail if the group's home volume set is not mounted. When the normal (default) MPE security provisions are in effect, the file must belong to the log-on account and group.

EXAMPLE

To restore the security provisions previously in effect for the file named FILE1, enter:

```
:SECURE FILE1
```

:SEGMENTER

Calls MPE Segmenter.

SYNTAX

:SEGMENTER [*listfile*]

PARAMETERS

listfile Actual designator of file to receive listable output from Segmenter. Can be any ASCII output file. Formal designator is SEGLIST. Because this file usually is a line printer, it is defined in a :FILE command and back referenced as follows:

```
:FILE LISTFL;DEV=LP
:SEGMENTER *LISTFL
```

Default is $STDLIST.

NOTE

The formal file designator used in this command (SEGLIST) cannot be back referenced as actual file designators in the command parameter list. For further information, see OPERATION section of :FILE command.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Calls MPE Segmenter.

EXAMPLE

To call the Segmenter from a session and transmit the listable output to a line printer instead of the current list device, enter:

```
:FILE LISTFL;DEV=LP
:SEGMENTER *LISTFL
```

NOTE

To use a relocatable library (RL) via the :SEGMENTER command, the user is required to have read and lock capability.

ADDITIONAL DISCUSSION

MPE Segmenter Reference Manual.

:SETCATALOG

Causes the Command Interpreter to search a catalog of user-defined commands and to establish a directory entry for each command in the catalog.

SYNTAX

```
                                                                  [;ACCOUNT]
:SETCATALOG [catfilename[,catfilename,...,[catfilename]]][;SHOW][;SYSTEM  ]
```

PARAMETERS

catfilename — Catalog file name. Normally, this file would be prepared using the Editor. When more than one user-defined command resides in the catalog file, the commands must be separated from each other by one line, the first character of which must be an asterisk.

SHOW — Lists catalogs and UDCs as the user-defined commands are initialized. This parameter is useful for listing additional information if there is an error in UDC initialization. SHOW lists each UDC as it is checked. If an error occurs, it is listed after the erroneous UDC.

ACCOUNT — Specifies that the catalog file being defined or reset is at the account level. Requires AM capability.

SYSTEM — Specifies that the catalog file being defined or reset is at the system level. Requires SM capability.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	NO
Breakable?		NO

OPERATION

The :SETCATALOG command causes the Command Interpreter to search a specially-prepared disc file for user-defined commands. The Command Interpreter then establishes a directory entry for each command in the file.

EXAMPLE

To create a user-defined command (UDC) and have the Command Interpreter establish a directory entry for it in the file MYCMNDS, enter:

```
:EDITOR

/ADD
      1  S                    (UDC name)
      2  SHOWJOB              (MPE command)
      3  **                   (Separates UDC's)
      4  R                    (UDC name)
      5  RESUME               (MPE command)
      6  //
/KEEP MYCMNDS
/END

:SETCATALOG MYCMNDS
```

Thus, S and R are established as user-defined commands. If you enter S, the :SHOWJOB MPE command will execute; if you enter R, :RESUME will execute.

Example for altering catalog:

```
:SETCATALOG (Closes existing UDC file)
:EDITOR

/TEXT MYCMNDS
/ADD
      6  **
      7  E
      8  EDITOR
      9//
/KEEP MYCMNDS
/END

:SETCATALOG MYCMNDS
```

ADDITIONAL DISCUSSION

See Section III of this manual.

:SETDUMP

Enables Stackdump facility on abort.

SYNTAX

```
                    DB [,ST] [,QS]
:SETDUMP [ { ST [,DB]          } [;ASCII]]
                    QS [,DB]
```

PARAMETERS

DB	Dump memory from DL to Q(initial) address. Default is a display of the stack marker trace and all registers at time of abort.
ST	Dump memory from Q(initial) to S address. Default is a display of the stack marker trace and all registers at time of abort.
QS	Dump memory from Q-63 to S address. This parameter is ignored if ST parameter is used. Default is a display of the stack marker trace and all registers at time of abort.
ASCII	Display ASCII conversion of octal values requested by DB, ST, or QS parameters (along with octal display). Default is octal.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

Arms Stackdump Facility as described in the *MPE Debug/Stackdump Reference Manual.*

In an interactive session, all parameters of the :SETDUMP command are ignored and the only effect is the arming of the Stackdump Facility in order for the process to call Debug if an abort occurs. To be able to access Debug, you must have read and write access to the program file.

EXAMPLE

To arm the Stackdump Facility to display the memory area from the Q(initial) to S address, with accompanying ASCII conversion of the octal data, enter:

```
:SETDUMP ST;ASCII
```

ADDITIONAL DISCUSSION

MPE Debug/Stack Dump Reference Manual.

:SETJCW

Scans JCW table for a specified JCW name and updates the value of this JCW.

SYNTAX

```
:SETJCW jcwname char value
```

PARAMETERS

jcwname The name of a Job Control Word (JCW). If *jcwname* is that of an existing JCW, the JCW table is scanned for this name. If a name matching *jcwname* is not found in the JCW table, *jcwname* is added to the table. (REQUIRED PARAMETER)

char Any special character except %, used as a delimiter. (REQUIRED PARAMETER)

value One of the following:

1. Octal number between 0 and %177777.
2. Decimal number between 0 and 65535.
3. A name which has been defined as being equivalent to certain numeric values, as follows:

   OK = 0
   OK1 = OK + 1 = 1
   OK100 = OK + 100 = 100 = %144

   WARN = 16384 = %40000 (= OK16384)
   WARN100 = WARN + 100 = 16484 = %40144

   FATAL = 32768 = %100000 (= WARN16384 = OK32768)
   FATAL85 = FATAL + 85 = 32853 = %100125

   SYSTEM = 49152 = %140000
   SYSTEM200 = SYSTEM + 200 = 49352 = %140310

4. The name of an existing JCW.

(REQUIRED PARAMETER)

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

There are two predefined job control words, "JCW" and "CIERROR". At the beginning of a job or session the value of "CIERROR" is set to zero. Whenever a warning/error occurs as a result of executing an MPE command, "CIERROR" is set to the value of the error message reported, and retains this value until another error/warning occurs or the :SETJCW command is executed. The "JCW" is usually set by a subsystem when some type of fatal error occurs. For a description of "JCW" see the *Intrinsics Manual.*

If a specified *jcwname* is found in the JCW table, the *jcwname* is set equal to the value passed with :SETJCW. If the *jcwname* is not found, it is inserted in the JCW table.

EXAMPLE

To set the Job Control Word CURR1 to 100, enter:

```
:SETJCW CURR1,100
```

To set CURR1 to WARN, enter:

```
:SETJCW CURR1/WARN
```

:SETMSG

Disables or re-enables receipt of user or operator messages at standard list device.

SYNTAX

```
              OFF
:SETMSG {           }
              ON
```

PARAMETERS

OFF — Sets job or session to refuse :TELL command messages from other users.

ON — Re-enables message operation.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

Allows job or session to run in quiet mode, such that :TELL messages from other users are refused. :WARN messages from the Console Operator override quiet mode and are accepted.

EXAMPLE

To prevent messages, enter:

```
:SETMSG OFF
```

To re-enable message reception, enter:

```
:SETMSG ON
```

:SHOWALLOW

Display which operator commands have been allowed.

SYNTAX

:SHOWALLOW [{*username* / @}.{*acctname* / @}]

PARAMETERS

username Defines a particular user.

account name Defines a particular account.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

This command will display for the user which operator commands that (s)he or the specified users have been allowed. Either zero or the two parameters must be specified. If none are specified, then only that user's allows are listed. Account manager capability is required to specify "@" for *username* and system manager capability is required to specify "@" for *acctname*. In addition, the command separately lists which operator commands have been globally allowed.

EXAMPLE

```
:SHOWALLOW
MANAGER.SYS

    USER HAS NO COMMANDS ALLOWED.

THERE ARE NO GLOBAL ALLOWS DEFINED.
```

:SHOWCATALOG

Lists user-defined command (UDC) files.

SYNTAX

```
:SHOWCATALOG [listfile]
```

PARAMETERS

listfile An arbitrary file name, causing the listing to be sent to a line printer. Default: If omitted, the listing is sent to $STDLIST. If specified, the listing is sent to device class LP (line printer) unless directed elsewhere with a prior :FILE command.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	NO
Breakable?		YES (Aborts)

OPERATION

The :SHOWCATALOG command lists user-defined command (UDC) files and specifies at which levels (USER, ACCOUNT, SYSTEM) the files have been defined.

EXAMPLE

To list UDC files and have the listing appear on the standard list device, enter:

```
:SHOWCATALOG
```

To have the listing sent to the line printer, enter:

```
:SHOWCATALOG LFILE
```

To have the listing sent to a disc device, enter:

```
:FILE LFILE;DEV=DISC
:SHOWCATALOG *LFILE

UDC CATALOG LIST SENT TO LIST FILE.  (CI 1932)
```

ADDITIONAL DISCUSSION

See Section III of this manual.

:SHOWDEV

Reports status of input/output devices.

SYNTAX

```
                 ldev
:SHOWDEV [               ]
                 classname
```

PARAMETERS

ldev — Logical device number of device for which status information is to be displayed. Unique for each individual device. Default is if both *ldev* and *classname* are omitted, status information for all devices on the system is displayed.

classname — Device class name of device(s) for which status information is to be displayed. May apply to several devices. Default is if both *ldev* and *classname* are omitted, status information for all devices on the system is displayed.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		YES (Aborts)

OPERATION

Displays status information for any input/output device on the system. The following items may appear in the listing, always displayed on the standard list device:

Logical device number.

Input type allowed:

- J Accepts jobs.
- D Accepts data.
- A Accepts jobs and data.

Availability:

AVAIL	Available as a real, non-sharable device.
SPOOLED	Available via input or output spooling.
UNAVAIL	Not available (owned by job or session).
DISC	Device is a disc, always available.

Device ownership (if applicable):

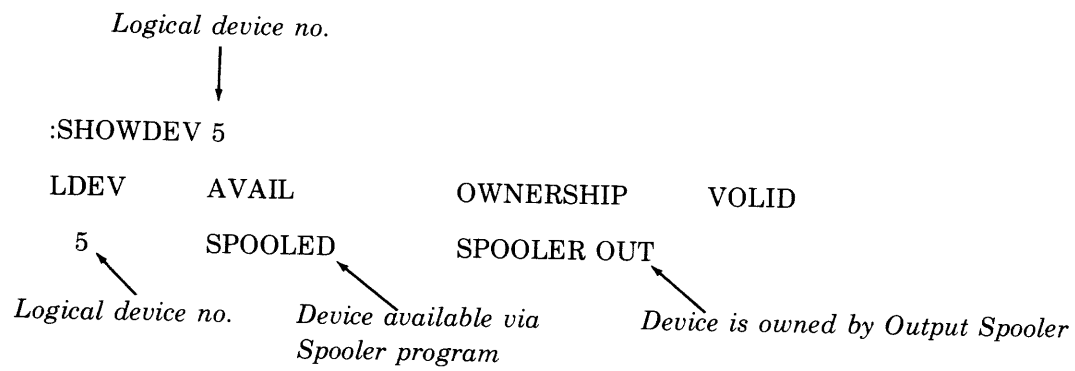
SYS	Owned by system. If *#nnn* appears, it specifies Process Identification Number (PIN) of the owning process (running program).
SPOOLER IN	Input spooling in effect, owned by spooler.
SPOOLER OUT	Output spooling in effect, owned by spooler.
#Jnnn	Owned by indicated job.
#Snnn	Owned by indicated session.
DIAG	Allocated to diagnostic testing by Console Operator via :GIVE command.
nn FILES	Indicates number of files currently in use on a disc.
DOWN	Device is offline, requested by Console Operator via :DOWN command.
DP	Device is being taken offline (:DOWN command operation pending).
VOLID	The MPE file label for labeled tapes. If the tape is unlabeled, (NOLABEL) is displayed.

The display appears in the following format:

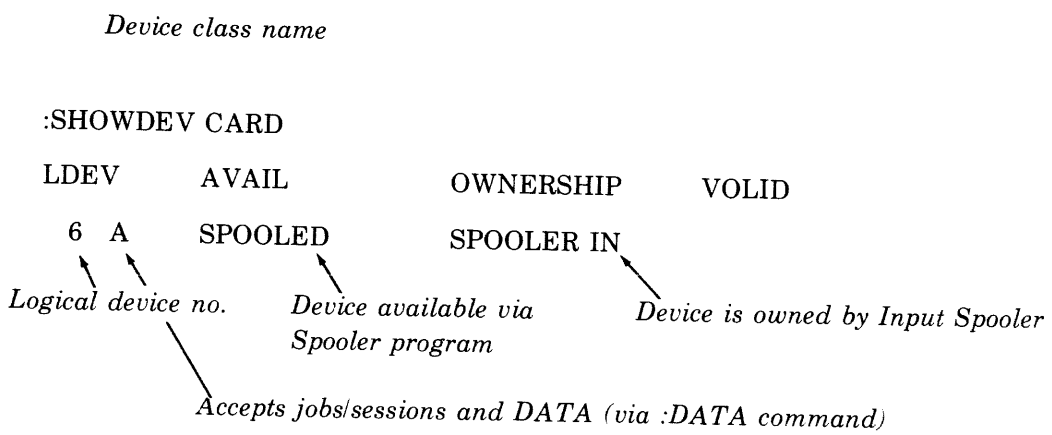
```
:SHOWDEV
LDEV  AVAIL      OWNERSHIP      VOLID

  1    DISC       113 FILES
  2    DISC       0 FILES
  4    DISC       38 FILES
  6    SPOOLED    SPOOLER OUT
  7    AVAIL
  8    AVAIL
  9    AVAIL
 10 A  AVAIL
 15    UNAVAIL    SYS #3
 16    UNAVAIL    SYS #3
 20 A  UNAVAIL    #S91: 2 FILES
```

EXAMPLE

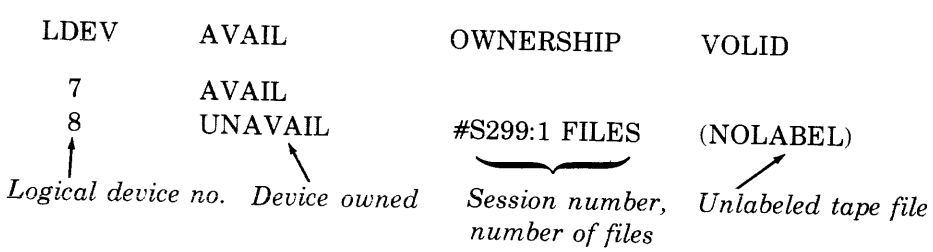
To display the status of the device identified by logical device number 5 enter:



To display the status of all devices of the device class CARD, enter:



To display all devices with the class name TAPE, enter:



Reports status of input devicefiles.

SYNTAX

```
                [#Innn  ]
                [STATUS ]
:SHOWIN [                                ]
        [SP ][;item ][;item ][;item ]
```

PARAMETERS

#Innn Identifier of particular input device file for which information is to be displayed. The information appears in following format:

```
DEV/CL   DFID   JOBNUM   FNAME   STATE   FRM   SPACE   RANK   PRI   #C

45       #I25   #S16     $STDIN  OPENED
```

Default: MPE displays information for all input devicefiles used by this job.

STATUS Request to summarize status information for all current input devicefiles. The information appears in following format:

```
8 FILES:
    0 ACTIVE
    0 READY;INCL 0 SPOOFLES, 0 DEFERRED
    8 OPENED; INCL 0 SPOOFLES
    0 SPOOFLES; 0 SECTORS
```

Default: MPE displays information for all input devicefiles used by this job.

item Request to display status of current input devicefiles as identified by the following subparameters:

```
[DEV=ldev ]

          @S
          @J
[JOB= { @       } ]
          [#]Snn
          [#]Jnnn

 ACTIVE
[READY   ]
 OPENED
```

Each set of brackets, above, defines an item. You cannot use the same item more than once in the parametr list. The subparametrs are:

ldev	Display status of input devicefile residing on device identified by logical device number *ldev*.
@S	Display status of input devicefiles for all sessions.
@J	Display status of input devicefiles for all jobs.
@	Display status of input devicefiles for all jobs and sessions.
#Snnn	Display status of all input devicefiles for session indicated.
#Jnnn	Display status of all input devicefiles for job indicated.
ACTIVE	Display status of ACTIVE devicefiles.
READY	Display status of READY devicefiles.
OPENED	Display status of OPENED devicefiles.

Do not use duplicate item keywords in this command. That is, you can specify

:SHOWIN DEV =25;ACTIVE;@J but not
:SHOWIN DEV=25;ACTIVE;OPENED.

