HP 3000 Computer Systems

EDIT/3000

Reference Manual

HEWLETT
PACKARD

19447 PRUNERIDGE AVE., CUPERTINO, CALIFORNIA 95014

Part No. 03000-90012

Printed in U.S.A. 8/80

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.

LIST OF EFFECTIVE PAGES

The List of Effective Pages gives the date of the curent edition and of any pages changed in updates to that edition. Within the manual, any page changed since the last edition is indicated by printing the date the changes were made on the bottom of the page. Changes are marked with a vertical bar in the margin. If an update is incorporated when an edition is reprinted, these bars are removed but the dates remain. No information is incorporated into a reprinting unless it appears as a prior update.

Third Edition . . . . . Aug 1980

PRINTING HISTORY

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The date on the title page and back cover of the manual changes only when a new edition is published. When an edition is reprinted, all the prior updates to the edition are incorporated. No information is incorporated into a reprinting unless it appears as a prior update. The edition does not change.

The software product part number printed alongside the date indicates the version and update level of the software product at the time the manual edition or update was issued. Many product updates and fixes do not require manual changes, and conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one to one correspondence between product updates and manual updates.

First Edition	Nov 1972	32201A.04
Second Edition	Aug 1975	32201A.04
Update Package #1	Jun 1976	32201A.04
Update #1 Incorporated	Dec 1976	32201A.05
Update Package #2	May 1977	32201A.06
Update #2 Incorporated	Oct 1977	32201A.06
Update Package #3	Apr 1978	32201A.07
Update Package #4	Aug 1978	32201A.07
Update #4 Incorporated	Jan 1979	32201A.07
Update Package #5	Feb 1979	32201A.07
Update #5 Incorporated	Apr 1979	32201A.07
Third Edition	Aug 1980	32201A.07

PREFACE

This publication is the reference manual for EDIT/3000, a subsystem of the Multiprogramming Executive III operating system (MPE III) used to create and manipulate ASCII files.

The content of this publication is:

Section I - introduces EDIT/3000. Discusses the concepts of editing files and explains the basic structure of EDIT/3000. A summary of all EDIT/3000 commands is included.

Section II - explains how to operate EDIT/3000. Included are instructions explaining how to initiate interactive sessions and batch jobs, and discussions of MPE and EDIT/3000 special controls.

Section III - explains EDIT/3000 basic commands. Examples are provided which demonstrate the use of each command.

Section IV - explains the use of EDIT/3000 advanced commands. Examples are provided for all advanced commands, and include instructions on creating library procedures which can be called by EDIT/3000 to perform complex and reiterative editing functions.

Section V - contains error messages and recovery procedures.

Section VI - discusses customizing EDIT/3000 through the use of three interface procedures.

Appendix A - contains a table of ASCII characters.

Appendix B - contains a summary of EDIT/3000 commands.

Appendix C - contains a WORK file recovery technique.

Other publications which should be available for reference when using this manual are:

MPE Intrinsics Reference Manual (30000-90010)
MPE Commands Reference Manual (30000-90009)
FORTRAN Reference Manual (30000-90040)
COBOL/3000 Reference Manual (32213-90001)

For 3000 systems which do not use MPE III, refer to the following manuals:

- *MPE Intrinsics Reference Manual* (30000-90087)
- *MPE Commands Reference Manual* (30000-90088)
- *FORTRAN/3000 Reference Manual* (32102-90001)

CONTENTS

Section I — Page

INTRODUCING EDIT/3000

What is EDIT/3000 1-1
EDIT/3000 Features 1-1
EDIT/3000 Editing Modes 1-2
EDIT/3000 File Definitions. 1-2
- INPUT File 1-2
- OUTPUT File 1-2
- WORK File 1-2
- TEXT File 1-3
- JOIN File. 1-3
- HOLD File. 1-3
- USE File 1-3

EDIT/3000 Commands. 1-4

Section II — Page

OPERATING EDIT/3000

Initiating an Interactive Session 2-1
Terminating an Interactive Session 2-3
Initiating a Batch Job. 2-4
Terminating a Batch Job. 2-6
Running EDIT/3000 in an Interactive Session. 2-6
Running EDIT/3000 in Batch Mode 2-7
MPE/3000 and EDIT/3000 Special Controls. 2-9
Entering EDIT/3000 Commands 2-12
EDIT/3000 Special Entry Point 2-13

Section III — Page

EDIT/3000 COMMANDS

Definition of Terms 3-1
- Position. 3-1
- Range 3-1
- Line 3-1
- Column 3-2
- String 3-2

Command Descriptions. 3-2
- ADD Command 3-6
  - Purpose 3-6
  - Form 3-6
  - Description 3-6
  - Limitations 3-6
  - Examples. 3-7
- CHANGE Command 3-10
  - Purpose 3-10
  - Form 3-10
  - Description 3-10
  - Limitations 3-10
  - Examples. 3-12
- COPY Command. 3-14
  - Purpose 3-14
  - Form 3-14
  - Description 3-14
  - Limitations 3-14
  - Examples. 3-14
- DELETE Command. 3-17
  - Purpose 3-17
  - Form 3-17
  - Description 3-17
  - Limitations 3-17
  - Examples. 3-18
- END Command. 3-20
  - Purpose 3-20
  - Form 3-20
  - Description 3-20
  - Limitations 3-20
  - Example 3-20
- FIND Command 3-21
  - Purpose 3-21
  - Form 3-21
  - Description 3-21
  - Limitations 3-21
  - Examples. 3-22
- GATHER Command 3-23
  - Purpose 3-23
  - Form 3-23
  - Description 3-23
  - Limitations 3-24
  - Examples. 3-25
- HOLD Command 3-27
  - Purpose 3-27
  - Form 3-27
  - Description 3-27
  - Limitations 3-27
  - Examples. 3-28
- INSERT Command 3-29
  - Purpose 3-29
  - Form 3-29
  - Description 3-29
  - Limitations 3-30
  - Examples. 3-31
- JOIN Command 3-35
  - Purpose 3-35
  - Form 3-35
  - Description 3-35
  - Limitations 3-36
  - Examples. 3-36
- KEEP Command. 3-39
  - Purpose 3-39
  - Form 3-39
  - Description 3-39
  - Limitations 3-40
  - Examples. 3-42
- LIST Command 3-47
  - Purpose 3-47
  - Form 3-47
  - Description 3-47
  - Limitations 3-47
  - Examples. 3-48

MODIFY Command 3-51
Purpose 3-51
Form 3-51
Description 3-51
Limitations 3-51
Examples 3-52
Q Command 3-54
Purpose 3-54
Form 3-54
Description 3-54
Limitation 3-54
Example 3-54
REPLACE Command 3-55
Purpose 3-55
Form 3-55
Description 3-55
Limitations 3-55
Examples 3-56
SET Command 3-59
Purpose 3-59
Form 3-59
Description 3-59
Limitations 3-64
Examples 3-65
TEXT Command 3-67
Purpose 3-67
Form 3-67
Description 3-67
Limitations 3-68
Examples 3-70
USE Command 3-75
Purpose 3-75
Form 3-75
Description 3-75
Limitations 3-75
Examples 3-76
VERIFY Command 3-79
Purpose 3-79
Form 3-79
Description 3-79
Limitations 3-79
Examples 3-80
XPLAIN Command 3-81
Purpose 3-81
Form 3-81
Description 3-81
Limitations 3-81
Examples 3-82
Z:: = Command 3-83
Purpose 3-83
Form 3-83
Description 3-83
Limitations 3-83
Examples 3-84
:Command 3-86
Purpose 3-86
Form 3-86
Description 3-86
Limitations 3-87
Examples 3-87

Section IV Page
EDIT/3000 ADVANCED COMMANDS
WHILE Command 4-2
Purpose 4-2
Form 4-2
Description 4-3
Limitations 4-3
Examples 4-4
BEGIN Command 4-8
Purpose 4-8
Form 4-8
Description 4-8
Limitations 4-8
Examples 4-9
NOT Command 4-12
Purpose 4-12
Form 4-12
Description 4-12
Limitations 4-12
Examples 4-13
OR Command 4-16
Purpose 4-16
Form 4-16
Description 4-16
Limitations 4-16
Examples 4-17
YES Command 4-20
Purpose 4-20
Form 4-20
Description 4-20
Limitations 4-20
Examples 4-21
PROCEDURE Command 4-24
Purpose 4-24
Form 4-24
Description 4-24
Limitations 4-25
Examples 4-26

Section V Page
MESSAGES AND RECOVERY PROCEDURES 5-1

Section VI Page
CUSTOMIZING EDIT/3000
User Interface Procedures 6-1
HP32201 'USERINIT Procedure 6-2
HP32201 'USERCOMMAND Procedure 6-2
HP32201 'USERADD Procedure 6-3
Examples 6-3

APPENDIX A A-1

APPENDIX B B-1

APPENDIX C C-1

INDEX I-1

ILLUSTRATIONS

Title	Page
Running EDIT/3000 in an Interactive Session	2-6
Running EDIT/3000 in Batch Mode	2-8
Using the BREAK Key	2-10
Using CONTROL Y	2-11
Using ; and & When Entering EDIT/3000 Commands	2-12

TABLES

Title	Page
EDIT/3000 Commands	1-4
Default EDIT/3000 Operating Conditions	3-60
Summary of Parameter Descriptions	3-88
Error Messages	5-2
Command Language Reference Chart	B-2

INTRODUCING EDIT/3000

SECTION I

1-1. WHAT IS EDIT/3000?

EDIT/3000 is a subsystem of the HP 3000 Multiprogramming Executive III operating system (MPE III) that is used to create and manipulate ASCII files.

Characters, strings of characters, or entire lines of characters can be inserted, deleted, replaced, modified, searched for, and otherwise manipulated, by using EDIT/3000 *commands*.

The files to be edited can be source programs written in computer languages such as FORTRAN and COBOL, or text material such as letters, reports, or technical manuals.

1-2. EDIT/3000 FEATURES

With EDIT/3000, it is possible to

- Create and build a new file (called a *WORK* file) by entering commands and lines of text from the standard input device, or text can be entered from special disc files called the *HOLD* file and *JOIN* file.
- Copy the WORK file contents into a permanent file called the *TEXT* file.
- Call the TEXT file contents back into the WORK file for further additions and/or editing.
- Change all occurrences of a character string in the WORK file to another character string with one command.
- Delete characters and lines from the WORK file.
- Locate a string in the WORK file.
- Move and renumber portions of text from one location to another in the WORK file.
- Copy text from the WORK file into a HOLD file.
- Insert text into the WORK file from the standard input device or from the HOLD file.
- Add or insert all or a portion of an external file (called the *JOIN* file) to the WORK file.
- Display all or any portion of the WORK file on the standard output device or write it to any other specified file.
- Modify lines in the WORK file.
- Replace characters, strings of characters, or lines in the WORK file.
- Call external, user-written procedures to process text in the WORK file.
- Execute pre-stored EDIT/3000 commands from a user file (called a USE file).
- Use advanced commands to provide Boolean logic for conditional and repetitive editing.

1-3. EDIT/3000 EDITING MODES

EDIT/3000 can be run in either of two modes:

- Interactive session. In an interactive session, commands and text records normally are entered through an interactive terminal. Messages and other output (such as prompt characters) from EDIT/3000 are listed on the terminal.

- Batch mode. In batch mode, commands and text records are supplied through a batch input medium such as punched cards or magnetic tape. Messages and output from EDIT/3000 are listed on the standard output device (usually a line printer).

1-4. EDIT/3000 FILE DEFINITIONS

Seven files are used by EDIT/3000, as discussed below.

1-5. INPUT FILE

The INPUT file normally is used to enter commands and text records to EDIT/3000. Generally this file is a terminal in interactive mode and a batch input device (such as a card reader) in batch mode. EDITIN is EDIT/3000's formal designator for the input file.

1-6. OUTPUT FILE

EDIT/3000 normally transmits messages (and prompt characters in interactive sessions) to the OUTPUT file. Generally, this file is a terminal in interactive mode and a line printer in batch mode. EDITOUT is EDIT/3000's formal designator for the output file.

1-7. WORK FILE

The WORK file contains the information to be modified. To minimize the possibility of a user accidentally destroying an only copy, EDIT/3000 operates only on the WORK file. Thus, when a file is created and text is added, or when an external file is copied into the EDIT/3000 subsystem for modification, the text is written into the WORK file and all modifications are performed on this file. At any time, you may save this file under a new file name and leave the old file in its previous state, or save this modified file and purge the old file.

The size of the WORK file can be specified by you with a SET SIZE command (see Section III), or the size can be determined by EDIT/3000 by default. The default size of WORK file is determined as follows:

1. If the TEXT file is on disc, its size is known and the WORK file is set to that size plus an allowance for expansion. (See the discussion of file size along with the SET SIZE command in paragraph 3-95.)

2. If the TEXT file is on punched cards, magnetic tape, or any device other than disc, such that the size is not known to the system, or if the text is to be entered from the standard INPUT file, EDIT/3000 accepts approximately 1,767 lines of text (unless the SET SIZE command is used to specify a different size).

Note that only one quarter (4 extents) of the WORK file's disc space is initially allocated. The MPE File System searches for and allocates more extents as needed.

1-8. TEXT FILE

The TEXT file is an external file which is brought into the EDIT/3000 subsystem (copied into the WORK file) with a TEXT command (see Section III). The contents of the TEXT file become the contents of the WORK file and all modifications then are performed on the WORK file. The WORK file can be saved with the present name of the TEXT file, in which case the modifications are incorporated and the previous version of the TEXT file is purged; or the WORK file can be saved under a new name and the previous version of the TEXT file will remain unchanged.

1-9. JOIN FILE

An external file which is brought into the EDIT/3000 subsystem (copied into the WORK file) with a JOIN command (see Section III) is called the JOIN file. All or any portion of a file may be copied into the WORK file with the JOIN command. The information can be added to the end of the current WORK file or it can be inserted into the WORK file at any point. For example, seven lines from a JOIN file could be inserted between lines 13 and 14 of the current WORK file. The contents of the existing JOIN file are not altered by the JOIN command.

1-10. HOLD FILE

The HOLD file serves as a temporary storage file for EDIT/3000 and is generally used for holding interim information (for example, copying parts of the WORK file to the HOLD file and then adding or inserting the HOLD file into other places within the WORK file at some later time).

EDIT/3000 sets the HOLD file to the same size as the WORK file.

The HOLD file is created by EDIT/3000 when the first HOLD command (see Section III) is entered. When subsequent HOLD commands are entered, specifying that another part of the WORK file is to be held, the old HOLD file contents are replaced unless the APPEND parameter is added to the HOLD command, in which case the new contents are added to the end of the current HOLD file.

1-11. USE FILE

The USE file is an external user file containing EDIT/3000 commands (and, optionally, text records) which is called with a USE command (see Section III).

When a USE command is entered, EDIT/3000 reads all commands from the USE file. Any messages from EDIT/3000 are sent to the OUTPUT file, as are requests for text records. Text records then are entered through the INPUT file. (It also is possible, however, to specify that all information, including text records, will be found in the USE file.)

USE files may be nested (for example, a USE file may call another USE file) to perform complicated editing functions.

> Note: Since files are constantly being created and copied while using EDIT/3000, it is possible that the File System might not be able to allocate the disc space needed for a particular operation. There are two possible reasons why this might occur: (1) the system has used up all its disc space, or (2) the disc space allowed for the file's group or account is used up. In the latter case, you will want to either ask users to store and purge files in that group or account, or ask your system manager to increase the disc space limits set for that group or account.

1-12. EDIT/3000 COMMANDS

A summary of EDIT/3000 commands is presented in table 1-1. Included is the command name, the purpose of the command, and the page number in this manual where a complete description of the command can be found.

Table 1-1. EDIT/3000 Commands

COMMAND NAME	PURPOSE	PAGE NO.
ADD	Enters text into the WORK file from the standard input device and/or from the HOLD file.	3-6
BEGIN	Used as the first expression in a BEGIN-END pair.	4-7
CHANGE	Changes existing contents of the WORK file.	3-10
COPY	Copies text from one location to another in the WORK file.	3-14
DELETE	Deletes characters and/or lines from the WORK file.	3-17
END	Terminates EDIT/3000 operation. Or, when used with a matching BEGIN command, terminates a BEGIN-END pair.	3-20
FIND	Finds a specific position or a character string in the WORK file.	3-21
GATHER	Moves portions of text from one location to another in the WORK file and renumbers the lines. (The text is deleted from its original location.) Also can be used to renumber all lines in the WORK file.	3-23
HOLD	Copies part or all of the WORK file into the HOLD file for subsequent re-copying into one or more locations of the WORK file.	3-27
INSERT	Inserts text into the WORK file from the INPUT file or from the HOLD file at a specific position.	3-29
JOIN	Copies all or part of the JOIN file to the WORK file.	3-35
KEEP	Saves all or part of the WORK file into an MPE/3000 file.	3-39
LIST	Lists all or part of the WORK file to the OUTPUT file or to any other specified file.	3-47
MODIFY	Modifies text in the WORK file using one or more subcommands (Delete, Insert, and Replace) of this command.	3-51
NOT	Reverses a flag after executing the command immediately following the NOT command.	4-12
OR	Sets the flag true, or skips the OR command and the command immediately following it if the flag already is true.	4-16
PROCEDURE	Calls and executes a procedure previously written and stored in a segmented library (SL) file.	4-24
Q	Displays a user-defined message at the terminal.	3-54
REPLACE	Replaces one or more lines in the WORK file with new text from the standard input file or from the HOLD file.	3-55
SET	Alters EDIT/3000 default operating criteria.	3-59

Table 1-1. EDIT/3000 Commands (Continued)

COMMAND NAME	PURPOSE	PAGE NO.
TEXT	Copies the contents of a TEXT file into the WORK file, deleting the current WORK file contents.	3-67
USE	Instructs EDIT/3000 to receive commands from the USE file and to send messages to the OUTPUT file and, generally, to expect input from the INPUT file.	3-75
VERIFY	Reports the current EDIT/3000 operating conditions declared in a SET command (or the default conditions not declared in a SET command).	3-79
WHILE	Causes EDIT/3000 to repeat commands in a pre-defined command block.	4-2
XPLAIN	Lists an explanation of all or part of the EDIT/3000 commands.	3-81
YES	Sets a flag for a WHILE command block true.	4-20
Z::= or Z::	Assigns the value of a character string variable to Z::= and uses that value whenever Z:: appears as a part or all of a command.	3-83
:	Instructs EDIT/3000 to pass the rest of the record to MPE. .	3-86

OPERATING EDIT/3000

SECTION II

EDIT/3000 can be run during either a batch job or an interactive session. Initiation of batch jobs and interactive sessions is covered in this section to the extent necessary for you to operate EDIT/3000; however, more extensive descriptions of these procedures can be found in the *MPE Commands Reference Manual.*

In addition to information regarding the initiation of jobs and sessions and the operation of EDIT/3000, MPE and EDIT/3000 special controls are described. These special controls allow you to correct typing errors when information is being entered from a terminal,or to temporarily suspend EDIT/3000 operation.

EDIT/3000 commands are not covered in this section. See Sections III and IV for descriptions of these commands.

2-1. INITIATING AN INTERACTIVE SESSION

To initiate an interactive session

1. Turn the terminal on.
2. Dial the HP 3000 computer system (if the terminal is connected over switched telephone lines).
3. Press the RETURN key (or equivalent) to produce a prompt character (:).
4. Enter the MPE/3000 :HELLO command.

The :HELLO command has the form

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

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

        [;PRI={BS|CS|DS|ES},]

        ;INPRI=inputpriority
        [                   ]
        ;HIPRI
```

Note: Throughout this manual, *optional parameters are shown enclosed in brackets* [ ]. *Required* parameters are shown enclosed in braces { }.

where

sessionname
is an arbitrary name used in conjunction with *username* parameters to form a session identity. The parameter contains from 1 to 8 alphanumeric characters, beginning with a letter. Default: no session name is assigned.

username
is the user's name, established by the Account Manager, which allows you to log on under this account. The parameter contains from 1 to 8 alphanumeric characters, beginning with a letter. (REQUIRED PARAMETER)

userpass
is the user's password, optionally assigned by the Account Manager. It contains from 1 to 8 alphanumeric characters, beginning with a letter.

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

acctpass
is the account password, optionally assigned by the System Manager. It contains from 1 to 8 alphanumeric characters, beginning with a letter.

groupname
is the name of the group to be used for the local file domain and CPU time charges. The parameter *groupname* is established by the Account Manager and contains from 1 to 8 alphanumeric characters, beginning with a letter. Default: the home group if one is assigned by the Account Manager. (Optional if the user has a home group required if a home group is not assigned.)

grouppass
is the group password, optionally assigned by the Account Manager. It contains from 1 to 8 alphanumeric characters, beginning with a letter. (Required if assigned and the user is logging on under other than the home group, not required if the user is logging on under the home group.)

termtype
is the type of terminal used for input. MPE uses this parameter to determine device-dependent characteristics such as delay factors for carriage returns. This parameter must be a number from 0 to 16; the number meanings are shown below:

0 = HP 2749B ASR 33 EIA-compatible Terminal (10 characters per second (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 (10-240 cps).

5 = Memorex 1240 Communication Terminal (10, 15, 30, 60 cps). Note: must have even parity-checking option.

6 = HP 2762A/B (GE 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 2615A Terminal (Beehive MiniBee) (10-240 cps).

10 = HP 2640A/B, HP 2641A, HP 2644A or HP 2645A Interactive Display Terminal (character mode or full program control of block mode transmission) (10-240 cps).

11 = HP 2640A/B, HP 2641A, HP 2644A or HP 2645A Interactive Display Terminal (block mode without program control) (10-240 cps).

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).

If the *termtype* parameter is omitted, a default is assigned according to the termtype configured for that input-output port. Default: For hardwired terminals, the default is determined by the System Supervisor during system configuration. There is no default for terminals that are not hardwired. (Required parameter to insure correct listings if the terminal is not hardwired or if the terminal is not the default *termtype*.)

cpusecs
is the maximum CPU time that a session can use, entered in seconds; it must be a value from 1 to 32767. When the limit is reached, the session is aborted. To specify no limit, enter a question mark (?) or omit the parameter. Default: no limit.

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

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

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

HIPRI
is the request for maximum session-selection input priority, causing the 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. Default: the current *jobfence* and execution limit.

For a more detailed description of the :HELLO command, see the *MPE Commands Reference Manual.*

An example of the :HELLO command, using only the (required) parameters *username.acctname*, is as follows:

```
:HELLO MANAGER.SCR

HP3000 / MPE III B.01.02.  TUE, JUN  3, 1980,  1:49 PM
```

(Displayed on the terminal by MPE/3000)

Note that the *acctname* parameter is preceded by a period. This is the required form for this parameter (see the form example).

2-2. TERMINATING AN INTERACTIVE SESSION

To terminate an interactive session, use the MPE/3000 :BYE command. Following this, MPE/3000 outputs the number of seconds of central processor unit (CPU) time used, the number of minutes connected, and the date and time.

For example,

```
:BYE

CPU=4. CONNECT=1. TUE, JUN  3, 1980,  1:50 PM
```

(Displayed on the terminal by MPE/3000)

2-3. INITIATING A BATCH JOB

The MPE/3000 :JOB command is used to initiate a batch job. Note that the colon must be supplied in the first position of a command record in batch mode (MPE/3000 prompts with the colon in interactive mode).

The form of the :JOB command is

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

     [;TIME=cpusecs]

     [;PRI={BS|CS|DS|ES}]

     ;INPRI=inputpriority
     [                   ]
     ;HIPRI

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

where

jobname
is an arbitrary name used with *username* and *acctname* parameters to form a job identity. The parameter contains from 1 to 8 alphanumeric characters, beginning with a letter. Default: no job name is assigned.

username, *userpass*, *acctname*, *acctpass*, *groupname*, *grouppass*
Are the same as described for the :HELLO command. See paragraph 2-1.

cpusecs
is the maximum CPU time allowed a job, in seconds; it must be a value from 1 to 32767. When this limit is reached, the job is aborted. To specify no limit, enter a question mark (?) or omit this limit. Default: no limit.

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

inputpriority
is the relative input priority used in checking against access restrictions imposed by *jobfence*, if one exists; it takes effect when a job is entered. This parameter must be a value from 1 (lowest priority) to 13 (highest priority). If a value is less than or equal to the *jobfence* set by the Console Operator, the job is deferred. Default: 8.

HIPRI
is the request for maximum input priority, causing the job to be scheduled regardless of current *jobfence* or execution limit for jobs; it overrides *inputpriority*. This parameter can be specified only if you have System Manager or System Supervisor Capability. Default: 8 or *inputpriority* if this parameter is specified.

RESTART
is the request to restart a spooled job interrupted by system termination/restart. This parameter takes effect automatically when the 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: spooled jobs are not restarted after system termination/restart.

device
is the class name or logical device number of the device to receive the listing output. It cannot specify a magnetic tape unit. Default: $STDLIST.

> Note: A non-sharable device (ND) file access attribute is required in order to use this parameter.

outputpriority
is the output priority for the job list file, if destined for a spooled line printer or card punch. It is used to select the next spooled device file (on disc) for output, among all those contending for a specific printer or punch. The parameter must be a value from 1 (lowest priority) to 13 (highest priority). Note that when *outputpriority* is 1, output always is deferred. Thus, an *outputpriority* of 2 or greater should be used for output written from disc.

This parameter applies only to output destined for spooled output devices, and is ignored for other output. Default: 8.

numcopies
is the number of copies of a job listing to be produced. This parameter applies only when listing is directed to a spooled device, and is ignored in other cases. It must be a value from 1 to 255. Default: 1.

For a more detailed description of the :JOB command, see the *MPE Commands Reference Manual.*

An example of the :JOB command, using only the (required) parameters *username.acctname* is as follows:

```
:JOB JOE.VOLLMER,PUB
PRIORITY = DS; INPRI = 8; TIME = UNLIMITED SECONDS
JOB NUMBER = #J2
TUE, JUN  3, 1980,  1:53 PM
HP3000 / MPE III B.01.DC
```

(Printed on the standard job listing device by MPE/3000)

Note that the *acctname* parameter is preceded by a period. This is the required form for this parameter (see the form example). Also note that in a batch job, MPE/3000 prints the *groupname* (PUB in the example) after the *acctname*, whether or not this parameter was included in the :JOB command (it was not included in the :JOB card in the example).

2-4. TERMINATING A BATCH JOB

The MPE :EOJ command is used to terminate a batch job. MPE displays the number of seconds of CPU time used, the number of minutes the job was active, and the date and time.

For example,

```
:EOJ
CPU SEC. = 1.  ELAPSED MIN. = 1.  TUE, JUN  3, 1980,  1:53 PM
```

(Printed on the standard job listing device by MPE)

2-5. RUNNING EDIT/3000 IN AN INTERACTIVE SESSION

To run EDIT/3000 in an interactive session, initiate the session with the MPE :HELLO command, then use the MPE :EDITOR command as shown in figure 2-1.

```
:HELLO MANAGER.SCR

HP3000 / MPE III B.01.02.  TUE, JUN  3, 1980, 1:49 PM


:EDITOR
HP32201A.7.08 EDIT/3000  TUE, JUN  3, 1980,  11:28 AM
(C) HEWLETT-PACKARD CO. 1980
/ADD
    1      THIS IS THE FIRST LINE OF THE WORK FILE
    2      (A NEW FILE IN THIS EXAMPLE)
    3      //
...
/LIST ALL
    1      THIS IS THE FIRST LINE OF THE WORK FILE
    2      (A NEW FILE IN THIS EXAMPLE)
/END
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? YES

 END OF SUBSYSTEM
```

Figure 2-1. Running EDIT/3000 in an Interactive Session

Once accessed, EDIT/3000 displays a slash (/) as a prompt character and waits for a command. The ADD command causes EDIT/3000 to prompt for input by printing a line number. Input then is entered from the terminal. To terminate the ADD command, two slashes may be used as in the example, or CONTROL Y may be used (press and hold CONTROL (or equivalent), and press Y). EDIT/3000 responds by displaying . . . , then displays another prompt for another command. The LIST ALL command causes EDIT/3000 to copy all records in the WORK file to the OUTPUT file (the standard list device—the terminal in this case—unless specified otherwise). The END command terminates EDIT/3000 operation. Note that EDIT/3000 asks if it is OK to clear (the WORK file). If the response is YES (or Y), EDIT/3000 operation terminates and MPE/3000 prints the message END OF SUBSYSTEM. If the response is other than YES (or Y), EDIT/3000 displays a message stating that the clear is not confirmed and then displays a prompt character for another command.

If the contents of the WORK file have not been saved with a KEEP command, the WORK file contents will be lost when a CLEAR is confirmed.

See Sections III and IV for descriptions of EDIT/3000 commands.

2-6. RUNNING EDIT/3000 IN BATCH MODE

In batch mode, MPE/3000 and EDIT/3000 commands (and text if desired) are included with the :JOB command. Unlike sessions, jobs require that the colon prompt character be included with MPE/3000 commands. EDIT/3000 commands in batch mode, however, need not be preceded with the EDIT/3000 prompt character (/). If prompt characters are included with EDIT/3000 commands in batch mode, they will be listed (two slashes will be printed by MPE/3000 instead of just one) on the standard list device but ignored otherwise by EDIT/3000.

An example of running EDIT/3000 in batch mode is shown in figure 2-2. (This job includes the same editing operations as illustrated for the session in figure 2-1.)

A card reader was used as the input device for the job shown in figure 2-2 and the cards submitted with the job were as follows:

```
:JOB MANAGER.SCR
:EDITOR
ADD
THIS IS THE FIRST LINE OF THE WORK FILE
(THE DATA WAS ENTERED FROM CARDS)
// (This card with the two slashes was necessary to terminate the ADD command)
LIST ALL
END
:EOJ
```

Note that the colon prompt characters were included as the first character with the MPE/3000 :JOB, :EDITOR, and :EOJ commands but slash prompt characters were not included with the EDIT/3000 ADD, LIST, and END commands. EDIT/3000 listed the slash prompt characters on the standard list device (and would have output two slashes if each command had been preceded by a slash). Also note that EDIT/3000 listed the line numbers (1 and 2) even though these (like the slashes) had not been included in the data. If line numbers had been included in the data, they would have been considered to be the first characters of the record.

```
:JOB JOE.VOLLMER,PUB
PRIORITY = DS; INPRI = 8; TIME = UNLIMITED SECONDS
JOB NUMBER = #J2
TUE, JUN  3, 1980,  1:53 PM
HP3000 / MPE III B.01.DC



:EDITOR



HP32201A.4.02 EDIT/3000  WED, JUL 30, 1975,  8:39 AM
/ADD
    1      THIS IS THE FIRST LINE OF THE WORK FILE
    2      (THE DATA WAS ENTERED FROM CARDS)


/LIST ALL
    1      THIS IS THE FIRST LINE OF THE WORK FILE
    2      (THE DATA WAS ENTERED FROM CARDS)
/END



 END OF SUBSYSTEM
:EOJ
CPU SEC. = 4.  ELAPSED MIN. = 1.  TUE, JUN 3, 1980,  1:53 PM
```

Figure 2-2. Running EDIT/3000 in Batch Mode

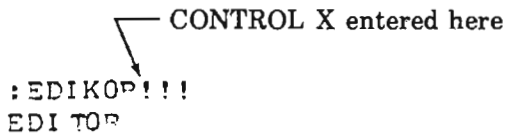
Note: EDITOR input files processed in batch mode do not need a semicolon (;) at the end of each command. Semicolons may still be used to ignore sequence numbers, although this practice is not recommended.

2-7. MPE/3000 and EDIT/3000 SPECIAL CONTROLS

In an interactive session, the following MPE/3000 special controls allow you to correct typing errors, temporarily suspend EDIT/3000 operation, and type continuation lines.

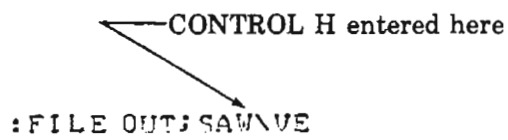
CONTROL X
To delete a line that you have typed in from the terminal (and before you press the RETURN key), type CONTROL X or the equivalent (press and hold CONTROL or its equivalent and press X). MPE/3000 prints !!! and returns the carriage or cursor. Now you can type the corrected line.

For example,



CONTROL H
To erase the last character typed, type CONTROL H or its equivalent (press and hold CONTROL or its equivalent and press H). MPE/3000 prints an indication of the erasure (such as a back slash, or a backspace and a back slash, depending on the terminal type). It is now possible to type the correct character. To erase more than one character, type CONTROL H once for each character to be erased.

An example of CONTROL H is as follows:



The corrected line is entered into the computer as:

```
:FILE OUT;SAVE
```

BREAK
To suspend EDIT/3000 operation temporarily, press the BREAK key or its equivalent. Control is passed to MPE/3000, which prints a colon prompt character and waits for a command. When you are ready to resume EDIT/3000 operation, type :RESUME. MPE/3000 prints the message READ PENDING. Now, enter data or commands to resume EDIT/3000 operation.

An example appears in figure 2-3. Here, the user has suspended EDIT/3000 operation to display files to which he may have access by entering the :LISTF command.

See also the : command, described in Section III, which allows you to enter certain system commands from within EDIT/3000.

LINE FEED
To continue any input line to the next line, press LINE FEED or its equivalent. MPE/3000 will space one line and you can continue to type. (Note: This is most useful when editing text lines which exceed terminal line length.)

```
:EDITOR
HP32201A.7.08 EDIT/3000  TUE, JUN 3, 1980, 11:28 AM
(C) HEWLETT-PACKARD CO. 1980
/ADD
    1       ABC
    2       DEF
    3 <---------- BREAK KEY PRESSED
:LISTF


FILENAME


BATCH1      BATCH2      BATCH3      EDIT3       EDITPROC    FTRAN1
FTRAN10     FTRAN11     FTRAN12     FTRAN13     FTRAN14     FTRAN15
FTRAN16     FTRAN17     FTRAN18     FTRAN19     FTRAN2      FTRAN20
FTRAN21     FTRAN22     FTRAN23     FTRAN24     FTRAN25     FTRAN26
FTRAN27     FTRAN28     FTRAN29     FTRAN3      FTRAN30     FTRAN31
FTRAN32     FTRAN33     FTRAN34     FTRAN35     FTRAN36     FTRAN37
FTRAN38     FTRAN39     FTRAN4      FTRAN40     FTRAN41     FTRAN42
FTRAN43     FTRAN44     FTRAN5      FTRAN6      FTRAN7      FTRAN8
FTRAN9      JUST        K1751515    NAMES       SL          TRACE1
TRACE2      TRACE3      XMPL1       XMPL2       XMPL3
:RESUME
READ PENDING <---------- DISPLAYED BY MPE/3000
GHI
    4       //
...
/LIST ALL
    1       ABC
    2       DEF
    3       GHI
/END
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
 END OF SUBSYSTEM
:
```

Figure 2-3. Using the BREAK Key

If the terminal being used is a device not normally furnished by Hewlett-Packard, the keys used for CONTROL X, CONTROL H, BREAK, and LINE FEED may have other labels. See the *MPE Commands Reference Manual* for terminal characteristics.

In addition to the MPE special controls described above, CONTROL Y is a special control provided by EDIT/3000. It can be used during EDIT/3000 operation only and is useful solely in interactive sessions.

CONTROL Y
To terminate EDIT/3000 command execution during an interactive session on HP terminals, press CONTROL Y (hold control key and press Y) or its equivalent. EDIT/3000 displays . . . and a slash to prompt for the next command.

An example of CONTROL Y usage is shown in figure 2-4.

```
:EDITOR

HP32201A.7.08 EDIT/3000  TUE, JUN 3, 1980, 11:28 AM
(C) HEWLETT-PACKARD CO. 1980
/ADD
    1       ABCDE
    2       FGHIJ
    3       ^Y...        <-- CONTROL Y entered here
/END
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? YES


 END OF SUBSYSTEM
```

Figure 2-4. Using CONTROL Y

2-8. ENTERING EDIT/3000 COMMANDS

";"
To use more than one command in an EDIT/3000 command record, follow each command (except the last one) with a semicolon, as follows:

```
ADD;LIST ALL;END
```

EDIT/3000 will execute the commands in the order in which they are given.

"&"
To continue a command or text entry from one record to the following record, use an ampersand (&) as the last character in the record to be continued.

Examples are shown in figure 2-5.

Note: If it is desired that the & be the last character in a record, instead of being interpreted as a continuation character, enter a blank character after the & (and before pressing RETURN in a session.)

The maximum length for a command string is 256 characters.

A comment may be added as part of a command string, but it may not be embedded in the command. It may appear before a command or follow it after a semicolon. A comment must begin with two adjacent less-than signs (<<) and end with two adjacent greater-than signs (>>). For example,

```
/FIND FIRST;<< MOVE POINTER TO BEGINNING OF FILE. >>

          LINE 1
          (1)
/
```

```
:EDITOR

HP32201A.7.08 EDIT/3000  TUE, JUN 3, 1980, 11:28 AM
(C) HEWLETT-PACKARD CO. 1980
/ADD;LIST ALL;&
/END
    1      THIS LINE AND THE NEXT LINE &
ARE BOTH PART OF RECORD 1
    2      THIS IS RECORD 2
    3      //
...
    1      THIS LINE AND THE NEXT LINE ARE BOTH PART OF RECORD 1
    2      THIS IS RECORD 2
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? YES

  END OF SUBSYSTEM
```

Figure 2-5. Using ; and & When Entering EDIT/3000 Commands.

2-9. EDIT/3000 SPECIAL ENTRY POINT

An entry point to EDIT/3000 is availalbe that will automatically text in a file before the first prompt (/) is displayed and will keep the file when EXIT is typed. (See Section III for a description of the TEXT and KEEP commands.)

In order to specify the file desired, first enter the MPE :FILE command in the form :FILE EDTTEXT= *filename*. Next run EDIT/3000 by using the MPE :RUN command, with BASICENTRY specified for the *entrypoint* parameter. (See the *MPE Commands Reference Manual* for an explanation of the MPE :FILE and :RUN commands.) An example is shown in figure 2-6.

The MPE :FILE command must be used even if the *filename* is EDTTEXT itself. In addition, when this special entry is used, the TEXT and KEEP commands are disallowed since they are executed automatically to the *filename* specified in the MPE :FILE command. If the texted-in file was unnumbered, it will be kept unnumbered.

```
:FILE EDTTEXT=AFILE
:RUN EDITOR.PUB.SYS,BASICENTRY


HP32201A.7.02 EDIT/3000  MON, JUL 31, 1978,  4:36 PM
(C) HEWLETT-PACKARD CO. 1978
/LIST ALL
    1      LINE ONE OF AFILE
    2      LINE TWO OF AFILE
    3      LINE THREE OF AFILE
    4      LINE FOUR OF AFILE
    5      LINE FIVE OF AFILE
/ADD
    6      LAST LINE OF AFILE
    7      //
...
/END
 END OF SUBSYSTEM
```

Figure 2-6. Using the BASICENTRY entry point to EDIT/3000

A simple User Defined Command could be created to facilitate initiating EDIT/3000 with this entry point. For example:

```
ED F1
OPTION LIST
FILE EDTTEXT=!F1
RUN EDITOR.PUB.SYS,BASICENTRY
```

See Section IX of the *MPE Commands Reference Manual* for an explanation of creating and using User Defined Commands.

EDIT/3000 COMMANDS

SECTION III

EDIT/3000 is controlled by commands which specify the type of operation performed, such as adding text (ADD command), or listing the contents of the WORK file (LIST command).

In batch mode, commands are entered as records following the MPE/3000 :EDITOR command; in interactive session, EDIT/3000 displays a prompt character (/) after it begins execution and the commands are entered from the terminal.

3-1. DEFINITION OF TERMS

Terms used in describing EDIT/3000 commands are described in the following paragraphs.

3-2. POSITION

A *position* describes the location in the WORK file of

- A line.
- A column in a line.
- A string.
- The WORK file pointer (EDIT/3000 monitors the edit cycle to determine where in the work file any modifications will occur). See paragraph 3-7.

3-3. RANGE

A *range* expression in a command informs EDIT/3000 which part of the WORK file is affected by the command. A range expression can consist of only one position or it can consist of all characters between and including two positions.

If two positions are declared as a range, the first position must sequentially precede the second position. A slash separates the two positions. A *range list* consists of several ranges, separated by commas. For example, 7/10,50,100/200 is a range list that includes lines 7 through 10, line 50, and lines 100 through 200. A null range in a range list is indicated by two commas. If ALL is specified as the range in a command, the range includes all text currently in the WORK file.

3-4. LINE

A *line* refers to a record in the WORK file. Each line has a unique number that indicates its position. A line number is of the form nnnnn.nnn. For example, 10000.001.

3-5. COLUMN

Each line of the WORK file has a predefined number of columns as defined either by the SET LENGTH command or by the EDIT/3000 subsystem default of 72 characters per line. If text is entered beyond the right margin (default : column 72), a warning message is displayed.

A column position in a range expression specifies a particular column in a line of the WORK file.

3-6. STRING

A *string* refers to a string of characters enclosed within a pair of special (non-alphanumeric) characters used as delimiters. Delimiters may be any of these special (non-alphanumeric) characters except apostrophes ('), commas (,), semicolons (;), periods (.), parentheses, and slashes (/). It is recommended that #, +, -, and = should not be used as string delimiters when advanced commands such as BEGIN, END, WHILE, OR, or NOT are used. A null string is represented by two consecutive delimiters, for example, "".

> Note: Whenever a character is used as a delimiter, it is important that it does not appear in the string.

Nonprinting characters can appear outside the string, if represented by their decimal numeric equivalent (in ASCII code) and preceded by apostrophes ('). Thus the string "EDIT/3000" '13 '10 represents the characters EDIT/3000, a carriage return (13), and linefeed (10). ASCII characters and their decimal representations are shown in Appendix B.

> Note: Blanks beyond the last non-blank character in a line are not recognized as part of a string.

3-7. COMMAND DESCRIPTIONS

To help clarify the command descriptions presented in this manual, the following rules are observed:

- *Optional* command spelling and parameters in the form shown for a command are denoted by being enclosed in brackets [ ].

- *Required* parameters for a command are shown enclosed in braces { }. If two or more parameters are enclosed within the same set of braces, as, for example,

  $$\text{CHANGE} \left\{ \begin{array}{l} \textit{oldstring} \\ \\ \textit{colnum} \end{array} \right\}$$

  then one (and only one) of the parameters *must* be specified.

The following common items apply to all EDIT/3000 commands:

- Command names may be entered using uppercase or lowercase letters. For example, the ADD command can be entered as add.

- The first letter of a command *name* may be used instead of the complete command name, except in the case of the COPY command, which requires CO to prevent conflict with C[HANGE]. For example, you may use A (or a) for ADD, L (or L) for LIST. *Command parameters, however, must be spelled out in full,* unless specified otherwise in the command description. For example, you can use A, HOLD, NOW to abbreviate the ADD command, but you may not use A, H, N, because those parameters may not be abbreviated.

- Blank characters are denoted by the letter b where necessary for clarity.

- An asterisk (*) can be used with some commands to signify the current location of the EDIT/3000 pointer. (This pointer is set and incremented by EDIT/3000 to monitor the edit cycle and to show where in the WORK file any change requested at this time will begin.) For example, a LIST * command immediately following an ADD command will list the last record entered with the ADD command.

- The letter Q (to specify QUIET, or no prompting) may not be separated from the command name with a blank.

Command descriptions are presented in alphabetical order. The advanced commands BEGIN, NOT, OR, PROCEDURE, WHILE, and YES are not included in this section. See Section IV for these commands.

Parameters used in the command descriptions have the same meanings wherever they appear. These meanings are described in the following paragraphs.

> Note: For ease of reference (and so that you do not have to keep turning back to these pages), abbreviated parameter descriptions are provided on page 3-77 at the end of this section.

colnum
any integer from column number 1 through the column number established by the SET RIGHT and SET LENGTH = *colnum* options in effect.

filename
the MPE/3000 file name of any file that you can access. See the *MPE Commands Reference Manual* for a discussion of files. A file equation may be back-referenced by placing an asterisk in front of the file designator.

increment
the value by which line numbers are incremented. The value is from .001 through 99999.999 when SET FORMAT = DEFAULT or from .001 through 999.999 when SET FORMAT = COBOL. However, the line 99999.999 cannot contain any text when FORMAT = DEFAULT is in effect and the line 999.999 cannot contain text if FORMAT = COBOL.

integer
an integer in the range from 1 through 99999 when SET FORMAT = DEFAULT or from 1 through 999 when SET FORMAT = COBOL.

limit
an integer from 1 through 9999.

linenumber
the value assigned to line numbers. The value is from .001 through 99999.999 when SET FORMAT = DEFAULT or from .001 through 999.999 when SET FORMAT = COBOL. As noted above, the line 99999.999 or 999.999, as applicable, cannot contain any text.

newstring
a character string that replaces *oldstring*.

oldstring
a character string to be replaced in the WORK file.

position

position is specified in an EDIT/3000 command as follows:

36	line 36.
36(10)	column number 10 in line 36. This represents the *absolute* column number 10. Absolute column numbers are enclosed in parentheses and do not include a sign.
36(+10)	the 11th (not 10th) non-blank character in line 36. This represents the 10th non-blank position after column 1, or the *relative* column number 10 which is the 11th non-blank column in line 36. Relative column numbers are enclosed in parentheses and include a sign.
36+10	the 10th line after line 36.
36 -10	the 10th line before line 36.
36 (FIRST)	the first non-blank character in line 36.
36(FIRST +6)	the sixth character in line 36, starting with the first non-blank character in line 36.
36(LAST)	the last non-blank character in line 36.
36(LAST -6)	the sixth to last character in line 36, starting with the last non-blank character in line 36.
36(RIGHT -2)	the position in line 36 which is two columns before the right margin.
FIRST	first line.
LAST	last line.
FIRST +10	the 10th line after the first line.
LAST -10	the 10th line before the last line.
"ARRAY"(+5)	the 15th non-blank character after the *first* character in ARRAY.
*(LAST)	the last non-blank character in the current line.
LAST(LAST -5)	the fifth to last non-blank character in the last line.
"AND"(2)	column number 2 of the next line that contains the string "AND."

range

range may reference: 1) a single position in a record, 2) a continuous string of positions, 3) a single record (line), 4) a continuous string of records, 5) a character, or 6) a string of characters. For example,

36(10)	a *range* of column 10 in line 36 only.
36/45	lines 36 through 45.
36(10)/45(9)	a *range* from column 10 in line 36 through column 9 in line 45.
36(+10)/45(9)	a *range* from the 11th non-blank character in line 36 through the 9th column in line 45.
36+10/65 -9	a *range* from the 10th line after line 36 through the 9th line before line 65.
5+3(15)/23 -6(45)	a *range* from column 15 of the 3rd line after line 5 through column 45 of the 6th line before line 23.
16	a *range* of line 16 only.

ALL all lines in the WORK file.

"A" a *range* of the character A only.

"ALL" a *range* of the string ALL only.

"ARE" a *range* of the string ARE only.

"ARE"/"POSSIBLE" a *range* from the first character of the next appearance in the WORK file of the string ARE through the last character of the next appearance of the string POSSIBLE. Note that if the last position in a *range* is a string, *range* includes the last character of that string.

rangelist
one or more ranges. For example, 10/20,23/46 signifies two *ranges*, one consisting of lines 10 through 20 and one consisting of lines 23 through 46; "WRITE"/"READ","KEY"/"NOW" signifies two *ranges*, one starting with the first character of WRITE and extending through the next appearance of READ and one starting with the first character of KEY and extending through the next appearance of NOW.

recnum
a logical record number contained in a file referenced by *filename*. **The first record is number 0.** See the *MPE Commands Reference Manual* for a discussion of logical record numbers.

startcolumn
the starting column number of a portion of a record. This may be any number within the LEFT-RIGHT margins (see the SET command, paragraph 3-92) or an expression of the form FIRST ±n, LAST ±n, LEFT +n, or RIGHT -n, whose value is within these margins.

startline
the first line of a group of lines.

stopcolumn
the ending column number of a portion of a record. This may be any number as described above for startcolumn.

stopline
the last line of a group of lines.

string
a group of characters written exactly as it appears or is to appear. A *string* must be delimited by (enclosed in) any special (non-alphanumeric) characters except: apostrophes, commas, semicolons, periods, parentheses, asterisks, slashes. As noted in paragraph 3-6, #, +, -, and = should not be used as delimiters when advanced commands are also used. For example,

"THIS IS A STRING"
#THIS IS A STRING#

Nonprinting characters can appear outside the string, if represented by their decimal numeric equivalent (in ASCII code — see Appendix A) and preceded by apostrophes ('). For example,

'13"ABC""DEF"'13

3-8. ADD COMMAND

3-9. Purpose. The ADD command is used to enter text into the WORK file from an INPUT file or from the HOLD file.

3-10. Form. The form of the ADD command is

A[DD][Q] [*linenumber*] [,HOLD[Q] [,NOW]]

3-11. Description. The ADD command can add text to the WORK file from an input file or from the HOLD file. The command adds text between existing lines of text in the WORK file by declaring a specific *linenumber* in the ADD command, or to the end of the WORK file if *linenumber* is absent. If an asterisk is entered as the *linenumber*, text will be added immediately following the line on which the pointer is situated. The new *linenumber* is determined in the same way as in the example above.

3-12. Limitations. The ADD command cannot be used to replace existing lines of text in the WORK file. For example, if line 5 already exists in the WORK file, the command ADD 5 will cause the text to be added between line 5 and the line following it.

If the *linenumber* parameter is absent from the ADD command and the WORK file is empty, text begins at the line specified by the SET FROM = *linenumber* command (default = 1). The left and right margins are set by the SET LEFT = *colnum* and SET RIGHT = *colnum* commands. See paragraph 3-92 for a description of the SET command.

3-13. Examples. Examples are provided for several variations of the ADD command. The position of the pointer for each variation is shown in these examples.

- ADD

In the following example, the ADD command adds text records to the end of the WORK file. (The text begins at line 1 in the example because the WORK file is empty and SET FROM is equal to the default value of 1.)

EDIT/3000 prompts with line numbers and text is added. Two slashes (//) are used to terminate the ADD command. The ADD command also can be terminated by CONTROL Y in an interactive session.

The FIND * command (see paragraph 3-32) locates the pointer, which is set to position 1 of the last record entered. EDIT/3000 prints the last line entered, an arrow, and the column number (position) in parentheses. Thus, the pointer is at the first character of the last line entered. Note that the line number is not considered to be part of the record. Note also that when another ADD command is entered, the pointer will be incremented so that text will be added beginning with the *next* line (line 5 in this example).

```
/ADD
    1      I=I.  "WHAT IS EDIT/3000?
    2      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    3      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    4      (MPE/3000) THAT IS USED TO CREATE AND
    5      //
...
/FIND*
    4      (MPE/3000) THAT IS USED TO CREATE AND
           ^(1   )
```

- ADDQ

When the ADDQ command is used, as in the following example, EDIT/3000 does not prompt with line numbers. Text is added as before, and CONTROL Y or // is used to terminate the command.

```
/ADDQ
MANIPULATE ASCII FILES.
CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
LINES OF CHARACTERS CAN BE INSERTED, DELETED,
BY USING EDIT/3000 COMMANDS.
//
...
/FIND*
    3      BY USING EDIT/3000 COMMANDS.
           ^(1   )
```

ADD

- ADD *linenumber*

If the *linenumber* parameter is part of the ADD command, a specific line number is added to the WORK file. Line number 7.1 is added between lines 7 and 8 in the following example.

```
/LIST 7/8
    7     LINES OF CHARACTERS CAN BE INSERTED, DELETED,
    8     BY USING EDIT/3000 COMMANDS.
/ADD 7.1
    7.1   REPLACED, MODIFIED, AND SEARCHED FOR
    7.2   //
...
/FIND*
    7.1   REPLACED, MODIFIED, AND SEARCHED FOR
          *(1  )
/LIST 7/8
    7     LINES OF CHARACTERS CAN BE INSERTED, DELETED,
    7.1   REPLACED, MODIFIED, AND SEARCHED FOR
    8     BY USING EDIT/3000 COMMANDS.
```

- ADD,HOLD

The ADD, HOLD command adds text to the WORK file from an input file, then, after input from this file is terminated with // or CONTROL Y, the command adds text from the HOLD file.

In the following example, lines 9 and 10 are added from the terminal, input is terminated with //, and lines 11 and 12 are added from the HOLD file. (The HOLD file had been created previously with the HOLD command, described in paragraph 3-44.) The pointer is located at position 1 of the last line entered from the HOLD file.

(If the ADD,HOLDQ form of the command had been used, the HOLD file contents added (lines 11 and 12) would not have been listed.)

```
/ADD,HOLD
    9     THE FILES TO BE EDITED CAN BE SOURCE PROGRAMS
   10     WRITTEN IN COMPUTER LANGUAGE SUCH AS FORTRAN
   11     //
...
   11     AND COBOL, OR TEXT MATERIAL SUCH AS LETTERS,
   12     REPORTS, TECHNICAL MANUALS, AND SO FORTH.
...
/FIND*
   12     REPORTS, TECHNICAL MANUALS, AND SO FORTH.
          *(1  )
```

- ADD,HOLD,NOW

The ADD,HOLD,NOW command adds text to the WORK file from the HOLD file only, as shown in the following example. (Note that a new HOLD file has been created.)

```
/ADD,HOLD,NOW
   13      1-2.  EDIT/3000 FEATURES
   14      WITH EDIT/3000, IT IS POSSIBLE TO
   15        CREATE AND BUILD A NEW FILE (CALLED A WORK FILE)
   16        BY ENTERING COMMANDS AND LINES OF TEXT FROM AN
...
/FIND*
   16        BY ENTERING COMMANDS AND LINES OF TEXT FROM AN
            '(1  )
```

- ADD,HOLDQ,NOW

The ADD,HOLDQ,NOW command adds text to the WORK file from the HOLD file without listing the text added. Instead, EDIT/3000 displays the message LAST LINE = *linenumber*, where *linenumber* is the last line added from the HOLD file. In this case, it happens to be the last line in the WORK file. The HOLD file used in this example is the same as that used in the previous example.

```
/ADD,HOLDQ,NOW
...
LAST LINE =     16
/FIND*
   16        BY ENTERING COMMANDS AND LINES OF TEXT FROM AN
            '(1  )
```

- ADD *linenumber*,HOLD,NOW

When the ADD *linenumber*,HOLD,NOW variation of the ADD command is used, the HOLD file contents are added to the WORK file beginning at the line number specified by the *linenumber* parameter.

In the following example, four lines of text are added to the WORK file starting at line 12 (the lines are inserted between lines 12 and 13 of the WORK file). A listing of the complete WORK file shows the result of the various ADD commands.

```
/ADD 12.1,HOLD,NOW
   12.1    1-2.  EDIT/3000 FEATURES
   12.2    WITH EDIT/3000, IT IS POSSIBLE TO
   12.3      CREATE AND BUILD A NEW FILE (CALLED A WORK FILE)
   12.4      BY ENTERING COMMANDS AND LINES OF TEXT FROM AN
...
/FIND*
   12.4      BY ENTERING COMMANDS AND LINES OF TEXT FROM AN
            '(1  )
/LIST ALL
     1      1-1.  WHAT IS EDIT/3000?
     2      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
     3      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
     4      (MPE/3000) THAT IS USED TO CREATE AND
     5      MANIPULATE ASCII FILES.
     6      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
     7      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
     7.1    REPLACED, MODIFIED, SEARCHED FOR, AND
     7.2    OTHERWISE MANIPULATED
     8      BY USING EDIT/3000 COMMANDS.
     9      THE FILES TO BE EDITED CAN BE SOURCE PROGRAMS
    10      WRITTEN IN COMPUTER LANGUAGE SUCH AS FORTRAN
    11      AND COBOL, OR TEXT MATERIAL SUCH AS LETTERS,
    12      REPORTS, TECHNICAL MANUALS, AND SO FORTH.
    12.1    1-2.  EDIT/3000 FEATURES
    12.2    WITH EDIT/3000, IT IS POSSIBLE TO
    12.3      CREATE AND BUILD A NEW FILE (CALLED A WORK FILE)
    12.4      BY ENTERING COMMANDS AND LINES OF TEXT FROM AN
    13        INPUT FILE AND/OR FROM A SPECIAL DISC FILE
    14        (CALLED A HOLD FILE).
```

If ADD *linenumber*, HOLDQ,NOW is specified, only the number of the last line added from the HOLD file is displayed.

CHANGE

3-14. CHANGE COMMAND

3-15. Purpose. The CHANGE command is used to alter specific portions of the WORK file contents.

3-16. Form. The form of the CHANGE command is

$$C[HANGE][Q]\left\{\begin{matrix} oldstring \\ startcolumn[/stopcolumn] \end{matrix}\right\}\left\{\begin{matrix} TO \\ , \end{matrix}\right\}\ \{newstring\}\ [\left\{\begin{matrix} IN \\ , \end{matrix}\right\}\ [rangelist]]$$

For example,

```
CHANGEQ "SPL/3000" TO "FORTRAN/3000" IN 1/10
```

oldstring *newstring* *rangelist*

The example will change all occurrences of SPL/3000 to FORTRAN/3000 in lines 1 through 10. The use of Q suppresses the listing of all changed lines.

3-17. Description. The CHANGE command searches through the range of positions specified in *rangelist*, locating each occurrence of *oldstring* or *startcolumn*. When each occurrence of the target (either *oldstring* or *startcolumn*) is located, *newstring* is substituted for *oldstring*, or *newstring* is substituted for the value contained in the *startcolumn/stopcolumn* area, or *newstring* is inserted before *startcolumn* (if *stopcolumn* was not specified in the command).

3-18. Limitations. The CHANGE command operates within the LEFT-RIGHT margins (see the SET command, paragraph 3-92); however, if *newstring* is longer than the *oldstring* it replaces, the CHANGE command can produce lines that exceed LENGTH by as much as 50 percent.

If the CHANGE *oldstring* TO *newstring* form of the CHANGE command is used and *rangelist* is omitted, the command will not execute unless *oldstring* is located first with a FIND *oldstring* command.

For example, the first line below contains the string "EDIT/3000". The commands FIND FIRST;CHANGE "EDIT/3000" TO "EDITOR/3000";LIST FIRST demonstrate that the command CHANGE "EDIT/3000" TO "EDITOR/3000" did not execute.

```
    1     1-2.  WHAT IS EDIT?


    1      1-2.  WHAT IS EDIT/3000?
    2
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
/FIND FIRST;CHANGE "EDIT/3000" TO "EDITOR/3000;LIST FIRST
    1      1-2.  WHAT IS EDIT/3000?
           ^(1 )
*21*STRING NOT FOUND BEFORE LIMIT
    1      1-2.  WHAT IS EDIT/3000?
```

The *rangelist* parameter can be omitted, however, if the FIND *string* command is used first, where *string* is the string to be changed. See below.

```
/FIND3 FIRST;FIND "EDIT/3000";CHANGE "EDIT/3000" TO "EDITOR/3000"
    1      1-2.  WHAT IS EDIT/3000?
                 (15 )?
    1      1-2.  WHAT IS EDITOR/3000?
```

If the *startcolumn/stopcolumn* parameters are included in the CHANGE command (following a FIND command to locate the line containing the string to be changed), the CHANGE command also will execute, even if the *rangelist* parameter is omitted, as shown by the sequence

FIND 3;CHANGE 1/9 TO "EDITOR/3000";LIST 3

```
/FIND 3;CHANGE 1/9 TO "EDITOR 3000";LIST 3
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
           ?(1   )
    3      EDITOR 3000 IS A SUBSYSTEM OF THE HP 3000
    3      EDITOR 3000 IS A SUBSYSTEM OF THE HP 3000
```

Note: To avoid a time-consuming search of all lines in the WORK file (unless, of course, this is desired) when an *oldstring, newstring* type of change is invoked, use specific line numbers in *rangelist.* In addition, if it is known that items to be changed are contained within specific columns (or to limit the columns for the CHANGE command execution), use the SET LEFT and SET RIGHT commands (see paragraph 3-92).

3-19. Examples. The following examples show several variations of the CHANGE command. The location of the pointer is illustrated at the conclusion of each CHANGE command.

- CHANGE *oldstring* TO *newstring* IN *rangelist*

In the following example, the characters ID (*oldstring*) are changed to EGO (*newstring*). Note that *every* occurrence of the characters ID was changed. To change only the word ID, the *newstring* parameter should be entered as "bIDb".

> Note: Where necessary for clarity, blank spaces are signified by the letter b in this manual.

After another change to change EGO back to ID, a new CHANGE command is entered to obtain the desired result.

The pointer is located one position beyond the last character of *newstring* at the conclusion of the CHANGE command.

```
/LIST ALL
    1     THE FOLLOWING LINES WILL BE EDITED
    2     USING THE CHANGE COMMAND.
    3
    4     IT HAS BEEN SAID THAT THE ID MAY
    5     BE IDENTIFIED AS PART OF THE PSYCHE.
/CHANGE "ID" TO "EGO" IN 4/5
    4     IT HAS BEEN SAEGO THAT THE EGO MAY
    5     BE EGOENTIFIED AS PART OF THE PSYCHE.
/FIND*
    5     BE EGOENTIFIED AS PART OF THE PSYCHE.
            (7  )↑
/CHANGE "EGO" TO "ID" IN 4/5
    4     IT HAS BEEN SAID THAT THE ID MAY
    5     BE IDENTIFIED AS PART OF THE PSYCHE.
/FIND*
    5     BE IDENTIFIED AS PART OF THE PSYCHE.
            (6  )↑
/CHANGE " ID " TO " EGO " IN 4
    4     IT HAS BEEN SAID THAT THE EGO MAY
/FIND*
    4     IT HAS BEEN SAID THAT THE EGO MAY
                                  (31 )↑
```

- CHANGE *startcolumn* TO *newstring* IN *rangelist*

The following CHANGE command example is used to insert the characters FOR EXAMPLE, BY before column 1 in line 2. Note that at the conclusion of the CHANGE command, the pointer is set immediately after the characters that were changed.

```
/LIST 2
    2     USING THE CHANGE COMMAND.
/CHANGE 1 TO "FOR EXAMPLE, BY " IN 2
    2     FOR EXAMPLE, BY USING THE CHANGE COMMAND.
/FIND*
    2     FOR EXAMPLE, BY USING THE CHANGE COMMAND.
                       (17 )^
```

- CHANGE *oldstring,newstring,rangelist*

The next example illustrates the use of commas instead of the words TO and IN in the CHANGE command.

```
/LIST 4
    4     IT HAS BEEN SAID THAT THE EGO MAY
/CHANGE " EGO "," ID ",4
    4     IT HAS BEEN SAID THAT THE ID MAY
/FIND*
    4     IT HAS BEEN SAID THAT THE ID MAY
                                  (30 )↑
```

If the CHANGE *startcolumn/stopcolumn* TO *newstring* form of the CHANGE command is used and *newstring* contains a different number of characters than that specified in *startcolumn/stopcolumn*, EDIT/3000 will contract or extend the line to accommodate the characters specified in *newstring*. See below.

```
/F "EDIT/3000"
    3     EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
          ↑(1  )
/C 1/9 TO "EDITOR/3000"
    3     EDITOR/3000 IS A SUBSYSTEM OF THE HP 3000
/C 1/11 TO "EDITOR"
    3     EDITOR IS A SUBSYSTEM OF THE HP 3000
```

- CHANGE *oldstring* TO *newstring* IN ALL

When ALL is specified in the *rangelist* parameter of the CHANGE command, EDIT/3000 searches through the entire WORK file for all occurrences of *oldstring* within the columns specified by SET LEFT and SET RIGHT and changes all such occurrences to *newstring*.