Default: MPE displays status information for all input devicefiles used by this job.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		YES (Aborts)

OPERATION

Displays status information about one or more currently-defined input devicefiles. This information reflects the status at the time the command is entered, and always appears on the standard list device. The information includes:

Logical device number of device.

Devicefile identification in the form *#Innn*.

Job/session number (*jsnum*) of job/session using the devicefile, if not used for READY or ACTIVE data; otherwise, job/session name appears on line following standard device information.

Filename associated with the devicefile.

State:

ACTIVE	Input being read from spooled device to disc.
READY	Input spooling completed, and file is now ready for use by a program.
OPENED	File is being accessed by a program.

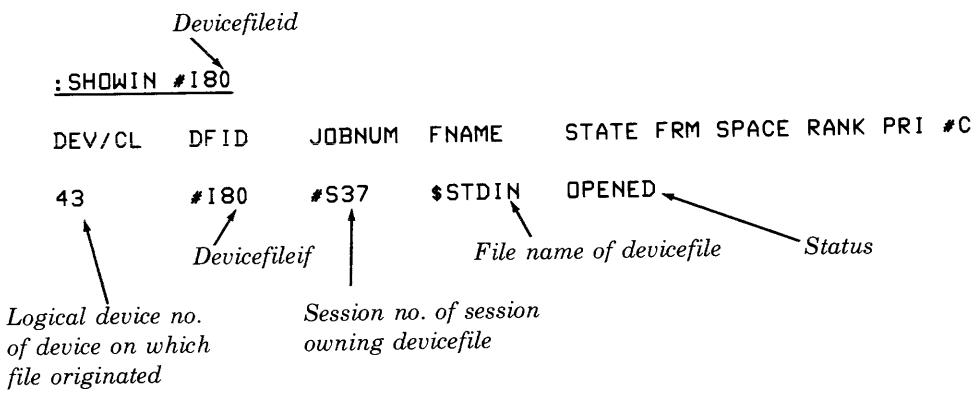
Approximate disc space currently used (in sectors), for spooled input devicefiles only.

Rank, indicating the order in which the file is entered in the system with respect to other files.

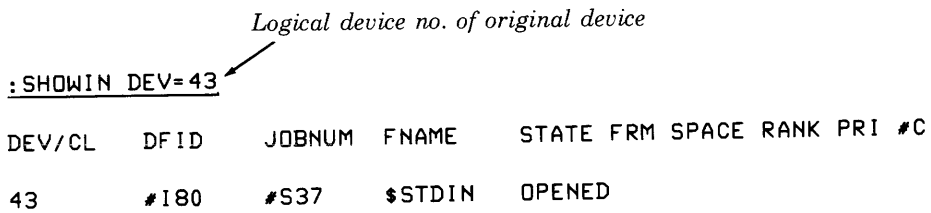
Input priority requested by user (1 = lowest, 13 = highest, blank = current default priority).

EXAMPLE

The following is an example of how you would determine the status of an individual input devicefile:



If you do not know the *devicefileid* of the device file whose status you want to determine, you may request the status display by entering either the logical device number or device class name of the device on which the file originated:



You can also request displays of devicefile information by various combinations of qualifications (devices, jobs/sessions, and states). As an example, suppose you wanted to display information about all OPENED input devicefiles used by all sessions (but not jobs) in the system. You would enter:

Requests data on OPENED devicefiles used by all sessions

```
:SHOWIN JOB=@S;OPENED

DEV/CL    DFID     JOBNUM   FNAME     STATE FRM SPACE RANK PRI #C

7         #I81     #S38     MASTER    OPENED
26        #I36     #S18     $STDIN    OPENED
32        #I85     #S41     $STDIN    OPENED
34        #I58     #S26     $STDIN    OPENED
42        #I64     #S28     $STDIN    OPENED
43        #I80     #S37     $STDIN    OPENED
50        #I84     #S40     $STDIN    OPENED
51        #I35     #S17     $STDIN    OPENED

8 FILES (DISPLAYED):
    0 SPOOFLES: 0 SECTORS
```

:SHOWJCW

Displays current state of Job Control Word.

SYNTAX

:SHOWJCW [*jcwname*]

PARAMETERS

jcwname The name of a Job Control Word (JCW). If omitted, the status of all JCW's is displayed.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		YES (Aborts)

OPERATION

The :SHOWJCW command is used to display the current state of any or all Job Control Words. If no particular JCW is specified, all JCWs and CIERRORs are shown.

EXAMPLE

```
:showjcw
JCW = 0
CIERROR = 0
```

To display the current state of JCW named JCW1, enter:

```
:SHOWJCW JCW1
```

:SHOWJOB

Displays status information about jobs/sessions.

SYNTAX

```
                [#]Snnn
                [#]Jnnn
:SHOWJOB [ STATUS]
                id[;state]
                state[;id]
```

PARAMETERS

#Snnn The session number (assigned by MPE) of the session for which status information is to be displayed. The information appears in Type I format, described under OPERATION. Default is that the status information for all jobs/sessions is displayed.

#Jnn The job number (assigned by MPE) of the job for which status information is to be displayed. The information appears in Type I format, described under OPERATION. Default is that the status information for all jobs/sessions is displayed.

STATUS A request to display a summary of status information for all jobs/sessions. The information is in Type II format, described under OPERATION. Default is that the status information for all jobs/sessions is displayed.

id A list of jobs/sessions/session whose status information is to be displayed. This parameter is written in this format:

```
      @
      @J
      @S
JOB= { [jsname,]username.acctname }
      @,username.acctname
      [@,]@.acctname
```

The symbol @ indicates all. Thus @S means all sessions, @J means all jobs, and @.@.*acctname* means all sessions and jobs for all users in the specified account. If only one job/session is displayed, output appears in Type I format; if more than one job/session is displayed, the output appears in Type III format. (Format types are described under OPERATION.) Default is status information for all jobs/sessions is displayed.

state A particular job/session state, specified as a further restriction on which jobs/sessions are to be displayed. The format of this parameter is:

```
INTRO
WAIT [,N]
EXEC [,D]
SUSP
```

N and D, which are requests for only non-deferred or deferred jobs/sessions, respectively, apply only to the WAIT state.

If only one job/session is displayed, output appears in Type I format; if more than one job/session is displayed, output appears in Type III format. (Format types are described under OPERATION.) Default is that status information for all jobs/sessions is displayed.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		YES (Aborts)

OPERATION

This command enables you to determine the number of jobs and sessions in each processing state, the current jobfence and job/session limits, and allows you to keep track of individual spooled and streamed jobs that are entered in the system.

The output appears in three possible formats:

Type I:

```
JOBNUM   STATE   IPRI   JIN   JLIST   INTRODUCED   JOB   NAME

#S16     EXEC           45    45      TUE  7:08A   TEST.PUBS

JOBFENCE= 0; JLIMIT= 3; SLIMIT= 16
```

Type II:

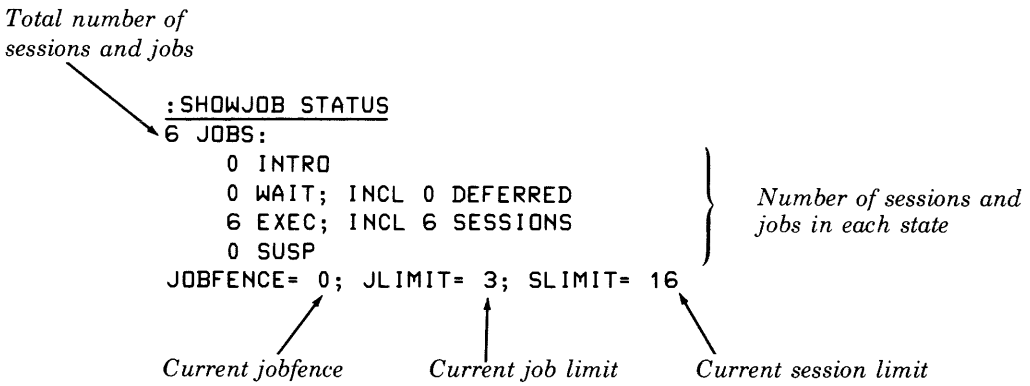
```
7 JOBS:
    0 INTRO
    0 WAIT; INCL 0 DEFERRED
    7 EXEC; INCL 7 SESSIONS
JOBFENCE= 0; JLIMIT= 3; SLIMIT= 16
```

Type III:

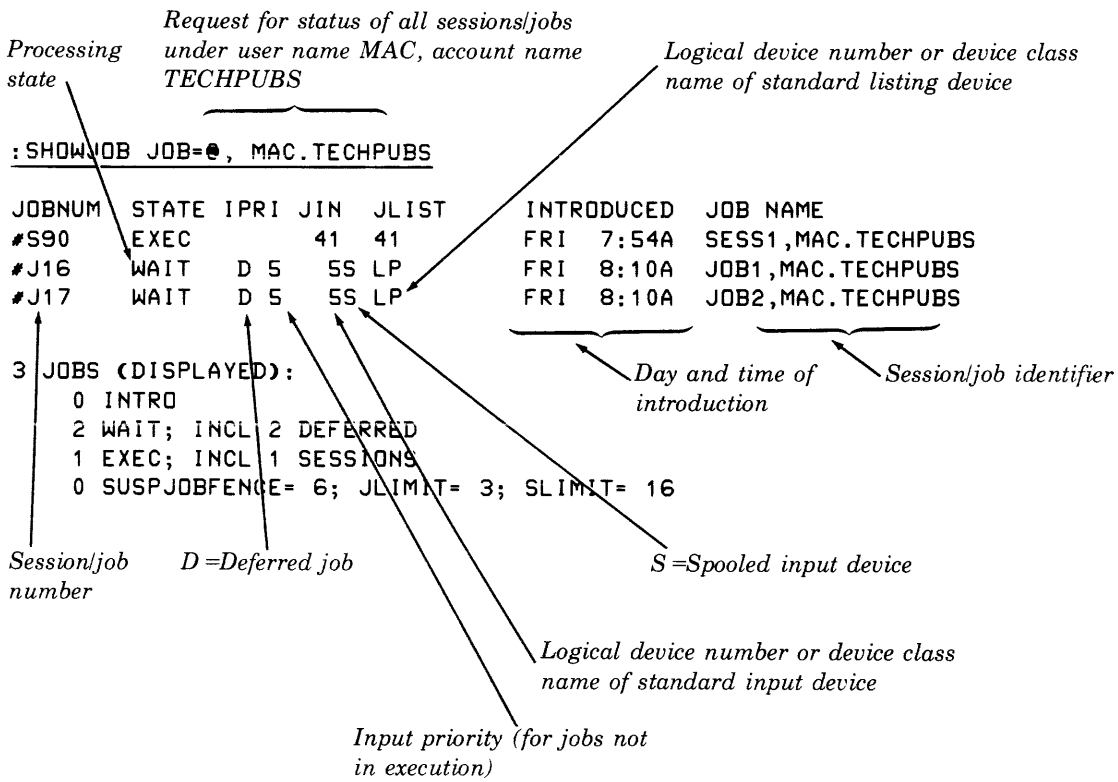
(Type I followed by Type II.)

EXAMPLE

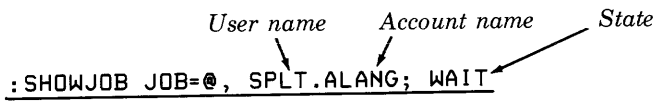
To determine the number of jobs and sessions in each processing state and the current jobfence and job/session limits, enter:



To produce a status report for all jobs and sessions running under the identifier MAC.TECHPUBS and show which jobs are streamed or spooled, enter:



If you wish to determine which jobs/sessions under your user name and account are in a particular state (such as WAIT), you can request a report on them in this way:




```
:SHOWJOB JOB=@, SPLT.ALANG; WAIT

JOBNUM  STATE IPRI JIN  JLIST    INTRODUCED  JOB NAME

#J392   WAIT:2  13   6S FASTLP   TUE  2:05P  SPLT.ALANG

JOBFENCE= 2; JLIMIT= 1; SLIMIT= 16
```

To get a report on all jobs and sessions in the system, enter:

```
:SHOWJOB

JOBNUM  STATE IPRI JIN  JLIST    INTRODUCED  JOB NAME

#S745   EXEC        29  29       TUE  2:53P  DL,SPL.ALANG
#S746   EXEC        26  26       TUE  2:53P  CLI.AOPSYS
#S747   EXEC QUIET  42  42       TUE  2:53P  MAC.TECHPUBS
#S748   EXEC        47  47       TUE  2:53P  FIELD.SUPPORT
#S749   EXEC        34  34       TUE  2:55P  SPLT.ALANG
#S750   EXEC        28  28       TUE  2:55P  MIKE.DCA
#J396   INTRO   13   6S FASTLP   TUE  2:55P  SPLT.ALANG

7 JOBS:
    1 INTRO
    0 WAIT; INCL 0 DEFERRED
    6 EXEC; INCL 6 SESSIONS
    0 SUSP
JOBFENCE= 2; JLIMIT= 1; SLIMIT= 16
```

:SHOWLOGSTATUS

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

SYNTAX

:SHOWLOGSTATUS *logid*

PARAMETERS

logid The logging identifier created by the :GETLOG command.

USE

Available	In Sessions?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	NO
Breakable?		NO

OPERATION

This command is used to list the status of currently running logging processes. The status includes the total number of records written by the process and the current number of users accessing the logging file.

This command gives the following information about all currently running logging processes:

LOGID — The identifier of the logging process.
USERS — The number of users.
STATE — Active or inactive.
RECORDS — The number of records in the log file.

EXAMPLE

To display the status of an open log file named LEN, enter:

```
:SHOWLOGSTATUS LEN
    LOGID          USERS      STATE      RECORDS
     LEN             0         INACT        160
```

Reports job/session status.

SYNTAX

```
:SHOWME
```

PARAMETERS

None

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		YES (Aborts)

OPERATION

The :SHOWME command reports the following information pertaining to your job or session:

Job/session number.

User job/session identity.

The message IN BREAK or NOT IN BREAK, indicating if :SHOWME was entered during a break or not.

MPE version.

Current time.

Log-on time.

Central processor time used by this job/session.

Amount of time job/session has been connected.

Standard input device number.

Standard list device number.

Current welcome message.

EXAMPLE

```
:showme
USER: #92,CARRI.USERS,RUTH       (NOT IN BREAK)
MPE VERSION: HP32002C.00.00
CURRENT: MON, AUG 25, 1980,  4:32 PM
LOGON:   MON, AUG 25, 1980,  9:46 AM
CPU SECONDS: 22          CONNECT MINUTES: 407
$STDIN LDEV: 75          $STDLIST LDEV: 75
```

Reports status of output devicefiles.

SYNTAX

```
                [#Onnn  ]
                [STATUS ]
:SHOWOUT
                [SP] [;item ] [;item ] [;item ]
```

PARAMETERS

#Onnn — Identifies particular output devicefile for which information is to be displayed. The information is displayed in Type I format, described under OPERATION. Default is a display of status information for all output device files.

STATUS — Request to summarize status information for all current output devicefiles. The information is displayed in Type II format, described under OPERATION. Default is a display of status information for all output device files.

SP — Request to display status information for currently spooled output devicefiles associated with the logon job or session. The information is displayed in Type III format, described under OPERATION. Default is a display of status information for all output device files.

item — Request to display status of all current output devicefiles. This parameter is written in the following format:

```
[          ldev      ]
[DEV= {               }]
[          classname ]

[          @J        ]
[          @S        ]
[JOB= { @            }]
[          [#]Jnnn   ]
[          [#]Snnn   ]

[ACTIVE                ]
[                      ]
[READY [,N]            ]
[      [,D]            ]
[                      ]
[OPENED                ]
[LOCKED                ]
```

Each set of brackets defines an item. You cannot use the same item more than once in the parameter list. The subparameters are:

ldev	Display status of output devicefile residing on device identified by logical device number *ldev*.
classname	Display status of output devicefiles residing on all devices having device class name *classname*.
@J	Display status of output devicefiles for all jobs.
@S	Display status of output devicefiles for all sessions.
@	Display status of output devicefiles for all jobs and sessions. JOB = @ is the default on the current console.
#Jnnn	Display status of all output devicefiles for specified job.
#Snnn	Display status of all output devicefiles for session indicated.
ACTIVE READY OPENED LOCKED	Display status of output devicefiles in specified state.
N	Display status of non-deferred READY devicefiles only.
D	Display status of deferred READY device-files only.

If information for only one devicefile is displayed, output is in Type I format; if information for more than one devicefile is displayed, output is in Type III format. (Format types are described under OPERATION.)

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		YES (Aborts)

OPERATION

Displays status information for one or more currently-defined output devicefiles. The information reflects the status at the time the command is entered and always appears on the standard list device. The information includes:

Logical device number or device class name of device.

Devicefile identification, in form *#Onnn*.

Job/session number (*jsnum*) of job/session using devicefile, if not used for READY or ACTIVE data. Otherwise, job/session name appears on line following standard device information.

Filename assigned to devicefile.

State:

ACTIVE	Spooled devicefile on disc is actually being written to a printer, plotter, or card punch.
READY	Devicefile on disc is ready for output.
LOCKED	READY, but system is using file with exclusive access.
OPENED	Devicefile on disc is being accessed by a program.

Forms message indicator (the letter Y), appearing only if a forms alignment message applies to this devicefile.

Approximate disc space currently used (in sectors), for spooled output devicefiles only.

Rank, indicating the order in which the file is entered in the system with respect to other files.

D, indicating a deferred file, for spooled devicefiles only.

Outputpriority, requested by user, for spooled devicefiles only (1 = lowest, 13 = highest, blank = current default priority).

Number of copies needed (#C), for spooled devicefiles only.

Output may appear in three possible formats:

Type I:

```
DEV/CL  DFID  JOBNUM  FNAME     STATE   FRM  SPACE  RANK  PRI  #C

45      #032  #S16    $STDLIST  OPENED
```

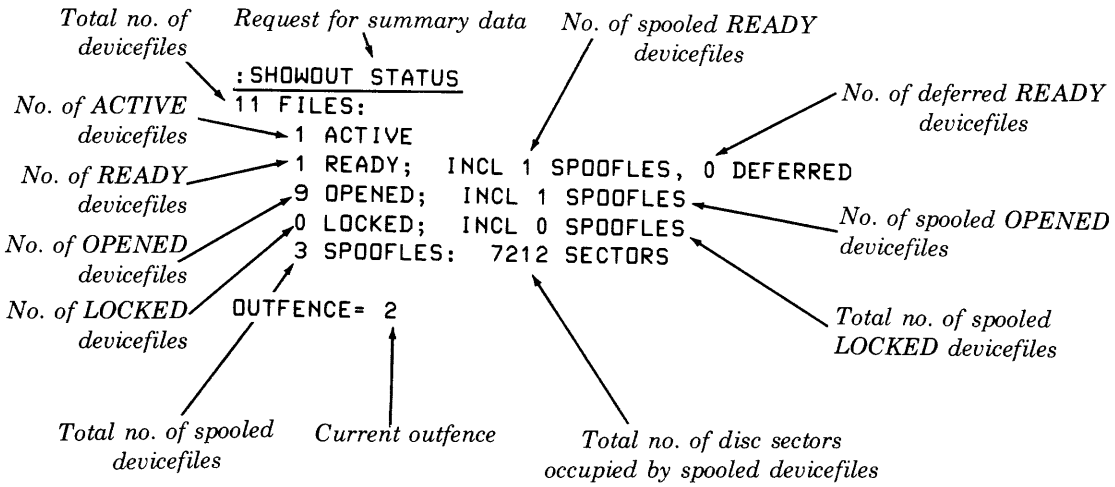
TYPE II:

```
19 FILES
   0 ACTIVE
   2 READY; INCLUDING 2 SPOOFLES, 2 DEFERRED
  17 OPENED; INCLUDING 1 SPOOFLE
   0 LOCKED; INCLUDING 0 SPOOFLES
   3 SPOOFLES: 1572 SECTORS
OUTFENCE = 6
OUTFENCE = 2 FOR LDEV 13
```

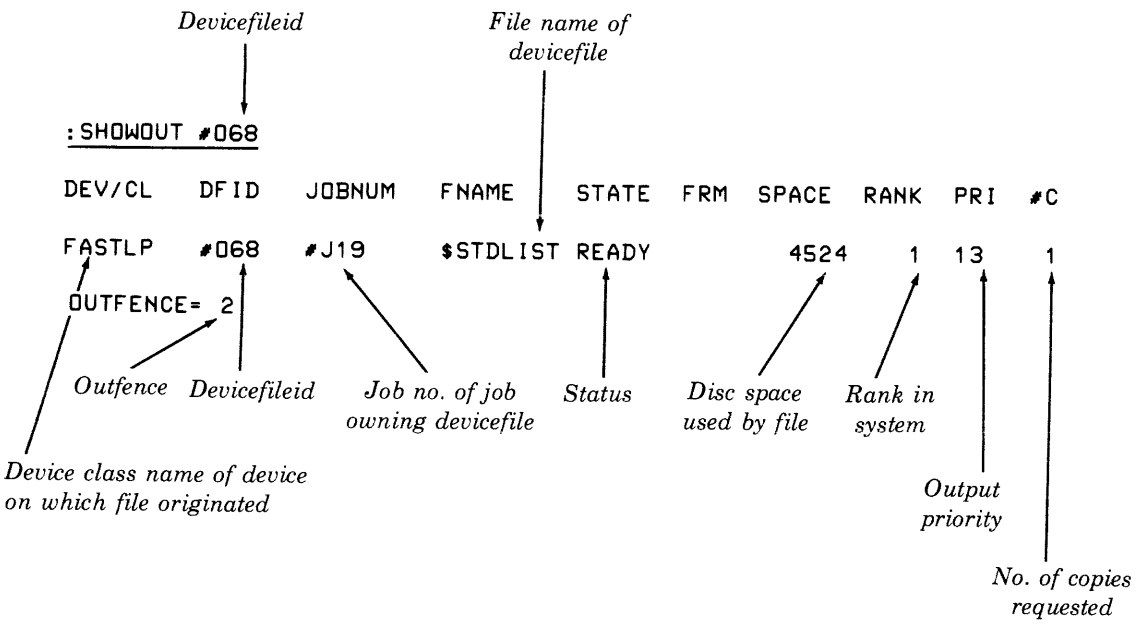
Type III:

(Type I followed by Type II.)

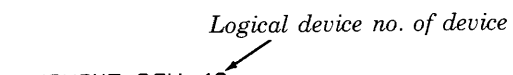
To display the total number of output devicefiles currently existing, the number of those that are spooled, and the states that they are in, enter:



To list information about an individual output devicefile, you can reference its devicefile identifier in the :SHOWOUT command:

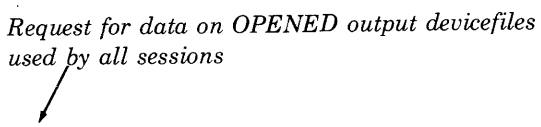


You can also request status of a device file by using the logical device number or device class name of the device for which the file is destined in the :SHOWOUT command:




```
:SHOWOUT DEV=43

DEV/CL    DFID     JOBNUM  FNAME      STATE   FRM  SPACE  RANK  PRI  #C

43        #089     #S37    $STDLIST   OPENED

OUTFENCE= 2
```

You can request displays of output devicefile information by various combinations of qualifications (devices, jobs/sessions, and states). As an example, suppose you wanted to list information about all OPENED output devicefiles used by all sessions in the system. You would enter:




```
:SHOWOUT JOB=@S; OPENED

DEV/CL    DFID     JOBNUM    FNAME      STATE   FRM  SPACE  RANK  PRI  #C

20        #090     #S38      $STDLIST   OPENED
26        #037     #S18      $STDLIST   OPENED
27        #095     #S42      $STDLIST   OPENED
32        #093     #S41      $STDLIST   OPENED
34        #064     #S26      $STDLIST   OPENED
42        #073     #S28      $STDLIST   OPENED
43        #089     #S37      $STDLIST   OPENED
50        #092     #S40      $STDLIST   OPENED
51        #036     #S17      $STDLIST   OPENED

9 FILES (DISPLAYED):
    0 SPOOFLES: 0 SECTORS

OUTFENCE= 2
```

:SHOWTIME

Prints current time and date.

SYNTAX

```
:SHOWTIME
```

PARAMETERS

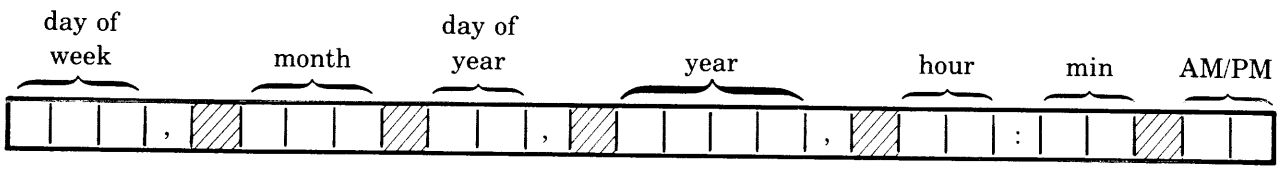
None

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

Prints current time and date, as indicated by system clock in the following format:



EXAMPLE

To display the time and date, enter;

```
:SHOWTIME
TUE, JUN 17, 1980, 11:08
```

Changes terminal operating speed.

SYNTAX

```
:SPEED newinspeed,newoutspeed
```

PARAMETERS

newinspeed New input speed, in characters per second. Must be 10, 14, 15, 30, 60, 120, or 240.

newoutspeed New output speed, in characters per second. Must be 10, 14, 15, 30, 60, 120, or 240.

NOTE

Newinspeed and *newoutspeed* must be the same.

USE

Available	In Session?	YES
	In Job?	NO
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

MPE automatically senses the input/output speed of a terminal when you log on at that terminal. If your terminal has speed-adjustment controls, you can change the input and output speeds after log on with the :SPEED command. Note that this command is not valid for terminals that operate on one speed only.

When the command is entered, MPE outputs the following message at the old output speed:

CHANGE SPEED AND INPUT MPE

Manually change the speed control on the terminal and verify the new speed by entering:

MPE

If the characters MPE cannot be verified, the system assumes that the terminal is to continue at the old speed. (To continue, you must reset the terminal to the old speed.)

You can also change the terminal speed programmatically by using the FCONTROL intrinsic. (See the MPE *Intrinsics Reference Manual*.)

EXAMPLE

To change the input and output speeds to 240 cps, enter:

```
:SPEED 240,240
```

MPE outputs:

```
CHANGE SPEED AND INPUT MPE
```

Manually change the speed and input MPE.

Compiles an SPL program.

SYNTAX

:SPL [*textfile*][,[*uslfile*][,[*listfile*][,[*masterfile*][,*newfile*]]]]

PARAMETERS

textfile Actual designator of input file from which source program is read. Can be any ASCII input file. Formal designator is SPLTEXT. Default is $STDIN.

uslfile Actual designator of USL file on which object program is written. Can be any binary output file. Formal designator is SPLUSL. If entered, this parameter must indicate a file created previously in one of four ways:

1. By saving a USL file (with the MPE :SAVE command) which was created by a previous compilation where the default value was used for the *uslfile* parameter.
2. By building the USL with the MPE Segmenter -BUILDUSL command. (See the *MPE Segmenter Reference Manual.*)
3. By creating a new USL file with the MPE :BUILD command with a *filecode* of USL or 1024.
4. By having the statement $CONTROL USLINIT in your program.

Default: $NEWPASS is assigned.

listfile Actual designator of file on which program listing is written. Can be any ASCII output file. Formal designator is SPLLIST. Default is $STDLIST.

masterfile Actual designator of master file which is merged against *textfile* to produce composite source. Can be any ASCII input file. Formal designator is SPLMAST. Default is that the master file is not read; input is read from *textfile*.

newfile Actual designator of merged *textfile* and *masterfile*. Can be any ASCII output file. Formal designator is SPLNEW. Default is that no file is written.

NOTE

The formal file designators used in this command (SPLTEXT, SPLUSL, SPLLIST, SPLMAST, SPLNEW) cannot be back referenced as actual file designators in the command parameter list. For further information see the OPERATION section of the :FILE command.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Compiles SPL program onto a User Subprogram Library (USL) file on disc. If *textfile* is not specified, MPE expects input from your standard input device.

EXAMPLE

To compile an SPL program entered from the job input device into an object program in the USL file $NEWPASS, and write the listing to your standard list device, enter:

```
:SPL
```

If the next command is one to prepare an object program, $NEWPASS can be passed to that command by referencing $OLDPASS.

To compile an SPL program residing on the disc file SOURCE into an object program on the USL file OBJECT, with program listing generated on the disc file LISTFL, enter:

```
:SPL SOURCE,OBJECT
:SAVE OBJECT
```

ADDITIONAL DISCUSSION

SPL Reference Manual.

Compiles, prepares, and executes an SPL program.

SYNTAX

```
:SPLGO [textfile][,[listfile][,[masterfile][,newfile]]]
```

PARAMETERS

textfile Actual designator of input file from which source program is read. Can be any ASCII input file. Formal designator is SPLTEXT. Default is $STDIN.

listfile Actual designator of file on which program listing is written. Can be any ASCII output file. Formal designator is SPLLIST. Default is $STDLIST.

masterfile Actual designator of master file which is merged against *textfile* to produce composite source. Can be any ASCII input file. Formal designator is SPLMAST. Default is that the master file is not read; input is read from *textfile*.

newfile Actual designator of merged *textfile* and *masterfile*. Can be any ASCII output file. Formal designator is SPLNEW. Default is that no file is written.

NOTE

The formal file designators used in this command (SPLTEXT, SPLLIST, SPLMAST, SPLNEW) cannot be back referenced as actual file designators in the command parameter list. For further information see OPERATION section of :FILE command.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Compiles, prepares, and allocates/executes an SPL program. If *textfile* is omitted, MPE expects input from your standard input device. This command creates a temporary User Subprogram Library (USL) file ($NEWPASS) that you cannot access, and a temporary program file that you can access under the name $OLDPASS.

EXAMPLE

To compile, prepare, and execute an SPL program entered from your standard input device, with the program listing sent to your standard list device, enter:

```
:SPLGO
```

To compile, prepare, and execute an SPL program read from the disc file SOURCE and send the resulting program listing to the disc file LISTFL, enter:

```
:SPLGO SOURCE,LISTFL
```

ADDITIONAL DISCUSSION

SPL Reference Manual.

Compiles and prepares an SPL program.

SYNTAX

```
:SPLPREP [textfile][,[progfile][,[listfile][,[masterfile][,newfile]]]]
```

PARAMETERS

textfile Actual designator of input file from which source program is read. Can be any ASCII input file. Formal designator is SPLTEXT. Default is $STDIN.

progfile Actual designator of program file on which program segments are written. Can be any binary output file with *filecode* of PROG or 1029. If entered, this parameter must indicate a file created on one of two ways:

1. By creating a new program file with the MPE :BUILD command with a *filecode* of PROG or 1029 and a *numextents* parameter value of 1.

2. By specifying a non-existent file in the *progfile* parameter, in which case a job/session temporary file of the correct size and type is created.

Default: $NEWPASS is assigned.

listfile Actual designator of file on which program listing is written. Can be any ASCII output file. Formal designator is SPLLIST. Default is $STDLIST.

masterfile Actual designator of master file which is merged against *textfile* to produce composite source. Can be any ASCII input file. Formal designator is SPLMAST. Default is that the master file is not read; input is read from *textfile*.

newfile Actual designator of merged *textfile* and *masterfile*. Can be any ASCII output file. Formal designator is SPLNEW. Default: no file is written.

NOTE

The formal file designators used in this command (SPLTEXT, SPLPROG, SPLLIST, SPLMAST, SPLNEW) cannot be back referenced as actual file designators in the command parameter list. For further information see OPERATION section of :FILE command.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	NO
	Programmatically?	NO
Breakable?		YES (Suspends)

OPERATION

Compiles and prepares an SPL program onto a program on disc. If *textfile* is not specified, MPE expects input from your standard job/session input device. The User Subprogram Library (USL) file created during compilation is a temporary file passed directly to the preparation mechanism; you can access it under the name $OLDPASS only if the program file is not $NEWPASS. (Note that if you want to save the compiled USL, you must specify some other file as the second positional parameter in this command and then use a :SAVE command. See the third example below).

EXAMPLE

To compile and prepare an SPL program entered through the job input device, onto the file $NEWPASS, with the listing printed on the job list device, enter:

```
:SPLPREP
```

If the next command is one to execute the program, the file $NEWPASS can be passed to this command by referencing $OLDPASS.

To compile and prepare an SPL source program from a source file named SFILE into a program named MYPROG, with the resulting listing sent to the job/session list device, enter:

```
:SPLPREP SFILE,MYPROG
```

Note that when the first positional parameter is omitted, *textfile* is read from the standard input device:

```
:SPLPREP,FILEZ      Compiles into $NEWPASS, prepares into FILEZ.
:SAVE $OLDPASS, NUSL    Saves USL, changing file name from $OLDPASS
                        (formally $NEWPASS) to NUSL.
```

ADDITIONAL DISCUSSION

SPL Reference Manual.

:STORE

Stores disc files onto magnetic tape or serial disc.