```
/LIST ALL
    1     1-2.  WHAT IS EDIT/3000?
    2
    3     EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4     MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5     (MPE/3000) THAT IS USED TO CREATE AND
    6     MANIPULATE ASCII FILES.
    7
    8     CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9     LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10     REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11     MANIPULATED BY USING EDIT/3000 COMMANDS.
/CHANGE "EDIT/3000" TO "HP 3000 TEXT EDITOR" IN ALL
    1     1-2.  WHAT IS HP 3000 TEXT EDITOR?
    3     HP 3000 TEXT EDITOR IS A SUBSYSTEM OF THE HP 3000
   11     MANIPULATED BY USING HP 3000 TEXT EDITOR COMMANDS.
/FIND*
   11     MANIPULATED BY USING HP 3000 TEXT EDITOR COMMANDS.
                                          (41 )↑
```

COPY

3-19a. COPY COMMAND

3-19b. **Purpose.** The COPY command copies text from one location into another in the WORK file.

3-19c. **Form.** The form of the COPY command is

$$\text{CO[PY] [Q]}\textit{range}\left\{\begin{matrix}\text{TO}\\ ,\end{matrix}\right\}\{\textit{linenumber}\}[\left\{\begin{matrix}\text{BY}\\ ,\end{matrix}\right\}\textit{increment}]$$

3-19d. **Description.** The COPY command copies the lines specified by *range* to the *linenumber* indicated. The new line numbers are incremented by the value specified in the command. If *increment* is not included, it defaults to the SET DELTA = *increment* in effect. If necessary, in order to fit the lines to be copied into the space available between existing lines, EDIT/3000 will adjust the increment in effect until the minimum increment of .001 is reached. If the lines to be copied will not fit into the available space, the error message "*18* INSUFFICIENT LINE NUMBERS OR WORK SPACE — COMMAND NOT PERFORMED" will be displayed and no text will be copied.

During the copy, the line numbers of the lines being copied from and copied to are displayed thus:

5→50

where 5 represents the line number *from* which data is copied and 50 represents the line number *to* which the data is copied. If this output is not desired, it can be inhibited by adding Q to the COPY command.

In an interactive session, a COPY command may be terminated by entering CONTROL Y.

3-19e. **Limitations.** EDIT/3000 will reject a COPY command and print an error message if:

1. A *linenumber* specified in the TO portion of a COPY command already exists in the WORK file.
2. The SET DELTA = *increment* in effect cannot be adjusted by EDIT/3000 to allow as many line numbers as there are lines to be copied.
3. A BY *increment* (or ,*increment*) parameter is specified that does not allow as many line numbers as there are lines to be copied.

3-19f. **Examples.** Several variations of the COPY command are shown in the following examples. The location of the pointer is illustrated at the conclusion of each COPY command.

- COPY ALL TO *linenumber* BY *increment*

The following example uses the command COPY ALL TO 2.1 BY .01 to copy all lines to 2.1, 2.11, 2.12, 2.13, 2.14, 2.15, and 2.16. The pointer is located at the first position of the last line copied (2.16).

```
/L ALL
    1       1/2.  WHAT IS EDIT/3000?
    2
    3       EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4       MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5       (MPE/3000) THAT IS USED TO CREATE AND
    6       MANIPULATE ASCII FILES.
    7
/COPY ALL TO 2.1 BY .01
    1       =>      2.1
    2       =>      2.11
    3       =>      2.12
    4       =>      2.13
    5       =>      2.14
    6       =>      2.15
    7       =>      2.16
/FIND*
    2.16
            ^(1 )
/L ALL
    1       1/2.  WHAT IS EDIT/3000?
    2
    2.1     1/2.  WHAT IS EDIT/3000?
    2.11
    2.12    EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    2.13    MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    2.14    (MPE/3000) THAT IS USED TO CREATE AND
    2.15    MANIPULATE ASCII FILES.
    2.16
    3       EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4       MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5       (MPE/3000) THAT IS USED TO CREATE AND
    6       MANIPULATE ASCII FILES.
    7
/
```

- COPYQ *range*, *linenumber*

The next example copies lines 2, 2.1, 2.11, 2.12, and 2.13 to a location starting at line 6.01.

```
/COPYQ 2/2.13, 6.01
/L ALL
     2
     2.1    1/2.  WHAT IS EDIT/3000?
     2.11
     2.12   EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
     2.13   MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
     2.14   (MPE/3000) THAT IS USED TO CREATE AND
     2.15   MANIPULATE ASCII FILES.
     2.16
     3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
     4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
     5      (MPE/3000) THAT IS USED TO CREATE AND
     6      MANIPULATE ASCII FILES.
     6.01
     6.02   1/2.  WHAT IS EDIT/3000?
     6.03
     6.04   EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
     6.05   MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
     7
/
```

3-20. DELETE COMMAND

3-21. Purpose. The DELETE command is used to delete lines and/or character strings from the WORK file.

3-22. Form. The form of the DELETE command is

D[ELETE][Q] [*rangelist*]

For example,

DELETEQ 1/10

3-23. Description. The DELETE command will delete complete lines, specific character strings from a line, or the entire contents of the WORK file. If no *rangelist* is specified, EDIT/3000 deletes the line in which the pointer is located. A *rangelist* that specifies a *startline/stopline* pair deletes these lines and all lines in between. A *rangelist* that specifies a *startline,stopline* pair deletes only these two lines. A *rangelist* that specifies positions within lines, separated by a slash, results in the deletion of all characters contained between these two columns, including any lines that fall between the two positions.

When the DELETEQ form of the command is used,EDIT/3000 does not display the lines deleted. Instead, a message is displayed showing the number of lines deleted. For example,

```
/S SHORTIT EDIT3:DO 1/4
NUMBER OF LINES DELETED = 4
/S
CLEAR? Y
```

In interactive session, you may terminate an active DELETE command at any time by typing CONTROL Y on the terminal.

3-24. Limitations. WORK file contents removed with a DELETE command are lost. When a DELETE ALL command is entered (deleting the entire WORK file), EDIT/3000 will ask if it is OK to clear, thus confirming your intent.

> Note: EDIT/3000 does not list the lines removed when the DELETE ALL command is used.

The DELETE command operates between LEFT and RIGHT margin values unless the *rangelist* specifies *lines* (without column numbers). In this case, entire lines are deleted, and the DELETE command ignores SET LEFT and SET RIGHT commands. See paragraph 3-92 for a discussion of the SET command.

3-25. Examples. Several variations of the DELETE command are shown in the following examples. The location of the pointer is illustrated at the conclusion of each DELETE command.

- DELETE *startline/stopline*

The first example specifies a startline/stopline pair of 8/10 in *rangelist* and lines 8 through 10 are deleted.

```
/LIST ALL
    1     1-2.  WHAT IS EDIT/3000?
    2
    3     EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4     MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5     (MPE/3000) THAT IS USED TO CREATE AND
    6     MANIPULATE ASCII FILES.
    7
    8     CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9     LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10     REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11     MANIPULATED BY USING EDIT/3000 COMMANDS.
/DELETE 8/10
    9     CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9     LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10     REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
/FIND*
   11     MANIPULATED BY USING EDIT/3000 COMMANDS.
          *(1  )
```

- DELETE *startline (colnum)/stopline (colnum)*

The following example specifies columns within lines. This results in the deletion of the two characters contained in the two columns and all characters between the two positions.

```
/FIND 4(16)
    4     MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
                  (16 )*
/FIND 5(11)
    5     (MPE/3000) THAT IS USED TO CREATE AND
                (11 )*
/DELETE 4(17)/5(11)
    4     MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5     (MPE/3000) THAT IS USED TO CREATE AND
/FIND*
    5     THAT IS USED TO CREATE AND
          *(1  )
/LIST 4/5
    4     MULTIPROGRAMMING
    5     THAT IS USED TO CREATE AND
```

- DELETE *startline,stopline*

The startline,stopline form of the *rangelist*, as in the following example, causes only the two lines named in *rangelist* to be deleted. (Note that more than two lines, separated by commas, could be specified, in which case those lines named would be deleted.)

```
/DELETE 3,6
    3     EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    6     MANIPULATE ASCII FILES.
/FIND *
    7
          *(1  )
```

- DELETE *string/string*

When the *rangelist* parameter consists of two strings, as, for example,

DELETE "SUBSYSTEM"/"ASCII"

EDIT/3000 deletes all characters between the *first* character of the *first* string and the *last* character of the *last* string, including any lines that are between these two characters.

```
/LIST ALL
    1     1-2.  WHAT IS EDIT/3000?
    2
    3     EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4     MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5     (MPE/3000) THAT IS USED TO CREATE AND
    6     MANIPULATE ASCII FILES.
    7
    8     CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9     LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10     REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11     MANIPULATED BY USING EDIT/3000 COMMANDS.
/FIND FIRST
/DELETE "SUBSYSTEM"/"ASCII"
    3     EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4     MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5     (MPE/3000) THAT IS USED TO CREATE AND
    6     MANIPULATE ASCII FILES.
/FIND*
    6      FILES.
          *(1   )
/LIST 3/6
    3     EDIT/3000 IS A
    6      FILES.
```

If the same string is used for both ranges in *rangelist,* as, for example,

DELETE "SUBSYSTEM"/"SUBSYSTEM"

EDIT/3000 deletes this string only.

```
/LIST ALL
    1     1-2.  WHAT IS EDIT/3000?
    2
    3     EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4     MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5     (MPE/3000) THAT IS USED TO CREATE AND
    6     MANIPULATE ASCII FILES.
    7
    8     CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9     LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10     REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11     MANIPULATED BY USING EDIT/3000 COMMANDS.
/FIND FIRST
/DELETE "SUBSYSTEM"/"SUBSYSTEM"
    3     EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
/FIND*
    3     EDIT/3000 IS A  OF THE HP 3000
                 (16 )?
/LIST 3
    3     EDIT/3000 IS A  OF THE HP 3000
```

3-26. END COMMAND

3-27. Purpose. The END command is used to terminate EDIT/3000 operation, thus returning control to MPE/3000, or to end BEGIN/END pairs. (See Section IV for a discussion of the END command used in a BEGIN/END pair.)

3-28. Form. The form of the END command is

E[ND]

3-29. Description. The END command terminates EDIT/3000 operation and returns control to MPE/3000. When an END command is entered in an interactive session, EDIT/3000 will display the message:

```
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR?
```

If the answer is YES (or Y), EDIT/3000 will terminate and MPE/3000 will display END OF SUBSYSTEM. If the answer is other than YES (or Y), EDIT/3000 displays the following message

```
CLEAR NOT CONFIRMED - TEXT IS UNCHANGED
```

and prompts for another command. Both the HOLD and WORK files remain unchanged.

If the work file is not altered or it if is altered but kept, the message

```
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR?
```

does not appear, and EDIT/3000 is terminated without requiring a response.

In batch mode, the END command is executed unconditionally.

3-30. Limitations. Unless a KEEP command (see paragraph 3-62) is entered before an END command, the contents of the WORK file will be lost when the END command is executed. (The HOLD file is always lost when the END command executes.)

END in a USE file causes the USE command to terminate (see the description of the USE command in this section).

3-31. Example. The following examples show the use of the END command.

```
/END
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? YES

 END OF SUBSYSTEM
:
```

If SHORT=TRUE, then the full string, IF IT IS OK TO CLEAR RESPOND "YES," is not displayed.

```
/SET SHORT;END
CLEAR? YES

 END OF SUBSYSTEM
:
```

3-32. FIND COMMAND

3-33. Purpose. The FIND command is used to locate a specific position in the WORK file.

3-34. Form. The form of the FIND command is

F[IND][Q] [* / *range*]

3-35. Description. The FIND command searches *forward* in the WORK file from the present position of the pointer until the *range* parameter is found. If *range* is a character string, FIND searches forward until the first occurrence of the string is found. The pointer is set at the first character of the string. If *range* is a line number, EDIT/3000 moves the pointer to the first column in that line. If *range* is not a single position, but is of the form *startline/stopline*, EDIT/3000 will find the first position specified (*startline*) or the first existing line before the second position specified (*stopline*). For example, FIND 1/200 will find the first position of the first line (1) if it exists or of the first existing line before line 200. If *range* is specified as a position in a line, the pointer is moved to that position. FIND * finds the current location of the pointer. The position of the pointer is always indicated by EDIT/3000 unless the Q parameter is specified in the FIND command.

3-36. Limitations. If a SET command (see paragraph 3-92) has been used to set left and right margins in the WORK file, then a search for a string will only operate within these margins. For example, if the margins are set with the command SET LEFT = 50, RIGHT = 72, a FIND command will only locate a string that occurs totally within columns 50 through 72.

Unless it is desired that the entire contents of the WORK file from the present position of the pointer to the end of the file be searched, a FIND command should be entered with a *range,* as for example,

```
FIND "AJAX"/* +10
      or
FIND "AJAX"/100
```

The first of the foregoing commands will search only the 10 lines immediately following the *present* location of the pointer, while the second example will search from the pointer to line 100.

If *range* is not forward of the present location of the pointer, the remainder of the file will be searched and *range* will not be found. A FIND FIRST;FIND *range* command may be used in this case. The pointer will return to the first position of the first line in the file and search from there.

A message will be printed if EDIT/3000 cannot comply with a FIND command. Reasons for failure include specifying a location or string that does not exist, specifying a *range* in the wrong order (last location before first location), specifying a *range* behind the current location of the pointer in the file, specifying a *range* outside the left or right margins established by a SET command, and specifying a *range* consisting of a string which does not exist. Note that specifying ALL for the *range* is invalid with the FIND command.

3-37. Examples. The following examples illustrate the use of the FIND command.

The FIND FIRST command finds the first line in the WORK file, then positions the pointer to the first position in this line.

FIND " ENTIRE" finds the string bENTIRE (b signifies a blank character) and positions the pointer to the first character in this string.

The next command, FIND " HP 3000" results in error message number 21, "STRING NOT FOUND BEFORE LIMIT" because EDIT/3000 only searches *forward* in the WORK file and HP 3000 occurs before line 8 (the previous location of the pointer).

The command FIND FIRST;FIND "HP 3000" causes EDIT/3000 to locate the first line, then the line containing the string HP 3000. To prevent EDIT/3000 from listing the first line, the command FINDQ FIRST;FIND "HP 3000" is used.

Finally, the command FIND 8(45) finds the 45th position in line 8.

```
/LIST ALL
    1      1-2.  WHAT IS EDIT/3000?
    2
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    7
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
/FIND FIRST
    1      1-2.  WHAT IS EDIT/3000?
           ↑(1   )
/FIND " ENTIRE"
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
                                              (38 )↑
/FIND " HP 3000"
*21 *STRING NOT FOUND BEFORE LIMIT
/FIND FIRST;FIND " HP 3000"
    1      1-2.  WHAT IS EDIT/3000?
           ↑(1   )
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
                                     (32 )↑
/FINDQ FIRST;FIND " HP 3000"
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
                                     (32 )↑
/FIND 8(45)
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
                                                (45 )↑
```

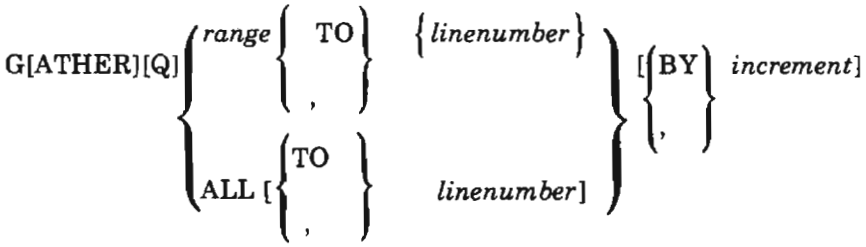
The next example illustrates that FIND *startline/stopline* finds the first existing line (in the WORK file) before the second linenumber specified.

```
/DQ 1/7
NUMBER OF LINES DELETED = 7
/L ALL
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
/FIND 1/10
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
           ^(1   )
/
```

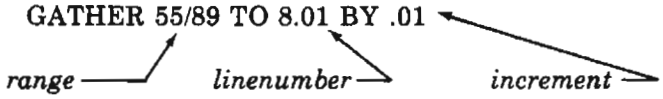
3-38. GATHER COMMAND

3-39. Purpose. The GATHER command removes text from one location to another in the WORK file.

3-40. Form. The form of the GATHER command is



For example,



Note: Use the SET SIZE command (see paragraph 3-92) to set the SIZE to a large value if extensive gathering is to be done.

3-41. Description. When the *range* parameter is specified in a GATHER command, the *linenumber* parameter must be specified also, as for example,

GATHER 80/100 TO 230.1

or

GATHER 80/100,230.1 (a comma may be used instead of TO)

EDIT/3000 will move lines 80 through 100 to 230.1, incrementing each line by the SET DELTA = *increment* in effect (the default is 1 if SET FORMAT = DEFAULT and 0.1 if SET FORMAT = COBOL), and deleting lines 80 through 100 from their original locations. The SET DELTA = *increment* in effect could have been overridden by using the *increment* parameter in the GATHER command (see paragraph 3-92 for a discussion of the SET command).

EDIT/3000 will adjust the increment in effect until the minimum increment of .001 is reached in order to fit the lines to be moved into the space available between existing lines. If lines to be moved will not fit into the available space, the operation will halt, an error message will be displayed, and no text will be moved.

When text to be moved would overlap existing lines, line numbers must be changed before the GATHER command is used. One method of accomplishing this is with a HOLD command as follows:

```
HOLD 10/15 (Hold 6 lines of text)

DELETE 10/15 (Delete the 6 lines)

GATHER 100/1500 TO 10 (Add 1401 lines between lines 10 and 15)

FIND *

ADD linenumber,HOLD,NOW (Add the 6 lines held in the HOLD file to the end of the 1401 lines
                         which were moved. linenumber = last line + increment in effect)
```

To override the SET DELTA = *increment* in effect, the BY *increment* parameter is used in the GATHER command.

To renumber all lines in the WORK file, the GATHER ALL command is used. New line numbers are assigned, starting with the SET FROM = *linenumber* in effect (default = 1). Each line is incremented by the SET DELTA = *increment* in effect (default = 1 if SET FORMAT = DEFAULT and 0.1 if SET FORMAT = COBOL). The new line numbers are not displayed.

In an interactive session, a GATHER command (except GATHER ALL) can be terminated before it is complete by typing CONTROL Y.

3-42. Limitations. EDIT/3000 will reject a GATHER command and print an error message if:

1. A TO *linenumber* (or *,linenumber*) parameter is not provided. Exception: when a GATHER ALL command is used, this parameter is optional.

2. A *linenumber* specified in the TO portion of a GATHER command already exists in the WORK file.

3. The SET DELTA = *increment* in effect cannot be adjusted by EDIT/3000 to allow as many line numbers as there are lines to be moved.

4. A BY *increment* (or *,increment*) parameter is specified that does not allow as many line numbers as there are lines to be moved.

3-43. Examples. Several variations of the GATHER command are shown in the following examples. The location of the pointer is illustrated at the conclusion of each GATHER command.

- GATHER *range* TO *linenumber* BY *increment*

The following example uses the command GATHER 8/11 to 2.1 BY .01 to move lines 8 through 11 to 2.1, 2.11, 2.12, and 2.13. The pointer is located at the first position of the last line moved (2.13). Lines 8 through 11 are deleted.

```
/LIST ALL
    1      1-2.  WHAT IS EDIT/3000?
    2
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    7
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
/GATHER 8/11 TO 2.1 BY .01
    8      =>     2.1
    9      =>     2.11
   10      =>     2.12
   11      =>     2.13
/FIND*
    2.13   MANIPULATED BY USING EDIT/3000 COMMANDS.
           '(1   )
/LIST ALL
    1      1-2.  WHAT IS EDIT/3000?
    2
    2.1    CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    2.11   LINES OF CHARACTERS CAN BE INSERTED, DELETED,
    2.12   REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
    2.13   MANIPULATED BY USING EDIT/3000 COMMANDS.
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    7
```

- GATHERQ *range* TO *linenumber*

The next example moves lines 2.1, 2.11, 2.12, and 2.13 from their present positions to a location starting at line 7.1 and the GATHER ALL command renumbers all lines to integer-type numbers. The WORK file is now in the same condition as it was before the GATHER commands were used.

```
/GATHERQ 2.1/2.13 TO 7.1
/FIND*
    7.4    MANIPULATED BY USING EDIT/3000 COMMANDS.
           '(1   )
/LIST ALL
    1      1-2.  WHAT IS EDIT/3000?
    2
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    7
    7.1    CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    7.2    LINES OF CHARACTERS CAN BE INSERTED, DELETED,
    7.3    REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
    7.4    MANIPULATED BY USING EDIT/3000 COMMANDS.
/GATHER ALL;LIST ALL
    1      1-2.  WHAT IS EDIT/3000?
    2
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    7
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
```

GATHER

3-44. HOLD COMMAND

3-45. Purpose. The HOLD command copies lines of text from the WORK file to the HOLD file.

3-46. Form. The form of the HOLD command is

H[OLD][Q][[*range*] [,APPEND]]

3-47. Description. The HOLD command is used to copy all or a portion of the WORK file contents into the HOLD file. A listing of all lines transferred into the HOLD file is displayed (unless the HOLDQ form of the command is used). The WORK file contents are unaffected.

To clear an existing HOLD file of all contents, the HOLD (or H) command, with no *range* specified, is used. In an interactive session, EDIT/3000 will display the message

CLEAR HOLD?

If the response is YES (or Y), the HOLD file is cleared. Any other response is interpreted as NO. In batch mode, EDIT/3000 clears the HOLD file automatically.

3-48. Limitations. If a HOLD file exists when a HOLD command is used, the contents of the existing HOLD file are deleted and replaced by the new contents unless the HOLD *range*,APPEND form of the HOLD command is used, in which case the new contents are added to the end of the existing HOLD file. If the APPEND parameter is not included with the HOLD command, the following message is displayed in an interactive session:

CLEAR HOLD?

If the response is YES or Y, the existing HOLD file is cleared. Any other response is interpreted as NO. In batch mode, EDIT/3000 clears the HOLD file automatically.

The HOLD command operates only within the column boundaries established by the SET LEFT and SET RIGHT margins in effect. See paragraph 3-92 for a discussion of the SET LEFT and SET RIGHT commands.

3-49. **Examples.** Several variations of the HOLD command are shown in the following examples.

- HOLD *range*

The following example uses the HOLD 1/6 command to create a HOLD file and copy lines 1 through 6 from the WORK file into this file. The lines are listed by EDIT/3000.

The WORK file contents are not altered.

```
/LIST ALL
    1      1-2.  WHAT IS EDIT/3000?
    2
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    7
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
/HOLD 1/6
1-2.  WHAT IS EDIT/3000?

EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
(MPE/3000) THAT IS USED TO CREATE AND
MANIPULATE ASCII FILES.
```

- HOLDQ *range*

The next command, HOLDQ 1/6, copies the same lines to the HOLD file but does not list the lines. Instead, the message

HOLD FILE LENGTH IS 6 RECORDS

is printed.

Note that since a HOLD file exists, EDIT/3000 asks if it is OK to clear the HOLD file.

```
/HOLDQ 1/6
CLEAR HOLD? YES
HOLD FILE LENGTH IS 6 RECORDS
```

- HOLD *range*, APPEND

The HOLD 8/11,APPEND command in the next example adds lines 8 through 11 to the end of the existing HOLD file.

```
/HOLD 8/11,APPEND
CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
LINES OF CHARACTERS CAN BE INSERTED, DELETED,
REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
MANIPULATED BY USING EDIT/3000 COMMANDS.
```

- CLEARING THE HOLD FILE

To clear the HOLD file, the HOLD command, with no range specified, is used as illustrated in the following example.

```
/HOLD
CLEAR HOLD? Y
```

3-50. INSERT COMMAND

3-51. Purpose. The INSERT command inserts character strings or lines of text into the WORK file at a specified position, irrespective of line boundaries.

3-52. Form. The form of the INSERT command is

$$\text{I[NSERT][Q]}\ [\textit{position}]\ [\left\{\begin{matrix}\text{BY}\\ \\ ,\end{matrix}\right\}\ [\textit{increment}]]\ [,]\ [\text{HOLD[Q]}\ [,]\ [\text{NOW}]]$$

3-53. Description. The INSERT command is used to insert character strings in a line of text in the WORK file or to insert complete lines of text in the WORK file.

In an interactive session, EDIT/3000 prompts by displaying the line number referenced in the *position* parameter and the contents of the line, then displays an arrow under the column, if a column was specified in the *position* parameter. (If the Q parameter is included in the INSERT command the prompt is not displayed.) A specific column in a line is specified in the *position* parameter by enclosing the column number in parentheses. For example,

```
INSERT 5(10)
```

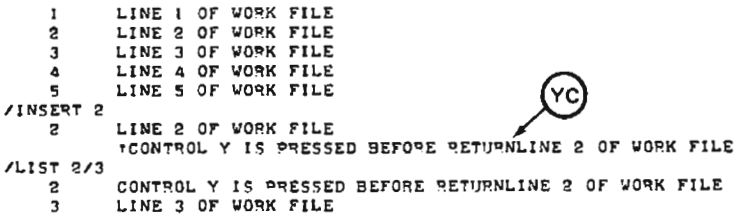
references the 10th column of line 5.

If no column is specified, as in INSERT 5, the leftmost column is assumed by EDIT/3000 as determined by the SET LEFT = *colnum* option in effect (default = column 1). If a column number that is outside the SET LEFT, SET RIGHT options in effect is specified, the INSERT command will not execute and EDIT/3000 will display an error message. (See paragraph 3-92 for a discussion of the SET command.) If the text generated using an INSERT command consists of more characters than are allowed in a line, new lines are created by EDIT/3000 to accommodate the excess characters (and the existing line). In this case, the Editor attempts to break the line at a blank character. New line numbers are determined by the SET DELTA = *increment* option in effect. The SET DELTA option can be overridden with the BY *increment* (or ,*increment*) parameter of the INSERT command.

If new lines are being inserted before a particular line in the WORK file, the insert command saves that line. This saved line is then added after the new lines, once control Y has been pressed (see example 2 below). If the SET DELTA=*increment* option in effect (or the BY *increment* parameter) does not allow enough new line numbers to be added when new lines are being inserted, EDIT/3000 will insert new lines until it can no longer add new line numbers, then terminate the INSERT mode and print an error message, as well as the saved line which was lost from the work file. See the example of this in paragraph 3-55.

In an interactive session, the INSERT command may be terminated with CONTROL Y or // in any one of four ways:

1. Press CONTROL Y after the last character has been inserted and before RETURN is pressed, as follows:



The characters are inserted on the same line as the line specified (2) in the *position* parameter. No characters will be moved to the next line unless more characters are inserted than the length of the line can accept.

2. Press CONTROL Y after the last character has been inserted and after RETURN is pressed, as follows:

```
/INSERT 3
    3       LINE 3 OF WORK FILE
            ↑CONTROL Y IS PRESSED AFTER RETURN  (CR) (YC)
    3.1     LINE 3 OF WORK FILE
/LIST 3/4
    3       CONTROL Y IS PRESSED AFTER RETURN
    3.1     LINE 3 OF WORK FILE
    4       LINE 4 OF WORK FILE
```

The text to be inserted replaces the line specified in the *position* parameter and the text previously contained in this line is given a new line number. Thus, the INSERT command caused a new line to be inserted before an existing line.

3. Enter // after the last character has been inserted and before RETURN is pressed, as follows:

```
/INSERT 3
    3       LINE 3 OF WORK FILE
            ↑// IS ENTERED BEFORE RETURN // (CR)
LINE 3 OF WORK FILE
/LIST 3/4
    3       // IS ENTERED BEFORE RETURN LINE 3 OF WORK FILE
    4       LINE 4 OF WORK FILE
```

Entering // in this manner has the same effect as that described for CONTROL Y in example 1.

4. Enter // after the last character has been inserted and after RETURN is pressed, as follows:

```
/INSERT 1
    1       LINE 1 OF WORK FILE
            ↑// IS ENTERED AFTER RETURN  (CR)
    1.1     //
LINE 1 OF WORK FILE
/LIST 1/2
    1       // IS ENTERED AFTER RETURN
    1.1     LINE 1 OF WORK FILE
    2       LINE 2 OF WORK FILE
```

Entering // in this manner has the same effect as that described for CONTROL Y in example 2.

In batch mode, only // may be used either as the last two characters in a line of characters to be inserted or as the only two characters in a record. The results are the same as shown in the preceding examples 3 and 4.

3-54. **Limitations.** Text can only be inserted in a line beginning at a column within the SET LEFT, SET RIGHT margins established for that line. An existing line number must be specified in the *position* parameter for the INSERT command (unlike the ADD command, which can specify ADD 5.1 to insert the line 5.1 between the existing lines 5 and 6).

3-55. Examples. The following examples show several variations of the INSERT command. The location of the pointer is illustrated after the conclusion of each INSERT command.

- INSERT *linenumber*

The following example illustrates how to insert a line into the WORK file. To insert a new line before line 7, the command INSERT 7 is used. EDIT/3000 prompts with the number 7 and the new line is entered and RETURN is pressed. When 7.1 is displayed by EDIT/3000, // is entered. The pointer is located at the first position of line 7.1, which is the old line 7.

The LIST 6/8 command shows the positions of the inserted line and the previous line 7.

```
/LIST ALL
    1      1-2.  WHAT IS EDIT/3000?
    2
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    7
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
/INSERT 7
    7
            ↑(EDIT/3000 COMMANDS ARE USED.)
    7.1     //

/FIND*
    7.1
          ↑(0   )
/LIST 6/8
    6      MANIPULATE ASCII FILES.
    7      (EDIT/3000 COMMANDS ARE USED.)
    7.1
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
```

The following example illustrates the error message which is printed if the SET DELTA=*increment* option in effect does not allow new line numbers when new lines are being inserted. TEST is listed and INSERT 1.998 is used to insert two new lines. A carriage return is pressed after each line. Since the SET DELTA= *increment* option does not allow any more new line numbers, a warning message is printed out, as well as the saved line (the line originally on line 1.998) which is now lost from the WORK file.

```
/T TEST
/LIST ALL
    1      THIS IS LINE ONE
    1.998  THIS IS A LINE INBETWEEN
    2      THIS IS LINE TWO
/INSERT 1.998
    1.998 THIS IS A LINE INBETWEEN
          "ABCDEFGHIJKLMNOP
    1.999 123456789
**** WARNING - LINE BROUGHT FORWARD LOST:
THIS IS A LINE INBETWEEN
*15*COMMAND WILL NOT REPLACE OR INTERLEAVE LINES
/LIST ALL
    1      THIS IS LINE ONE
    1.998  ABCDEFGHIJKLMNOP
    1.999  123456789
    2      THIS IS LINE TWO
/END
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? YES
```

INSERT

- INSERT *colnum*

To insert text before a position in a line, the position number is specified in parentheses as shown in the next example.

The text (HELD,) is entered and // is entered *before* RETURN.

```
/INSERT 10(20)
   10     REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
                              ' HELD, //
 SEARCHED FOR, AND OTHERWISE
/FIND *
   10     REPLACED, MODIFIED, HELD,  SEARCHED FOR, AND OTHERWISE
                               (27 )'
/LIST 10/11
   10     REPLACED, MODIFIED, HELD,  SEARCHED FOR, AND OTHERWISE
   11     MANIPULATED BY USING EDIT/3000 COMMANDS.
```

- SPLITTING A LINE INTO TWO LINES

If you wish to make two lines out of an existing line, use the INSERT command as shown in the following example.

The FINDQ FIRST;FIND "OPERATING" commands locate the desired position where the line is to be split.

Entering the INSERT 4(28) command and then // *after* RETURN creates a new line (4.1) and the previous line 4 is now two lines.