SYNTAX

```
:STORE [fileset][,fileset][,. . .];storefile
                                          [;DATE>=moddate]
       [;SHOW][;FILES=maxfiles][          [;DATE<=accdate]]
```

PARAMETERS

fileset Signifies a set of files to be stored. Default is @. This parameter is of the form:

filedesignator [.*groupdesignator* [.*acctdesignator*]]

fileset can be entered in any of the following formats and may use wild card characters, in any order, as replacements.

file.group.account	Store specified file in specified group and account.
file.group	Store specified file in specified group under log-on account.
file	store specified file under log-on group.
@.group.account	Store all files in specified group and account.
@.group	Store all files in specified group under log-on account.
@.@.account	Store all files in all groups in specified account.
@	Store all files in log-on group. Default.
@.@	Store all files in all groups under log-on account.
@.@.@	Store all files in system.
file.@.account	Store specified file in any group of specified account.

NOTE

The characters @, #, and ? can be used as wild card characters in the *fileset* parameter. These wild card characters have the following meanings:

@ — specifies zero or more alphanumeric characters.

— specifies one numeric character.

? — specifies one alphanumeric character.

The characters can be used as follows:

n @	Store all files starting with the character *n*.
@n	Store all files ending with the character *n*.

n @*n*	Store all files starting with the character *n* and ending with the character *n*.
n # #.. #	Store all files starting with the character *n* followed by up to seven digits.
?*n* @	Store all files whose second character is *n*.
n?	Store all two-character files starting with *n*.
?*n*	Store all two-character files ending with *n*.

storefile Name of destination tape or serial disc file onto which the stored files are to be written. Can be any magnetic tape or serial disc from the output set. This file must be named in a :FILE command and back referenced as shown in the following examples:

```
:FILE STORTAPE;DEV=TAPE
:STORE @.@;*STORTAPE
```

or

```
:FILE STORDISC;DEV=SDISC
:STORE @.@;*STORDISC
(REQUIRED PARAMETER)
```

SHOW Request to list names of files stored. Default is total number of files stored, names of files not stored, and number of files not stored are listed. The listing is sent to $STDLIST (formal designator is SYSLIST) unless a :FILE command is entered to send the listing to some other device. For example,

```
:FILE SYSLIST;DEV=LP
```

entered before the :STORE command would send the listing to a line printer.

maxfiles Maximum number of files that may be stored. Default is 4000.

<=accdate Store only files not accessed since date specified.

>=moddate Store only files which have been modified on or after date specified.

When no date is specified all files specified by the *filesets* will be stored. The date is expressed *mm/dd/yy*.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		YES (Aborts)

OPERATION

Enables you to copy offline onto magnetic tape or serial disc a particular user disc file or set of files.

Files currently open for output, input/output, update or append, and files currently being stored or restored, cannot be acted upon by the :STORE command. However, files loaded into memory (containing currently running programs) and files open for input only can be stored, since their contents cannot be altered! You can only store those files whose home volume sets are mounted.

Users with System Manager or System Supervisor capability can store any user file in the system. Users with Account Manager capability can store any file in the account (but cannot dump those with negative file codes unless they have Privileged Mode capability also).

Before entering a :STORE command, you must identify *storefile* as a magnetic tape or serial disc file with a :FILE command written in the following format:

:FILE *formaldesignator*[=*filereference*];DEV=*device*[;REC=*recsize*][;BUF / ;NOBUF]

device — Must indicate the device class name or logical unit number of a magnetic tape or serial disc unit.

recsize — Sets the size of the blocks on the store tape. Record size may be any multiple of 256 words between 256 and 4096 words for tapes, and 256 to 8192 words for serial disc. Default is 4096 for non-programmatic calls to :STORE, and 1024 words for programmatic calls.

BUF — Causes the tape and disc I/O to overlap.

NOBUF — Will not allow the tape and disc I/O to overlap. It is suggested that BUF always be used with tape, and NOBUF always be used with serial disc. Default is BUF. NOBUF must be specified if *recsize* is greater than 4096.

Other parameters for storefile are supplied by the :STORE command. If you attempt to supply them, MPE will reject the :STORE command.

NOTE

To use a device class name when storing to serial disc, you must have answered "YES" to the SYSDUMP question of "SERIAL DISC CLASS?". If not, you will receive file system error #46, "OUT OF DISC SPACE". The same requirement applies to the :RESTORE command, in this case receiving file system error #52, "NON-EXISTENT DEVICE".

While a file is being dumped, it is locked by MPE so that it cannot be altered or deleted until safely copied to tape or serial disc. If a job/session running a :STORE/:RESTORE function is aborted by yourself or the console operator, those files not yet stored or restored will be unlocked during the processing of the abort. If the BREAK. Key is pressed furing a store operation the operation stops after storing the current file and further output is suppressed.

EXAMPLE

To copy all files in the group GP4X in your logon account, to a tape file named BACKUP, enter:

```
:FILE BACKUP;DEV=TAPE
:STORE@.GP4X;*BACKUP;SHOW
```

The console operator receives a request to mount the tape identified as BACKUP. A listing of the files copied appears on your standard list device.

If a file is in use by the system, the file name is displayed, along with the message BUSY.

Spools batch jobs or data from session or job.

SYNTAX

```
:STREAM [inputfile][,character]
```

PARAMETERS

inputfile File designator of ASCII input file from which jobs or data are spooled to disc. May be a disc file created with the Editor and containing commands or data; or a file input from a card reader, tape unit, or terminal. Records in this file cannot exceed 255 characters. All character positions within each record are significant. Default is $STDIN.

character Character used in place of colon (:) to identify MPE commands within *inputfile.* When inputfile is entered on device configured to accept jobs sessions or data input via the :DATA command, this character can be any ASCII special (non-alphanumeric) character except a colon. Default is exclamation point (!).

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		YES (Command and partially-streamed job abort.)

OPERATION

This command allows you to initiate jobs while in an interactive session by allowing you to make up your job from your terminal or to read these records from a card or tape file. Alternatively, you can read these records from a disc file previously prepared via the Editor. When the job is read, MPE spools it onto a disc file, assigns it a job number, and processes it independently as a completely separate entity from your session. In the meantime, MPE allows you to continue with your session.

You can only initiate jobs in this way if the Console Operator or a user who has been given operator capabilities has enabled the MPE Streaming Facility by entering the console command :STREAMS. This command also specifies a streaming device, which (to MPE) appears to be the source of your job input regardless of what device you actually use for this input. Thus, the job number assigned by MPE and the listing generated by the job appear on the standard listing device that corresponds to the streaming device, rather than on your terminal.

Note that to stream data to disc the input device used must be configured to accept the :DATA command.

When you enter :STREAM with the terminal as the default input device, whether during a session or job, MPE prompts you for input by displaying a greater than (>) character. When you enter :STREAM for another input device, MPE does not print the prompt character. In any case, you terminate *inputfile* by entering the :EOD command. At the terminal, this halts prompting for job input and returns control to your job/session.

Begin each job in *inputfile* with the :JOB command and terminate it with the :EOJ command. Begin each data file with the :DATA command and terminate this file with the :EOD command. Begin all commands with an appropriate substitute (other than colon) character, as in !JOB or #DATA. When *inputfile* is spooled to disc, MPE replaces the substitute command identifier with a colon so that command reports are properly interpreted when executed.

Following the !EOJ command that terminates the job, MPE assigns each job a unique job number and prints this number on the standard list device. MPE also assigns each job a preset priority unless you specify otherwise in the :JOB command, and processes the job independent of the initiating job. Regardless of which device you use to submit the *inputfile,* all jobs in that file are treated as though they originated on a unique streaming device designated by the Console Operator (with the :STREAMS command). The listing for each spooled job is written to the standard list device that corresponds to the streaming device, unless you use the OUTCLASS= parameter of the :JOB command to direct the listing to another device.

To terminate job input, enter a colon (:). In response, MPE terminates prompting for batch job input and instead prompts you for another MPE command:

>:	*Denotes end of batch job input*
:	*MPE prompts for next command*

Pressing the Break key aborts the :STREAM command and any job currently being entered through the command, and incompletely spooled disc space is returned to the system.

If you make an error while entering the JOB command, you receive an error message on your job listing device. The Console Operator, however, receives no indication of either the job or the error.

EXAMPLE

To use the Editor to enter a job into a disc file and then stream the job from that file, you must name the input file in the :STREAM command:

```
:STREAM DISCFILE
```

Name of file on disc that contains job input

If you use a character other than an exclamation point as the substitute command identifier in your job input, you must identify that character in the :STREAM command. Because you enter this character as the second positional parameter in this command, you must always precede it with a delimiting comma, even when you omit the input file name (the first parameter):

```
                    Delimiting comma
                   /
:STREAM ,* <------------ Asterisk used as substitute command identifier
>*JOB MAC.TECHPUBS
>*FORTGO MYPROG
>*EOJ
#J74
>:
:
```

If your job input file contains subsystem commands, such as commands directed to the Editor, do not enter any command identifier at the beginning of these commands. For instance, when using the Editor, enter the subsystem commands as follows:

```
:STREAM                          Indicates streaming.
>#JOB MAC.TECHPUBS               Initiates job.
>#EDITOR                         Invokes Editor.
>TEXT MYFILE
>ADD                             Subsystem commands (without # as
      .                          subsystem command identifier).
      .
      .
>KEEP MYFILE,UNN
>END
>#EOJ                           Terminates job.
  #J76                          Job number.
>:                              Terminates job input.
:                               Prompt for new MPE command.
```

If you wish the job listing to appear on a device other than the standard listing device associated with the streaming device, you can specify this other device in the JOB command in this way:

```
:STREAM
>!JOB MAC.TECHPUBS;OUTCLASS=12
                              \
                    Logical device number of alternate device
```

To initiate, from a session, a batch job that compiles, prepares, and executes the FORTRAN program MYPROG, enter:

```
:STREAM
>!JOB SOFT.PUBS
>!FORTGO MYPROG
>!EOJ
  #J73    (Job number returned by MPE)
>!EOD
```

ADDITIONAL DISCUSSION

Using Files Manual.

Sends a message to another job or session.

SYNTAX

```
                [#]Jnnn
:TELL { [#]Snnn                          }[;][text]
        [jsname,]username.acctname
        @
        @.acctname
        @J
        @S
```

PARAMETERS

#Jnnn — Job number (as assigned by MPE) for the job that is to receive the message. (Obtained by entering the :SHOWJOB command).

#Snnn — Session number (as assigned by MPE) of the session that is to receive the message. (Obtained by entering the :SHOWJOB command).

jsname,username.acctname — Names of the job/session and user to receive the message, and the account name under which they are running. (This parameter is the same as the job or session identity entered with the :JOB or :HELLO command.) (Obtained by entering the :SHOWJOB command).

If several users are running under the same job/session identity, MPE will send the message to all of them.

@ — All users.

@.acctname — All users under acctname.

@J — All jobs.

@S — All sessions.

text — Message text, consisting of any string of ASCII characters. The default is that no message is printed; however, MPE prints the FROM message as follows:

FROM/*sessionid*

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

Transmits a message from the sender's session to one or more jobs/sessions currently running. The message appears on the receiving job/session list device.

If a message is sent to a terminal that is currently interacting with a program, MPE queues the message as high as possible among the current input/output requests but does not interrupt reading or writing in progress. If the job/session or user designated to receive the message is not running, or the job is spooled, the transmitting job/session receives a system message indicating this. MPE rejects the :TELL command if the receiving device is operating in the quiet mode (see :SETMSG command) and informs the sender with:

USER *username* NOT ACCEPTING MESSAGES

EXAMPLE

To send a message to a user identified as BROWN, logged on under account A, running a session named BROWNSES, enter:

:TELL BROWNSES,BROWN.A; USE PROGA, NOT PROGB *message text*

To send a message to all users logged on in account A:

:TELL @.A; PLEASE LOG OFF

:TELLOP

Sends message to Console Operator.

SYNTAX

```
:TELLOP [text]
```

PARAMETERS

text Message text, consisting of any string of ASCII characters. Default is that no message is printed; however, MPE prints the FROM as follows:

FROM/*sessionid*

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	YES
Breakable?		NO

OPERATION

Sends a message from the sender's job/session to the operator's console. The message text appearing on the operator's console, preceeded by the time it was transmitted and your job/session number. Like messages transmitted between users (:TELL command), this message is printed as soon as possible without interrupting any console input/output currently in progress.

EXAMPLE

To send a message to the operator instructing him to mount a tape, you could enter:

```
:TELLOP PLS MOUNT MYTAPE, VERSION 1
```

:VSUSER

Prints a listing of all users of a currently mounted volume set.

SYNTAX

```
:VSUSER [vsname]
```

PARAMETERS

vsname A fully-qualified volume set name.

USE

Available	In Session?	YES
	In Job?	YES
	In Break?	YES
	Programmatically?	NO
Breakable?		NO

OPERATION

The :VSUSER command lists all users who have explicitly or implicitly mounted a volume set.

EXAMPLE

```
:VSUSER
```

ADDITIONAL DISCUSSION

MPE System Manager/System Supervisor Manual.

USER DEFINED COMMANDS

SECTION III

MPE allows you to define your own commands which can be used to fit your specific needs. A user-defined command, or UDC, is a command built from standard MPE commands. When you combine several MPE commands into a single list and assign a name to that list, you create a UDC. In this way it is possible to enter a UDC name in response to the MPE prompt (:) and cause several MPE commands to be executed. A user-defined command may be used in most places where an MPE command may be used, including another UDC. However, UDCs cannot be executed through the COMMAND intrinsic.

UDCs may be created by any user for personal use, by the Account Manager for use by anyone in the account, and by the System Manager for use by any user on the system. When identically named UDCs occur at different levels, the user UDC takes precedence over the account level UDCs which in turn take precedence over the system level UDCs. (An exception to this rule is discussed in Nesting UDCs, this section).

Note that to establish account or system level UDCs you must have special capabilities. For information on these capabilities, see the *System Manager/System Supervisor Reference Manual.*

SYNTAX OF USER-DEFINED COMMANDS

A user-defined command consists of two parts: a *header* section containing control information, and a *body* consisting of MPE commands.

HEADER

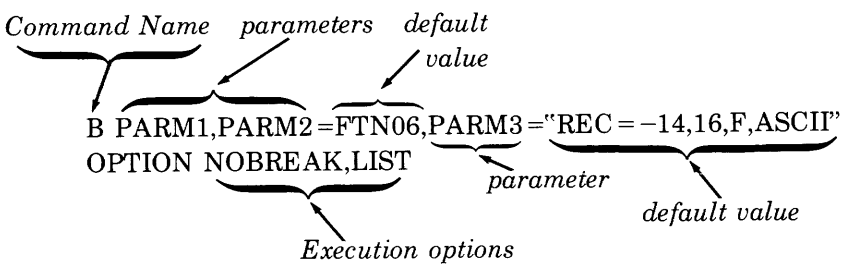
The control information contained in the header section of a UDC consists of:

1. The command name, composed of alphanumeric characters (16 maximum) beginning with a letter. You cannot use the reserved word RFA as a UDC name.

2. Parameters and their optional default values (maximum 16).

3. Execution options (LIST/NOLIST, BREAK/NOBREAK, LOGON/NOLOGON, HELP/NOHELP).

An example of a UDC header section is



Any line of a UDC header may be continued up to 320 characters, and each continuation line, when needed, must contain an ampersand (&) as the last non-blank character.

BODY

The body contains a set of MPE commands which are executed when the UDC name is entered in place of an MPE command. Any line in a UDC body may be continued using the ampersand (&) character up to a maximum of 320 characters. The following is the body of user-defined command "B".

```
FILE INPUT=!PARM2,OLD
FILE LISTFILE=!PARM1,NEW;!PARM3;DISC=1000,16,2
RUN ABC
```

Thus, the complete user-defined command B is:

```
Header ---> B PARM1,PARM2=FTN06,PARM3="REC=-14,16,F,ASCII"
            OPTION NOBREAD,LIST
Body   ---> FILE INPUT=!PARM2,OLD
            FILE LISTFILE=!PARM1,NEW;!PARM3;DISC=1000,16,2
            RUN ABC
```

USING UDC's

The first line (including continuation lines) in the header of a UDC supplies to the Command Interpreter the name of the command and any parameters and their defaults. Thus an absolutely minimal first line could be

```
C
```

In this example, the command would be named C and would have no parameters. The name given a parameter in the header of a UDC is the formal name by which the parameter is known in the body of the command. Any occurance of this parameter in the command body preceeded by an exclamation point (!) signifies that MPE should substitute the value of that parameter into the command. Thus, in the previous example of UDC "B", the line

```
FILE INPUT=!PARM2,OLD
```

informs MPE that you wish to substitute FTN06 for !PARM2

The body part of a user-defined command may contain MPE commands and other user-defined commands, but may not contain any non-command material such as data for MPE subsystems or user programs.

NOTE

The :REDO command is not allowed from within a UDC. Using :DATA, :JOB or :HELLO within a UDC will cause your current job or session to log off.

Notice that the command body of UDC "B" contains only the MPE commands :FILE and :RUN, and parameters for these commands:

```
FILE INPUT=!PARM2,OLD
FILE LISTFILE=!PARM1,NEW;!PARM3;DISC=1000,16,2   ---->  Body of UDC "B"
RUN ABC
```

To execute a UDC, the user enters the command name followed by a parameter list. The parameter list supplies values for the parameters defined in the UDC command header. Required parameters are those for which no default values are supplied in the command header. If a value for a required parameter is not supplied in the parameter list, an error is reported and the UDC is not executed. If there are no required parameters in the UDC header, a parameter list does not have to be supplied when the command is executed. In the previous example, the parameters PARM2 and PARM3 will be passed to the command body either as default values as specified in the command header, or as values entered in the parameter list with the command. The value for PARM1 must be supplied in the parameter list.

The parameters DISC=1000,16,2 (for the FILE LISTFILE command) and ABC (for the RUN command), of course, cannot be changed by the user-entered command call. Thus, if the command

```
:B FTN77
```

is entered, the effect is the same as if the following MPE commands had been entered:

```
:FILE INPUT=FTN06,OLD
:FILE LISTFILE=FTN77,NEW;REC=40,16,F,ASCII;DISC=1000,16,2
:RUN ABC
```

When supplying values for UDC parameters you can use either *keyword* parameters or *positional* parameters, but not both at the same time.

A *keyworded* parameter is one in which the formal name of the parameter is used, and may appear in any position in the parameter list. A *positional* parameter is one in which the formal name of the parameter is not used. Instead, a substitute name or value is used in the same relative position in the parameter list as the parameter's formal name appears in the parameter declaration.

The following are examples of different parameter lists for a UDC with the header

```
B PARM1,PARM2=FTN06,PARM3="REC=40,16,F,ASCII"
```

EXAMPLE 1:

```
:B FTN05
```

The *required* parameter PARM1 (specified) would have the value FTN05; PARM2 (not specified) its default value FTN06; and PARM3 (not specified) its default value REC=40,16,F,ASCII.

EXAMPLE 2:

```
:B FTN05,FTN77
```

PARM1 would have the value FTN05, and PARM2 the value FTN77. The default value for PARM2 would not be used because the call contained a new value. The default value (REC=40,16,F,ASCII) would be used for PARM3 because PARM3 was not specified. Both FTN05 and FTN77 are positional parameters because they substitute for PARM1 and PARM2.

EXAMPLE 3:

```
:B PARM2=FTN77,PARM1=FTN05
```

Example 3 illustrates entering the command B with *keyworded* parameters. Note that this call is identical in effect to the call in Example 2.

EXAMPLE 4:

```
:B FTN05,,"REC=-88"
```

Example 4 illustrates the use of quote marks when a parameter containing embedded blanks or special characters is used. Also note that the second parameter is not specified (denoted by the adjacent commas), thus the default value FTN06 is used for PARM2.

Usually an exclamation point (!) in the body of a UDC must be followed by a formal parameter name. If a parameter name does not follow the exclamation point, then an error is reported to the user.

The user can, however, change the meaning of the exclamation point if, for example, an exclamation point is part of his UDC and he does not wish it to signify a parameter value. When exclamation points occur in a pair (!!), the pair is replaced by a single exclamation point and MPE does not treat the characters which follow as a parameter name. If the number of consecutive exclamation points is greater than two, then each pair is replaced by a single exclamation point. If the number of points is odd, MPE expects a parameter name to follow the last exclamation point. For example:

```
TESTUDC PARM
OPTION LIST
COMMENT "!!!!PARM"
COMMENT "!!!!!PARM"
```

If the user executes this UDC with ":TESTUDC ABC", the two comments will be printed as:

```
COMMENT "!!PARM"
COMMENT "!!ABC"
```

In previous versions of MPE, a UDC parameter name had to be terminated by a non-alphanumeric character in the body of a UDC to allow the UDC parser to find the end of the name. Now the UDC parameter can be delimited by the use of double quotes (") enabling the user to insert the parameter into the middle of another string.

Consider, for example, that a user has three application programs that he compiles and preps daily:

SOURCE	USL	PROGRAM
APPL1SRC	APPL1USL	APPL1
APPL2SRC	APPL2USL	APPL2
APPL3SRC	APPL3USL	APPL3

Assuming a file equation :FILE SLP;DEV=SLOWLP a UDC could be set up that would easily combine the steps to prepare any of the above three source files into a program file:

```
COMPAPPL     MODULE,LISTING=*SLP
OPTION LIST                                          quotes used to identify parameter
PURGE APPL!MODULE
PURGE APPL!"MODULE"USL
BUILD APPL!"MODULE"USL;CODE=USL;DISC=255,16
SPL APPL!"MODULE"SRC,APPL!"MODULE"USL,!LISTING       parameter delimited
PREP APPL!"MODULE"USL,APPL!MODULE;PMAP                        by semicolon
SAVE APPL!MODULE                                  parameter delimited by a blank
```

After adding the COMPAPPL UDC to the catalog, you can compile and prepare Application 1 by entering

```
:COMPAPPL1
```

Entering this UDC is the same as entering the following MPE commands:

```
:PURGE    APPL1
:PURGE    APPL1USL
:BUILD    APPL1USL;CODE=USL;DISC=255,16
:SPL      APPL1SRC,/APPL1USL,*SLP
:PREP     APPL1USL,/APPL1;PMAP
:SAVE     APPL1
```

OPTIONS

The following options can be specified in a UDC declaration. The underlined options are defaults and need not be specified.

LIST/<u>NOLIST</u> — The LIST option of a user-defined command allows the text of the body part of the UDC, as modified by its parameters and other variables, to be listed on the standard list device. Also, with the LIST option, errors are reported with a caret (^) under the error. Default is NOLIST.

EXAMPLE OF **LIST** OPTION:

```
B PARM1,PARM2=FTN06,PARM3="REC=14,16,F,ASCII"
OPTION LIST
FILE INPUT=!PARM2,OLD
FILE LISTFILE=!PARM1,NEW;!PARM3;DISC=1000,16,2
RUN ABC

:B FTN27
FILE  INPUT=FTN06,OLD
FILE  LISTFILE=FTN27,NEW;REC=14,16,F,ASCII;DISC=1000,16,2
RUN ABC

THIS IS PROGRAM ABC, AN EXAMPLE
RUN BY A USER-DEFINED COMMAND

END OF PROGRAM
:
```

BREAK/NOBREAK If NOBREAK is specified, the command will be non-breakable.

If the NOBREAK option is not specified, all breakable commands within the UDC are breakable.

If a non-program command is broken (see table 1-1), the UDC terminates. If a program command is broken, and you use the :RESUME command to resume the program, the UDC will also be resumed. (An exception to this is if the :SETCATALOG command is used while you are in BREAK. In this case, the UDC will not be resumed. See Using SETCATALOG, this section.)

Default is BREAK.

NOTE

The CAUSEBREAK intrinsic overrides the UDC NO-BREAK option; a program containing CAUSEBREAK will break even if executed from a UDC which specifies the NOBREAK option.

EXAMPLE OF **NOBREAK** OPTION:

```
:EDITOR
HP32201A.7.0H EDIT/3000  THU, SEP 22, 1977,  3:36 PM
(C) HEWLETT-PACKARD CO. 1976
/T UDC1;L ALL
    1      S
    2      SHOWJOB
/ <---------------------------- Break
:SETCATALOG UDC1
:S
```

```
JOBNUM  STATE IPRI JIN  JLIST    INTRODUCED  JOB NAME

*S55    EXEC        20  20       THU  2:49P  FIELD.SUPPORT
*S57    EXEC        52  52
:SETCATALOG                                  Break Hit Here
:RESUME
READ PENDING        must be done so that UDC1 can be modified & saved.
A 1.1               (See ENTERING UDCs INTO AN EDIT FILE.)
    1.1   OPTION NOBREAK
    1.2   //
...
/L ALL
    1     S
    1.1   OPTION NOBREAK
    2     SHOWJOB
/K UDC1
UDC1 ALREADY EXISTS - RESPOND YES TO PURGE OLD AND THEN KEEP
PURGE OLD?Y
/
:SETCATALOG UDC1                  Break Hit Here
:S                                To NO Effect

JOBNUM  STATE IPRI JIN  JLIST    INTRODUCED  JOB NAME

*S55    EXEC        20  20       THU  2:49P  FIELD.SUPPORT
*S57    EXEC        52  52       THU  2:50P  BOB.DATAMGT
*S59    EXEC       100 100       THU  2:59P  YU.UTILITY
*S65    EXEC       110 110       THU  3:24P  RICH.COBOL74
*S61    EXEC       111 111       THU  3:01P  SMITH.COBOL74
*S62    EXEC       102 102       THU  3:10P  STEVE.PASCAL
*S63    EXEC        25  25       THU  3:12P  ED.MPE
*S66    EXEC        54  54       THU  3:25P  MIKE.DATAMGT

8 JOBS:
    0 INTRO
    0 WAIT; INCL 0 DEFERRED
    8 EXEC; INCL 8 SESSIONS
    0 SUSP
JOBFENCE= 2; JLIMIT= 2; SLIMIT= 16
```

LOGON/NOLOGON The LOGON option specifies that the UDC will be executed at log-on automatically.

Only one log-on UDC is executed for a user. If UDC files with option LOGON exist at more than one level, the user level log-on UDC takes precedence over the account level log-on UDC, which in turn takes precedence over the system level log-on UDC. If more than one log-on UDC has been defined at the same level, and no log-on UDCs exist at a lower level, the first one encountered in the file(s) is executed.

Default is NOLOGON.

EXAMPLE OF **LOGON** OPTION:

```
    1     S
    2     OPTION LOGON
    3     SHOWJOB
    4     ...
/E

:HELLO BARBARA.USERS

CPU=10. CONNECT=7. THU, JUL 12, 1979,  9:43 AM

HP 3000 / MPE III B.00.02. THU, JUL 12, 1979,  9:13 AM


JOBNUM  STATE IPRI JIN  JLIST     INTRODUCED  JOB NAME

#S55    EXEC        20  20        THU  6:43A  FIELD.SUPPORT
#S57    EXEC        52  52        THU  6:45A  BARBARA.USERS
#S59    EXEC       100 100        THU  9:32A  AD.CRESSMAN
#S68    EXEC        43  43        THU  9:59A  PAUL.REYNOSO

4 JOBS:
    0 INTRO
    0 WAIT; INCL 0 DEFERRED
    4 EXEC; INCL 4 SESSIONS
    0 SUSP
```

HELP/NOHELP

The HELP subsystem can be used to obtain information about user-defined commands unless NOHELP is specified as an option. The Command Interpreter locates the appropriate catalog entry for the specified UDC and prints the complete command (header and body).

You cannot use the HELP subsystem to obtain information on a UDC from within a subsystem or program.

Default is HELP.

EXAMPLE OF USING THE **HELP** SUBSYSTEM FOR A UDC:

```
:HELP B
USER DEFINED COMMAND:

B PARM1,PARM2=FTN06,PARM3="REC=14,16,F,ASCII"
OPTION LIST
FILE INPUT=!PARM2,OLD
FILE LISTFILE=!PARM1,NEW;!PARM3;DISC=1000,16,2
RUN ABC
:
```

If you specify both the NOLIST and NOHELP options and an error occurs, the error will be reported, but the line containing the error will not be echoed.

USING THE :SETCATALOG COMMAND

The :SETCATALOG command is used to inform MPE that a specified file name(s) contains user defined commands. The Command Interpreter then searches the file and establishes a directory of all commands contained in the file(s). The file name(s) is stored in a system catalog of all UDC users (COMMAND.PUB.SYS). A user must have read and lock access for the UDC file in order to use the :SETCATALOG command and to logon while that UDC file is in effect. Therefore, for account and system level UDCs, the account and group file security in which the file resides must allow read and lock access to all users (R,L:ANY). Note that the default file access for the PUB group gives lock access only to users with AL, GL, or home group (GU) capability.

The system UDC catalog, COMMAND.PUB.SYS, must exist for the :SETCATALOG command to properly execute. If this file does not exist on your system, the system manager or operator must build it. The file should normally be built with a record size of 20; the size of the file will determine how many users will simultaneously have UDCs in effect. A file for the system catalog might be built in the following way:

```
:BUILT COMMAND;REC=20;DISC=2000
```

If you want to have the file named MYUDC1 searched by the Command Interpreter, and have the user-defined commands contained therein entered into a directory enter:

```
:SETCATALOG MYUDC1
```

Note that once a UDC file is entered into the system catalog with the :SETCATALOG command, that file cannot be purged with the :PURGE command nor modified and kept under the same name with the Editor while any user that has that UDC file as part of their directory is logged on. Any attempt to do either of these two operations will result in an error message (Exclusive Violation).

In order to purge or modify such a file, it must be removed from the system catalog by issuing the :SETCATALOG command with a parameter list that excludes that UDC file.

Entering :SETCATALOG with no parameters removes all your UDC files from the system catalog. Note that the files are not purged, but merely deleted from the catalog. For example:

```
:SETCATALOG
```

To reenter UDC files into the catalog, reenter the :SETCATALOG command for these files.

Removing UDCs with the :SETCATALOG command (for example, :SETCATALOG;SYSTEM to remove system-wide UDCs) does not have an immediate effect upon jobs/session that are still logged on. These jobs/sessions may continue to access the UDCs cancelled until that job or session terminates, or that job or session forces the creation of a new UDC directory with a :SETCATALOG command.

If the :SETCATALOG command is part of a user-defined command, the :SETCATALOG command will be the last command executed in the UDC. In addition, if a program command is part of your UDC and you break, do a :SETCATALOG and a :RESUME, that program command is the last command that your UDC will execute.

For example, suppose your UDC calls the Editor subsystem. Once in Editor you break and perform a :SETCATALOG command, followed by a :RESUME command. The :RESUME command will allow you to again enter the Editor subsystem, but the :EDITOR command will be the last command in the UDC to execute.

BUILDING AND MODIFYING A UDC FILE USING THE EDITOR

Once you have defined a command or set of commands which you would like to save, you can enter them into an ASCII disc file using the Editor. This file should have a record length of 80 bytes for numbered files (standard editor file), or 72 bytes for unnumbered files. You may create several such files, each consisting of one or more user-defined commands. If a file contains more than one user-defined command, the commands must be separated from each other by one line containing one or more asterisks, starting in the first column.

The Editor can be used to modify commands stored in a file, if that file is not being accessed. Note, however, that a UDC file once entered into the system catalog with the :SETCATALOG command, cannot be modified and kept under the same name with the Editor while any user that has that UDC file as part of their directory is logged on.

For example, suppose you are the only user with file UDC1 as your UDC file, and you wish to modify that file:

```
:showcatalog

UDC1.SMR.OSE
   S                    USER
:editor
HP32201A.7.08 EDIT/3000  FRI, AUG 29, 1980,  3:44 PM
(C) HEWLETT-PACKARD.CO. 1980
/t udc1
/l all
    1      s
    2      showjob
/a 1.1
    1.1    option nobreak
    1.2    //
...
/l all
    1      s
    1.1    option nobreak
    2      showjob
/k udc1
+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
!  ERROR NUMBER: 90    RESIDUE: 0                 !
!  BLOCK NUMBER: 0             NUMREC: 0          !
+----------------------------------------------+
*41*FAILURE TO OPEN KEEP FILE   (90)
EXCLUSIVE VIOLATION: FILE BEING ACCESSED (FSERR 90)
/ <-------------- Break
:setcatalog <-------------- releases file from system UDC catalog.
:resume
READ PENDING
k udc1
UDC1 ALREADY EXISTS - RESPOND YES TO PURGE OLD AND KEEP NEW
PURGE OLD?y
/e
END OF SUBSYSTEM
:setcatalog udc1
```

The following is an example showing how to create User-Defined Commands with the Editor:

```
:EDITOR
HP32201A.7.0H EDIT/3000  THU, Sep 22, 1977,  3:17 PM
(C) HEWLETT-PACKARD CO. 1976
/ADD
    1      S <---------- Head
    2      SHOWJOB <---- List of Commands
    3      //
...
/KEEP UDC1
/DELETE ALL
/ADD
    1      L
    2      LISTF
    3      **** <---------- Separates UDC's
    4      R
    5      RESUME
    6      //
...
/KEEP UDC2
/ <---------------------- Break
:SETCATALOG UDC1,UDC2
:S

JOBNUM  STATE IPRI JIN  JLIST    INTRODUCED  JOB NAME

*S55    EXEC     20   20       THU  2:49P  FIELD.SUPPORT
*S57    EXEC     52   52       THU  2:50P  BOB.DATAMGT
*S59    EXEC    100  100       THU  2:59P  YU.UTILITY
*S61    EXEC    111  111       THU  3:01P  SMITH.COBOL74
*S62    EXEC    102  102       THU  3:10P  STEVE.PASCAL
*S63    EXEC     25   25       THU  3:12P  ED.MPE

6 JOBS:
    0 INTRO
    0 WAIT; INCL 0 DEFERRED
    6 EXEC; INCL 6 SESSIONS
    0 SUSP
JOBFENCE= 2; JLIMIT= 2; SLIMIT= 16

:L

FILENAME
F53Q        JMAT        K2651517    U53         UDC1        UDC2
:R
READ PENDING

/END

END OF SUBSYSTEM
:
```

USING THE :SHOWCATALOG COMMAND

The :SHOWCATALOG command is used to list the UDCs currently in effect, the level at which they are defined, and the file in which they are contained.

```
:SHOWCATALOG

UDCU.TOM.MPE <----filename
    WELCOME<------UDC        USER <---------- UDC level
    T                        USER
UDCA.TOM.MPE
    LIST                     ACCOUNT
    FCOPY                    ACCOUNT
UDCS.TOM.MPE
    R                        SYSTEM
```

NESTING USER-DEFINED COMMANDS

User-defined commands can be nested; that is, they can be defined so that entering one UDC will cause several UDCs to execute. For example,

```
DOALL
L
S
****
L
LISTF
****
S
SHOWJOB
```

If the above set of UDCs is entered into a UDC directory, then each time DOALL is entered, the MPE commands :LISTF and :SHOWJOB will execute.