```
/FINDQ FIRST;FIND "OPERATING"
    4     MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
                                    (28 )'
/INSERT 4(28)
    4     MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
                                     '
    4.1   //
OPERATING SYSTEM
/LIST 4/5
    4     MULTIPROGRAMMING EXECUTIVE
    4.1   OPERATING SYSTEM
    5     (MPE/3000) THAT IS USED TO CREATE AND
```

Another way to accomplish the same thing is to use the FIND "OPERATING" and INSERT * commands, then enter // after RETURN. For example

```
/FINDQ FIRST;FIND "OPERATING"
    4     MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
                                    (28 )'
/INSERT *
    4     MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
                                     '
    4.1   //
OPERATING SYSTEM
/LIST 4/5
    4     MULTIPROGRAMMING EXECUTIVE
    4.1   OPERATING SYSTEM
    5     (MPE/3000) THAT IS USED TO CREATE AND
```

- INSERT *linenumber*,HOLD

When the INSERT *linenumber*, HOLD form of the INSERT command is used, text is entered from the input file and, when the end of this text is signalled, the contents of the HOLD file are inserted.

Note from the example below that EDIT/3000 *compresses* the contents of the HOLD file by adding words from the next record to the current record. (The word "LINES" from the second record is added to the end of the first record, and "REPLACED" from the third record is added to the end of the second record.) Whenever an insertion from the HOLD file is made, it is similar to a justification process. The EDITOR tries to fill out lines with words up to the RIGHT margin. If the line is longer than the value of the RIGHT parameter, the EDITOR breaks it at a blank and inserts the remainder on the next line. If the new line becomes full, the EDITOR breaks it to continue on the next line and so forth. If the HOLD file contains a blank line, it causes the current line to be completed but no blank line is inserted into the work file. Insertion would then continue to the next line. If the HOLD file has two blank lines, only one blank line gets inserted. If the HOLD file has three blank lines, 2 blank lines are inserted and so on.

The original line 5 starts at the end of the last HOLD file record. (The string (MPE/3000) from the original line 5 is appended to the end of the last HOLD file record entered into the WORK file.)

```
/LIST ALL
     1      1-2.  WHAT IS EDIT/3000?
     2
     3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
     4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
     5      (MPE/3000) THAT IS USED TO CREATE AND
     6      MANIPULATE ASCII FILES.
     7
     8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
     9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
    10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
    11      MANIPULATED BY USING EDIT/3000 COMMANDS.
/HOLD# 8/10
HOLD FILE LENGTH IS 3 RECORDS
/INSERT# 5,HOLD#
(MPE/3000) THAT IS USED TO CREATE AND
*THE HOLD FILE CONTENTS WILL BE INSERTED
AFTER TEXT FROM THE TERMINAL IS ENDED
BY ENTERING DOUBLE SLASH OR CONTROL Y.
//
CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE LINES
OF CHARACTERS CAN BE INSERTED, DELETED, REPLACED,
MODIFIED, SEARCHED FOR, AND OTHERWISE (MPE/3000)
THAT IS USED TO CREATE AND
/LIST 5/6
     5      THE HOLD FILE CONTENTS WILL BE INSERTED
     5.1    AFTER TEXT FROM THE TERMINAL IS ENDED
     5.2    BY ENTERING DOUBLE SLASH OR CONTROL Y.
     5.3    CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE LINES
     5.4    OF CHARACTERS CAN BE INSERTED, DELETED, REPLACED,
     5.5    MODIFIED, SEARCHED FOR, AND OTHERWISE (MPE/3000)
     5.6    THAT IS USED TO CREATE AND
     6      MANIPULATE ASCII FILES.
/E
```

INSERT

- INSERT *linenumber*, HOLD, NOW

When this form of the INSERT command is used, the HOLD file contents are inserted before the line specified in the *linenumber* parameter and no text is inserted from the input file first (as it was in the previous example).

Note that the HOLD file contents, however, are *compressed* the same as in the previous example.

```
/LIST ALL
     1      1-2.  WHAT IS EDIT/3000?
     2
     3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
     4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
     5      (MPE/3000) THAT IS USED TO CREATE AND
     6      MANIPULATE ASCII FILES.
     7
     8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
     9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
    10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
    11      MANIPULATED BY USING EDIT/3000 COMMANDS.
/HOLD 8/10
HOLD FILE LENGTH IS 3 RECORDS
/INSERT 5,HOLD,NOW
CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE LINES
OF CHARACTERS CAN BE INSERTED, DELETED, REPLACED,
MODIFIED, SEARCHED FOR, AND OTHERWISE (MPE/3000)
THAT IS USED TO CREATE AND
/LIST 5/6
     5      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE LINES
     5.1    OF CHARACTERS CAN BE INSERTED, DELETED, REPLACED,
     5.2    MODIFIED, SEARCHED FOR, AND OTHERWISE (MPE/3000)
     5.3    THAT IS USED TO CREATE AND
     6      MANIPULATE ASCII FILES.
```

3-56. JOIN COMMAND

3-57. Purpose. The JOIN command appends or merges the contents of another MPE/3000 file into the WORK file.

3-58. Form. The form of the JOIN command is

```
J[OIN] [Q] {filename} [ (linenumber[/linenumber]) ] [ {TO} {linenumber} [ {BY} increment] ] [[,] UNN[UMBERED] ]
                      [ (#recnum/#recnum)         ]   {, }                {, }
```

The second occurrence of the *linenumber* is optional. If only one *linenumber* is specified, the entire file, starting with that linenumber, is joined. If no *linenumber* or *#recnum* is mentioned, the range of the JOIN command would include all of the records of the file. If the *#recnum* form is used and onlv one record is to be joined, the same *#recnum* must appear on both sides of the slash mark.

For example,

```
JOINQ ADDFILE (2/234) TO 34.001 BY .001
```

3-59. Description. The JOIN command is used to copy into the WORK file all or a portion of an MPE/3000 file to which you have access.

If the SET FORMAT = DEFAULT option is in effect, EDIT/3000 assumes that the first or last eight bytes (depending on whether FRONT or REAR is currently set) of each record in the file to be copied contain the sequence number. If the SET FORMAT = COBOL option is in effect, EDIT/3000 disallows the SET REAR option and assumes that the first six bytes of each record in the file contain the sequence numbers. In either case, these bytes are not copied into the WORK file. If these bytes contain text that should be copied in the WORK file (as is usually the case with an unnumbered file), you will want to use the UNN option with the JOIN command to copy the entire record of each line. Alternatively, you may use a SET command with the appropriate SET options for the bytes which are to be copied. See paragraph 3-92 for a discussion of the SET command.

EDIT/3000 assigns new line numbers to all lines copied by a JOIN command. If the TO *linenumber* (or *,linenumber*) parameter is not specified, the new line numbers start with the value currently set for FROM (see paragraph 3-92 for a discussion of the SET command) or with the first available line in the WORK file (the last linenumber in the WORK file incremented by DELTA). If the TO parameter is used, the linenumbers start with the number specified. Succeeding lines are incremented by the SET DELTA = *increment* option in effect or by the BY *increment* (or *,increment*) parameter specified in the JOIN command.

In an interactive session, each line copied is displayed on the terminal (unless the Q parameter is included with the command). You may terminate the JOIN command by entering CONTROL Y. Any lines copied before CONTROL Y is entered remain in the WORK file.

The contents of the joined file are unaffected.

3-60. Limitations. Any file to be copied into the WORK file with a JOIN command must

1. Consist of ASCII-coded records.

2. Have a *filecode* of from 0 to 1023, 1040 XLSAV, 1060 RJEPN, or 1052 EDTCT. (See the *MPE Commands Reference Manual* for a discussion of *filecode*.)

3. Be available to you for read access. (See the *MPE Commands Reference Manual*.)

4. Exist as a disc-resident temporary or permanent file.

   Note: If the file to be copied is on a device other than disc (such as a magnetic tape device), use the MPE/3000 File Copier to copy the file to disc.

   Records copied from the JOIN file are entered into the WORK file within the limits established by column 1 and the value of the LENGTH parameter. If the length of records in the JOIN file is greater than the WORK file LENGTH limit in effect, the JOIN file records will be truncated to the WORK file length when they are joined.

   If one very large or many smaller files are to be joined, use the SET SIZE command (see paragraph 3-92) to establish a large enough WORK file before using the TEXT command to text in the main file.

If the JOIN command is attempted on a KSAM file (see the *KSAM/3000 Reference Manual*), the following warning message is produced:

* * * * WARNING FILE IS KSAM

KSAM files are joined according to the primary key order. If the (*linenumber/linenumber*) range is specified and the line numbers are not the primary key, the results will be unpredictable.

3-61. Examples. Several examples of the JOIN command are presented in the following examples. The location of the pointer is shown at the conclusion of each JOIN command.

- JOIN *filename* (*linenumber/linenumber*)

The following example joins line numbers 2 through 4 from the file ADDFILE to the WORK file starting at line 2.1.

```
/TEXT ADDFILE
/LIST ALL
    1       LINE 1 OF JOIN FILE
    2       LINE 2 OF JOIN FILE
    3       LINE 3 OF JOIN FILE
    4       LINE 4 OF JOIN FILE
    5       LINE 5 OF JOIN FILE
/TEXT FILE
/LIST ALL
    1       LINE 1 OF WORK FILE
    2       LINE 2 OF WORK FILE
    3       LINE 3 OF WORK FILE
    4       LINE 4 OF WORK FILE
    5       LINE 5 OF WORK FILE
/JOIN ADDFILE (2/4) TO 2.1 BY .1
    2.1     LINE 2 OF JOIN FILE
    2.2     LINE 3 OF JOIN FILE
    2.3     LINE 4 OF JOIN FILE
/FIND*
    2.3     LINE 4 OF JOIN FILE                                    (61 )*

/LIST 2/3
    2       LINE 2 OF WORK FILE
    2.1     LINE 2 OF JOIN FILE
    2.2     LINE 3 OF JOIN FILE
    2.3     LINE 4 OF JOIN FILE
    3       LINE 3 OF WORK FILE
```

- JOIN *filename* (*#recnum*/*#recnum*)

To join logical record numbers from a file to the WORK file, the (*#recnum*/*#recnum*) *rangelist* form is used as in the following example.

Note that line numbers 1 through 3 (which are logical record numbers 0 through 2) are added to the WORK file.

```
/JOIN ADDFILE (#0/#2) TO 3.1
    3.1    LINE 1 OF JOIN FILE
    3.2    LINE 2 OF JOIN FILE
    3.3    LINE 3 OF JOIN FILE
/FIND *
    3.3    LINE 3 OF JOIN FILE
           *(1   )
/LIST 3/4
    3      LINE 3 OF WORK FILE
    3.1    LINE 1 OF JOIN FILE
    3.2    LINE 2 OF JOIN FILE
    3.3    LINE 3 OF JOIN FILE
    4      LINE 4 OF WORK FILE
```

- JOINQ

If the JOIN *filename* form with no *rangelist* is used, as in the following example, the entire contents of the JOIN file are added to the end of the WORK file.

In addition, since the TO *linenumber* was not included, the JOIN file contents are copied to the end of the WORK file.

```
/LIST ALL
     1     LINE 1 OF WORK FILE
     2     LINE 2 OF WORK FILE
     3     LINE 3 OF WORK FILE
     4     LINE 4 OF WORK FILE
     5     LINE 5 OF WORK FILE
/JOINQ ADDFILE
NUMBER OF LINES JOINED = 5
/FIND*
    10     LINE 5 OF JOIN FILE
           *(1   )
/LIST ALL
     1     LINE 1 OF WORK FILE
     2     LINE 2 OF WORK FILE
     3     LINE 3 OF WORK FILE
     4     LINE 4 OF WORK FILE
     5     LINE 5 OF WORK FILE
     6     LINE 1 OF JOIN FILE
     7     LINE 2 OF JOIN FILE
     8     LINE 3 OF JOIN FILE
     9     LINE 4 OF JOIN FILE
    10     LINE 5 OF JOIN FILE
```

JOIN

3-62. KEEP COMMAND

3-63. Purpose. The KEEP command saves the contents of the WORK file.

3-64. Form. The form of the KEEP command is

K[EEP][*filename* [(range)] [,UNN[UMBERED]]]

or

K[EEP] Q *filename*

For example,

KEEP KFILE (45/678),UNNUMBERED

3-65. Description. If the *filename* parameter of the KEEP command refers to a new file, a file is opened under this name by MPE/3000, and the WORK file contents are stored on disc in this file. When *range* is specified in a KEEP command, only those lines in the WORK file specified by *range* are saved. (If a column number is specified in *range*, the entire line containing that column is saved.)

The file specified by *filename* must be in the log-on account.

In an interactive session, if an existing file is named in the KEEP command, EDIT/3000 asks if you wish to purge the old contents of the file. A positive response (YES or Y) causes the contents of the old file to be purged and the contents of the WORK file to be written into the kept file. Any other response terminates the KEEP operation.

If the KEEP command is used without any parameters, the *filename* is implied from the previous KEEP or TEXT command (whichever was performed last). If the UNNUMBERED option was specified in the previous KEEP or TEXT command, then this is also implied for the current KEEP command. EDIT/3000 produces an echo of the *filename* (and possibly the UNNUMBERED option) and copies the entire WORK file contents to that file.

The following example shows the KEEP command used without any parameters. In this case, both the *filename* and the UNNUMBERED option are implied:

```
:EDITOR
HP32201A.7.01 EDIT/3000   WED, MAY 3, 1978,  1:33 PM
(C) HEWLETT-PACKARD CO. 1976
/TEXT KEPFIL,UNN
/a
    4       MODIFICATIONS
    5       MORE TO COME
    6       ...
/K
 KEPFIL,UNN
KEPFIL ALREADY EXISTS - RESPOND YES TO PURGE OLD AND THEN KEEP
PURGE OLD?Y
/END

END OF SUBSYSTEM
:
```

Another example follows. This time only the *filename* is implied:

```
:EDITOR
HP32201A.7.01 EDIT/3000  WED, MAY 3, 1978,  1:43 PM
(C) HEWLETT-PACKARD CO. 1976
/TEXT KEPFIL
/A
    7       several changes in addition
    8       MORE TO COME
    9       ...
/K
 KEPFIL
KEPFIL ALREADY EXISTS - RESPOND YES TO PURGE OLD AND THEN KEEP
PURGE OLD?Y
/END
END OF SUBSYSTEM
```

If the file in question belongs to a group and/or account other than that of the log-on account, the *filename*, group, and account are echoed.

In batch mode the contents of the old file are purged automatically when an existing old file is named in a KEEP command.

Records in the WORK file are saved as fixed-length records unless the SET VARIABLE option of the SET command is used, in which case the records are saved as variable-length records. (See paragraph 3-92 for a discussion of the SET command.)

An MPE/3000 error message is displayed if access to a file is not possible. See the *MPE Commands Reference Manual* for a discussion of file access.

Unless the UNNUMBERED parameter is specified in the KEEP command, the line numbers in the WORK file are placed in the kept file in ASCII format. The SET FRONT, SET REAR, and SET FORMAT commands determine where the line number information will be kept (front or rear of each record). See paragraph 3-92 for discussions of these commands. Line numbers in the WORK file are not copied to the kept file when the UNNUMBERED parameter of the KEEP command is declared.

Note that the default KEEP file is limited to disc files on the local computer. If a device or remote file is being used, an MPE file equation must have been issued and an asterisk (*) must be used to back-reference the file. (See the example of keeping to a card punch device in paragraph 3-67.)

The KEEPQ *filename* command saves the WORK file in its "edit" format, instead of in the format in which files normally are stored by the MPE/3000 file system. This is accomplished by appending EDIT/3000 variables to the file and assigning a *filecode* of 1050 or 1051 to the file. A *filecode* of 1050 signifies an EDIT/3000 KEEPQ file (non-COBOL) and a *filecode* of 1051 signifies an EDIT/3000 KEEPQ file (COBOL). All such files are intended to be read only by EDIT/3000 although KEEPQ files can also be read by SPL/3000. To add the *lockword* to the new file would require EDIT/3000 to save it internally and thus cause a possible security violation.

3-66. Limitations. The *range* and UNNUMBERED parameters of the KEEP command may be used *only* with the KEEP *filename* (or K *filename*) variation of the KEEP command. These parameters may *not* be used with the KEEPQ *filename* (or KQ *filename*) variation of the command.

Note that the KEEP command does not put a lockword on the file unless you explicitly specify one. For example, suppose you have a file named MYFILE with the *lockword* LOCKED. Issuing the command KEEP MYFILE/LOCKED will purge the old file and put the *lockword* LOCKED on the new file. However, if you issue the command KEEP MYFILE or if you issue the command K[EEP] with no parameters, you will be prompted for the *lockword* and EDIT/3000 will purge the old file (if you have supplied the correct *lockword*) and create a new file named MYFILE without a *lockword*.

A TEXT command (see paragraph 3-98) to copy the contents of an EDIT/3000 KEEPQ file into the WORK file renames that file with an EDIT/3000 K*nnn* name. To prevent destruction of such a file, enter the commands TEXT *filename* and KEEP *filename* in succession.

If SET FIXED is in effect, the KEEP file will have

record size = SET LENGTH if the file is unnumbered

record size = SET LENGTH + 6 bytes if the file is numbered in the COBOL format

record size = SET LENGTH + 8 bytes if the file is numbered in the default format

The blocking factor is calculated to allow the optimal trade-off between disc space used and time required for input-output.

If SET VARIABLE is in effect, the KEEP file will have

record size = 1276 bytes

blocking factor = 1

Using this large record size maximizes the disc space used by allowing many variable-length records to be entered in a single block.

Any of these attributes can be overridden by back-referencing a file equation for the KEEP file in the KEEP command.

> Note: If a file equation is used to decrease the record size for a variable-length file, it may be necessary also to include the DISC parameter to increase the file limit. The default limit allows only as many blocks as are necessary to store all records when the block size is 1276 bytes.

Trying to KEEP a numbered file which is greater than 256 bytes wide (including 8 bytes for sequence numbers) will result in error #75, RECORDS TOO WIDE — KEEP UNNUMBERED.

3-67. Examples. Several variations of the KEEP command are shown in the following examples.

- KEEP *filename*

The first example uses the KEEP EDIT2 command to keep the file EDIT2. Because EDIT2 is an existing file, EDIT/3000 asks if it is OK to purge the old file.

```
HP32201A.4.01 EDIT/3000  WED, MAY 14, 1975,  3:43 PM
/T EDIT2;L ALL
    1      1-2.  WHAT IS EDIT/3000?
    2
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    7
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
/KEEP EDIT2
EDIT2 ALREADY EXISTS - RESPOND YES TO PURGE OLD AND THEN KEEP
PURGE OLD?Y
```

- KEEP *filename (range)*

The KEEP *range* form, as in the following example, keeps only those lines specified in *range* (lines 8 through 11 in the example).

Note: SET SHORT was used in the next three examples.

```
/KEEP TEST (8/11)
/TEXT TEST
CLEAR? Y
/LIST ALL
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
```

- KEEP *filename*, UNNUMBERED

When the KEEP *filename*, UNNUMBERED form of the KEEP command is used, as in the following example, the file can be copied back into the work file either with a TEXT *filename*, UNNUMBERED or a TEXT *filename* command.

Note that EDIT/3000 assigns line numbers to the unnumbered file TEST (as shown by the LIST ALL command).

```
/KEEP TEST,UNNUMBERED
PURGE OLD?Y
/TEXT TEST,UNNUMBERED
/LIST ALL
    1      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    2      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
    3      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
    4      MANIPULATED BY USING EDIT/3000 COMMANDS.
```

- SEQUENCING INFORMATION

When the KEEP *filename* form of the KEEP command is used, EDIT/3000 appends eight bytes to the end of each record for sequencing information. When the TEXT file is copied back into the WORK file with a TEXT *filename* command, these eight bytes are deleted by EDIT/3000; however, if you do not wish the information contained in the eight bytes to be saved in the TEXT file, use the KEEP *filename,* UNNUMBERED form of the KEEP command.

In the next example, the SET RIGHT and SET LENGTH options are set to 52, then lines 1 through 10 of the WORK file are saved under the *filename* TEST. Once EDIT/3000 operation is terminated, the MPE/3000 command:LISTF TEST,1 shows that the length has been increased by eight bytes to 60 bytes.

```
:EDITOR

HP32201A.4.01 EDIT/3000  TUE, JUN  3, 1975,  3:13 PM
/S SHORT;T EDIT3;L 1/10
    1      1-2.  WHAT IS EDIT/3000?
    2
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    7
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
/S RIGHT=52,LENGTH=52
/K TEST(1/10)
PURGE OLD?Y
/E

 END OF SUBSYSTEM
:LISTF TEST,1
ACCOUNT=  GOODWIN      GROUP=  PUB

FILENAME  CODE  ------------LOGICAL RECORD--------
                    SIZE  TYP         EOF       LIMIT

TEST              60B  FA           10          10
```

length

- KEEPQ *filename*

The next example demonstrates the use of the KEEPQ command. In the first part of the example, the TEXT file EDIT2 is copied into the WORK file and listed. The VERIFY FILES command shows a WORK file of K18111336 and the MPE/3000 LISTF command displays the file parameters.

In the second part of the example, the same file is copied into the WORK file and listed. Then the KEEPQ command is entered. Note that now when the VERIFY FILES is entered, there is no entry for the WORK file and an attempt to LIST ALL results in the error message

UNDEFINED TEXT

The MPE/3000 LISTF command shows that EDIT2 now has a *filecode* of 1050 (an EDIT/3000 KEEPQ file). Such a file can be read only by EDIT/3000 and can be used for no other purpose (no other EDIT/3000 commands will execute for such a file. For example, the LIST ALL command would not execute.)

The last part of the example shows how to use such a file, by entering the TEXT *filename* and KEEP *filename* commands in succession. Now the LIST ALL command executes.

```
:EDITOR

HP32201A.4.01 EDIT/3000  MON, JUN 30, 1975,  1:36 PM
/SET SHORT;TEXT EDIT2;LIST ALL
    1     LINE 1 OF WORK FILE
    2     LINE 2 OF WORK FILE
    3     LINE 3 OF WORK FILE
    4     LINE 4 OF WORK FILE
    5     LINE 5 OF WORK FILE
/VERIFY FILES
FILES:
    WORK: K1911336
    KEEP:
    TEXT: EDIT2.PUB.GOODWIN          MON, JUN 30, 1975,  1:36 PM
    JOIN:
/END


  END OF SUBSYSTEM
:LISTF EDIT2,2
ACCOUNT=  GOODWIN     GROUP=  PUB

FILENAME  CODE  ------------LOGICAL RECORD-----------  ----SPACE---- ACC
                  SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX

EDIT2              80B  FA           5          5  16       10  1  1 ???

:EDITOR

HP32201A.4.01 EDIT/3000  MON, JUN 30, 1975,  1:38 PM
/SET SHORT;TEXT EDIT2;LIST ALL
    1     LINE 1 OF WORK FILE
    2     LINE 2 OF WORK FILE
    3     LINE 3 OF WORK FILE
    4     LINE 4 OF WORK FILE
    5     LINE 5 OF WORK FILE
/KEEP EDIT2
PURGE OLD?Y
/VERIFY FILES
FILES:
    WORK:
    KEEP: EDIT2                      MON, JUN 30, 1975,  1:39 PM
    TEXT: EDIT2.PUB.GOODWIN          MON, JUN 30, 1975,  1:39 PM
    JOIN:
/LIST ALL
*40*UNDEFINED TEXT
/END

  END OF SUBSYSTEM
:LISTF EDIT2,2
ACCOUNT=  GOODWIN     GROUP=  PUB

FILENAME  CODE  ------------LOGICAL RECORD-----------  ----SPACE---- ACC
                  SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX

EDIT2     1958    112B  FA          22       2000   9       79  1 16 ???

:EDITOR

HP32201A.4.01 EDIT/3000  MON, JUN 30, 1975,  1:41 PM
/SET SHORT;TEXT EDIT2;KEEP EDIT2
/LIST ALL
    1     LINE 1 OF WORK FILE
    2     LINE 2 OF WORK FILE
    3     LINE 3 OF WORK FILE
    4     LINE 4 OF WORK FILE
    5     LINE 5 OF WORK FILE
/VERIFY FILES
FILES:
    WORK: K1911341
    KEEP: EDIT2.PUB.GOODWIN          MON, JUN 30, 1975,  1:41 PM
    TEXT: EDIT2.PUB.GOODWIN          MON, JUN 30, 1975,  1:41 PM
    JOIN:
/END


  END OF SUBSYSTEM
:
```

EDIT/3000 monitors whether you make changes to the TEXT file or not. Thus, when using the TEXT or END command, you are not burdened with the "OK TO CLEAR?" message unless the file has been altered. The "VERIFY ALL" and "VERIFY FILES" commands indicate if the WORK file has been altered with the following message:

WORK FILE HAS BEEN ALTERED

For example,

```
:EDITOR
HP32201A.7.01 EDIT/3000  WED, MAY 3, 1978,  1:42 PM

/TEXT KEEPFIL,UNN
/V FILES
FILES:
    WORK: K1231342
    KEEP:
    TEXT: KEEPFIL.PUB.SHELL            WED, MAY 3, 1978,  1:42 PM
    JOIN:
/ADD
    6       ADDITIONS
    7       MORE ADDITIONS
    *       ...
/V FILES




FILES:
    WORK: K1231342
          WORK FILE HAS BEEN ALTERED
    KEEP:
    TEXT: KEEPFIL.PUB.SHELL            WED, MAY 3, 1978  1:42 PM
    JOIN:
/KEEP KEEPFIL,UNN
KEEPFIL ALREADY EXISTS - RESPOND YES TO PURGE OLD AND THEN KEEP
PURGE OLD?Y
/V FILES
FILES:
    WORK: K1231342
    KEEP: KEEPFIL.PUB.SHELL            WED, MAY 3, 1 9978  1:48 PM
    TEXT: KEEPFIL.PUB.SHELL            WED, MAY 3, 1978  1:42 PM
    JOIN:
/END

END OF SUBSYSTEM
:
```

- KEEPING THE WORK FILE CONTENTS ON A CARD PUNCH DEVICE

In the next example, a FORTRAN/3000 source program, which has been copied into the WORK file with a TEXT command, is sent to a card punch device. To prevent the line numbers from being punched on the cards, the KEEP *filename*, UNNUMBERED form of the KEEP command is used.

```
:FILE OUT;DEV=PUNCH
:EDITOR

HP 32201A.4.01 EDIT/3000  MON, JUN 30, 1975,  2:42 PM
/S SHORT;TEXT FTRAN33
/LIST ALL
    1            PROGRAM INTRINSICS
    2      C
    3      C FOPEN, FREAD, AND FCLOSE EXAMPLE
    4      C
    5       100  FORMAT(72,S)
    6            CHARACTER*72 BUFFER,FILENAME*16
    7            LOGICAL LBUFFER(36)
    8            INTEGER FILENUMBER,FOPEN,FREAD
    9            EQUIVALENCE (LBUFFER,BUFFER)
   10            FILENAME="MAILLIST"
   11      C
   12      C THE NEXT STATEMENT CALLS THE FOPEN INTRINSIC
   13      C
   14            FILENUMBER=FOPEN(FILENAME,\1\,\%105\,0,0,0,0,0,\0,0\,
   15           #0,0,0,\%1600\)
   16            IF(.CC.)40,10,40
   17       10   DISPLAY "FILENUMBER = ",FILENUMBER
   18      C
   19      C THE NEXT STATEMENT CALLS THE FREAD INTRINSIC
   20      C
   21       20   N=FREAD(\FILENUMBER\,LBUFFER,\36\)
   22            IF(.CC.)50,30,60
   23       30   WRITE(6,100)BUFFER[1:N]
   24            GOTO 20
   25       40   DISPLAY "FOPEN FAILURE"
   26            STOP
   27       50   DISPLAY "FREAD ERROR"
   28            STOP
   29      C
   30      C THE NEXT STATEMENT CALLS THE FCLOSE INTRINSIC
   31      C
   32       60   CALL FCLOSE(\FILENUMBER\,\1\,\0\)
   33            IF(.CC.)70,80,70
   34       70   DISPLAY "FCLOSE FAILURE"
   35            STOP
   36       80   DISPLAY "FILE CLOSED SUCCESSFULLY"
   37            STOP
   38            END
/KEEP *OUT,UNN
```

- USER LABELS

If the file is kept with a name that already exists, the old file is purged, but the user labels are not lost. They are passed on to the new file.

3-68. LIST COMMAND

3-69. Purpose. The LIST command displays all or a portion of the WORK file.

3-70. Form. The form of the LIST command is

L[IST][Q] [*range*] [,UNN[UMBERED]][,OFFLINE][,TRANSLATE][,NOTEXT]

3-71. Description. The LIST command lists the lines of the WORK file contained in *range*. If *range* is not specified or if *range* is an asterisk (*), the line containing the pointer is displayed. If *range* is ALL, the entire contents of the WORK file are displayed.

If the Q or UNNUMBERED parameters are specified, only the text of each line in *range* is listed, without the line numbers.

Unless the OFFLINE parameter is specified, the listing is displayed on the standard output device (OUTPUT file). If OFFLINE is specified, the WORK file is listed on the device class LP, unless an alternative output device is specified with the MPE :FILE command. (See paragraph 3-73, EXAMPLES.) The number of lines printed on each page of an offline listing will be 60, unless otherwise specified by the SET LINES parameter. (See paragraph 3-95.)

If the system does not have a device class of LP, the message "*OFFLINE LIST DEVICE NOT AVAILABLE" appears. You must, in this case, use a file equation with the proper device class.

> Note: This feature must be included at configuration and may vary from installation to installation.

The TRANSLATE parameter causes all lowercase characters to be shifted to uppercase for the listing only. The contents of the WORK file are not altered, only the listing is affected.

The NOTEXT parameter causes a listing of line numbers only, without the text of the lines. (This is useful for determining what line numbers exist within a range.)

Parameters for the LIST command may be specified in any order as long as they are separated by commas. Some combinations, however, such as UNNUMBERED (or Q) and NOTEXT, obviously are incompatible.

3-72. Limitations. The LIST command displays only those characters within the SET LEFT, SET RIGHT margins in effect. See paragraph 3-92 for a discussion of the SET command.

3-73. **Examples.** Several variations of the LIST command are shown in the following examples. The location of the pointer is illustrated after the conclusion of each LIST command.

- LIST *range*

This example specifies a *range* of 3/6 and only lines 3 through 6 are listed. After the command is executed, the pointer is located at position 1 of line 7 (the line following the last line listed).

```
    1     1-2.  WHAT IS EDIT/3000?
    2
    3     EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4     MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5     (MPE/3000) THAT IS USED TO CREATE AND
    6     MANIPULATE ASCII FILES.
    7
    8     CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9     LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10     REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11     MANIPULATED BY USING EDIT/3000 COMMANDS.
/LIST 3/6
    3     EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4     MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5     (MPE/3000) THAT IS USED TO CREATE AND
    6     MANIPULATE ASCII FILES.
/FIND *
    7
          ↑(1  )
```

- LISTQ *range* and LIST *range* UNN[UMBERED]

When the LISTQ and LIST UNNUMBERED forms of the command are used as in the following example, the text, but not the line numbers, of each line in *range* is listed.

```
/LISTQ 3/6
EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
(MPE/3000) THAT IS USED TO CREATE AND
MANIPULATE ASCII FILES.
/FIND *
    7
          ↑(1  )
/LIST 3/6,UNNUMBERED
EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
(MPE/3000) THAT IS USED TO CREATE AND
MANIPULATE ASCII FILES.
/FIND*
    7
          ↑(1  )
```

- LIST *range*, NOTEXT

This form of the LIST command causes EDIT/3000 to display line numbers only; the text of the lines in *range* is not listed.

```
/LIST 3/6,NOTEXT
    3
    4
    5
    6
/FIND *
    7
          ↑(1  )
```

- LIST *range*,OFFLINE

The LIST *range*, OFFLINE command sends the WORK file records to *listfile* instead of displaying the records on the standard list device.

The *listfile* can be declared in the MPE/3000 :EDITOR and :FILE commands and equated to an alternative output device. In the first example following, the file OUT is declared and equated to a line printer (which is not the standard list device in a session). When EDIT/3000 is accessed, the :EDITOR *OUT command is used, back-referencing the file OUT. The LIST ALL,OFFLINE command causes the contents of the WORK file to be listed on the line printer.

```
:FILE OUT;DEV=FASTLP
:EDITOR *OUT

HP32201A.4.01 EDIT/3000  TUE, MAY 13, 1975, 10:12 AM
/TEXT EDIT2
/LIST ALL,OFFLINE
*** OFF LINE LISTING BEGUN. ***
```

Note: The listing below was printed on the line printer.

```
1       1-2.  WHAT IS EDIT/3000?
2
3       EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
4       MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
5       (MPE/3000) THAT IS USED TO CREATE AND
6       MANIPULATE ASCII FILES.
7
8       CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
9       LINES OF CHARACTERS CAN BE INSERTED, DELETED,
10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
11      MANIPULATED BY USING EDIT/3000 COMMANDS.
```

If EDIT/3000 has been accessed without specifying an offline list file, an offline listing can be obtained on devices other than LP as follows:

a. Enter the following :FILE command from within EDIT/3000:

   :FILE EDTLIST;DEV= FASTLP

   where FASTLP is the desired device class name (FASTLP is a line printer on the system used in the example) and EDTLIST is the formal file designator for EDIT/3000 offline listings.

b. Enter the EDIT/3000 command.

   LIST ALL,OFFLINE

LIST

EDTLIST is EDITOR's formal designator for the offline output list file.

EDTLIST is used for the LIST, OFFLINE and XPLAIN, OFFLINE commands.

EDITOUT is EDITOR's formal designator for the output list file ($STDLIST).

EDITIN is EDITOR's formal designator for the input file ($STDINX).

3-74. MODIFY COMMAND

3-75. Purpose. The MODIFY command modifies text records in the WORK file through the use of three MODIFY subcommands.

3-76. Form The form of the MODIFY command is

M[ODIFY][Q][*rangelist*]

3-77. Description. The three MODIFY subcommands are as follows:

D Deletes the character directly above it. A consecutive string of characters can be deleted by typing a D below each character or below the first and last characters in the string (with blank characters in between) to be deleted.

I Inserts characters in front of the character directly above the I.

R Replaces characters starting with the character directly above the R.

In an interactive session, the MODIFY command begins operation for each line in *rangelist* by displaying

MODIFY *linenumber*

then the contents of the line.

If *rangelist* is not included in a MODIFY command, or if * is specified in *rangelist,* EDIT/3000 modifies the line containing the current location of the pointer.

After one of the subcommands has been used to modify the line and RETURN has been pressed, EDIT/3000 displays the line again in its modified form unless the MODIFYQ form of the command has been used, in which case the modified line is not displayed. If the line is satisfactory, press RETURN and EDIT/3000 will print the next line to be modified (or terminate the MODIFY command if the last line in *rangelist* has been modified). If the line requires further modification, enter the appropriate subcommand.

In an interactive session, to cancel the effect of a MODIFY subcommand, press CONTROL Y before RETURN. To cancel a MODIFY command for all lines of a *range* not yet modified, press CONTROL Y after EDIT/3000 displays the line to be modified. EDIT/3000 then proceeds to the next *range* (if any) in *rangelist* or terminates the MODIFY command if this was the last (or only) *range* in *rangelist.* Lines already modified remain modified and all other lines remain unchanged.

3-78. Limitations. The MODIFY command operates only on text within the SET LEFT, SET RIGHT options in effect. (See paragraph 3-92 for a discussion of the SET command.)

Only one subcommand may be used to modify a line each time it is displayed, with this exception: an I (insert) subcommand can follow a D (delete) or an I can be used as the second D in a range of Ds.

For example,

```
/MODIFY 1
MODIFY     1
THIS IS LINE 1
        DDDDIITEM
THIS IS ITEM 1
        D   ILINE
THIS IS LINE 1
```

As you recall, if the ampersand (&) is used as the last character in a line, EDIT/3000 interprets this as meaning that the line is to be continued. Thus, when an ampersand is to be entered as the last character in a line to be modified with a REPLACE or INSERT subcommand, you must enter a blank character after the ampersand.

3-79. Examples. Several variations of the MODIFY command are shown in the following examples.

- DELETE SUBCOMMAND

The DELETE (D) subcommand causes any character above the D (or a string of Ds or two Ds with blank characters between) to be deleted. In the example, the blank character between "HP" and "3000" is deleted by moving the cursor under the blank character and typing D, then the entire string "HP 3000" is deleted by the D, a string of blanks, and another D.

```
/LIST ALL
    1     1-2.  WHAT IS EDIT/3000?
    2
    3     EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4     MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5     (MPE/3000) THAT IS USED TO CREATE AND
    6     MANIPULATE ASCII FILES.
    7
    8     CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9     LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10     REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11     MANIPULATED BY USING EDIT/3000 COMMANDS.
/MODIFY 3
MODIFY     3
EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
                                 D
EDIT/3000 IS A SUBSYSTEM OF THE HP3000

/FIND*
    4     MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
          *(1   )
/MODIFY 3
MODIFY     3
EDIT/3000 IS A SUBSYSTEM OF THE HP3000
                                   D    D
EDIT/3000 IS A SUBSYSTEM OF THE

/LIST 3/4
    3     EDIT/3000 IS A SUBSYSTEM OF THE
    4     MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
```

- REPLACE AND INSERT SUBCOMMANDS

The REPLACE (R) subcommand replaces characters starting with the character directly above the R with new characters entered from the INPUT file. In the example, the characters "HP3000" are replaced by "HEWLETT-PACKARD". The INSERT subcommand (I) is used to insert the characters "3000" after "HEWLETT-PACKARD". The I is positioned directly under the character where the inserted information is to begin.

```
/MODIFY 3
MODIFY     3
EDIT/3000 IS A SUBSYSTEM OF THE HP3000
                                 RHEWLETT-PACKARD
EDIT/3000 IS A SUBSYSTEM OF THE HEWLETT-PACKARD
                                                I3000
EDIT/3000 IS A SUBSYSTEM OF THE HEWLETT-PACKARD 3000
```

Note: The R subcommand can be used to extend line length by entering R, a series of blank characters, then RETURN.

- DELETE AND INSERT SUBCOMMANDS

The DELETE (D) and INSERT (I) subcommands can be used on the same line. The I can be used to insert characters after a D (or a range of Ds), or the I can be used as the last D in a range of Ds. Whatever is entered after the I will be inserted in the line after the last deleted character.

In the example, the characters "HEWLETT-PACKARD" are deleted by the range of Ds and the characters "HP" then are inserted with the I subcommand.

```
/MODIFY 3
MODIFY     3
EDIT/3000 IS A SUBSYSTEM OF THE HEWLETT-PACKARD 3000
                                  D              DIHP
EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
```

- USE OF AMPERSAND (&)

In the following example, an & is inserted as the last character and RETURN is pressed. EDIT/3000 interprets this as a continuation character and the command LIST 3 is added to the end of line 3.

The second part of the example deletes the characters "LIST 3" and inserts an & *and* a blank character. Now EDIT/3000 interprets the modification correctly.

```
/MODIFY 3
MODIFY     3
EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
                                        I&
LIST 3
EDIT/3000 IS A SUBSYSTEM OF THE HP 3000 LIST 3

/MODIFY 3
MODIFY     3
EDIT/3000 IS A SUBSYSTEM OF THE HP 3000 LIST 3
                                        D    DI&  (CR)
EDIT/3000 IS A SUBSYSTEM OF THE HP 3000 &

/LIST 3
    3     EDIT/3000 IS A SUBSYSTEM OF THE HP 3000 &
```

3-80. Q COMMAND

3-81. Purpose. The Q command displays a message on the standard list device (OUTPUT file).

3-82. Form. The form of the Q command is

Q [*string* / *"Z::"*]

3-83. Description. The Q command prints any message selected by you, or a blank line if no string is supplied. You can use the Q command, for example, to remind yourself to perform certain editing functions when a specific point is reached in an edit cycle. This command is most useful in WHILE blocks and USE files.

3-84. Limitation. The Q command is used to display a message only to the standard list device (a terminal in an interactive session or a line printer in batch mode).

3-85. Example.

```
/Q "THIS IS A MESSAGE"
THIS IS A MESSAGE
```

3-86. REPLACE COMMAND

3-87. **Purpose.** The REPLACE command replaces one or more lines in the WORK file.

3-88. **Form.** The form of the REPLACE command is

R[EPLACE][Q] [*rangelist*] [,HOLD[Q] [,NOW]]

3-89. **Description.** Entire lines are replaced by a REPLACE command. (Use the CHANGE or MODIFY commands if you want to change characters within a line.) If an entry in *rangelist* is a *colnum* or a *string* (signifying that only that column or string is to be replaced), the REPLACE command ignores this and replaces the entire line. If no *rangelist* is specified in the REPLACE command, or if an asterisk (*) is specified in *rangelist,* the REPLACE command replaces the line containing the current location of the pointer.

In an interactive session, EDIT/3000 displays the line number and the contents of the line to be replaced, then displays the line number again on the next line (unless the REPLACEQ form of the command has been used, in which case only the line number is displayed). Enter the replacement text and press RETURN. The new text replaces the old line. Pressing RETURN only (with no replacement text) deletes the contents of the line but not the line itself (the line number still exists; the contents are blank).

When all lines of a *range* have been replaced, EDIT/3000 prompts for the first line in the next *range* (if any) in *rangelist,* or terminates the command if this was the last (or only) *range* in *rangelist.*

Pressing CONTROL Y in an interactive session will terminate the replacement of lines in a *range,* or terminate the REPLACE command if this is the last (or only) *range* in *rangelist.*

In batch mode, the REPLACE command replaces lines with new text records entered following the REPLACE command.

Text from the HOLD file can be used to replace existing lines in the WORK file by adding the HOLD[Q] or HOLD[Q],NOW parameters to the REPLACE command. See paragraph 3-91, EXAMPLES.

3-90. Limitations. The REPLACE command operates only within the margins established by the SET LEFT, SET RIGHT (see paragraph 3-92) options in effect. Only that text within the left and right margins is displayed and only that part of the line is replaced. The replacement line is padded with blanks if it is shorter, while a warning message is displayed if the string is longer than the LEFT/RIGHT range.

When the HOLD parameter is used with the REPLACE command, text is replaced on a line-for-line basis. For example, if three lines are to be replaced, then only three lines will be accepted from the HOLD file. Or, if there are only three lines in the HOLD file, only that many lines will be replaced in the WORK file even if *range* specifies more than three lines.

3-91. Examples. The following examples illustrate the use of the REPLACE command and show the location of the pointer at the conclusion of each command.

- REPLACE *range*

With this form of the REPLACE command (in an interactive session), EDIT/3000 displays the line to be replaced, then prompts for the new text with the line number. The new text is entered and replaces the text originally contained in the line.

```
/LIST ALL
    1      1-2.  WHAT IS EDIT/3000?
    2
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    7
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
/REPLACE 5
    5      (MPE/3000) THAT IS USED TO CREATE AND
    5      THIS IS A NEW LINE 5.
/FIND*
    6      MANIPULATE ASCII FILES.
           *(1   )
/LIST 4/6
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      THIS IS A NEW LINE 5.
    6      MANIPULATE ASCII FILES.
```

- REPLACEQ *range*

With this form, EDIT/3000 does not display the line to be replaced, but does prompt with the line number.

In the example, the characters "(MPE/3000) THAT IS USED TO CREATE AND" are entered after the prompt (5) is displayed. These characters replace the old contents (THIS IS A NEW LINE) of line 5.

```
/LIST 5
    5      THIS IS A NEW LINE 5.
/REPLACEQ 5
    5      (MPE/3000) THAT IS USED TO CREATE AND
/FIND*
    6      MANIPULATE ASCII FILES.
           *(1   )
/LIST 4/6
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
```

- REPLACE *range*, HOLD,NOW

This form of the REPLACE command (in an interactive session) causes EDIT/3000 to display the line to be replaced, then the line from the HOLD file which replaces it.

In the example, lines 3 through 6 are replaced by four lines from the HOLD file.

```
    1      1-2.  WHAT IS EDIT/3000?
    2
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    7
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
/HOLD 8/11
CLEAR HOLD? Y
HOLD FILE LENGTH IS 4 RECORDS
/REPLACE 3/6,HOLD,NOW
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    3      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    4      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
    5      (MPE/3000) THAT IS USED TO CREATE AND
    5      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
    6      MANIPULATE ASCII FILES.
    6      MANIPULATED BY USING EDIT/3000 COMMANDS.
/FIND*
    7
           *(1  )
/LIST ALL
    1      1-2.  WHAT IS EDIT/3000?
    2
    3      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    4      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
    5      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
    6      MANIPULATED BY USING EDIT/3000 COMMANDS.
    7
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
```

- REPLACEQ,HOLD,NOW and REPLACEQ,HOLDQ,NOW

The REPLACEQ,HOLD,NOW form of the REPLACE command does not display the lines to be replaced but does display the lines from the HOLD file. In the following example, lines 3 through 6 are replaced by four lines from the HOLD file.

The REPLACEQ,HOLDQ,NOW form does not display the lines to be replaced *nor* the lines from the HOLD file. See the second command in the following example.

```
/REPLACEQ 3/6,HOLD,NOW
    3      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    4      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
    5      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
    6      MANIPULATED BY USING EDIT/3000 COMMANDS.
/REPLACEQ 3/6,HOLDQ,NOW
/LIST ALL
    1      1-2.  WHAT IS EDIT/3000?
    2
    3      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    4      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
    5      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
    6      MANIPULATED BY USING EDIT/3000 COMMANDS.
    7
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
```

REPLACE

- CREATING A TWO-COLUMN FORMAT WITH THE REPLACE COMMAND

The example below shows how to replace portions of lines with lines from the HOLD file to create a two-column format.

```
/SET LEFT=1,RIGHT=30
/ADD
     1      LINE 1 OF WORK FILE
     2      LINE 2 OF WORK FILE
     3      LINE 3 OF WORK FILE
     4      LINE 4 OF WORK FILE
     5      LINE 5 OF WORK FILE
     6      LINE 6 OF WORK FILE
     7      LINE 7 OF WORK FILE
     8      LINE 8 OF WORK FILE
     9      LINE 9 OF WORK FILE
    10      LINE 10 OF WORK FILE
    11      //
...
/HOLD^ 6/10
CLEAR HOLD? Y
HOLD FILE LENGTH IS 5 RECORDS
/SET RIGHT=40,LEFT=21
/REPLACE^ 1/5,HOLD^,NOW
/SET LEFT=1
/DELETE^ 6/10
NUMBER OF LINES DELETED = 5
/LIST ALL
     1      LINE 1 OF WORK FILE LINE 6 OF WORK FILE
     2      LINE 2 OF WORK FILE LINE 7 OF WORK FILE
     3      LINE 3 OF WORK FILE LINE 8 OF WORK FILE
     4      LINE 4 OF WORK FILE LINE 9 OF WORK FILE
     5      LINE 5 OF WORK FILE LINE 10 OF WORK FILE
```

3-92. SET COMMAND

3-93. Purpose. The SET command alters default conditions normally established by EDIT/3000.

3-94. Form. The form of the SET command is

S[*ET*] [{ BATCH / *POLL* }] [,DELTA = *increment*] [,DEPTH = *limit*] [, { FIXED / *VARIABLE* }]

[, FORMAT = { COBOL / *DEFAULT* }] [,FROM = *linenumber*] [, { FRONT / *REAR* }]

[, LEFT = *colnum*] [,LENGTH = *colnum*] [, { QUIET / *DISPLAY* }] [,LINES = *maxlines*]

[,RIGHT = *colnum*] [, { SHORT / *LONG* }] [,SIZE = *integer*] [,TIME[S] = *limit*]

[{ ,TABS [=(*colnum* [,*colnum*] . . .)] / ,NOTABS }] [,TABCHAR [=*string*]]

For example,

```
SET BATCH, SHORT, QUIET
```

3-95. Description. The default parameters for the SET command are underlined in the example. Before using a SET command, enter a VERIFY ALL command to display the conditions in effect.

The SET command can be entered on as many lines as necessary (end each line to be continued with an ampersand (&) or you may repeat the SET portion of the SET command at the start of the next line). Separate the parameters in a SET command with commas.

The SET command parameters are explained in the following paragraphs, in alphabetical order.

- SET BATCH/POLL. SET BATCH and SET POLL relate to where EDIT/3000 expects to find commands, text records, and where output will be sent. SET POLL is the default condition and means that EDIT/3000 expects to find commands and text records in the INPUT file. To operate in an interactive session in full batch mode (with text records following command records) in the USE file, SET BATCH is declared. SET POLL can be set to return to interactive mode and read text from the INPUT file.
- SET DELTA = *increment*. SET DELTA defines the interval between lines in the WORK file. The default increment is 1 or 0.1, depending on the SET FORMAT option in effect. If SET FORMAT = DEFAULT (the default condition), the increment is 1; if SET FORMAT = COBOL, the increment is 0.1. The SET DELTA value establishes line number increments which are used in the ADD, GATHER, and JOIN

commands, and (when the UNNUMBERED parameter is used) in TEXT commands. The SET DELTA value in effect can be overridden with the BY *increment* parameter of GATHER, INSERT, and JOIN commands.

- SET DEPTH = *limit*. SET DEPTH defines the maximum nesting of USE commands (how many times a USE command can call itself or another USE command); and the number of BEGIN-END pairs allowed in a WHILE block. The default limit is 10.

- SET FIXED/VARIABLE. SET FIXED and SET VARIABLE specify whether the WORK file contents are saved with a KEEP command into a file with fixed-length or variable-length records. The default condition is FIXED.

- SET FORMAT = COBOL/DEFAULT. SET FORMAT = COBOL and SET FORMAT = DEFAULT are used to inform EDIT/3000 whether EDIT/3000 operation is related to a COBOL source program or not. The default condition is DEFAULT.

  SET FORMAT = COBOL informs EDIT/3000 that the current WORK file contents, when kept, must be acceptable to a COBOL compiler. The chosen SET FORMAT option also determines certain other SET command default conditions as shown in table 3-1.

Table 3-1. Default EDIT/3000 Operating Conditions

SET COMMAND OPTION	SET FORMAT = DEFAULT	SET FORMAT = COBOL
FROM =	1	1
DELTA =	1	0.1
LEFT =	1	1
RIGHT =	72	74
LENGTH =	72	74
LINES =	60	60
QUIET/DISPLAY	DISPLAY	DISPLAY
SHORT/LONG	LONG	LONG
BATCH/POLL	POLL	POLL
DEPTH =	10	10
TIMES =	50	50
SIZE =	0	0
FRONT/REAR	REAR	FRONT
FIXED/VARIABLE	FIXED	FIXED
TABS/NOTABS	NOTABS	NOTABS
TABCHAR=	CONTROL I ('9)	CONTROL I ('9)

In addition, when SET FORMAT = DEFAULT, the smallest increment allowed for line number intervals is .001. The maximum line number count allowed is 99999.999. This restricts sequence numbers derived from line numbers with the KEEP command to eight bytes, with a range from 00000001 through 99999999. If SET FORMAT = COBOL, the smallest increment allowed for line number intervals is .001 and the maximum line number count allowed is 999.999. This restricts sequence numbers derived from line numbers with the KEEP command to six bytes, with a range from 000001 to 999999.

- SET FROM = *linenumber*. SET FROM = *linenumber* specifies the starting line number for text entered into the WORK file with an ADD command, or for text copied by the GATHER, JOIN, or TEXT *filename*, UNNUMBERED commands. The default is 1 (new text will start at line 1 in the WORK file).

- SET FRONT/REAR. SET FRONT and SET REAR request EDIT/3000 to form sequence numbers from the front bytes (SET FRONT) of each record or from the rear bytes (SET REAR) of each record. The default condition depends on the SET FORMAT option in effect. If SET FORMAT = DEFAULT, the last eight bytes of each record are used. If SET FORMAT = COBOL, the first six bytes of each record are used.

- SET LEFT = *colnum*. This parameter specifies the left margin for text records in the WORK file. The default condition is 1 and text will start in column number 1 of each record. SET LEFT must be less than or equal to SET RIGHT and SET LENGTH.

- SET LENGTH = *colnum*. This parameter specifies the maximum length (record size) of all records in the WORK file and may not exceed 255 bytes. The default length depends on the SET FORMAT option in effect. If SET FORMAT = DEFAULT, length = 72; if SET FORMAT = COBOL, length = 74. SET LENGTH must never be less than SET RIGHT.

  Some EDIT/3000 commands can increase the length of a line in the WORK file up to 50 percent larger than the length in effect (a warning message will be displayed but the command will execute). If a command attempts to increase the length of a line by more than 50 percent, however, an error message is displayed and the command terminates. In batch mode, EDIT/3000 operation also is terminated.

- SET LINES = *maxlines*. This parameter specifies the number of lines that will be printed on each page of an offline listing. The default is 60 lines. A minimum of 10 and maximum of 9999 lines can be specified; the number includes 3 lines for the page title.

- SET QUIET/DISPLAY. DISPLAY is the default condition (EDIT/3000 will display line numbers for lines to be added, and so forth). SET QUIET is equivalent to using the Q option for EDIT/3000 commands (except KEEP).

- SET RIGHT = *colnum*. This parameter specifies the right margin for text records in the WORK file. The default is 72 if the SET FORMAT = DEFAULT parameter has been declared and 74 if the SET FORMAT = COBOL parameter has been declared. *Colnum* must be greater than or equal to that set with a SET LEFT command and may not exceed the *length* established for records. SET RIGHT must never exceed SET LENGTH.

- SET SHORT/LONG. SET SHORT and SET LONG determine the length of EDIT/3000 messages. SET LONG is the default condition. SET SHORT cancels SET LONG and causes messages to be shortened. For example,

```
/END
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? YES

 END OF SUBSYSTEM

/SET SHORT;END
CLEAR? YES

 END OF SUBSYSTEM
```

SET SHORT also is used to inhibit display of commands in a WHILE block.

- SET SIZE = *integer*. SET SIZE establishes the actual size of the WORK file, in lines. The default is 0, when EDIT/3000 begins operation. With the default condition in effect, EDIT/3000 sets the size of the WORK file according to the source of text records, as follows:

  If the source device is a terminal, a card reader, or a magnetic tape unit, EDIT/3000 sets the size to allow up to 2000 records (unless the SET SIZE command has set the size to some other value in lines). If the source device is a disc file, EDIT/3000 sets the size as shown below:

TEXT DISC FILE SIZE (DFS) (RECORDS)	ACTUAL SIZE OF WORK FILE (RECORDS)
up to 1000	2000
1000 to 4000	DFS + 50%
greater than 4000	DFS + 2000

  Note that for every 8 lines of text, EDIT/3000 uses one record for its own purpose. A numbered file also contains a directory at the end of the file. Because of this, approximately 88% of the records can be used for text lines. For example, if you do not SET SIZE explicitly before texting in a file with less than 1000 records, EDIT/3000 allows up to 2000 records in your WORK file, which means that approximately 1,767 (88% of 2000) lines of text are allowed.

  Size should be set explicitly whenever possible, especially if less than 2000 records (approximately 1,767 lines of text) is sufficient space in the WORK file.

- SET TABCHAR = *string*. SET TABCHAR defines a tab character to be used when entering text beginning at a specific tab stop. The default tab character if none is specified is Control I. The length of *string* may not be greater than one character. A decimal value may be specified for *string* if the digits are preceded by an apostrophe. The decimal value for Control I is '9 (the ASCII Horizontal Tab character).

  When entering a line of text with the ADD or REPLACE command, the tab character is used to tab to the next column defined as a tab stop. To position at the second tab stop two tab characters are entered. For example, if the tab character is defined as follows:

```
/SET TABCHAR="!"
```

  and a line of text is to begin at the second tab stop, the addition looks like this:

```
/ADD
      1       !!45.85
      2
```

  When the text is moved into the work file, the tab characters are replaced with a sufficient number of blanks to position up to the second tab stop and the number 45.85 is moved into the next five columns.

  Note: Tab characters are recognized only in the ADD and REPLACE commands.

- SET TABS=(*colnum*,...)/NOTABS. SET TABS= defines up to 12 tab stops to allow text to be entered beginning at a specified column. If no SET TABS command is used, the default condition is SET NOTABS. If SET TABS is used but no column numbers are specified, the default column numbers are:

  - (4,7,10,13,16,19,22,25,28,31,65,71) if FORMAT=DEFAULT is set,

  - (6,10,14,18,22,26,30,34,38,46,54,67) if FORMAT=COBOL is set.

  Each *colnum* parameter must be larger than the preceding *colnum* parameter and must lie between the LEFT and RIGHT limits.

  When using the TAB key (provided on some terminals) in conjunction with the Editor SET TABS feature, the following considerations apply:

  - It is the user's responsibility to make sure that the tab stops set on the terminal correspond to the tab stop column numbers specified with the Editor SET TABS command. The column positions depend upon the command that is to be used. The ADD and REPLACE commands print ten characters on the terminal before the first column of the line. The ADDQ command accepts the first column of the line beginning with the first cursor position.

- When the TAB key is pressed, the Editor enters one tab character (Control I or a decimal 9) in the current line. The terminal positions the cursor to the next tab stop set on the terminal. When the line is moved into the WORK file, the Editor converts the tab character to the appropriate number of blanks to position the information at the proper columns. It is important to remember this when using the backspace (Control H). In this case, the terminal moves the cursor back one position but the Editor backspaces one character in the current line. If that character happens to be a tab character, the effect of backspacing is to return to the column position preceding the tab. For example, suppose a terminal tab stop is set at position 10 and the following commands are entered.

```
/SET TABS = (10)
/ADDQ
145I^c      H^c          Note: I^c indicates the tab key has been pressed and H^c indi-
        ↑                      cates Control H or backspace entered.
    Cursor position
```

The terminal moves the cursor one character position to the left when the backspace character is entered. However, the Editor considers the cursor position to be the 4th column since it backspaced over the tab character.

```
145I^c      Current line according to Editor
  ↑
```

- The tab stops may be set on the terminal either manually, or programmatically with an Editor USE file specified by the user. The code for TAB SET is Escape 1 (1^e). The following USE file (named TABFIL) sets tabs in columns 10 and 20 of lines to be entered with the ADD command:

```
SET TABS (10,20);
Q "          1^e          1^e  ";    (1^e indicates Escape 1)
```

To execute the USE file, enter the USE command and the file name. For example:

```
/USE TABFIL
```

Before setting new tabs, existing tabs should be reset with one of the following:

- Escape 2 (or CLEAR TABS)
- Escape 3 (or CONTROL CLEAR TABS (all) for HP 2645)
- RESET TERMINAL key

- SET TIME = *limit*. SET TIME defines the maximum number of times an EDIT/3000 WHILE block will execute. The default is 50 iterations.

3-96. **Limitations.** Parameters of the SET command may be included in any order (separated by commas), however, some parameters are dependent on others and this must be considered. For example,

The value used in a SET LEFT command may not exceed the value used in a SET RIGHT command or any tab setting.

The value used in a SET RIGHT command may not exceed the value used in a SET LENGTH command or less than any tab setting.

After a WORK file has been created (with TEXT or ADD), the value of LENGTH must be less than the WORK file record size. However, before a WORK file is created, the maximum value used in SET LENGTH may be 255.

Such SET command pairs as SET SHORT, SET LONG, and SET QUIET, SET DISPLAY are, of course, mutually exclusive. (The last occurrence of such a pair is in effect.)

Commands that can extend existing lines (such as MODIFY, INSERT, and CHANGE) shift characters to the right of the RIGHT margin, although only that text within the left and right margins is actually altered. REPLACE is an exception to this; characters are not shifted to the right of the RIGHT margin. The REPLACE command displays only that text within the margins and only replaces that part of the line. The replacement string is padded with blanks if it is shorter. A warning message is displayed if the string is longer than the LEFT/RIGHT range.

3-97. Examples. Examples are provided for several SET command options.

- SET DELTA

SET DELTA defines the interval between lines in the WORK file. In the following example, the interval is 1 (the default condition if SET FORMAT = DEFAULT) and lines entered with the ADD command are numbered 1, 2, and 3. After the SET DELTA = .001 command is entered, lines entered are numbered 1, 1.001, and 1.002.

```
/VERIFY DELTA
DELTA =      1
/ADD
     1      LINE 1
     2      LINE 2
     3      LINE 3
     4      //
...
/D ALL
CLEAR? Y
/SET DELTA=.001
/ADD
     1      LINE 1
     1.001  LINE 2
     1.002  LINE 3
     1.003  //
...
```

- SET FROM

SET FROM determines the starting line number for text entered into an empty WORK file.

In the example, lines start at 1 and 101, depending on the SET FROM = *linenumber* option in effect.

```
/VERIFY FROM
FROM =      1
/ADD
     1      LINE 1
     2      LINE 2
     3      //
...
/D ALL
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
/SET FROM=101
/ADD
   101      LINE 1
   102      LINE 2
   103      //
...
```

- SET QUIET AND SET DISPLAY

The SET QUIET command is equivalent to using the Q parameter for EDIT/3000 commands (except KEEP).

In the example, after SET QUIET is entered, EDIT/3000 does not prompt with line numbers when the ADD command is executed.

When the LIST command executes, line numbers are not displayed.

Using the SET DISPLAY command sets QUIET to FALSE.

```
/VERIFY QUIET
DISPLAY = TRUE (I.E. QUIET = FALSE)
/ADD
     1      LINE 1
     2      LINE 2
     3      //
...
/SET QUIET
/ADD
LINE 3
LINE 4
//
...
/LIST ALL
LINE 1
LINE 2
LINE 3
LINE 4

/SET DISPLAY
/LIST ALL
     1      LINE 1
     2      LINE 2
     3      LINE 3
     4      LINE 4
```

- SET TABCHAR AND SET TABS

These commands define a tab character and tab stops to allow text to be entered beginning at specific columns. In the example, the tab character is defined as a percent sign (%), and tab stops are set at columns 5 and 30. When line 1 is added, two percent signs are entered to tab over to the second tab stop (column 30). Then the data is entered. Line 3 begins at the first tab stop. When the text is listed, it appears in the correct columns. This is the way the information has been stored in the work file.