Note, however, that a UDC can only call another UDC if the UDC being called is defined after the calling UDC; UDCs can only be referenced forward, not backward. For example, for the command :SETCATALOG CAT1,CAT2 all commands in CAT2 are forward of CAT1.

If, in the above example, DOALL is put in the UDC file after the L and S user-defined commands, the UDC DOALL will not execute and an error message will result:

```
:EDITOR
HP32201A.7.0H EDIT/3000  MON, DEC  5, 1977, 10:39 AM
(C) HEWLETT-PACKARD CO. 1976
/A
       1        L
       2        LISTF
       3        ****
       4        S
       5        SHOWJOB
       6        ****
       7        DOALL
       8        L
       9        S
      10        //
...
/K DOALL
/E

END OF SUBSYSTEM
:SETCATALOG DOALL
:DOALL
L
^
UNKNOWN COMMAND NAME.  (CIERR 975)
:
```

System-level UDCs are forward of account-level UDCs, which in turn are forward of user-level UDCs. This means that an account-level UDC can call a system-level UDC and a user-level UDC can call either account or system-level UDCs.

If a system and user-level UDC exist with the same name, and an account-level UDC calls a UDC by this name, the system-level UDC will be called instead of the user-level UDC because in this case, forward of the user-level UDC.

ERRORS IN USER-DEFINED COMMANDS

A user-defined command contains an error if the execution of an MPE or a user-defined command within the body of the UDC results in an error, or the expansion of the UDC parameters result in an error. If an error or warning occurs as a result of executing a UDC, MPE may:

1. print the line in which the error occurs
2. print a caret (^) pointing to the error, and/or
3. print the appropriate error message

If option NOHELP is specified as the execution option, the caret is not printed. If options NOLIST and NOHELP are specified, then the line containing the error is not printed as well. The error message is always printed.

When a UDC contains an error (rather than a warning) none of the commands in the body of the UDC following the line which causes the error will be executed unless the line that caused the error is preceeded by a :CONTINUE. In a job, a UDC that contains an error when no :CONTINUE is in effect at any of the current nesting levels, will cause the job to terminate.

Consider the following example:

```
EDITOR
HP32201C.00.00 EDIT/3000  FRI, SEP  5, 1980,  8:46 AM
(C) HEWLETT-PACKARD CO. 1980
/A
      1        A
      2        OPTION LIST
      3        COMMENT A STARTS
      4        CONTINUE
      5        B
      6        COMMENT A ENDS
      7        **
      8        B
      9        OPTION LIST
     10        COMMENT B STARTS
     11        C
     12        COMMENT B ENDS
     13        **
     14        C
     15        OPTION LIST
     16        COMMENT C STARTS
     17        FILE A;CODE=XYZZY
     18        COMMENT C ENDS
     19        **
     20        //
...
/K TUDC
/E

END OF SUBSYSTEM
:SETCATALOG TUDC
:A <-------------------- begin execution of UDC A
COMMENT A STARTS
CONTINUE <-------------- CONTINUE in effect for A's call of B
B <--------------------- A calls B
COMMENT B STARTS
C <--------------------- B calls C
COMMENT C STARTS
FILE A;CODE=XYZZY <----- execution at :FILE results in an error

UNKNOWN FILE CODE TYPE. (CIERR 253)
COMMENT A ENDS <-------- The remainder of UDCs B and C are not executed.
```

UDC C contains an error because the :FILE command contains an invalid file code. Therefore, the :COMMENT following the :FILE command is not executed. Because the execution of UDC C results in an error, the last :COMMENT in UDC B is not executed. The last :COMMENT in UDC A is executed, however, because at that nesting level there is a :CONTINUE in effect for the execution of UDC B.

When a :CONTINUE is placed before the :FILE command in UDC C, all three UDCs execute completely.

```
:SETCATALOG
:EDITOR
HP32201C.00.00 EDIT/3000  FRI, SEP  5, 1980,  8:48AM
(C) HEWLETT-PACKARD CO. 1980
/T TUDC
/A 16.5
   16.5     CONTINUE
   16.6     //
...
/K;E
TUDC
TUDC ALREADY EXISTS - RESPOND YES TO PURGE OLD AND KEEP NEW
PURGE OLD?Y

END OF SUBSYSTEM
:SETCATALOG TUDC
:A
COMMENT A STARTS
CONTINUE
B
COMMENT B STARTS
C
COMMENT C STARTS
CONTINUE <-------------- CONTINUE in effect for :FILE command.
FILE A;CODE=XYZZY
                ^

UNKNOWN FILE CODE TYPE. (CIERR 253)
COMMENT C ENDS  }
COMMENT B ENDS  } <-------- The remainder of B and C are executed.
COMMENT A ENDS  }
```

TERMINALS SUPPORTED BY MPE

APPENDIX A

TERMINAL TYPE	DESCRIPTION
0	HP 2749B (ASR-33 EIA-compatible) Terminal (10 cps).
1	ASR-37 Teleprinter Terminal with Paper Tape Reader/Punch (10 cps).
2	ASR-35 EIA-compatible Terminal (10 cps).
3	Execuport 300 Data Communications Transceiver Terminal (10/15/30 cps).
4*	HP 2600A or Datapoint 3300 Keyboard Display Terminal (10/15/30/60/120/240 cps).
5	Memorex 1240 Communication Terminal (10/15/30/60 cps). NOTE: This terminal must be equipped with even parity-checking option.
6*	HP 2762A/B (General Electric Terminet 300 or 1200), or Data Communications Terminal, Model B (10/15/30/120 cps) with Paper Tape Reader/Punch, Option 2. NOTE: This terminal must be equipped for ECHO PLEX.
9*	HP 2165A Terminal (Beehive MiniBee) (10/15/30/60/120/240 cps).
10*	HP 2640A/B, HP 2641A, HP 2644A, or HP 2645A Character Mode or full program control of block mode transmission (10-240 cps).
11*	HP 2640A/B, HP 2641A, HP 2644A or HP 2645A. Allows user to use block mode without program control of block mode transmission. Requires user to position cursor before pressing ENTER. Recommended for speeds exceeding 30 cps when you expect to switch between character mode and block/line mode. May not be used in block/page mode.
12*	HP 2645K Katakana/Roman Data Terminal.
13*	Message switching network or other computer.
14	Multipoint Terminal.
15*	HP 2635A Printing Terminal. 8-bit protocol (for second character set).
16*	HP 2635A Printing Terminal. 7-bit protocol (standard character set).
19	2631B Remote Spooled printer.
* = Terminals that are supported on Series 30/33.	

For further information see the Communications Handbook, Section B (30000-90105).

SUBSYSTEM FORMAL FILE DESIGNATORS

APPENDIX B

Command	Parameters	Formal File Designator
:BASIC	*commandfile* *inputfile* *listfile*	BASCOM BASIN BASLIST
:BASICOMP	*textfile* *uslfile* *listfile*	BSCTEXT BSCUSL BSCLIST
:FORTRAN	*textfile* *uslfile* *listfile* *masterfile* *newfile*	FTNTEXT FTNUSL FTNLIST FTNMAST FTNNEW
:SPL	*textfile* *uslfile* *listfile* *masterfile* *newfile*	SPLTEXT SPLUSL SPLLIST SPLMAST SPLNEW
:COBOL	*textfile* *uslfile* *listfile* *masterfile* *newfile*	COBTEXT COBUSL COBLIST COBMAST COBNEW
:RPG	*textfile* *uslfile* *listfile* *masterfile* *newfile*	RPGTEXT RPGUSL RPGLIST RPGMAST RPGNEW
:BASICPREP	*textfile* *progfile* *listfile*	BSCTEXT BSCPROG BSCLIST
:FORTPREP	*textfile* *progfile* *listfile* *masterfile* *newfile*	FTNTEXT FTNPROG FTNLIST FTNMAST FTNNEW

Command	Parameters	Formal File Designator
:SPLPREP	*textfile* *progfile* *listfile* *masterfile* *newfile*	SPLTEXT SPLPROG SPLLIST SPLMAST SPLNEW
:COBOLPREP	*textfile* *progfile* *listfile* *masterfile* *newfile*	COBTEXT COBPROG COBLIST COBMAST COBNEW
:RPGPREP	*textfile* *progfile* *listfile* *masterfile* *newfile*	RPGTEXT RPGPROG RPGLIST RPGMAST RPGNEW
:BASICGO	*textfile* *listfile*	BSCTEXT BSCLIST
:FORTGO	*textfile* *listfile* *masterfile* *newfile*	FTNTEXT FTNLIST FTNMAST FTNNEW
:SPLGO	*textfile* *listfile* *masterfile* *newfile*	SPLTEXT SPLLIST SPLMAST SPLNEW
:COBOLGO	*textfile* *listfile* *masterfile* *newfile*	COBTEXT COBLIST COBMAST COBNEW
:RPGGO	*textfile* *listfile* *masterfile* *newfile*	RPGTEXT RPGLIST RPGMAST RPGNEW
:EDITOR	*listfile*	EDTLIST
:SEGMENTER	*listfile*	SEGLIST
:RJE	*commandfile* *inputfile* *listfile* *punchfile*	RJECOM RJEIN RJELIST RJEPUNCH

Command	Parameters	Formal File Designator
:PREP :PREPRUN	PMAP	SEGLIST
:PREPRUN :RUN	LMAP	LOADLIST
:RESTORE :STORE	SHOW	SYSLIST

DETAILS OF PROGRAM EXECUTION

APPENDIX C

In MPE, programs are run on the basis of *processes* created and handled by the system. A process is not a program itself, but the unique execution of a program by a particular user at a particular time. Therefore, if the same program is run by several users, or more than once by the same user, it is executed by several distinct processes.

The process is the basic executable entity in MPE. A process consists of a process control block (PCB) that defines and monitors the state of the process, a dynamically-changing set of code segments, and a data area (stack) upon which these segments operate. Thus, while a *program* consists of instructions (not yet executable) and data in a file, a *process* is an executing program with a data stack assigned. The code segments used by a process can be shared with other processes, but its data stack is private (Figure C-1). For example, each user working on-line through the BASIC language is running his program under a separate process; all use the same code (the only copy of the BASIC interpreter in the system), but each has his own stack.

Processes, and the elements that comprise them, are invisible to the programmer accessing MPE through standard capabilities; in other words, this programmer has no control over processes or their structure. For this user, MPE automatically creates, handles, and deletes all processes. The user with certain optional capabilities, however, can create and manipulate processes directly. See *MPE Intrinsics Reference Manual* for further details.

OVERVIEW OF ALLOCATION/EXECUTION

When a program is run, it is allocated and executed. Before allocation/execution takes place, MPE checks to verify the following points:

- *For program files that are permanent files,* the capabilities assigned to the program must not exceed those assigned to the group to which the program file belongs; otherwise, an error occurs.

- *For program files that are temporary files in the session/job temporary file domain,* the capabilities assigned to the program must not exceed those of the user running the program; otherwise, an error occurs.

- *For privileged segments, when the NOPRIV parameter is omitted from the :RUN (program execution) command,* the capabilities assigned to the program must include the privileged mode capability for the segment to be loaded in privileged mode; otherwise, an error occurs.

During *allocation*, the MPE loader binds the segments from the program file to referenced external segments from a segmented library (SL). *Execution* now begins. Then, the first code segment to be executed and the stack are moved into main memory. As it progresses, many processes may be created, run, and deleted. For each process in execution, one or more code segments and one stack, operating under control of a process control block (PCB), typically exist in main memory. Not all code segments belonging to a process in execution need exist in main memory simultaneously. Typically, a process operates on a set of segments that are dynamically swapped between memory and disc. MPE main-

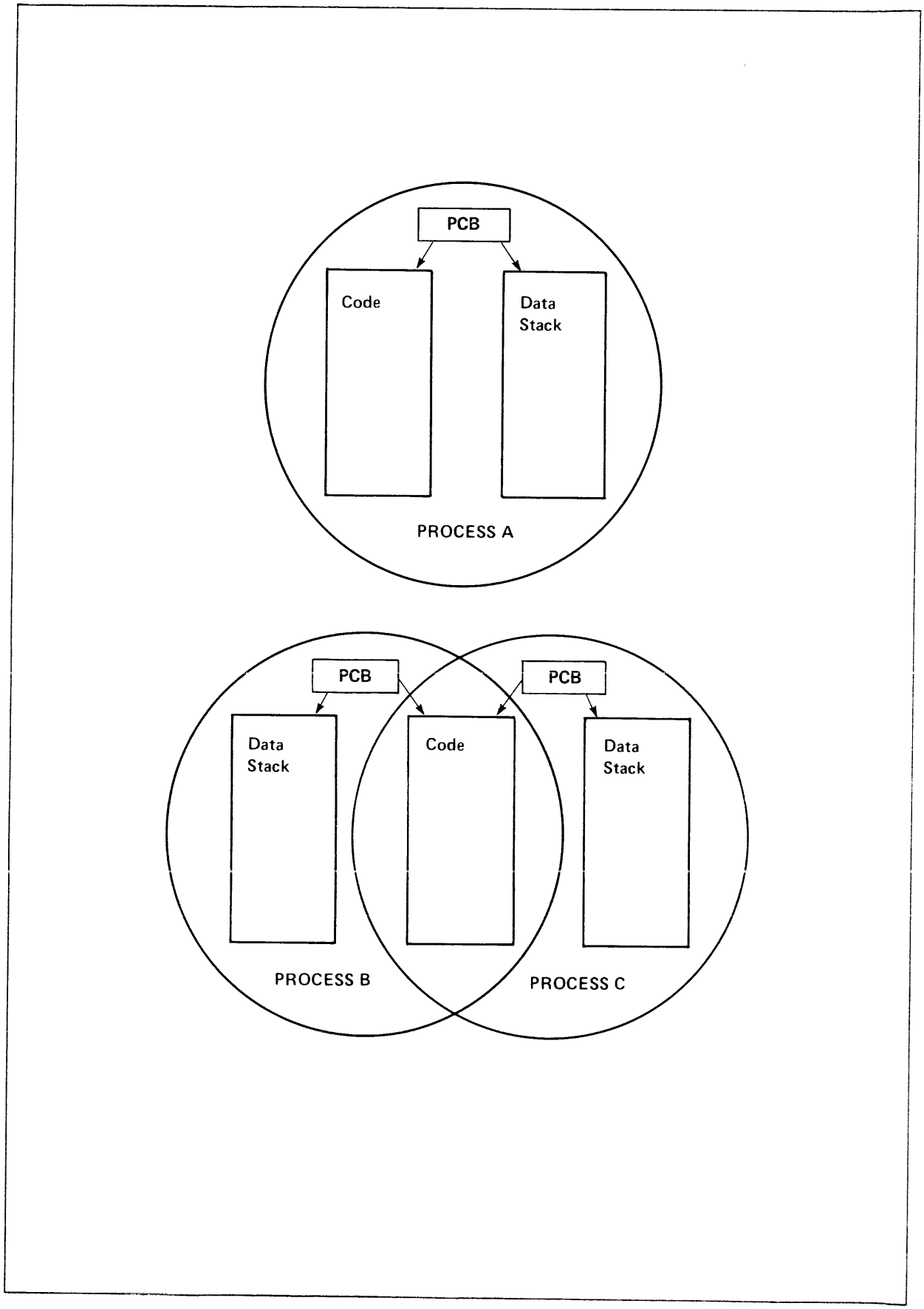


Figure C-1. Code-Sharing and Data Privacy

tains records of the frequency each segment is used, so that those used least frequently in main memory become the most eligible for swapping out. You never need to determine whether a segment is in main memory or on disc at any given time — this is always done for you automatically by MPE.

During allocation/execution, MPE keeps track of each code segment by maintaining information about its nature and current location in the code segment table (CST). Similarly, information about the stack is dynamically recorded in the data segment table (DST).

A particular program can be run by many user processes simultaneously, with all processes accessing the same copy of sharable code. But unlike the program code segments, the data segment containing the stack is private to each user's process and cannot be shared with others. As execution progresses, data enters and leaves the stack dynamically. Within the stack, data is arranged as a linear group of items accessed from one end (called the top-of-stack). When the last instruction in a process is executed, MPE releases those segments associated only with that process, including the stack segment, and deletes their related entries in the CST and DST. (However, shared code segments are not released until the last process using them is deleted.) All files opened by this process are closed.

PCB/CODE SEGMENT/STACK INTERACTION

Each process control block (PCB) contains information needed to control a process. This information includes the priority of the process and pointers to ancestor and descendent processes. (See *MPE Intrinsics Reference Manual* for a discussion of inter-process relationships.)

Each code segment executed by a process can contain one or more program units that include calls to procedures elsewhere in this segment or in another segment. When a code segment is executing in main memory, it is defined by pointers in three hardware registers: the Program Base (PB), Program Counter (P), and Program Limit (PL) registers (Figure C-2). There is only one set of these registers; at any particular instant, their contents refer to the code segment currently in execution. The PB register contains the absolute address of the starting location of the segment in main memory. The P register holds the absolute address of the instruction currently being executed. The PL register indicates the absolute address of the last location of the code segment. Execution can only be transferred from this segment by an interrupt or by a call or exit instruction referencing a procedure in another segment, in which case the PB, P, and PL registers are reset to reflect the characteristics of the new segment. Whenever an instruction is executed, the PL and PB registers are checked to ensure that referenced addresses fall within the proper segment boundaries. This *bounds check* guarantees that other programs and the system itself are protected against improper access. Since all addresses within a code segment are relative to the contents of the three program registers, a segment can be relocated anywhere in main memory and only the register contents and CST need be changed to reflect this transfer.

As with code segments, there are normally several stacks present in memory. (One stack exists for each process.) But since the execution of any process is interleaved with that of others, only one stack is active at any particular instant. Most dynamic computational operations take place on the stack. The last element added to the stack is placed in the word at the *top-of-the-stack*. In this position, it is the first element removed when the process associated with the stack requests data from the stack. (In other words, the last element in is the first one out.) Each time data is added to the stack, the previous top element becomes the second element from the top; each time the topmost element is removed, the second element returns to the top of the stack.

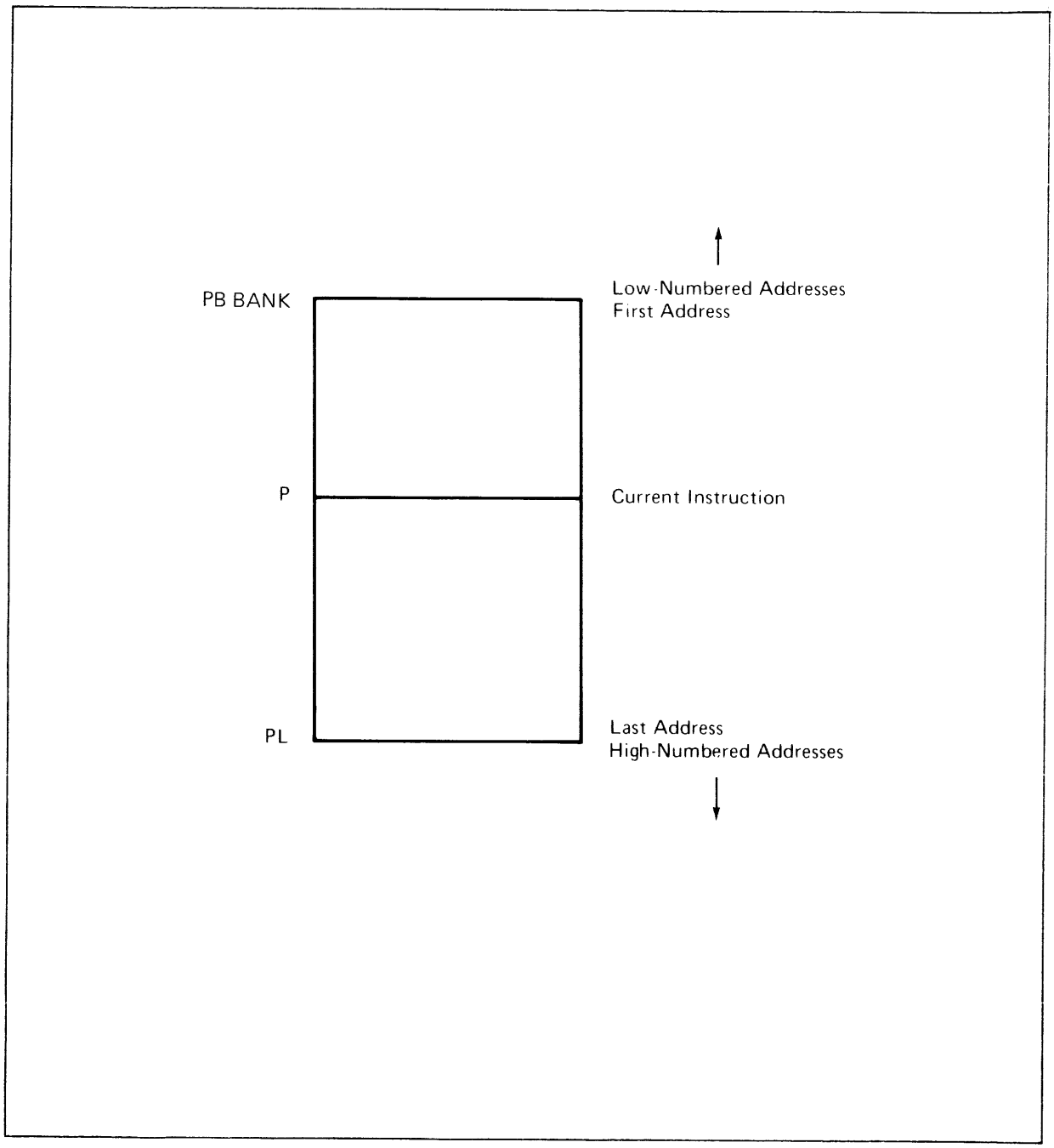
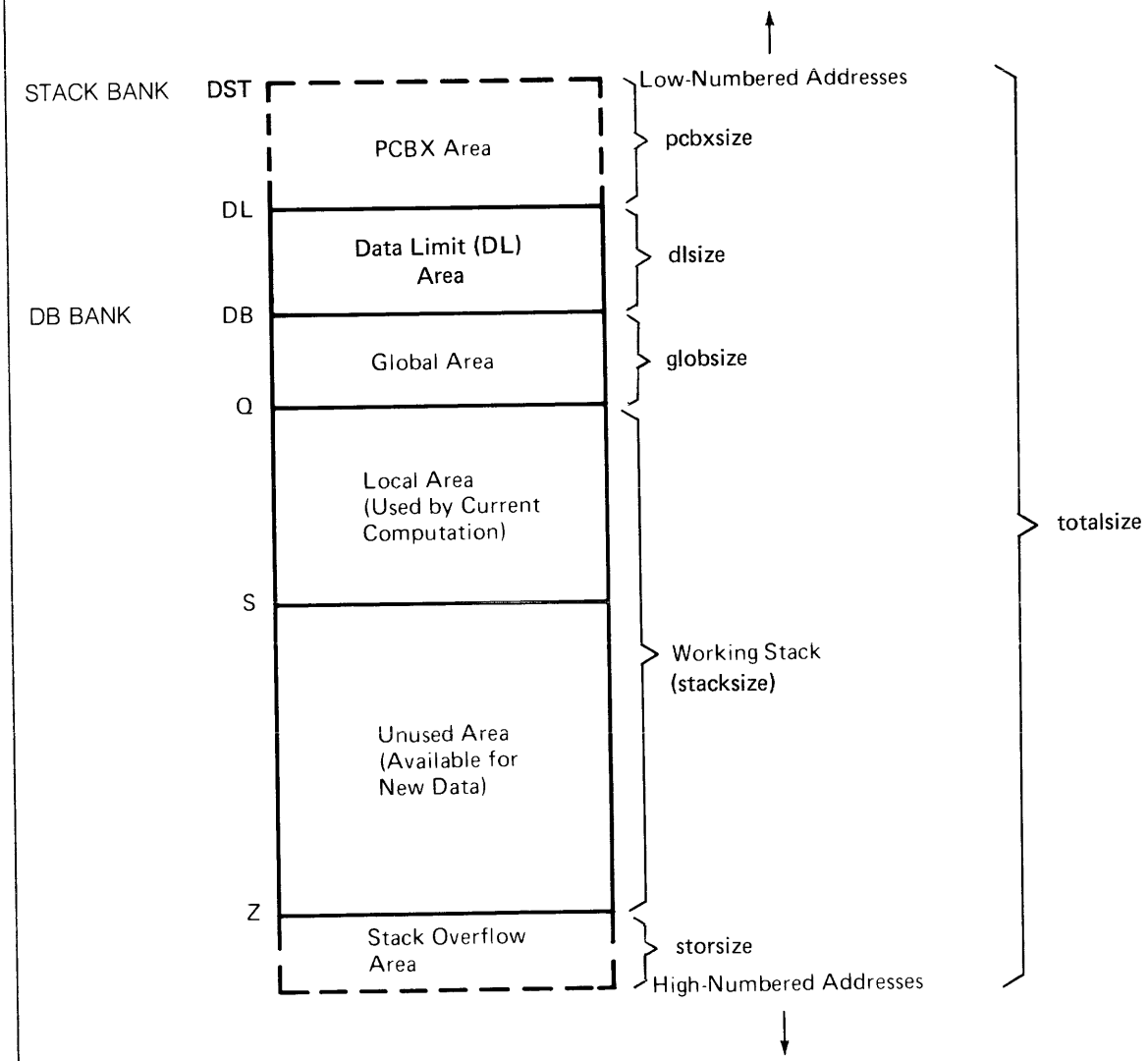


Figure C-2. Code Segment and Associated Registers

Programmers operate *directly* on elements in the stack only when using the SPL language. However, any program in any other language references the stack implicitly when it manipulates data, although the user need not be aware of this. The principal data areas within the stack segment and their purposes are summarized in table C-1. (The values of specific constants appearing in table C-1 are currently accurate but may be changed in future releases of the operating system.) The boundaries of these areas are delimited by the addresses stored in the registers indicated along the left side of the stack diagram appearing in figure C-3 (DST, DL, . . . Z registers). The PCBX, DL, and working stack areas are all subject to dynamic expansion or contraction as the process runs.



NOTE: DL, Q, S and Z are aligned with the Stack Bank register, and DB is aligned with the DB Bank register to allow split stack operations.

Figure C-3. Data (Stack) Segment and Associated Registers

Table C-1. Data Areas in Stack Segment

AREA	FUNCTION
PCBX	The Process Control Block Extension (PCBX) area contains information necessary for the system to control process activity efficiently, such as register settings and file pointers. In general, the PCBX is used when the process is running in main memory; the PCB is used when it is not. Initially, this area is %402 words long.
DL	This area contains user-managed information accessed and used only through SPL programs such as compilers. These programs sometimes require this space for buffers. This area is bounded by the addresses stored in the data limit (DL) and data base (DB) registers, and is accessed via DB-negative addressing. (A portion of the DL area, that between the addresses DB-10 and DB-1, is reserved for data used by subsystems during their operation.) The initial size of the DL area (*dlsize*) may be specified in the :RUN command or CREATE intrinsic. If omitted from the :RUN command, *dlsize* is taken from the specifications supplied with the :PREP command. If not supplied anywhere by user, *dlsize* is assigned a value of zero. The *dlsize* actually granted is calculated as follows: *dlsize* ← { [(specified *dlsize*) + %613] land - 128 } -%402 Note: In this calculation, *land* indicates a *logical-and* operation. This calculation assures that the area from the beginning of the segment to the DB address is an integral number of disc sectors. The resulting *dlsize* is checked to ensure that it is less than 32768 words.
Global	This area is used for global variables; these are variables declared within the data group of a main program and usable by any procedure within that program. It also contains global arrays and pointers to those arrays. The size of this area (*globsize*) is set during program preparation and entered in the program file. This size, and the content of the global area, are determined by the number of global variables and their initialization as specified by the programmer. The area is delimited by the contents of the DB register and the stack marker (Q) register.
Working Stack	From the standpoint of user programs, this is the most active area of the stack—it is the area where the user's temporary data is stored and manipulated. It is bounded by the addresses indicated by the initial contents of the Q register (Q-initial) and the Z-register. Its size (*stacksize*) is specified in the :RUN command or CREATE intrinsic, or else is taken from the :PREP command specifications. By default, in the absence of such information, *stacksize* is supplied by MPE. (This default *stacksize* is specified by the System Supervisor when he configures the system.) The *stacksize* must be greater than 512 words but less than 32768 words.
Local	Within the working stack area, the local area contains data relating only to the procedure currently in execution. The current contents of the Q-register denotes the beginning of this area and the contents of the top-of-stack (S) register indicates the end, pointing to the last item of data in the stack.
Unused	Also within the working stack area, the unused area is the space that remains available for new data. This area is bounded by the addresses in the S register and stack limit (Z) register. The Z register indicates the last main-memory location that can be used by user data in the stack area.

Table C-1. Data Areas in Stack Segment (Continued)

AREA	FUNCTION
Stack Overflow	This area is available for stack-overflow, a condition that occurs when the S-register pointer must be moved beyond the Z-register pointer and space must be provided for certain end-of-stack information. This buffer area lies between the Z-register contents and the end of the data segment. This area is currently %170 (120 decimal) words long.

When the stack is allocated for a process, its total size (*totalsize*) is calculated as follows:

totalsize ← *pcbxsize* + *dlsize* + *globsize* + *stacksize* + *storsize*

MPE ensures that *totalsize* is less than 32768 words and also less than or equal to *maxstack* (the maximum allowed stack size specified during system configuration). The entire size of the data segment allocated for the stack (*dssize*) is calculated as follows:

dssize ← (totalsize + *7) land − 4*

The dssize is effectively rounded upward to an integral multiple of four, and increased to include the four-word trailing main-memory link of the segment when present in memory. The segmenter descriptor (*dstsize*) in the Data Segment Table (DST) is:

dstsize ← *dssize/4*

Another stack-related value, *maxdata,* is maintained by MPE in the PCBX and is used to limit automatic stack expansion by fixing the maximum stack size. It may be specified in the :PREP command and recorded in the program file. It may also be specified in the :RUN command or CREATE intrinsic, in which case it overrides the :PREP command specification (if any). If *maxdata* is omitted from both :PREP and :RUN (or CREATE), MPE equates *maxdata* to *totalsize*. When specified, *maxdata* must be a positive number or zero. If the specified value exceeds the *maxstack* value defined during configuration, *maxdata* is equated to *maxstack*. During process execution, if the distance from the start of the data segment to the Z-address exceeds *maxdata* because of stack overflow, the process is aborted.

The size of the data segment as it exists in virtual memory on disc (*vsize*) is calculated as follows:

vsize ← *dssize* + 2058 words

The maximum virtual-memory space allowed (*maxsize*) is:

maxsize ← max(*dssize,maxdata*) + 2058

The above calculation includes three disc sectors for possible expansion within the PCBX area, and nine sectors for space used during possible stack-overflow abort. These extra twelve sectors, totaling 2058 words, imply a limit on *maxdata* of 30702 words.

Through bounds checks, MPE and certain hardware provisions ensure that data for the process remains within the limits defined by the contents of the DL and Z registers, and that no other user accesses this area. All locations are addressed relative to the DB, Q, and S registers.

Although the top-of-stack is logically indicated by the S register, up to four of the topmost elements of the stack can actually exist in central processor registers (the TR registers) rather than in main memory — in effect, the stack "spills over" from memory into these registers. This function is managed by the hardware and greatly enhances processing speed.