```
/SET TABCHAR = "%", TABS = (5,30)
/VERIFY TABS,TABCHAR
TABS = (  5, 30)
TAB CHARACTER = "%"
/ADD
     1      %%JANUARY 16, 1977
     2      DEAR MR. JONES:
     3      %WE ARE PLEASED TO ADVISE YOU THAT YOU HAVE WON
     4      A NEW POCKET CALCULATOR.  YOU WILL RECEIVE IT IN
     5      ...
/L.1/4
     1                                JANUARY 16, 1977
     2      DEAR MR. JONES:
     3          WE ARE PLEASED TO ADVISE YOU THAT YOU HAVE WON.
     4      A NEW POCKET CALCULATOR.  YOU WILL RECEIVE IT IN
```

3-98. TEXT COMMAND

3-99. Purpose. The TEXT command copies the contents of an MPE/3000 file into the WORK file.

3-100. Form. The form of the TEXT command is

$$\text{T[EXT]}\left[\textit{filename}\left[\begin{matrix}\textit{(linenumber/linenumber)}\\ \textit{(\#recnum/\#recnum)}\end{matrix}\right]\left[\text{,UNN[UMBERED]}\right]\right]$$

3-101. Description. A TEXT command that includes only the *filename* parameter copies the entire file into the WORK file. If the *(linenumber/linenumber)* parameter is included, only the records corresponding to the *line numbers* specified will be copied into the WORK file, and if the *(#recnum/#recnum)* parameter is included, only those records corresponding to the *logical record numbers* specified will be copied into the WORK file. (See paragraph 3-103, EXAMPLES.) If neither*(linenumber/linenumber)*nor *(#recnum/#recnum)* is used, the entire TEXT file contents are copied to the WORK file.

If the *filename* and the UNNUMBERED option are not specified, they are implied from the previous KEEP or TEXT command (whichever was performed last) for that file. If only the *filename*, but not the option, is specified, the UNNUMBERED option is implied if the text file is unnumbered, and the EDITOR responds with the message,

FILE UNNUMBERED

For example,

```
:EDITOR
HP32201A.7.01 EDIT/3000  MON, MAY3, 1976  1:51 PM
(C) HEWLETT-PACKARD CO. 1976
/TEXT KEEPFIL,UNN
/K KEEPFIL,UNN
KEEPFIL ALREADY EXISTS - RESPOND YES TO PURGE OLD AND THEN KEEP
PURGE OLD?Y
/T
 KEEPFIL,UNN
/TEXT KEEPFIL
 FILE UNNUMBERED
/
```

If the UNNUMBERED option was not specified in the last KEEP or TEXT command, only the *filename* is implied from that command when the current TEXT command is executed.

For example,

```
:EDITOR
HP 32201A.07.01 EDIT/3000  MON, MAY  8, 1978,  3:55 PM
(C) HEWLETT-PACKARD CO. 1976
/A
    1      OLD MATTER
    2      NEW MATTER
    3      ...
/K KEEPFIL
/T
 KEEPFIL
/END
END OF SUBSYSTEM
:
```

In an interactive session, if the TEXT command is used when the WORK file has been altered, EDIT/3000 asks if it is OK to clear the WORK file before the TEXT command is executed. If the response is other than YES or Y, the TEXT command is not executed. When a TEXT command is executed in batch mode, EDIT/3000 clears the WORK file automatically if it has been altered.

If FIXED is currently set when the TEXT command is issued, the value of LENGTH is taken from the TEXT file (the maximum record length). If VARIABLE is set, LENGTH remains set at its current value. EDIT/3000 then resets RIGHT to the value of LENGTH and LEFT to 1, and sets FIXED or VARIABLE depending on whether the TEXT file is a fixed or variable record file. This has the advantage of giving you more control over your operating environment. For example, if you desire to expand the records of a fixed length record file, you may issue the SET LENGTH= *colnum* and SET VARIABLE commands before texting in the file. These same commands should be used for variable length record files. EDIT/3000 cannot calculate the maximum record length of variable length files, so it uses the block size. By setting VARIABLE and LENGTH when texting in or keeping a variable length record file, less disc space is used for the WORK and TEXT files and better performance can be obtained.

3-102. Limitations. If a file is saved with a KEEP, UNNUMBERED command (or otherwise created without sequence numbers), the TEXT command usually will be of the form TEXT *filename*, UNNUMBERED. New line numbers will be assigned by EDIT/3000 starting with the SET FROM = *linenumber* option in effect and incremented by the SET DELTA = *integer* option in effect. If the text file has variable length records (type "V"), then VARIABLE will be set. (See the SET command, paragraph 3-92.)

When texting in unnumbered files from devices that cannot be rewound, the UNNUMBERED option should always be specified. Otherwise, one or more records will be lost from the beginning of the file.

Any sequence numbers contained in a file specified by *filename* must be in the front or rear bytes as specified in a SET FORMAT, SET FRONT, or SET REAR command (see the SET command, paragraph 3-92). The sequence numbers will be interpreted as line numbers instead of being copied as part of the text. If the sequence numbers are not in bytes that match the SET FORMAT or SET FRONT/REAR conditions in effect, the command TEXT *filename*, UNNUMBERED should be used. Then, the sequence numbers will be copied as part of the text and new line numbers will be assigned by EDIT/3000, starting from the SET FROM = *linenumber* in effect and incremented by the SET DELTA = *integer* option in effect.

If a file has been saved with the KEEPQ *filename* command, then the filename must be specified in a TEXT command for that file.

If a file has been saved with the KEEPQ *filename* command, the file is saved in its structured "edit" format, instead of in the sequential format in which files normally are stored by the MPE/3000 file system. This is accomplished by appending EDIT/3000 variables to the file and assigning a *filecode* of 1050 or 1051 to the file. A *filecode* of 1050 signifies an EDIT/3000 KEEPQ file (DEFAULT) and a *filecode* of 1051 signifies an EDIT/3000 KEEPQ file (COBOL). All such files are intended to be read only by EDIT/3000, although SPL/3000 can read KEEPQ files also. A TEXT command to copy the contents of such a file into the WORK file *renames* the file. To prevent destruction of such a file, enter the commands TEXT *filename* and KEEP *filename* in a succession. (See paragraph 3-62 for a discussion of the KEEP command and the *MPE Commands Reference Manual* for a discussion of *filecode* and the MPE/3000 file system.)

Texting in an empty file will result in error #23, FAILURE TO OPEN TEXT FILE (0), with File System error #0 (END OF FILE).

Texting in a file with records greater than 256 bytes will result in error #24, RECORD SIZE TOO LARGE.

The default TEXT file is limited to disc files on the local computer. If a device or remote file is being used, an MPE file equation (see the description of the:FILE command in the *MPE Commands Reference Manual*) must have been issued and an asterisk (*) must be used to back-reference the file. (See the example of texting in a card deck in paragraph 3-103).

When texting in a numbered file, an extra data segment is required. Since a two-word entry is made in the extra data segment for every eight records texted in, the number of records that can be texted in is limited by the configured maximum size for extra data segments. Consult your System Manager to determine the maximum size for extra data segments ($\leq$ 32K) and multiply it by 4 to determine the maximum number of records that can be texted in from a numbered file. There is no limit on the number of records when texting in unnumbered file.

Trailing blanks are truncated from the records before they are placed in the WORK file. Therefore, if a search is made for a string which ends with blanks, the string will not be found if it is located at the end of a record.

3-103. Examples. Several variations of the TEXT command are shown in the following examples. The location of the pointer is illustrated at the conclusion of each command.

- COPYING A NUMBERED FILE WITH A TEXT *filename* UNNUMBERED COMMAND

If a numbered TEXT file is copied into the WORK file with a TEXT *filename*, UNNUMBERED command, EDIT/3000 treats the entire text record as data and generates sequencing information.

In the following example, the file TEST (with a record size of 60 bytes) is copied into the WORK file with the T[EXT] TEST command, the first three lines are listed, and EDIT/3000 operation is terminated.

Next, EDIT/3000 is accessed and the same file is copied into the WORK file with a T TEST,UNN command.

When the first three lines are listed, note that EDIT/3000 has appended sequencing information to each record. When the WORK file is saved with the K TEST command, EDIT/3000 appends *another* set of sequencing information and now the record size is 68 bytes (as shown by the LISTF TEST,1 command).

If the KEEP *filename*, UNNUMBERED form of the command is used, the sequencing information generated by EDIT/3000 will not be appended to the end of the records.

```
:LISTF TEST,1
ACCOUNT= GOODWIN     GROUP= PUB

FILENAME  CODE  ------------LOGICAL RECORD--------
                  SIZE  TYP        EOF      LIMIT

TEST               60B  FA           3          3

:EDITOR

HP32201A.4.01 EDIT/3000  TUE, JUN  3, 1975,  3:59 PM
/S SHORT;T TEST;L 1/3
    1      1-2.  WHAT IS EDIT/3000?
    2
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
/E


 END OF SUBSYSTEM
:EDITOR

HP32201A.4.01 EDIT/3000  TUE, JUN  3, 1975,  4:00 PM
/S SHORT;T TEST,UNN;L 1/3
    1      1-2.  WHAT IS EDIT/3000?                          00001000
    2                                                        00002000
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000           00003000
/K TEST
PURGE OLD?T
PURGE OF OLD FILE NOT CONFIRMED - TEXT NOT KEPT
/K TEST
PURGE OLD?Y
/E


 END OF SUBSYSTEM
:LISTF TEST,1
ACCOUNT= GOODWIN     GROUP= PUB

FILENAME  CODE  ------------LOGICAL RECORD--------
                  SIZE  TYP        EOF      LIMIT

TEST               68B  FA           3          3
```

```
:EDITOR

HP32201A.4.01 EDIT/3000  TUE, JUN  3, 1975,  4:05 PM
/S SHORT;T TEST,UNN;L 1/3
    1      1-2.  WHAT IS EDIT/3000?                                  00001000
    2                                                                00002000
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000                   00003000
/K TEST,UNN
PURGE OLD?Y
/E
CLEAR? Y

 END OF SUBSYSTEM
:LISTF TEST,1
ACCOUNT=  GOODWIN      GROUP=  PUB

FILENAME  CODE  ------------LOGICAL RECORD--------
                   SIZE  TYP         EOF       LIMIT

TEST               60B   FA           10          10
```

- TEXT *filename*

The next example illustrates the use of the TEXT *filename* command. Note that the FIND * command causes an error message to be displayed. (The pointer location is at line zero after a TEXT command is executed.)

```
/S SHORT;TEXT EDIT3
/FIND*
*44*LINE NUMBER ZERO CAN NOT BE ACCESSED
/LIST FIRST;LIST LAST
    1      1-2.  WHAT IS EDIT/3000?
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
/LIST ALL
    1      1-2.  WHAT IS EDIT/3000?
    2
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    7
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
```

- TEXT *filename,range*

The *range* (3/8) specified in the following example copies lines 3 through 8 of file EDIT2 into the WORK file.

Note that the location of the pointer is still at line zero even though text starts at line 3.

```
/S SHORT;TEXT EDIT3 (3/8)
/FIND*
*44*LINE NUMBER ZERO CAN NOT BE ACCESSED
/LIST FIRST;LIST LAST
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
/LIST ALL
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    7
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
```

- COPYING A TEXT FILE FROM CARDS

To copy a TEXT file, located on cards, into the WORK file, perform the following steps as shown in the next example:

1. Log on to the system.

2. Enter a :FILE command, as follows:

   :FILE *filename;*DEV=CARD

   CARD is a device class assigned during system configuration to the card reader.

   for example

   :FILE INFILE;DEV=CARD

3. Place the card deck in the card reader. The first card in the card deck must be a :DATA card, as follows:

   :DATA[*jobname,*] {*username*} [*/upass*] {*.acctname*} [*/apass*] {*;filename*}

   where

   jobname
   is the name of the job or session that is to read the data. (Optional parameter.)

   username
   is the user's name, as established in MPE/3000 by the user with Account Manager Capability. (Required parameter.)

   upass
   is the user password. (Required if user has a password.)

   acctname
   is the name of the account, as established by the user with System Manager Capability. (Required parameter.)

   apass
   is the account password. (Required if account has a password.)

   filename
   is an additional qualifying name that can be used by the job or session to access the data. (Optional parameter.)

   The :DATA card used in the example was as follows:

   :DATA MANAGER.SCR

   Note that the *jobname* parameter was not included and that the *username.acctname* is the same as that used to log on (MANAGER.SCR).

   The last card in the card deck must be an :EOD card.

   See the *MPE Commands Reference Manual* for further discussions of the :DATA and :EOD commands.

4. Press the RESET switch (or equivalent) on the card reader to read the cards.

5. Enter the :EDITOR command to access EDIT/3000.

6. Enter the TEXT command. The command used in the example is

   TEXT *INFILE,UNNUMBERED

   Note: The asterisk is a *back reference* to the file defined in the :FILE command. The UNNUMBERED parameter is recommended because the file to be read did not contain line numbers. (Note that EDIT/3000 assigned line numbers as the file was texted in.)

```
:HELLO MANAGER.SCR
 SESSION NUMBER = #S179
 WED, MAY 28, 1975,  9:47 AM
 HP32000C.00.24

:FILE INFILE;DEV=CARD
:EDITOR

HP32201A.4.01 EDIT/3000  WED, MAY 28, 1975,  9:47 AM
/S SHORT
/TEXT *INFILE,UNNUMBERED
/LIST ALL
     1      JOHN BIGTOWN
     2      LOIS ANYONE
     3      ALI BABA
     4      JAMES DOE
     5      JOHN DOUGHE
     6      MARY MEEK
     7      SPACE MANN
     8      KING ARTHUR
     9      KARISSA GRANDTR
    10      JENNA GRANDTR
    11      SWASH BUCKLER
    12      KNEE BUCKLER
/KEEP NAMELIST
/END


 END OF SUBSYSTEM
:BYE

CPU (SEC) = 75
CONNECT (MIN) = 56
MON, JUN 30, 1975,  2:00 PM
END OF SESSION
```

TEXT

3-104. USE COMMAND

3-105. Purpose. The USE command designates a file (USE file) to be used by EDIT/3000 as a source of commands and/or text.

3-106. Form. The form of the USE command is

U[SE] [*filename*]

3-107. Description. The USE command causes EDIT/3000 to read commands from the USE file. In an interactive session, commands are read from the USE file, but any messages or requests for text records to be entered from the terminal are displayed on the terminal.

A USE command without the *filename* parameter functions as a return statement from the current EDIT/3000 USE command file in the same way that an end of data indication does. Note, however, that because the USE command will not be recognized if the FLAG is false (in a WHILE block for example,see Section IV), this type of return (unlike the end of data indication) is conditional. Thus, this conditional return may be used to write recursive EDIT/3000 command files. In batch mode, EDIT/3000 reads commands from the USE file but no interaction with the user occurs. Any text records can be in the USE file, or the USE file can transfer control to another file (INPUT or HOLD) to read the records. It is recommended that the Q command (see paragraph 3-80) be used in a USE file to remind you (or another user) of the purposes of the USE file. Text records can be included in a USE file and will be interpreted as such if the SET BATCH command is used. This allows you to operate in an interactive session in full batch mode (with text records following commands in the USE file). This option is useful when the same commands and text records are to be used in many positions in the WORK file.

A USE file may contain comments wherever a command may appear but not embedded in a command. A comment must begin with two adjacent less-than signs (<<) and end with two adjacent greater-than signs (>>). For example, a USE file may contain:

```
TEXT EDITFILE
FINDQ FIRST; FIND "#"; <<LOCATE LINE TERMINATED WITH #>>
LIST
```

3-108. Limitations. Any file to be read as a USE file must contain ASCII-coded records and must have a *filecode* between 0 and 1023. (See the *MPE Commands Reference Manual* for a discussion of *filecode.*)

The USE file should be kept unnumbered unless the commands are terminated with a semicolon, which signals to EDIT/3000 that the line number is not part of the command.

END in a USE file causes the USE command to terminate; the USE file is exited and EDIT/3000 prompts you with its usual slash. Note that END as the final line in a USE file is of no consequence — whereas if END appears anywhere else in the USE file, the lines following it are not executed.

3-109. **Examples.** Two examples are shown to illustrate the USE command and USE files.

- READING COMMANDS FROM A USE FILE

The USE file (USEFILE) in the following example contains the commands TEXT, CHANGE, FIND, MODIFY, and LIST. The command END is added later.

When the command USE USEFILE is entered, EDIT/3000 copies the file TEST into the WORK file, changes "EDIT/3000" to "HP 3000 TEXT EDITOR" for all occurrences of EDIT/3000, finds the first line, then searches forward from there to find the string "ASCII". The line (line 6) containing "ASCII" is displayed for modification and the characters "-CODED" are inserted. EDIT/3000 then lists the contents of the WORK file. When END is added in line 2.1 of USEFILE, the commands which follow line 2.1 are not executed when the USE command is used.

```
    1      TEXT TEST
    2      CHANGE "EDIT/3000" TO "HP 3000 TEXT EDITOR" IN ALL
    3      FINDO FIRST;FIND "ASCII"
    4      MODIFY *
    5      LIST ALL
    6      //
...
/KEEP USEFILE,UNN
/USE USEFILE
    1      1-2.  WHAT IS HP 3000 TEXT EDITOR?
    3      HP 3000 TEXT EDITOR IS A SUBSYSTEM OF THE HP3000
    6      MANIPULATE ASCII FILES.
                (12 )"
MODIFY     6
MANIPULATE ASCII FILES.
                I-CODED
MANIPULATE ASCII-CODED FILES.

    1      1-2.  WHAT IS HP 3000 TEXT EDITOR?
    2
    3      HP 3000 TEXT EDITOR IS A SUBSYSTEM OF THE HP3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII-CODED FILES.
/T USEFILE
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? YES
 FILE UNNUMBERED
/ADD 2.1
    2.1    END
    2.2    //
...
/K USEFILE,UNN
USEFILE ALREADY EXISTS - RESPOND YES TO PURGE OLD AND THEN KEEP
PURGE OLD?YES
/USE USEFILE
    1      1-2.  WHAT IS HP 3000 TEXT EDITOR?
    3      HP 3000 TEXT EDITOR IS A SUBSYSTEM OF THE HP3000
/
```

- READING COMMANDS AND TEXT FROM A USE FILE

USEFILE2, shown in the following example, contains the commands TEXT EDIT2, ADD, then 9 lines of text (including blank lines) to be added. The characters // signify the end of the text records.

> Note: If the USEfile is prepared using the EDIT/3000 ADD command, EDIT/3000 will interpret // as the end of the ADD command. To overcome this, enter //, then a blank character before RETURN so that line 13 (LIST ALL) can be added. When the USE file commands are executed, the blank character following // will be ignored.

Before the USE USEFILE2 command is entered, the SET BATCH command is entered to inform EDIT/3000 that the USE file contains text records.

EDIT/3000 copies the contents of file EDIT2 into the WORK file by executing the TEXT command, lists the lines that were added to EDIT2, executes the LIST ALL command, then saves the WORK file contents by executing the KEEP EDIT3 command.

```
    1      TEXT EDIT2
    2      ADD
    3
    4      1-2.  EDIT/3000 FEATURES
    5
    6      WITH EDIT/3000, IT IS POSSIBLE TO
    7
    8      CREATE AND BUILD A NEW FILE (CALLED A WORK FILE)
    9      BY ENTERING COMMANDS AND LINES OF TEXT FROM THE
   10      STANDARD INPUT DEVICE OR FROM A SPECIAL DISC FILE
   11      CALLED THE HOLD FILE.
   12      //
   13      LIST ALL
/KEEP USEFILE2,UNNUMBERED
/SET BATCH
/USE USEFILE2
   12
   13      1-2.  EDIT/3000 FEATURES
   14
   15      WITH EDIT/3000, IT IS POSSIBLE TO
   16
   17      CREATE AND BUILD A NEW FILE (CALLED A WORK FILE)
   18      BY ENTERING COMMANDS AND LINES OF TEXT FROM THE
   19      STANDARD INPUT DEVICE OR FROM A SPECIAL DISC FILE
   20      CALLED THE HOLD FILE.
...
    1      1-1.  WHAT IS EDIT/3000?
    2
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    7
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
   12
   13      1-2.  EDIT/3000 FEATURES
   14
   15      WITH EDIT/3000, IT IS POSSIBLE TO
   16
   17      CREATE AND BUILD A NEW FILE (CALLED A WORK FILE)
   18      BY ENTERING COMMANDS AND LINES OF TEXT FROM THE
   19      STANDARD INPUT DEVICE OR FROM A SPECIAL DISC FILE
   20      CALLED THE HOLD FILE.
/KEEP EDIT3
```

USE

3-110. VERIFY COMMAND

3-111. Purpose. The VERIFY command displays the EDIT/3000 options in effect.

3-112. Form The form of the VERIFY command is

```
V[ERIFY][AL[L]] [,B[ATCH]] [,DEL[TA]][,DE[PTH]] [,D[ISPLAY]] [,F[ILES]] [,FIX[ED]]
        [,FO[RMAT]] [,FR[OM]] [,FRON[T]] [,LE[FT]] [,LEN[GTH]] [,L[INES]]
        [,LO[NG]] [,P[OLL]] [,Q[UIET]] [,R[EAR]] [,RI[GHT]] [,SH[ORT]] [,S[IZE]]
        [,TABC[HAR]] [,T[ABS]] [,TI[MES]] [,TO[TAL]] [,V[ARIABLE]]
```

3-113. Description. The VERIFY command is used to display all the current EDIT/3000 options. Parameters may be added to the VERIFY command in any order but must be separated by commas. Parameters may be abbreviated to as few of their leading characters as needed to identify the parameters. Any abbreviation that includes characters beyond the required minimum will likewise be accepted. For example, you may abbreviate [,FORMAT] to [,FO], [,FOR], [,FORM], etc. All VERIFY parameters except FILES and TOTAL refer to those options which can be modified with a SET command (see paragraph 3-92). The FILES parameter causes EDIT/3000 to display all references to MPE/3000 files during the current EDIT/3000 cycle. The TOTAL parameter displays a count of the number of lines in the current WORK file (this count changes after an ADD, DELETE, INSERT, JOIN, or TEXT command is executed).

If the VERIFY ALL command is used, all options in effect are displayed.

3-114. Limitations. If the VERIFY command is entered with no parameters, it is interpreted as a FIND * command and will locate the current position of the pointer. Therefore, if you wish to verify any of the options in effect, you must append them to the VERIFY command or use the VERIFY ALL command.

3-115. Examples. The following examples illustrate the use of the VERIFY BATCH, VERIFY FROM, VERIFY, VERIFY FILES and VERIFY ALL commands. Note that, since the VERIFY command (with no parameters) is interpreted as a FIND * command, line 4 (the present location of the pointer) is displayed.

Note also that the the VERIFY ALL command displays the present location of the pointer before displaying all options in effect.

```
/VERIFY BATCH
POLL = TRUE  (I.E. BATCH = FALSE)
/VERIFY FROM
FROM =     1
/VERIFY
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
           *(I  )
/VERIFY ALL
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
           *(I  )
POLL = TRUE  (I.E. BATCH = FALSE)
REAR = TRUE (I.E. FRONT = FALSE)
DELTA =     1
CURRENT DEPTH = 0, THE DEPTH LIMIT = 10
RIGHT =72
LENGTH = 72
LONG = TRUE (I.E. SHORT = FALSE)
TIME = 50
TOTAL NUMBER OF CURRENT LINES = 11
FROM =     1
LEFT =  1
FIXED = TRUE (I.E. VARIABLE = FALSE)
SIZE = 0
DISPLAY = TRUE (I.E. QUIET = FALSE)
FORMAT=DEFAULT
FILES:
    WORK: KI331315
    KEEP:
    TEXT: EDIT2.PUB.GOODWIN          TUE, MAY 13, 1975,  1:16 PM
    JOIN:
```

If the WORK file has been changed, VERIFY ALL and VERIFY FILES will print an appropriate message beneath the name of the WORK file, as in the following:

```
/T TEST
/LIST ALL
    1      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    2      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
/ADD
    3      (MPE/3000) THAT IS USED TO CREATE AND
    4      MANIPULATE ASCII FILES.
    5      //
***
/V FILES
FILES:
    WORK: K2211204
          WORK FILE HAS BEEN ALTERED
    KEEP: TEST,KING,USERS            WED, AUG  9, 1978, 12:03 PM
    TEXT: TEST,KING,USERS            WED, AUG  9, 1978, 12:04 PM
    JOIN:
/K TEST
TEST ALREADY EXISTS - RESPOND YES TO PURGE OLD AND THEN KEEP
PURGE OLD?Y
/END

END OF SUBSYSTEM
```

The VERIFY TABS and VERIFY TABCHAR commands are illustrated in the next example. If a SET TABS or SET TABCHAR command has not yet been used, the Editor responds to a VERIFY TABS or TABCHAR command with NOTABS USED. If the default tab stops are set, a VERIFY yields the default column numbers. Note that the default tab character is Control I ('9). If you change the tab stops with another SET TABS command, the VERIFY TABS commands yields the new column numbers.

```
/VERIFY TABS
NO TABS USED
/VERIFY TABCHAR
NO TABS USED
/SET TABS
/VERIFY TABS,TABCHAR
TABS = (  4,  7, 10, 13, 16, 19, 22, 25, 28, 31, 65, 71)
TAB CHARACTER = '9
/SET TABS = (5,20,45)
/VERIFY TABS
TABS = (  5, 20, 45)
/
```

3-116. XPLAIN COMMAND

3-117. Purpose. The XPLAIN command displays summary descriptions of selected EDIT/3000 commands or all EDIT/3000 commands.

3-118. Form. The form of the XPLAIN command is

```
X[PLAIN][{ALL} [,A[DD]][,B[EGIN]][,C[HANGE]][,CO[PY]][,D[ELETE]]
         [,E[ND]][,F[IND]][,G[ATHER]][,H[OLD]][,I[NSERT]]
         [,J[OIN]][,K[EEP]][,L[IST]][,M[ODIFY]][,N[OT]][,O[R]]
         [,P]ROCEDURE]][,Q][,R[EPLACE]][,S[ET]][,T[EXT]]
         [,U[SE]][,V[ERIFY]][,W[HILE]][,X[PLAIN]]
         [,Y[ES]][,Z::=]
            [,OFFLINE]
```

3-119. Description. The XPLAIN command displays a brief description of each command included as a parameter with the XPLAIN command or all commands if the parameter ALL is included. Note that the parameters ALL and OFFLINE must be spelled completely, the parameter for COPY must be CO (or [,COP], [,COPY]), but all other parameters may be abbreviated to the first letter only (or the first letter plus any additional letters of the command; for example, [,LIST] may also appear as [,L], [,LI], and [,LIS]). If OFFLINE is specified, the descriptions are sent to the *listfile* as specified in the MPE/3000 :FILE command that defined *listfile*. (See paragraph 3-121, EXAMPLES.)

3-120. Limitations. When the XPLAIN command is used with no parameters, a description of the XPLAIN command itself is displayed.

3-121. **Examples.** The following examples illustrate the use of the XPLAIN ADD, XPLAIN OR, XPLAIN, and XPLAIN ALL,OFFLINE commands.

The XPLAIN command with no parameters displays a description of the XPLAIN command itself.

Before the command XPLAIN ALL,OFFLINE command can be executed, the *listfile* must be declared with an MPE/3000 :FILE command and equated to an output device, then the :EDITOR command must reference this file name. See the first two lines in the example.

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

HP32201A.4.01 EDIT/3000  TUE, MAY 13, 1975,  1:19 PM
/XPLAIN ADD
ADD:  TO ENTER LINES OF TEXT INTO THE TEXT FILE FROM THE $STDIN DEVICE
OR FROM THE HOLD FILE
EXAMPLE: ADD,60,HOLD

/XPLAIN OR
OR:  RESETS  FLAG  TO BE TRUE IF IT IS FALSE, OR TO FALSE IF FLAG IS
TRUE
EXAMPLE:  BEGINQ
          FINDQ"THIS"/*(+40)
          OR
          FINDQ"PROGRAM"/*(+20)
          END

/XPLAIN
XPLAIN:  TO OBTAIN AN EXPLANATION OF THE COMMANDS
EXAMPLE:  XPLAIN A,SET,F

/XPLAIN ALL,OFFLINE
*** OFF LINE LISTING BEGUN. ***
```

3-122. Z::= COMMAND

3-123. Purpose. The Z::= command is used to assign a character string value to Z::=.

3-124. Form. The form of the Z::= command is

```
Z::=
```

3-125. Description. When the Z::= command is entered in an interactive session, EDIT/3000 displays

```
ENTER Z::=
```

at which point you may enter the value to be assigned to Z::=. For example,

```
/Z::=
ENTER Z::=
"HP 3000"
```

In a batch job, EDIT/3000 assigns the record immediately following the Z::= command record to Z::=.

A series of EDIT/3000 commands, separated by semicolons, can be assigned to Z::= so that repetitious series of commands need not be repeated. For example,

```
/Z::=
ENTER Z::=
LIST LAST;FINDQ FIRST;FIND "EDIT/3000"
```

Once a value is assigned to Z::=, this value can be invoked in an EDIT/3000 cycle by entering Z:: (without the equals sign) or using Z:: as part of a command, and whatever has been assigned to Z::= will be substituted by EDIT/3000. Z:: is only effective as a command line or part of a command line, not to input data once in the ADD, INSERT, REPLACE, or MODIFY modes.

The default value for Z:: is "Z::". It can be reset to the default by entering CONTROL Y or "//" when the prompt ENTER Z::= is displayed.

To find the current value of Z::, enter

```
Q "Z::"
```

3-126. Limitations. No more than 255 characters may be assigned to Z::=. When an EDIT/3000 cycle begins, Z::= is undefined. You must define it by using the Z::= command, then entering the value to be assigned. The Z:: form of the command (which causes execution of the value assigned to Z::=) may not be used until you have defined Z::=.

You can change the current definition of Z::= by entering another Z::= command and assigning another definition. The old definition is lost. Any definition of Z::= is lost when EDIT/3000 terminates.

The Z:: command is not effective on the current record. That is, multiple commands on one line will use the previous value of Z::, even if Z::= is one of the commands on that line. For example, suppose Z:: contains the value FIRST, then FIND Z:: in the following command line locates the first column of the first line rather than the string "example":

```
/Z::=;FIND Z::
ENTER Z::=
"example"
    1      This is an example using EDITOP.
           "(1  )
```

Z:: =

3-127. Examples. The following examples illustrate the use of the Z::= command.

The first example sets Z::= equal to the string "(120-I)". The command CHANGE Z:: TO "I" IN ALL then searches the file JUST (a FORTRAN/3000 source program copied into the WORK file with the TEXT command) for all occurrences of the string "(120-I)". Each time the string is found, it is changed to "I".

The next example sets Z::= equal to

FINDQ FIRST;FIND "58:";CHANGE "58:" TO "28:"

Each time Z:: is entered, EDIT/3000 executes these three commands.

```
/TEXT JUST
/Z::=
ENTER Z::=
"(120-I)"
/CHANGE Z:: TO "I" IN ALL
  112       216  L=1-61
  126       236  L=1-59
  143       256  L=1-62
  150       266  L=1-61
  159       276  L=1-57
  177       296  L=1-59
  185       306  L=1-60
/Z::=
ENTER Z::=
FINDQ FIRST;FIND "58:";CHANGE "58:" TO "28:"
/Z::
   50       120  IF(BUFF![58:3].EQ."   ".AND.BUFF![57:1].NE." ")GOTO 130
                                (16 )!
   50       120  IF(BUFF![28:3].EQ."   ".AND.BUFF![57:1].NE." ")GOTO 130
/Z::
   51            IF(BUFF![59:3].EQ."   ".AND.BUFF![58:1].NE." ")GOTO 150
                                                (41 )!
   51            IF(BUFF![59:3].EQ."   ".AND.BUFF![28:1].NE." ")GOTO 150
```

The Z::= command can be used to alter commands in a WHILE loop during the execution of the loop. For instance, you can specify a different *line number* or *string* to be edited each time through the loop. In the following example, Z:: allows the user to select which lines are to be replaced for each iteration of the loop. The loop is terminated by entering CONTROL Y; in this example, CONTROL Y was entered just before the point where error #53 was displayed. (Make sure that Z:: has been cleared before entering the WHILE loop.)

```
/SET SHORT
/Z::=
ENTER Z::=
//
/WHILE FLAG
/   BEGIN
/      Z::=
/      R Z::/Z::+1,HOLD,NOW
/      END
ENTER Z::=
3
    3      LINE 3
    3      HOLD LINE 1
    4      LINE 4
    4      HOLD LINE 2
ENTER Z::=
9
    9      LINE 9
    9      HOLD LINE 1
   10      LINE 10
   10      HOLD LINE 2
ENTER Z::=
*53*TIME-OUT ON WHILE ITERATION
/L ALL
    1      LINE 1
    2      LINE 2
    3      HOLD LINE 1
    4      HOLD LINE 2
    5      LINE 5
    6      LINE 6
    7      LINE 7
    8      LINE 8
    9      HOLD LINE 1
   10      HOLD LINE 2
   11      LINE 11
   12      LINE 12
   13      LINE 13
   14      LINE 14
   15      LINE 15
   16      LINE 16
```

3-128. : COMMAND

3-129. Purpose. The : command is used to enter system commands from within EDIT/3000.