Before a process begins execution, stack space is first reserved for global data, beginning at the DB-address and terminating at the Q-address, which denotes the beginning of the dynamic *working stack* (figure C-4A). At this time, no data is stored beyond the Q-address, and so the S-register also points to the same address as the Q register — that is, the bottom and top of the working stack coincide. But, as the process begins execution, data is added to the stack, and the S-pointer (top-of-stack) moves away from the Q-pointer (figure C-4B). If, at some point, the process encounters a procedure call, a new area for data local to that procedure must be defined. To do this, the system hardware places a group of four words called the *stack marker* on top of the stack to save information necessary to re-create the currently-defined local area later. The Q and S registers are then pointed to the top word of the stack marker, which also delimits the beginning of a new, fresh and unique local area for the procedure just called (figure C-4C).

The words in the stack marker preserve the state of the machine at the time of the procedure call. These words contain the following information, (shown in order ascending toward the Q-address):

Word	Contents
Q-3	Current contents of the Index (X) register.
Q-2	The return address for the code segment, denoted by P+ 1 (relative to the PB register).
Q-1	The current contents of the Status register (which includes the number of the code segment containing the calling procedure).
Q-0	The delta-Q value, which is the number of words between the new and previous Q-locations, enabling the later re-setting of Q to its old value.

As data is added to the stack during execution of the new procedure, the S-pointer moves away from the new Q-pointer, reflecting the latest data added (figure C-4D). When the procedure exits back to the main program, the new local data area is deleted from the stack, the stack marker is used to restore the Q-pointer to its previous setting, any value returned by the procedure is left at the new top-of-stack, and the S-pointer is set to indicate the new top-of-stack (figure C-4E). This results in a "clean" stack from which temporary data local to the called procedure is eliminated because it is no longer needed.

Whenever a procedure is called, the Q and S registers are manipulated in this manner. The Q register changes with each procedure call and exit; the S register may change when an instruction references data. Thus, when a process executes a main program (outer block) that calls three procedures, there will be a maximum of four local areas (one for the main program and three for the procedures called by it) on the stack. Each procedure's local area is delimited at its base by its own stack marker. Within these stack markers, the delta-Q words form a logical chain that links the present Q register setting back to its initial value.

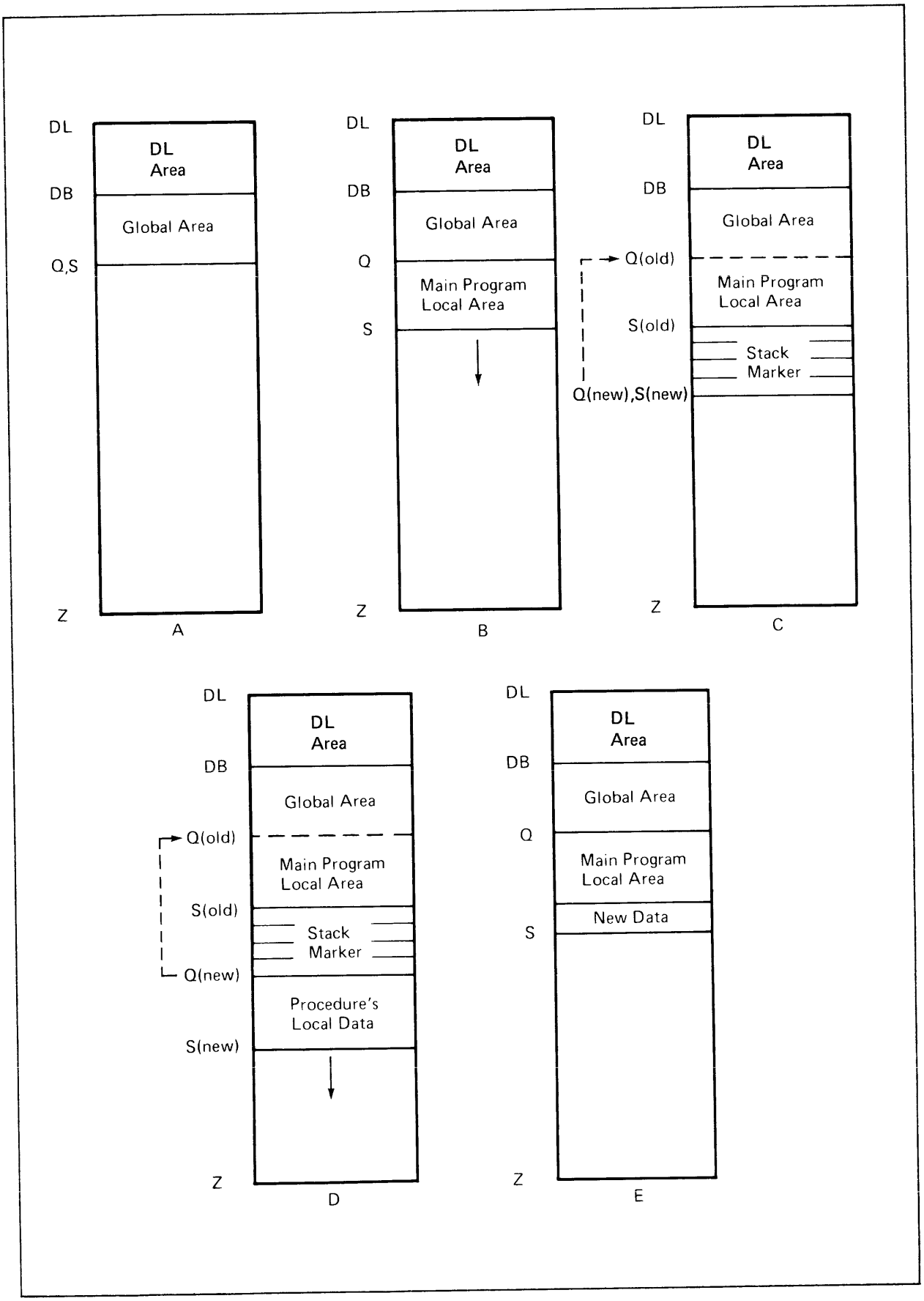


Figure C-4. Stack Operation

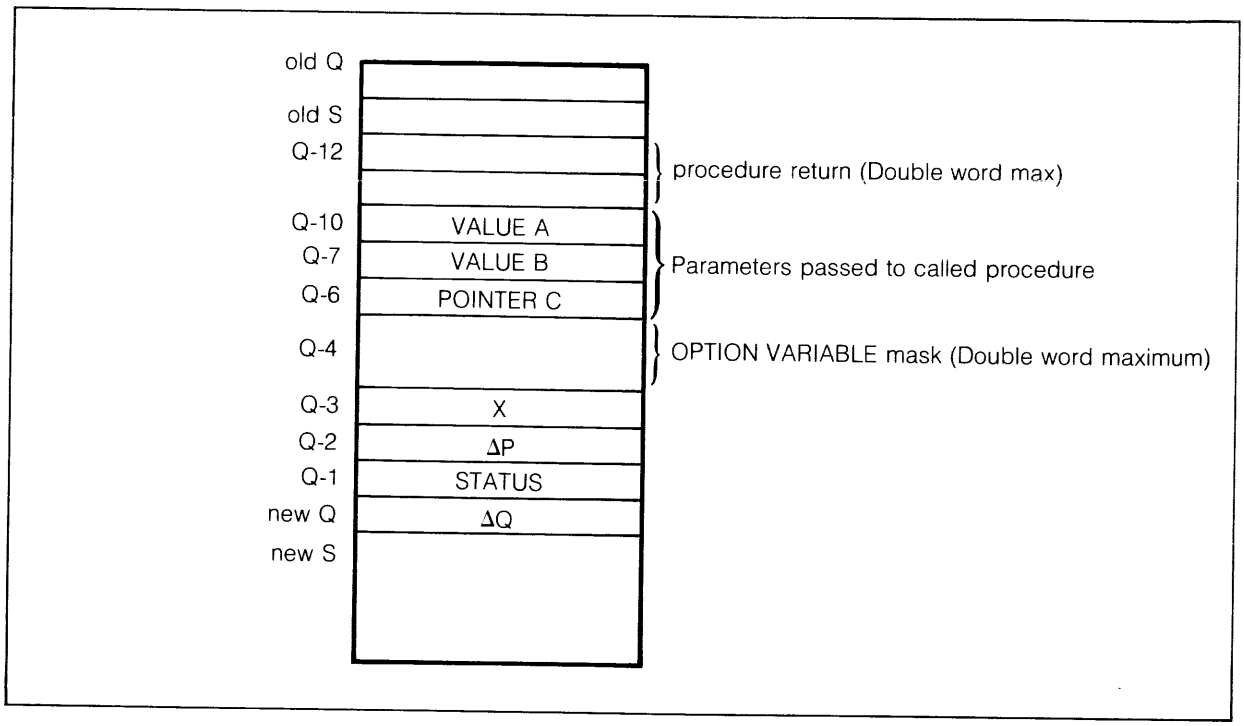


Figure C-5. Stack Operation Example

Figure C-5 shows what the top of stack area would look like if a procedure (or intrinsic) is called that has the following type of declaration:

```
DOUBLE PROCEDURE X (A,B,C);
VALUE A,B; INTEGER A,B;
LOGICAL C;
OPTION VARIABLE;
```

In this example the code increments the S-register by 2 to save a double word area for the procedure double value return. Next, since parameters A and B are specified by value, the actual value of the parameters are pushed on TOS respectively. Parameter C is specified to be the address of a logical location so the DB relative address is placed on TOS. Since the procedure is OPTION VARIABLE, not all of the parameters must be passed. As a result, a method is provided for the procedure to find out which parameters are valid. This method utilizes a mask value which is the next word pushed on TOS. The mask value is a one bit if the parameter was passed and a zero bit if it was not passed. The parameter area exists whether or not the parameter is passed or not. Bit 15 of the mask word is a one if parameter C was passed, Bit 14 is a one if parameter B was passed and Bit 13 is a one if parameter A was passed. The mask value can be up to 31 bits and therefore a maximum of 31 parameters can be to any one procedure.

Next a PCAL to the procedure pushes a standard 4 word stack marker and moves the Q register down to prevent accidental destruction of the stack marker and parameters passed to the procedure.

The last instruction of the procedure, in this example, is an EXIT 4 instruction. During execution of this instruction the Q register is moved back to the "old Q" location and all values except the double returned word are taken (popped) off the stack. The next instruction in the user's program stores the double return value in a DB relative double word variable location. The Q negative relative locations shown in the figure are relative to the Q register while the procedure is executing.

PROGRAM LISTINGS

Listing Of Prepared Program

```
PROGRAM FILE SCR4.MPE.SYS
SEG40                (1) 0
   NAME              SIT  CODE ENTRY SEG
   LISTRL'           1    0    32
   MAKEROOMINDL      30              10
   FGETINFO          31              ?
   FREADMR''         32              10
   NTOA              33              10
   BLANKLINE         34              10
   TESTBIT           35              10
   DNTOA             36              10

   OPENRL            27   2523  2523
   FOPEN             70              ?
   FLOCK             71              ?
   FREADMR'          72              10
   SEGMENT LENGTH         3130
SEG30                1
   NAME              SET  CODE ENTRY SEG
   LISTSL'           1      0   106
   FGETINFO          33              ?
   NTOA              34              10
   BLANKLINE         35              10
   TESTBIT           36              10
   DNTOA             37              10
   PRINTLINE         40              10
   EJECTPAGE         41              10
   CLEANUPRTBUF      2    545   545
   FWRITEDIR'        42              10
   GETREFTABENTRY    3    556   556
   FEOF              43              10

   FPOINT            46              ?
   FCLOSE            47              ?
   NTOA              50              10
   EJECTPAGE         51              10
   SEGMENT LENGTH         2030

PRIMARY DB        350  INITIAL STACK   1440  CAPABILITY      101
SECONDARY DB      1216 INITIAL DL         0  TOTAL CODE      27320
TOTAL DB          1566 MAXIMUM DATA   40000  TOTAL RECORDS   155
ELAPSED TIME      00:14:29.887               PROCESSOR TIME  00:24.156
:
```

Prepared Program Listing Key

Item No.	Meaning
1	The name of the program file (filename.groupname.accountname).
2	The segment name.
3	The (logical) segment number.
4	The program unit entry-point name or external procedure name.
5	The assigned entry number in the Segment Transfer Table (STT).
6	The beginning location of the procedure code in the segment.
7	The location of the entry point in this segment.
8	The (logical) segment number of the segment containing this *external* procedure. If this entry is a number, then the procedure is external to the segment but internal to the program file; if it contains a question mark (?), then the procedure is external to the segment *and* external to the program file.
9	The segment length (in words).
10	The primary DB area size.
11	The secondary DB area size.
12	The total DB area size.
13	The time elapsed during preparation process.
14	The initial stack size.
15	The initial DL size.
16	The maximum area available for data (maximum Z-DL size).
17	Capability of program file.
18	Total code in file.
19	Total records in file.
20	Total central processor time used during preparation process.

Listing Of Loaded Program

```
PROGRAM FILE SCR4.MPE.SYS
                          (1)     (4)  (5)  (6)  (7)  (8)  (9)  (10)
TERMINATE (2)  (3)  PROG  0   50   11   SSL  0    2   37
SENDMAIL            PROG  0   46   11   SSL  0    2   40
DEBUG               PROG  0   44   11   SSL  0    1   52
RECEIVEMAIL         PROG  0    6   11   SSL  0    1   40
AWAKE               PROG  0    5   11   SSL  0    4   33
WHO                 PROG  0    4   11   SSL  0    4   45
FREADDIR            PROG  0   45   10   SSL  0    1    0
FWRITEDIR           PROG  0   44   10   SSL  0    2    0
FWRITE              PROG  0   43   10   SSL  0    3    0
DLSIZE              PROG  0   41   10   SSL  0   15   42
PRINT               PROG  0   15    7   SSL  0   11   45
SETJCW              PROG  0   13    7   SSL  0    1   45
GETJCW              PROG  0   12    7   SSL  0    2   45
ADJUSTUSLF          PROG  0   42    6   SSL  0   11   46
FPOINT              PROG  0   46    2   SSL  0    5    1
RESETDB             PROG  0   34    4   SSL  0    7   33
                              26    2
SETSYSDB            PROG  0   33    4   SSL  0   11   33
                              25    2
FCONTROL            PROG  0   14    2   SSL  0    3    1
PROCTIME            PROG  0    4    2   SSL  0   15   44
TIMER               PROG  0    3    2   SSL  0   23   33
GETUSERMODE         PROG  0   10   11   SSL  0    4   44
                              35    4
                              27    2
                              55    1
LOADEDSLSEG         PROG  0   53    1   SSL  0    2   47
GETPRIVMODE         PROG  0   45   11   SSL  0    5   44
                              32    4
                              24    2
                              52    1
QUIT                PROG  0    7   11   SSL  0   11   31
                              52   10
                              25    5
                              50    1
```

```
FGETINFO            PROG  0   14   11   SSL  0    4    2
                              46   10
                              32    6
                              56    3
                              15    2
                              33    1
                              31    0

301 302 303 304 305 306 307 310 311 312 (11)
```

Loaded Program Listing Key

Item No.	Meaning
1	The name of the program file (filename.groupname.accountname).
2	The name of the external procedure.
3	The type of the segment referencing the external procedure, where: PROG = program segment. GSL = group segmented library segment. PSL = public segmented library segment.
4	External parameter checking level.
5	External segment transfer table (STT) number.
6	External (logical) segment number.
7	Entry point segment type, where: GSL = group segmented library segment. PSL = public segmented library segment. SSL = system segmented library segment.
8	Entry point parameter checking level.
9	Entry point segment transfer table (STT) number.
10	Entry point (logical) segment number.
11	A list of the code segment table (CST) numbers to which the program file segments were assigned. The list is ordered by logical segment number.

INDEX

READER COMMENT SHEET

HP 3000 Computer Systems
MPE Commands
Reference Manual

30000-90009 **January 1981**

We welcome your evaluation of this manual. Your comments and suggestions help us improve our publications. Please use additional pages if necessary.

Is this manual technically accurate?	**Yes** ☐ **No** ☐	(If no, explain under Comments, below.)
Are the concepts and wording easy to understand?	**Yes** ☐ **No** ☐	(If no, explain under Comments, below.)
Is the format of this manual convenient in size, arrangement, and readability?	**Yes** ☐ **No** ☐	(If no, explain or suggest improvements under Comments, below.)

Comments:

FROM:

Name _____

Company _____

Address _____

_____

_____

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1070 CUPERTINO, CALIFORNIA

POSTAGE WILL BE PAID BY ADDRESSEE

Publications Manager
Hewlett-Packard Company
Computer Systems Division
19447 Pruneridge Avenue
Cupertino, California 95014

FOLD

FOLD

Part No. 30000-90009
Printed in U.S.A. 1/81
3MPE.320.30000-90009