3-130. Form. The form of the : command is

:MPE command

For example,

```
:PURGE AFILE
```

3-131. Description. The : command allows the user to enter certain MPE commands without using the BREAK key. The colon indicates to EDIT/3000 that it should pass the rest of the record to the MPE operating system.

A : command may appear in a USE file or a WHILE block. In addition, Z:: may be used in any : command as a substitute for a parameter.

Any errors or warnings are reported as follows:

```
**** COMMAND ERROR  xx, yy
** COMMAND WARNING  xx, yy
```

where xx indicates the message number and yy the parameter in error. In addition, the MPE messages are displayed. The following example illustrates an error message:

```
/:LISTF FILE1
**** COMMAND ERROR  907,  0
NON-EXISTENT FILE,  (CIERR 907)
/
```

where 907 indicates the message number and 0 the parameter in error.

An exception to this format occurs in Series I machines, in which case the MPE message is not displayed.

3-132. **Limitations.**

Using the BREAK key during the execution of a : command causes a break in EDIT/3000. When RESUME is typed, execution of the command continues.

Valid commands are those which can be executed programmatically (see the *MPE Intrinsics Manual*, Section IV, the subsection entitled "Executing MPE Commands Programmatically," for a list of such commands). Commands which are valid MPE commands but which cannot be executed programmatically (such as the MPE :RUN command) cause error #12. Commands which are not valid MPE commands cause error #975 (except on Series I machines), as do any User Defined Commands. Note, however, that User Defined Commands are valid during a BREAK.

If used in a multiple command sequence, the : command must appear last (and only once). For example:

```
/T AFILE;L ALL;:SHOWTIME
```

If the : command appears earlier in the sequence of commands, only the commands up to and including the : command will be executed.

3-133. **Examples.** The following example illustrates the use of the : command. The MPE :FILE command is used to equate EDTLIST to a line printer other than the standard list device and the LIST ALL, OFFLINE command is used from within EDIT/3000 to obtain an offline listing of the WORK file. The MPE :TELL command is used to send a message to all users logged on in the STUDENT account. A BREAK from EDIT/3000 was not necessary.

```
/TEXT EDIT3
/LIST 1/6
    1       1-2.  WHAT IS EDIT/3000?
    2
    3       EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4       MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5       (MPE/3000) THAT IS USED TO CREATE AND
    6       MANIPULATE ASCII FILES.
/:FILE EDTLIST;DEV=FASTLP
/LIST ALL,OFFLINE
*** OFF LINE LISTING BEGUN. ***
/:TELL @,STUDENTS;CLASS RESUMES IN 20 MINUTES.
```

Table 3-2. Summary of Parameter Descriptions

colnum	an integer specifying the column number in a line.
filename	the MPE/3000 file name.
increment	the value by which line numbers are incremented.
integer	an integer in the range from 1 through 99999 when SET FORMAT = DEFAULT or from 1 through 999 when SET FORMAT = COBOL.
limit	an integer from 1 through 9999.
linenumber	the value assigned to line numbers.
newstring	a character string that replaces *oldstring*.
oldstring	a character string to be replaced in the WORK file.
position	a specific location in the WORK file
range	one or more positions in the WORK file.
rangelist	one or more ranges.
recnum	a logical record number in a file.
startcolumn	the starting column number of a portion of a record.
startline	the first line of a group of lines.
stopcolumn	the ending column number of a portion of a record.
stopline	the last line of a group of lines.
string	a group of characters.

EDIT/3000 ADVANCED COMMANDS

SECTION IV

In addition to the commands described in Section III, EDIT/3000 allows you to use seven advanced commands, as follows:

BEGIN
The BEGIN command is used as the first expression in a BEGIN-END pair to delimit a compound command.

END
Terminates a BEGIN-END pair.

NOT
Reverses the normal control, or "flag", within a BEGIN-END pair for execution of the next command.

OR
In a BEGIN-END pair, the OR command resets the flag to true if it is false, or skips the OR command and the following command if the flag is true.

PROCEDURE
Calls a logical procedure written previously and stored in a segmented library (SL) for execution by EDIT/3000.

WHILE
Causes EDIT/3000 to repeat the next two expressions (or one if a flag is declared) until the first expression fails (the flag is set to false) or until the SET TIME limit is exceeded.

YES
Restores normal control within a BEGIN-END pair (resets the flag to true).

When the advanced commands are used in a logical sequence with the basic EDIT/3000 commands, they form command blocks with recursive properties. In addition, logical procedures can be written in SPL/3000 (Systems Programming Language for the HP 3000 Computer System) or FORTRAN/3000, and these procedures can be called for execution with the PROCEDURE command.

WHILE

4-1. WHILE COMMAND

4-2. PURPOSE

The WHILE command causes EDIT/3000 to repeat the two commands following subject to these conditions:

1. While a WORK file condition specified in the EDIT/3000 command immediately following the WHILE command can be met (the flag is true).

2. Until the number of iterations specified by the SET TIME = *limit* command has been met (the default is 50 iterations). See Section III, paragraph 3-92 for a discussion of the SET command.

For example, the WHILE block

```
WHILE
    FIND "EDIT/3000"
    LIST *
```

will cause EDIT/3000 to search forward in the WORK file from the current location of the pointer, find the string "EDIT/3000", and list the line containing the string. Once the line has been listed, the pointer is located at the first position of the next line and EDIT/3000 will search forward again for another occurrence of "EDIT/3000". When the end of the WORK file is reached, the condition specified in the first command (the flag command) will fail ("EDIT/3000" can no longer be located) and the WHILE block will terminate execution. (If the number of iterations specified in a SET command is reached before the end of the WORK file, execution will terminate also.)

If FLAG is used in the WHILE command, EDIT/3000 will repeat a single command (the command following the WHILE command) until the flag is set to false or until the number of iterations allowed by the SET TIME option in effect is reached.

For example,

```
WHILE FLAG
    FIND "EDIT/3000" (+8)
```

will find the string "EDIT/3000", position the pointer to the last character in the string (number of characters in string −1) and search forward from there for another occurrence of the same string. When the end of the WORK file is reached, the flag will be set to false and the iterations will terminate.

> Note: The character position (+8) is necessary in the FIND command to position the pointer to the last character in the string. Otherwise, EDIT/3000 would find the string, the pointer would be located at the first character of the string, and the next iteration would find the same string. In the above example, 1 can be used instead of 8.

4-3. FORM

The form of the WHILE command is

```
WHILE [FLAG]
```

4-4. DESCRIPTION

As long as the condition specified in the first command following a WHILE command can be met, EDIT/3000 sets a flag *true* and the commands in the WHILE block continue to execute. When the condition specified in the first command cannot be met, EDIT/3000 sets the flag *false*, execution of the commands in the WHILE block terminates, and a "soft" error message is displayed. (A "soft" error results in an error message but does not terminate EDIT/3000 operation; a "hard" error results in an error message and terminates EDIT/3000 operation. See Section V for a discussion of errors and error messages.)

> Note: When a soft error does occur, only a NOT, a YES, an OR or an END command can be executed during the WHILE or USE modes. See paragraph 4-13 for a discussion of the NOT command and paragraph 4-25 for a discussion of the YES command.

4-5. LIMITATIONS

In a WHILE block, a *command* may be a basic command or a set of basic commands preceded by a BEGIN command and terminated by an END command. (The BEGIN-END pair enclosing other commands comprises a *compound* command.)

The commands in a WHILE block will execute only as long as the SET TIME = *limit* option in effect is not exceeded, even if the flag remains true. The default *limit* is 50 iterations. (If the flag is set to false by EDIT/3000, of course, execution will terminate before the SET TIME value is reached.) The number of BEGIN-END pairs allowed in a WHILE block is determined by the SET DEPTH = *limit* command. The default *limit* is 10.

If the SET SHORT command is used, display of WHILE block commands during execution is inhibited.

If data is embedded in a WHILE block (using SET BATCH), it will be treated as if it were a command; slashes, numbers, and preceding blanks are removed.

4-6. EXAMPLES

Two examples of the WHILE command are provided. See paragraph 4-7, "BEGIN-END COMMAND," paragraph 4-13, "NOT COMMAND," paragraph 4-19, "OR COMMAND," and paragraph 4-25, YES COMMAND" for examples of more complex WHILE blocks which use BEGIN-END pairs, and the NOT, OR, and YES commands.

The WORK file is shown below.

```
/S SHORT;T EDIT3;L ALL
    1       1-2.  WHAT IS EDIT/3000?
    2
    3       EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4       MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5       (MPE/3000) THAT IS USED TO CREATE AND
    6       MANIPULATE ASCII FILES.
    7
    8       CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9       LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10       REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11       MANIPULATED BY USING EDIT/3000 COMMANDS.
   12
   13       1-2.  EDIT/3000 FEATURES
   14
   15       WITH EDIT/3000, IT IS POSSIBLE TO
   16
   17       CREATE AND BUILD A NEW FILE (CALLED A WORK FILE)
   18       BY ENTERING COMMANDS AND LINES OF TEXT FROM THE
   19       STANDARD INPUT DEVICE OR FROM A SPECIAL DISC FILE
   20       CALLED THE HOLD FILE.
   21
   22       SAVE THE WORK FILE INTO A PERMANENT FILE CALLED
   23       THE TEXT FILE.
   24
   25       CALL THE TEXT FILE BACK INTO THE WORK FILE FOR
   26       FURTHER ADDITIONS AND/OR EDITING.
   27
   28       CHANGE ALL OCCURENCES OF A CHARACTER STRING WITH
   29       ONE COMMAND.
   30
   31       DELETE CHARACTERS AND LINES FROM THE WORK FILE.
```

In the following example, EDIT/3000 searches forward from the first position of the first line, finds the string "EDIT/3000" in line 1, and lists it. EDIT/3000 then searches forward again and finds and lists the string in lines 3, 11, 13, and 15. The string is not found again before the end of the WORK file is reached and the condition specified in the FIND "EDIT/3000" command fails. EDIT/3000 sets the flag false and prints the error message

21 STRING NOT FOUND BEFORE LIMIT AT DEPTH 2

Note that after the WHILE command is entered in a session, EDIT/3000 displays the / prompt character, then indents three positions for the next command. This is *depth 2*.

```
/FINDQ FIRST
/WHILE
/   FIND "EDIT/3000"
/   LIST*
  /FIND "EDIT/3000"
    1      1-2.  WHAT IS EDIT/3000?
                    (15 )↑
  /LIST*
    1      1-2.  WHAT IS EDIT/3000?
  /FIND "EDIT/3000"
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
           ↑(1  )
  /LIST*
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
  /FIND "EDIT/3000"
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
                          (22 )↑
  /LIST*
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
  /FIND "EDIT/3000"
   13      1-2.  EDIT/3000 FEATURES
            (7  )↑
  /LIST*
   13      1-2.  EDIT/3000 FEATURES
  /FIND "EDIT/3000"
   15      WITH EDIT/3000, IT IS POSSIBLE TO
           (6  )↑
  /LIST*
   15      WITH EDIT/3000, IT IS POSSIBLE TO
  /FIND "EDIT/3000"
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 2
 -/LIST*
```

The next example illustrates the use of the WHILE FLAG command. The WHILE FLAG command causes the next command (instead of the next two commands) to be repeated until the condition specified in the command fails and the flag is set to false by EDIT/3000.

The command following WHILE FLAG (FIND "EDIT/3000"(+8)) searches forward, finds the string in line 1, and lists it. The pointer is set to the *last* character of the string because of the (+8) character position parameter. (To find the last character in a string, the character position parameter must be set equal to *number of characters in string −1.*)

The string is found and listed again in lines 3, 11, 13, and 15.

```
/FINDQ FIRST
/WHILE FLAG
/   FIND "EDIT/3000"(+8)
    1      1-2.  WHAT IS EDIT/3000?
                          (23 )'
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
             (9  )'
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
                                  (30 )'
   13      1-2.  EDIT/3000 FEATURES
                   (15 )'
   15      WITH EDIT/3000, IT IS POSSIBLE TO
                   (14 )'
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 2
```

The following example illustrates a FIND command in a WHILE block that does not use the character position parameter to position the pointer to the last character in the string being searched for.

EDIT/3000 positions the pointer to the first character in the string. Thus, subsequent iterations of the FIND command locate the same occurrence of the string because the pointer location is not changed. The iterations were terminated by typing CONTROL Y.

```
/FINDQ FIRST
/WHILE FLAG
/   FIND "EDIT/3000"
  /FIND "EDIT/3000"
    1      1-2.  WHAT IS EDIT/3000?
                   (15 )'
  /FIND "EDIT/3000"
    1      1-2.  WHAT IS EDIT/3000?
                   (15 )'
  /FIND "EDIT/3000"
    1      1-2.  WHAT IS EDIT/3000?
                   (15 )'
  /FIND "EDIT/3000"
    1      1-2.  WHAT IS EDIT/3000?
      *21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 2
```

4-7. BEGIN-END COMMANDS

4-8. PURPOSE

The BEGIN-END commands are used to block a set of EDIT/3000 commands within a WHILE command block.

4-9. FORM

The form of the BEGIN-END commands is

```
B[EGIN][Q]
E[ND]
```

4-10. DESCRIPTION

A BEGIN-END pair can include any number of EDIT/3000 commands. These commands will execute repeatedly until the flag becomes false or until the SET TIME = *limit* option is reached.

For example,

```
FINDQ FIRST
WHILE
    FINDQ "PROGRAMMER"
    BEGIN
        CHANGE "PROGRAMMER" TO "PGMR" IN *
        LIST *
        END
```

will find all occurrences of "PROGRAMMER" in the WORK file, change this string to "PGMR" and list the line where the string was found. When the end of the WORK file is encountered, EDIT/3000 sets the flag to false and the iterations terminate.

If the BEGINQ form of the command is used, the soft error message (which occurs when the flag is set to false) will not be displayed.

4-11. LIMITATIONS

BEGIN-END pairs should be used only within a WHILE command block. The number of BEGIN-END pairs allowed in a single WHILE block is determined by the SET DEPTH = *limit* option in effect. The default limit is 10.

The commands between a BEGIN-END pair (within a WHILE block) will execute as long as the flag is true or until the SET TIME = *limit* value is not exceeded. The default *limit* is 50 iterations. If used outside a WHILE block, the commands between the BEGIN-END pair will execute only once.

4-12. EXAMPLES

The WORK file is shown below.

```
/S SHORT;T EDIT;L ALL
    1      1-2.  WHAT IS EDIT/3000?
    2
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    7
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
   12
   13      1-2.  EDIT/3000 FEATURES
   14
   15      WITH EDIT/3000, IT IS POSSIBLE TO
   16
   17      CREATE AND BUILD A NEW FILE (CALLED A WORK FILE)
   18      BY ENTERING COMMANDS AND LINES OF TEXT FROM THE
   19      STANDARD INPUT DEVICE OR FROM A SPECIAL DISC FILE
   20      CALLED THE HOLD FILE.
   21
   22      SAVE THE WORK FILE INTO A PERMANENT FILE CALLED
   23      THE TEXT FILE.
   24
   25      CALL THE TEXT FILE BACK INTO THE WORK FILE FOR
   26      FURTHER ADDITIONS AND/OR EDITING.
   27
   28      CHANGE ALL OCCURENCES OF A CHARACTER STRING WITH
   29      ONE COMMAND.
   30
   31      DELETE CHARACTERS AND LINES FROM THE WORK FILE.
```

BEGIN

The following example shows a BEGIN-END pair nested within another BEGIN-END pair. Note that in a session EDIT/3000 indents after the WHILE command and after each BEGIN command.

The first BEGIN-END pair finds "SUBSYSTEM" and changes it to "SUB-SYSTEM." The second BEGIN-END pair finds "FEATURES", changes it to "CAPABILITIES", and lists the entire WORK file.

The next iteration of the command FIND "SUBSYSTEM" fails, the flag is set to false, causing the error message shown to be displayed, and the iterations terminate.

```
/FIND" FIRST
/WHILE
/   FIND "SUBSYSTEM"
/   BEGIN
/      CHANGE "SUBSYSTEM" TO "SUB-SYSTEM"
/      BEGIN
/         FIND "FEATURES"
/         CHANGE "FEATURES" TO "CAPABILITIES"
/         LIST ALL
/         END
/      END
    3     EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
                    (16 )*
    3     EDIT/3000 IS A SUB-SYSTEM OF THE HP 3000
   13     1-2.  EDIT/3000 FEATURES
                    (17 )*
   13     1-2.  EDIT/3000 CAPABILITIES
    1     1-2.  WHAT IS EDIT/3000?
    2
    3     EDIT/3000 IS A SUB-SYSTEM OF THE HP 3000
    4     MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5     (MPE/3000) THAT IS USED TO CREATE AND
    6     MANIPULATE ASCII FILES.
    7
    8     CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9     LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10     REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11     MANIPULATED BY USING EDIT/3000 COMMANDS.
   12
   13     1-2.  EDIT/3000 CAPABILITIES
   14
   15     WITH EDIT/3000, IT IS POSSIBLE TO
   16
   17     CREATE AND BUILD A NEW FILE (CALLED A WORK FILE)
   18     BY ENTERING COMMANDS AND LINES OF TEXT FROM THE
   19     STANDARD INPUT DEVICE OR FROM A SPECIAL DISC FILE
   20     CALLED THE HOLD FILE.
   21
   22     SAVE THE WORK FILE INTO A PERMANENT FILE CALLED
   23     THE TEXT FILE
   24
   25     CALL THE TEXT FILE BACK INTO THE WORK FILE FOR
   26     FURTHER ADDITIONS AND/OR EDITING.
   27
   28     CHANGE ALL OCCURENCES OF A CHARACTER STRING WITH
   29     ONE COMMAND.
   30
   31     DELETE CHARACTERS AND LINES FROM THE WORK FILE.
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 2
```

The next example finds the string "EDIT/3000" and changes it to "HP 3000 TEXT EDITOR". The second BEGIN-END pair finds "FEATURES" and changes it to "CAPABILITIES". So that EDIT/3000 can locate additional "EDIT/3000" strings (between the first "EDIT/3000" and "FEATURES"), the FINDQ FIRST command sets the pointer back to the first line in the WORK file.

EDIT/3000 then locates "EDIT/3000" in lines 3, 11, 13, and 15, and changes all occurrences to "HP 3000 TEXT EDITOR".

When the condition specified in the first FIND command fails, the iterations terminate and the error message is displayed.

```
/FINDQ FIRST
/WHILE
/   FINDQ "EDIT/3000"
/   BEGIN
/      CHANGE "EDIT/3000" TO "HP 3000 TEXT EDITOR"
/      BEGIN
/         FIND "FEATURES"
/         CHANGE "FEATURES" TO "CAPABILITIES"
/         FINDQ FIRST
/         END
/      END
    1      1-2.  WHAT IS HP 3000 TEXT EDITOR?
   13      1-2.  EDIT/3000 FEATURES
                      (17 )?
   13      1-2.  EDIT/3000 CAPABILITIES
    3      HP 3000 TEXT EDITOR IS A SUBSYSTEM OF THE HP 3000
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 4
   11      MANIPULATED BY USING HP 3000 TEXT EDITOR COMMANDS.
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 4
   13      1-2.  HP 3000 TEXT EDITOR CAPABILITIES
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 4
   15      WITH HP 3000 TEXT EDITOR, IT IS POSSIBLE TO
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 4
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 2
```

4-13. NOT COMMAND

4-14. PURPOSE

The NOT command reverses the flag setting after the command immediately following the NOT command is executed.

4-15. FORM

The form of the NOT command is

N[OT]

4-16. DESCRIPTION

The NOT command reverses the flag setting after the command following it is executed.

For example,

```
FINDQ FIRST
WHILE
     FIND "PROGRAMMER"(+10)
     BEGIN
          NOT
          FINDQ "PROBLEMS"/*(+10)
          LIST *
          END
```

lists a line only if the string "PROBLEMS" is *not* found within 10 character positions after the string "PROGRAMMER". In other words, the NOT command reverses the flag to false after the command FINDQ "PROBLEMS"/(+10), causing the LIST * command to be skipped. The entire WHILE block, however, repeats until the command FIND "PROGRAMMER" (+10) (the *first* flag) fails or until the SET TIME = *limit* value in effect is reached.

4-17. LIMITATIONS

The NOT command affects only the command immediately following it and can be used *only* in a WHILE block.

4-18. EXAMPLES

The WORK file is shown below.

```
/S SHORT;T EDIT3;L ALL
    1      1-2.  WHAT IS EDIT/3000?
    2
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    7
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
   12
   13      1-3.  EDIT/3000 FEATURES
   14
   15      WITH EDIT/3000, IT IS POSSIBLE TO
   16
   17      CREATE AND BUILD A NEW FILE (CALLED A WORK FILE)
   18      BY ENTERING COMMANDS AND LINES OF TEXT FROM THE
   19      STANDARD INPUT DEVICE OR FROM A SPECIAL DISC FILE
   20      CALLED THE HOLD FILE.
   21
   22      SAVE THE WORK FILE INTO A PERMANENT FILE CALLED
   23      THE TEXT FILE.
   24
   25      CALL THE TEXT FILE BACK INTO THE WORK FILE FOR
   26      FURTHER ADDITIONS AND/OR EDITING.
   27
   28      CHANGE ALL OCCURENCES OF A CHARACTER STRING WITH
   29      ONE COMMAND.
   30
   31      DELETE CHARACTERS AND LINES FROM THE WORK FILE.
```

In the following example, EDIT/3000 finds the string "EDIT/3000" in line 1, lists the line, then searches forward 20 character positions from the first character following the string (+9). The string "HP 3000" is not found within this limit, the error message

21 STRING NOT FOUND BEFORE LIMIT AT DEPTH 4

is displayed, and EDIT/3000 executes the LIST * command, displaying the line again.

The next occurrence of "EDIT/3000" *does* have the string "HP 3000" following it within 20 character positions. EDIT/3000 skips the LIST * command and the line is *not* listed.

The iterations continue until the FIND "EDIT/3000"(+9) command fails and the flag is set to false.

```
/FIND? FIRST
/WHILE
/   FIND "EDIT/3000"(+9)
/   BEGIN
/      NOT
/      FIND? "HP 3000"/*(+20)
/      LIST *
/      END
    1      1-2.  WHAT IS EDIT/3000?
                                (24 )!
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 4
    1      1-2.  WHAT IS EDIT/3000?
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
                 (11 )!
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
                                       (32 )!
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 4
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
   13      1-2.  EDIT/3000 FEATURES
                        (17 )!
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 4
   13      1-2.  EDIT/3000 FEATURES
   15      WITH EDIT/3000, IT IS POSSIBLE TO
                      (15 )!
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 4
   15      WITH EDIT/3000, IT IS POSSIBLE TO
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 2
```

4-19. OR COMMAND

4-20. PURPOSE

The OR command sets the flag true if the flag is false. If the flag is true, both the OR command and the command immediately following it are skipped.

4-21. FORM

The form of the OR command is

O[R]

4-22. DESCRIPTION

In the following example

```
FINDQ FIRST
WHILE
     FIND "EDIT/3000"(+9)
     BEGIN
          FIND "HP 3000"/*(+20)
          OR
          BEGIN
               CHANGE "HP3000" TO "HP 3000"
               END
END
```

the command FIND "EDIT/3000" (+9) finds the string "EDIT/3000". Next, the BEGIN and FIND "HP 3000"/*(+20) commands locate the string "HP 3000" if it is within 20 character positions of the end of the first string. If the string "HP 3000" is found, the flag is true and the OR command and the next BEGIN-END pair are skipped. (If the command immediately following an OR command is a BEGIN command to start a series of commands within a BEGIN-END pair, the entire BEGIN-END block (including any other BEGIN-END blocks it may contain is skipped.)

If the flag is false, the OR command sets it to true and the BEGIN-END block is executed, changing "HP3000" to "HP 3000".

4-23. LIMITATIONS

The OR command should be used only with a WHILE block.

4-24. EXAMPLES

The WORK file is shown below.

```
/S SHORT;T EDIT;;L ALL
    1      1-0.  WHAT IS EDIT/3000?
    2
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    7
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
   12
   13      1-0.  EDIT/3000 FEATURES
   14
   15      WITH EDIT/3000, IT IS POSSIBLE TO
   16
   17      CREATE AND BUILD A NEW FILE (CALLED A WORK FILE)
   18      BY ENTERING COMMANDS AND LINES OF TEXT FROM THE
   19      STANDARD INPUT DEVICE OR FROM A SPECIAL DISC FILE
   20      CALLED THE HOLD FILE.
   21
   22      SAVE THE WORK FILE INTO A PERMANENT FILE CALLED
   23      THE TEXT FILE.
   24
   25      CALL THE TEXT FILE BACK INTO THE WORK FILE FOR
   26      FURTHER ADDITIONS AND/OR EDITING.
   27
   28      CHANGE ALL OCCURENCES OF A CHARACTER STRING WITH
   29      ONE COMMAND.
   30
   31      DELETE CHARACTERS AND LINES FROM THE WORK FILE.
```

OR

The following example illustrates the OR command usage.

The first FIND command after the WHILE finds the string "EDIT/3000" and positions the pointer one character beyond the last character in the string.

If the second FIND command fails, the flag is set to false. The OR command sets the flag to true, the command LIST * is executed and the line is displayed. If the second FIND command executes ("HP 3000" is found within 20 character positions), the flag is true and the OR command and the command following it (LIST *) are skipped. The line is not displayed in this case.

The iterations continue until the first FIND command fails and EDIT/3000 sets its flag to false.

```
/FINDO FIRST
/WHILE
/   FIND "EDIT/3000"(+9)
/   BEGIN
/      FIND "HP 3000"/*(+20)
/      OR
/      LIST *
/      END
    1      1-2.  WHAT IS EDIT/3000?
                              (24 )↑
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 3
    1      1-2.  WHAT IS EDIT/3000?
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
                (11 )↑
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
                                      (33 )↑
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
                                      (32 )↑
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 3
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
   13      1-2.  EDIT/3000 FEATURES
                        (17 )↑
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 3
   13      1-2.  EDIT/3000 FEATURES
   15      WITH EDIT/3000, IT IS POSSIBLE TO
                     (15 )↑
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 3
   15      WITH EDIT/3000, IT IS POSSIBLE TO
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 2
```

4-25. YES COMMAND

4-26. PURPOSE

The YES command sets the flag true if it is false.

4-27. FORM

The form of the YES command is

Y[ES]

4-28. DESCRIPTION

The YES command has no effect other than setting the flag true if it is false. (For example, the YES command does not cause the next command to be skipped.) In the following example,

```
WHILE
    FIND "PROGRAMMING" (+10)
    BEGIN
        FIND "PROBLEMS"/*(+10)
        CHANGE "PROBLEMS" TO "PROB" IN *
        YES
        LIST *
        END
```

EDIT/3000 lists a line if the first FIND command (FIND "PROGRAMMING"(+10)) succeeds. The line is either one that contains "PROGRAMMING" (as found by the first FIND command) and "PROBLEMS" within 10 character positions of "PROGRAMMING", or one that contains "PROGRAMMING" and does not contain "PROBLEMS" within 10 character positions of "PROGRAMMING" (the second FIND command failed). In either case, the line is listed. If the second FIND command fails, setting its flag to false, the YES command sets it back to true and the LIST * command is executed. Thus, no commands are skipped. The entire WHILE block repeats until the first FIND command fails or until the SET TIME = *limit* value in effect is reached.

4-29. LIMITATIONS

The YES command is not limited to a WHILE block, and can be used to reset soft errors in a Batch job.

4-30. EXAMPLES

The WORK file is shown below.

```
/S SHORT;T EDIT3;L ALL
    1      1-2.  WHAT IS EDIT/3000?
    2
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    7
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
   12
   13      1-2.  EDIT/3000 FEATURES
   14
   15      WITH EDIT/3000, IT IS POSSIBLE TO
   16
   17      CREATE AND BUILD A NEW FILE (CALLED A WORK FILE)
   18      BY ENTERING COMMANDS AND LINES OF TEXT FROM THE
   19      STANDARD INPUT DEVICE OR FROM A SPECIAL DISC FILE
   20      CALLED THE HOLD FILE.
   21
   22      SAVE THE WORK FILE INTO A PERMANENT FILE CALLED
   23      THE TEXT FILE.
   24
   25      CALL THE TEXT FILE BACK INTO THE WORK FILE FOR
   26      FURTHER ADDITIONS AND/OR EDITING.
   27
   28      CHANGE ALL OCCURENCES OF A CHARACTER STRING WITH
   29      ONE COMMAND.
   30
   31      DELETE CHARACTERS AND LINES FROM THE WORK FILE.
```

YES

The first FIND command in the next example finds the string "EDIT/3000" in line 1. The second FIND command fails, EDIT/3000 sets its flag to false and displays an error message. The YES command sets the flag to true and the LIST * command executes, displaying the line.

If the second FIND command is successful, the LIST * command also is executed. Thus, the YES command allows the command following it to be executed by setting a false flag true.

The iterations continue until the first FIND command fails, setting its flag to false.

```
/FIND FIRST
/WHILE
/   FIND "EDIT/3000"(+9)
/   BEGIN
/      FIND "HP 3000"/*(+20)
/      CHANGE "HP 3000" TO "HEWLETT-PACKARD 3000" IN *
/      YES
/      LIST *
/      END
    1     1-2.  WHAT IS EDIT/3000?
                              (24 )↑
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 3
    1     1-2.  WHAT IS EDIT/3000?
    3     EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
               (11 )↑
    3     EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
                                    (33 )↑
    3     EDIT/3000 IS A SUBSYSTEM OF THE HEWLETT-PACKARD 3000
    3     EDIT/3000 IS A SUBSYSTEM OF THE HEWLETT-PACKARD 3000
   11     MANIPULATED BY USING EDIT/3000 COMMANDS.
                                   (32 )↑
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 3
   11     MANIPULATED BY USING EDIT/3000 COMMANDS.
   13     1-2.  EDIT/3000 FEATURES
                    (17 )↑
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 3
   13     1-2.  EDIT/3000 FEATURES
   15     WITH EDIT/3000, IT IS POSSIBLE TO
                  (15 )↑
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 3
   15     WITH EDIT/3000, IT IS POSSIBLE TO
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 2
```

YES

PROCEDURE

4-31. PROCEDURE COMMAND

4-32. PURPOSE

The PROCEDURE command calls and executes a procedure written previously in SPL/3000 (Systems Programming Language for the HP 3000 Computer System), FORTRAN/3000, or any other language which accepts parameters as specified and which produces a logical result and is stored in an MPE/3000 Segmented Library (SL). See the *MPE Segmenter Subsystem Reference Manual* for a description of SLs.

4-33. FORM

The form of the PROCEDURE command is

$$\text{P[ROCEDURE]}\ \left[\textit{pname}\ \left[\begin{matrix}\text{G}\\ \text{,P}\\ \text{S}\end{matrix}\right]\ \left[\textit{,rangelist}\right]\right]$$

where

pname
is the name (or *entry point*) assigned to the procedure in the SL.

G, P, S
specify which SL library is to be searched for the procedure. G specifies the *group* library; P the *public* library; and S the *system* library. See the *MPE Segmenter Subsystem Reference Manual* for a discussion of the group, public, and system libraries. Default for the SL library is S.

rangelist
is the same as that described for basic EDIT/3000 commands in Section III *except* that a procedure can affect only lines of the WORK file, even if *rangelist* includes a *position* or *character string*. The default for the *rangelist* is the line at which the cursor is currently positioned in the WORK file.

4-34. DESCRIPTION

The PROCEDURE command with *no* parameters specified will call the last loaded procedure, with the line at which the cursor is currently positioned.

For EDIT/3000 to be able to call a procedure, certain conventions must be followed when the procedure is written. The procedure must be logical. The order and type of the parameters used in the procedure are as shown in the following SPL/3000 and FORTRAN/3000 examples;

A sample header for an SPL/3000 logical procedure:

```
LOGICAL PROCEDURE SAMPLE (STRING, LENGTH, NUMBER, SPACE);
BYTE ARRAY STRING, NUMBER;
INTEGER LENGTH;
ARRAY SPACE;
```

A sample header for a FORTRAN/3000 logical function:

```
LOGICAL FUNCTION SAMPLE (STRING, LENGTH, NUMBER, SPACE)
INTEGER LENGTH, SPACE(20)
CHARACTER STRING*80, NUMBER*8
```

The procedure must be written to conform to the following requirements:

1. Be a type logical procedure (if written in SPL/3000) or a type logical function (if written in FORTRAN/3000).

2. Include these four parameters, in the order shown:
   STRING, LENGTH, NUMBER, SPACE

   where

   STRING
   is a byte array (in SPL/3000) or a character array (in FORTRAN/3000) that contains one line of the WORK file each time EDIT/3000 calls the procedure. (The line will be within the range specified in *rangelist*. Maximum size allowed is 255 bytes.)

   LENGTH
   is an integer value specifying the length of the current line being passed by STRING.

   NUMBER
   is a byte array (in SPL/3000) or a character array (in FORTRAN/3000) that contains the line number of the current line being passed by string. Maximum size is 8 bytes.

   SPACE
   is a scratch array of 20 words maintained by EDIT/3000 for the procedure's use.

   SPACE is initialized to null characters (binary zeros) when EDIT/3000 is first entered; thereafter, the SPACE array is not initialized for each call of the specified procedure as it iterates through the *rangelist* of the PROCEDURE command. Consequently, this array may be used to "remember" information in successive calls to the procedure within the rangelist specified.

   These four parameters must be declared by the procedure whether they are used or not.

3. The procedure must correct the parameter LENGTH if it shortens or lengthens the line in STRING and pass this value back to EDIT/3000. If LENGTH is negative, the line is deleted.

4. The procedure must return logical TRUE for each successful execution; EDIT/3000 then calls the procedure again for the next line in *rangelist*.

5. The procedure must return logical FALSE for an unsuccessful operation. EDIT/3000 then displays a "soft" error message and discontinues further calls to the procedure. Any lines in *rangelist* not yet processed by the procedure remain as they were before the PROCEDURE command was used.

4-35. Limitations. A PROCEDURE command can call only *logical* procedures written in a language which accepts parameters as specified and which produces a logical result.

The procedure must reside in a segmented library.

In addition to the array SPACE, the procedure called by the PROCEDURE command may access the array USERSPACE, as described for the User Interface Procedures (see paragraph 6-1). USERSPACE is a 10 word array which contains the PROCEDURE command's input and output MPE file numbers in the first two words. In order to access this array, the procedure must use the array SPACE indexed from -10 to -1.

4-36. EXAMPLES

A FORTRAN/3000 logical function (stored on disc under the name EDITPROC) to find lines containing ASCII blanks is shown below. The function is compiled using the MPE/3000 :FORTRAN command.

See the *MPE Commands Reference Manual* for a discussion of the :FORTRAN command.

```
:FORTRAN EDITPROC

PAGE 0001    HP32102A.01.4


00001000  $CONTROL USLINIT,SEGMENT=SEG1
00002000        LOGICAL FUNCTION FINDBLANKLINE(STRING,LENGTH,NUMBER,
00003000       *SPACE)
00004000        CHARACTER*255 STRING,MESSAGE*28,NUMBER*8
00005000        INTEGER SPACE(20),LMESSAGE(14)
00006000        EQUIVALENCE (LMESSAGE,MESSAGE)
00007000        IF(LENGTH.EQ.0)GOTO 20
00008000        DO 10 I=1,LENGTH
00009000        IF(STRING(I:1).NE." ")GOTO 30
00010000   10   CONTINUE
00011000   20   MESSAGE=" LINE NUMBER          BLANK"
00012000        MESSAGE(14:8)=NUMBER
00013000        CALL PRINT(LMESSAGE,\-28\,\0\)
00014000        FINDBLANKLINE=.FALSE.
00015000        RETURN
00016000   30   FINDBLANKLINE=.TRUE.
00017000        RETURN
00018000        END


****  NO ERRORS,  NO WARNINGS;   PROGRAM UNIT COMPILED  ****
COMPILATION TIME    0.950 SECONDS    ELAPSED TIME    87.075 SECONDS
TOTAL COMPILATION TIME  0:00:02
TOTAL ELAPSED TIME      0:01:41

 END OF COMPILE
```

Once the function has been compiled into a User Subprogram Library (USL, see the *MPE Segmenter Subsystem Reference Manual),* the Segmenter subsystem is accessed with the MPE/3000 :SEGMENTER command as shown below.

The Segmenter command –BUILDSL is used to create an SL file named SL, with 300 records maximum and one disc extent.

The –USL command is used to identify the USL file ($OLDPASS) which contains the function procedure to be added to the SL file.

The segment name containing the procedure (SEG1 — see the first statement of the FORTRAN/3000 function) is added to the SL file using the –ADDSL command.

The –LISTSL command shows the segment added and the name of entry points in this segment (FINDBLANKLINE is the only entry point — or procedure name — in this segment). Note that the SL file name contains PUB, meaning that the *public account* library will have to be searched for this procedure. The -EXIT command terminates Segmenter operation.

```
:SEGMENTER

SEGMENTER SUBSYSTEM (C.0)
-BUILDSL SL.300,1
-USL $OLDPASS
-ADDSL SEG1
-LISTSL

SL FILE SL.PUB.GOODWIN

SEGMENT   0 SEG1            LENGTH   120

   ENTRY POINTS     CHECK CAL STT  ADR
   FINDBLANKLINE      3    C    1    0

   EXTERNALS        CHECK STT SEG
   BLANKFILL'         0     4   ?
   CCHRLPB'           0     3   ?
   PRINT              3     2   ?

1

USED                 1600          AVAILABLE          111200
-EXIT

 END OF SUBSYSTEM
```

Once EDIT/3000 is accessed using the MPE/3000 :EDITOR command, the TEXT file is copied into the WORK file with the T[EXT] EDIT3 command and listed with the L[IST] ALL command.

```
/S SHORT;T EDIT3;L ALL
    1      1-2.  WHAT IS EDIT/3000?
    2
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    7
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
   12
   13      1-2.  EDIT/3000 FEATURES
   14
   15      WITH EDIT/3000, IT IS POSSIBLE TO
   16
   17      CREATE AND BUILD A NEW FILE (CALLED A WORK FILE)
   18      BY ENTERING COMMANDS AND LINES OF TEXT FROM THE
   19      STANDARD INPUT DEVICE OR FROM A SPECIAL DISC FILE
   20      CALLED THE HOLD FILE.
   21
   22      SAVE THE WORK FILE INTO A PERMANENT FILE CALLED
   23      THE TEXT FILE.
   24
   25      CALL THE TEXT FILE BACK INTO THE WORK FILE FOR
   26      FURTHER ADDITIONS AND/OR EDITING.
   27
   28      CHANGE ALL OCCURENCES OF A CHARACTER STRING WITH
   29      ONE COMMAND.
   30
   31      DELETE CHARACTERS AND LINES FROM THE WORK FILE.
```

The WHILE block uses the procedure FINDBLANKLINE to locate and delete blank lines as follows:

1. The FINDQ * command locates the current line in the WORK file.

2. The command PROCEDURE FINDBLANKLINE,P,* calls the procedure FINDBLANKLINE from the public library (P) to determine if the current line is blank.

3. If the current line is blank, the procedure returns logical FALSE and EDIT/3000 displays the message

   19 FALSE RETURN FROM EDITOR PROCEDURE AT DEPTH 3

   The OR command sets the flag to true and the DELETEQ * command deletes the current line from the WORK file. The message

   NUMBER OF LINES DELETED = 1

   is displayed.

4. If the current line is not blank, the procedure returns logical TRUE, the OR command and the command following it (DELETEQ *) are skipped, and the line is not deleted.

5. Once the last line of the WORK file is reached, the FINDQ * command continues to locate this line until the number of iterations specified in the SET TIME = *limit* is reached (default *limit* is 50 iterations).

6. The LIST ALL command verifies that the lines have been deleted.

```
/FIND? FIRST
/WHILE
/   FIND? *
/   BEGIN
/      PROCEDURE FINDBLANKLINE,?,*
/      ??
/      DELETE *
/      END
 LINE NUMBER      9     BLANK
*19*FALSE RETURN FROM EDITOR PROCEDURE
AT DEPTH 3
NUMBER OF LINES DELETED = 1
 LINE NUMBER      7     BLANK
*19*FALSE RETURN FROM EDITOR PROCEDURE
AT DEPTH 3
NUMBER OF LINES DELETED = 1
 LINE NUMBER     10     BLANK
*19*FALSE RETURN FROM EDITOR PROCEDURE
AT DEPTH 3
NUMBER OF LINES DELETED = 1
 LINE NUMBER     14     BLANK
*19*FALSE RETURN FROM EDITOR PROCEDURE
AT DEPTH 3
NUMBER OF LINES DELETED = 1
 LINE NUMBER     16     BLANK
*19*FALSE RETURN FROM EDITOR PROCEDURE
AT DEPTH 3
NUMBER OF LINES DELETED = 1
 LINE NUMBER     21     BLANK
*19*FALSE RETURN FROM EDITOR PROCEDURE
AT DEPTH 3
NUMBER OF LINES DELETED = 1
 LINE NUMBER     24     BLANK
*19*FALSE RETURN FROM EDITOR PROCEDURE
AT DEPTH 3
NUMBER OF LINES DELETED = 1
 LINE NUMBER     27     BLANK
*19*FALSE RETURN FROM EDITOR PROCEDURE
AT DEPTH 3
NUMBER OF LINES DELETED = 1
 LINE NUMBER     30     BLANK
*19*FALSE RETURN FROM EDITOR PROCEDURE
AT DEPTH 3
NUMBER OF LINES DELETED = 1
*51*L
TIME-OUT ON WHILE ITERATION
/LIST ALL
    1      1-2.  WHAT IS EDIT/3000?
    3      EDIT/3000 IS A SUBSYSTEM OF THE HP 3000
    4      MULTIPROGRAMMING EXECUTIVE OPERATING SYSTEM
    5      (MPE/3000) THAT IS USED TO CREATE AND
    6      MANIPULATE ASCII FILES.
    8      CHARACTERS, STRINGS OF CHARACTERS, OR ENTIRE
    9      LINES OF CHARACTERS CAN BE INSERTED, DELETED,
   10      REPLACED, MODIFIED, SEARCHED FOR, AND OTHERWISE
   11      MANIPULATED BY USING EDIT/3000 COMMANDS.
   13      1-2.  EDIT/3000 FEATURES
   15      WITH EDIT/3000, IT IS POSSIBLE TO
   17      CREATE AND BUILD A NEW FILE (CALLED A WORK FILE)
   18      BY ENTERING COMMANDS AND LINES OF TEXT FROM THE
   19      STANDARD INPUT DEVICE OR FROM A SPECIAL DISC FILE
   20      CALLED THE HOLD FILE.
   22      SAVE THE WORK FILE INTO A PERMANENT FILE CALLED
   23      THE TEXT FILE
   25      CALL THE TEXT FILE BACK INTO THE WORK FILE FOR
   26      FURTHER ADDITIONS AND/OR EDITING.
   28      CHANGE ALL OCCURENCES OF A CHARACTER STRING WITH
   29      ONE COMMAND.
   31      DELETE CHARACTERS AND LINES FROM THE WORK FILE.
/E
CLEAR? Y

 END OF SUBSYSTEM
```

MESSAGES AND RECOVERY PROCEDURES

SECTION V

An error indication is produced whenever an error occurs during EDIT/3000 operation.

In an interactive session or batch job, EDIT/3000 normally displays the error number followed by the error message. For example,

```
*14* INVALID LINE NUMBER
```

In an interactive session, the user has the option to request a different error reporting scheme by initiating EDIT/3000 with the MPE command

```
:RUN EDITOR.PUB.SYS; PARM = 8
```

In this case, only the error number is displayed. To skip the error message, merely press RETURN. To request the complete error message, type any character key except RETURN. For example,

```
*14*L

INVALID LINE NUMBER
```

To skip the error message, merely press RETURN.

This option is not available in a batch job. The complete error message is displayed by EDIT/3000 on the standard list device.

In case of a file system error, a tombstone and the file system error message are printed in addition to the EDIT/3000 message. For example:

```
/T FILE1
+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
|   ERROR NUMBER: 52   RESIDUE:0                  |
|   BLOCK NUMBER: 0            NUMREC: 0          |
+-------------------------------------------------+
*23*FAILURE TO OPEN TEXT FILE    (52)
NONEXISTENT PERMANENT FILE   (FSERR 52)
/
```

Three types of errors can occur: *warn*, *soft* and *hard*. *Warn* errors do not cause an EDIT/3000 command function to terminate; they are solely for the user's information. *Soft* errors do cause the command to fail, although EDIT/3000 operation does not terminate. *Hard* errors result in termination of EDIT/3000 operation.

Table 5-1 contains a list of all error messages that can occur during EDIT/3000 operation. The first column contains the error number. The type of error (warn, soft or hard) is shown in the second column. The error message is contained in the third column and the probable cause is shown in the fourth column. The last column contains corrective action information which will help you in initiating recovery procedures.

Each error that relates to an MPE/3000 file intrinsic is preceded by a File Information Display. See the *MPE Intrinsics Reference Manual* for descriptions of these displays. These types of errors contain the MPE/3000 *errorcode* in parentheses after the error message.

Table 5-1. Error Messages

ERROR NO.	TYPE	MESSAGE	CAUSE	CORRECTIVE ACTION
1	HARD	INVALID COMMAND NAME	Command name incorrect.	Re-enter the command.
2	HARD	INVALID OPTION	Command parameter incorrect.	Re-enter the command.
3	HARD	MISSING PARAMETER	Command parameter needed.	Re-enter the command.
4	HARD	FILE NOT ACCESSIBLE *(errorcode)*	File requested is not accessible due to the *errorcode* reported.	Check the *errorcode* in the *MPE Intrinsics Reference Manual*.
5	HARD	FILE NOT TYPE ASCII	File requested is a binary file.	Terminate EDIT/3000 operation, then run a program such as the HP 3000 File Copier to create an ASCII version of the file.
6	HARD	CHARACTER ADJUSTMENT RUNS OFF OF FILE	A character string is too long.	Verify LEFT, RIGHT, LENGTH options in effect and change accordingly, or change the string.
7	HARD	READ ERROR ON TEXT FILE *(errorcode)*	File cannot be read due to the *errorcode* shown.	Check the *errorcode* as shown in the *MPE Intrinsics Reference Manual* and try again. If error persists, copy error-free portions of TEXT file with TEXT command and use ADD command to correct the portion that is not read.
8	SOFT	ADJUSTMENT RUNS OFF OF FILE	Command would move pointer off either end of WORK file.	Verify FIRST and LAST options in effect and change accordingly or correct command.
9	HARD	SYNTAX ERROR IN TEXT POSITION EXPRESSION	A *position* parameter is incorrect.	Re-enter the command.
10	SOFT	ABSOLUTE COLUMN POSITION OUT OF RANGE	A *column* parameter is outside the LEFT, RIGHT margins.	Verify LEFT, RIGHT options in effect and change accordingly or re-enter the command.
11	SOFT	POSITION NOT FOUND	A *position* does not exist in the WORK file	List the lines where the *position* should be and re-enter the command accordingly.
12	HARD	OVERSIZED LINE *(linenumber)*	The line exceeds the LEFT, RIGHT, or LENGTH options in effect.	Correct the LEFT, RIGHT, or LENGTH options and re-enter the command.
13	HARD	TIMEOUT ERROR	WHILE command construct wrong.	Re-enter the command.
14	HARD	INVALID LINE NUMBER	A *linenumber* parameter is wrong under the SET FORMAT option in effect or contains an incorrect character.	Correct the parameter or change the SET FORMAT option. Text the file in UNNUMBERED, examine and correct; then KEEP, UNNUMBERED.
15	SOFT	COMMAND WILL NOT REPLACE OR INTERLEAVE LINES	A *linenumber* parameter in an ADD, GATHER, or JOIN command references a line that already exists.	List the line number, then re-enter the command with correct *linenumber* parameter.
16	WARN	WARNING - LINE *linenumber* EXTENDS PAST *colnum*	A line extends beyond the RIGHT margin option in effect.	Change the RIGHT or LENGTH options, or correct the command, or modify the line

Table 5-1. Error Messages (Continued)

ERROR NO.	TYPE	MESSAGE	CAUSE	CORRECTIVE ACTION
17	HARD	UNMATCHED END COMMAND	A BEGIN-END pair is incorrect.	Correct the BEGIN-END command pairs.
18	HARD	INSUFFICIENT LINE NUMBERS OR WORK SPACE – COMMAND NOT PERFORMED	Not enough line numbers are available in the WORK file for a GATHER or JOIN where a TO *linenumber* parameter has been specified.	Renumber the WORK file lines to accommodate the command, or re-enter the command with a smaller increment.
19	SOFT	FALSE RETURN FROM EDITOR PROCEDURE	A procedure has returned a value of false.	See PROCEDURE command in Section IV.
20	HARD	UNDELIMITED STRING	Incorrect delimiters enclosing a string.	Re-enter the command.
21	SOFT	STRING NOT FOUND BEFORE LIMIT	A string was not found within the *range* specified or does not exist in the WORK file.	Use the FIND FIRST command, then try again. Verify that string is correct.
22	SOFT	LINE DOES NOT EXIST	The line does not exist in the WORK file.	List the line before and the line after the line, with NO TEXT. ADD *linenumber*, if necessary.
23	HARD	FAILURE TO OPEN TEXT FILE *(errorcode)*	File requested with a TEXT command was not opened due to the *errorcode* reported.	Check the *error* code in the *MPE Intrinsics Reference Manual.*
24	HARD	RECORD SIZE TOO LARGE ***(length)***	The ***length*** of the file is greater than 255 bytes.	None within EDIT/3000.
25	HARD	SCRATCH FILE OPEN FAILURE *(errorcode)*	Work file could not be opened due to *errorcode.*	See the *MPE Intrinsics Reference Manual.*
26	HARD	GETDSEG FAILURE *reason (see below)* SEGMENT OF SAME IDENTITY ALREADY EXISTS ILLEGAL LENGTH SPECIFIED TOO MANY DATA SEGMENTS INSUFFICIENT STORAGE FOR SIZE	Stated in the *reason* appended to message.	See the ***MPE Commands Reference Manual.*** System reconfiguration may be necessary. Extra data segments are necessary only when texting in numbered files. Text in the file UNNUMBERED.
27	SOFT	AN E.O.D. RECEIVED BEFORE TEXT FILE END	An :EOD record was encountered before the physical end-of-file occured.	None - warning only.
28	HARD	SCRATCH FILE READ ERROR *(errorcode)*	WORK file not read due to *errorcode* reported.	See the *MPE Intrinsics Reference Manual.*
29	HARD	INVALID LINENUMBER ON NUMBERED TEXT OR JOIN FILE - RECORD *(number)*	Sequence numbers incorrect for the requested function.	**Use following commands: TEXT filename, UNN or JOIN filename, UNN**
30	HARD	DMOVOUT ERROR	Data segment management error.	Notify the System Manager.
31	HARD	SCRATCH FILE WRITE ERROR *(errorcode)*	WORK file could not be written to due to *errorcode.*	See the *MPE Intrinsics Reference Manual.*
32	HARD	DMOVIN ERROR	Data segment management error.	Notify the System Manager.
33	HARD	INVALID COLUMN RANGE (FIRST FOLLOWS SECOND)	The *range* parameter is not in ascending order.	Correct the *range* parameter of the command.

Table 5-1. Error Messages (Continued)

ERROR NO.	TYPE	MESSAGE	CAUSE	CORRECTIVE ACTION
34	WARN	WARNING - HOLD FILE IS EMPTY	HOLD file is empty.	None - warning, only.
35	HARD	FAILURE TO LOAD LIBRARY PROCEDURE	The procedure requested with a PROCEDURE command could not be loaded.	Check the SL library. See Section IV, PROCEDURE command.
36	HARD	SCRATCH FILE IS FULL. KEEP, THEN TEXT AGAIN	The WORK file has no more space available.	KEEP WORK file. TEXT file back into WORK file.
37	HARD	INVALID INTEGER	A *linenumber* parameter is incorrect or is outside the range defined by the SET FORMAT option in effect.	Correct the *linenumber* parameter.
38	HARD	INVALID RECORD-LINE PAIR (SUBSYSTEM PROBLEM)	Data segment management error.	Notify the System Manager.
39	HARD	RANGE IS NULL OR FIRST POSITION FOLLOWS SECOND	The *range* parameter is not in ascending order.	Correct the *range* parameter.
40	HARD	UNDEFINED TEXT	The WORK file is empty.	Use an ADD, JOIN, or TEXT command to enter text into WORK file.
41	HARD	FAILURE TO OPEN KEEP FILE *(errorcode)*	KEEP command could not be executed due to *errorcode* reported.	Check KEEP command. If command OK, see *MPE Commands Reference Manual.*
42	HARD	WRITE ERROR ON KEEP FILE *(errorcode)*	File requested by KEEP command could not be written to due to *errorcode* reported.	Check KEEP command. If command OK, see the *MPE Intrinsics Reference Manual.*
43	HARD	OUT OF SEQUENCE LINE NUMBER IN TEXT FILE *(linenumber)*	Line numbers in TEXT file are incorrect.	Use TEXT *filename*,UNNUMBERED command. Correct line numbers. KEEP *filename*,UNNUMBERED.
44	SOFT	LINE NUMBER ZERO CAN NOT BE ACCESSED	A command attempted to access line zero.	Re-enter the command as FIND FIRST.
45	HARD	READ ERROR ON COMMAND FILE *(errorcode)*	Commands could not be read from the file requested due to the *errorcode* reported.	Verify that file exists and is correct. If correct, see the *MPE Commands Reference Manual.*
46	HARD	COMMAND CONTINUES PAST 256 CHARACTERS	Command record too long.	Re-enter the command record.
47	HARD	FAILURE TO OPEN USE FILE *(errorcode)*	USE command could not open USE file referenced due to *errorcode* reported.	Check that *filename* is correct. If OK, see the *MPE Intrinsics Reference Manual.*
48	HARD	FAILURE TO OPEN WHILE FILE *(errorcode)*	WHILE command could not execute due to *errorcode* reported.	Check WHILE command. If OK, see the *MPE Intrinsics Reference Manual.*
49	HARD	WRITE ERROR ON WHILE FILE *(errorcode)*	WHILE command could not execute due to *errorcode* reported.	Check the WHILE command expressions. If OK, see the *MPE Intrinsics Reference Manual.*
50	HARD	WHILE FILE OVERFLOW	Too many expressions in a WHILE command block.	Write the WHILE command expressions into a USE file.

Table 5-1. Error Messages (Continued)

ERROR NO.	TYPE	MESSAGE	CAUSE	CORRECTIVE ACTION
51	HARD	FAILURE TO OPEN JOIN FILE *(errorcode)*	*JOIN file could not be opened due to errorcode* reported.	See the *MPE/3000 Operating System Reference Manual.*
52	HARD	READ ERROR ON JOIN FILE *(errorcode)*	JOIN file could not be read due to *errorcode* reported.	Check *filename* parameter, if OK, see the *MPE Intrinsics Reference Manual.*
53	HARD	TIME-OUT ON WHILE ITERATION	SET TIME = *limit* has been exceeded or you typed CONTROL Y.	Correct WHILE command or increase iterations allowed by altering SET TIME command.
54	HARD	SYNTAX ERROR IN ABSOLUTE COLUMN POSITION EXPRESSION	A *colnum* parameter is incorrect.	Correct the parameter.
55	HARD	READ ERROR ON HOLD FILE *(errorcode)*	HOLD file could not be read due to *errorcode* reported.	Check HOLD command, if OK, see the *MPE Intrinsics Reference Manual.*
56	HARD	READ ERROR ON INPUT *(errorcode)*	TEXT or JOIN file could not be read due to *errorcode* reported.	Check command and *filename* parameter, if OK, see the *MPE Intrinsics Reference Manual.*
57	SOFT	END OF INPUT FILE	Physical end-of-file reached.	None-information only.
58	HARD	FAILURE TO OPEN HOLD FILE *(errorcode)*	Hold file could not be opened due to *errorcode* reported.	Check HOLD command, if OK, see the *MPE Intrinsics Reference Manual.*
59	HARD	WRITE ERROR ON HOLD FILE *(errorcode)*	HOLD file could not be written to due to *errorcode* reported.	Check HOLD command, if OK, see the *MPE Intrinsics Reference Manual.*
60	HARD	FCLOSE FAILURE *(errorcode)*	A file could not be closed due to the *errorcode* reported.	Check *filename* parameter, if OK see the *MPE Intrinsics Reference Manual.*
61	HARD	MAXIMUM LINE NUMBER EXCEEDED ON COBOL FILE - RENUMBER AND KEEP AGAIN	SET COBOL is in effect and line number maximum has been exceeded.	Renumber WORK file and try again.
62	HARD	UNABLE TO EDIT FILES OF THIS CODE TYPE *(filecode)*	File cannot be edited with this file-code.	See the *MPE Commands Reference Manual.*
63	HARD	WRITE ERROR *(errorcode)* ON OUTPUT *FILE*	The *listfile* named could not be written on due to the *errorcode* reported.	See the *MPE Intrinsics Reference Manual.*
64	HARD	POSITIONING ERROR ON WHILE FILE *(errorcode)*	A command in a WHILE block could not locate a position in a file due to the *errorcode* reported.	Check the *position* parameter of the command, if OK, see the *MPE Intrinsics Reference Manual.*
65	HARD	POSITION BEYOND EOF ON WHILE FILE	A command in a WHILE block tried to go to a location beyond the end-of-file in the WORK file.	Correct the WHILE block.
66	HARD	FREEDSEG REQUEST DENIED	Data segment management error.	Notify the System Manager.
67	HARD	FRENAME ERROR ON TEXT FILE *(errorcode)*	The WORK file could not be renamed due to the *errorcode* reported.	Check the *filename* parameter, if OK, see the *MPE Intrinsics Reference Manual.*

Table 5-1. Error Messages (Continued)

ERROR NO.	TYPE	MESSAGE	CAUSE	CORRECTIVE ACTION
68	HARD	FRENAME ERROR ON KEEP FILE *(errorcode)*	The file could not be renamed due to the *errorcode* reported.	Check the *filename* parameter. If OK, see the *MPE Intrinsics Reference Manual.*
69	HARD	UNABLE TO UNLOAD PREVIOUS PROCEDURE	A procedure named in a PROCEDURE command could not be loaded because the previous procedure has not been unloaded.	Check the PROCEDURE command and any previous PROCEDURE commands. See the *MPE Commands Reference Manual.* Notify the System Manager.
70	WARN	**WARNING: WORK FILE IS TEMPORARY** *reason* (See below) INSUFFICIENT SPACE IN [GROUP / ACCOUNT] USER DOES NOT HAVE SF CAPABILITY	The WORK file is temporary due to *reason* specified.	Check your library of files; your disc space may be exceeded and you may need to purge some files to make space.
71	HARD	KEEP FILE MUST BE WITHIN LOG-ON ACCOUNT	File name used in KEEP command cannot be qualified by an account name other than log-on account.	KEEP the file within the log-on account.
72	HARD	INVALID TAB OPTION	Syntax of a SET command TABS or TABCHAR parameter is incorrect: Tab stops must be within LEFT and RIGHT margins and must be specified in increasing order, TABS keyword must be followed by = or comma, tab list must be enclosed within left and right parentheses, number of tab stops cannot exceed 12, TABCHAR must be enclosed in quotes and cannot exceed one character.	Check SET command and reenter command with correct syntax.
73	HARD	FREAD LABEL ERROR (errorcode)	It is not possible to read the user labels from the old file that is kept. This is due to the errorcode.	Re-try or use a different filename with the KEEP command.
74	HARD	FWRITE LABEL ERROR (errorcode)	It is not possible to write the user label to a new file that is kept. This is due to the errorcode.	Re-try or use a different filename with the KEEP command.
75	SOFT	RECORDS TOO WIDE – KEEP UNNUMBERED	File width exceeds maximum of 256 bytes when line numbers are appended.	KEEP the file UNNUMBERED or adjust SET LENGTH to smaller value.

CUSTOMIZING EDIT/3000

SECTION VI

6-1. USER INTERFACE PROCEDURES

To satisfy a number of needs for customizing EDIT/3000 in specific applications, three logical procedures can be written. These procedures are HP32201'USERINIT, HP32201'USERADD, and HP32201'USER-COMMAND. They are designed to allow you to tailor some of the externals of EDIT/3000 to your specific requirements.

A user may write one, two, or all three logical procedures in SPL/3000, FORTRAN/3000, or any other programming language which accepts parameters as specified, produces a logical result, and is stored in an MPE/3000 Segmented Library (see the *MPE Segmenter Reference Manual* for a description of segmented libraries). When EDIT/3000 is to be customized for use by everyone on the system, these user written procedures should be installed in the system segmented library (SL.PUB.SYS). If, however, EDIT/3000 is to be customized for those users of a particular account, they should be installed in the segmented library of the public group of that account (SL.PUB.*account*). Lastly, if EDIT/3000 is being customized for those users of an account who are logged on in a particular group, the procedures should be installed in the segmented library of the log-on group of that account (SL.*group.account*).

In all cases, in order for EDIT/3000 to call the procedures, it must be initiated with the MPE command

```
:RUN EDITOR.PUB.SYS; PARM=16
```

This is the *only* way to activate the procedures, *even* if they are in the system segmented library. At run time, the segmented libraries are then searched for the procedures in the following order: first the SL of the log-on group of the log-on account, next the SL of the public group of the log-on account, and last the SL of the public group of the system account.

Note that a simple User Defined Command could be written to facilitate initiating EDIT/3000 as described above. For example:

```
MYEDIT
OPTION LIST
RUN EDITOR.PUB.SYS; PARM=16
```

See Section IX of the *MPE Commands Reference Manual* for an explanation of creating and using User Defined Commands.

Each procedure includes the following four parameters, in the order shown: STRING, LENGTH, USER-SPACE, PROCSPACE. For example, written in SPL/3000, the header for such a procedure is:

```
LOGICAL PROCEDURE HP32201'procname
      (STRING,LENGTH,USERSPACE,PROCSPACE)
BYTE ARRAY STRING;
INTEGER LENGTH;
ARRAY USERSPACE;
ARRAY PROCSPACE;
```

where *procname* is either USERINIT, USERCOMMAND, or USERADD.

The parameters are defined as follows (additional requirements are given in the descriptions of each procedure):

STRING is a byte array containing the characters of one line, which may consist of up to 255 characters.

LENGTH is an integer value specifying the number of characters in the current line being passed by STRING.

USERSPACE is a 10 word array that is available for user "global" storage. This space is shared with the other user procedures. The first two words contain the EDIT/3000 command input and output MPE file numbers, in that order. These file numbers could, for example, be used for diagnostic messages by the three procedures. The rest of the array is initialized to binary zeros when EDIT/3000 is invoked and will not be changed by EDIT/3000 between calls to the procedures.

PROCSPACE is a 20 word array maintained by EDIT/3000 for the procedure's use. It is shared with the EDIT/3000 PROCEDURE command (see the description of the array SPACE, along with the PROCEDURE command, in paragraph 4-31) and is initialized to binary zeros.

Descriptions of each procedure follow.

6-2. **HP32201'USERINIT PROCEDURE.** USERINIT will allow for a user written procedure to return a string which is executed as the first command(s) when an EDIT/3000 process is begun. SET commands (see Section III), for example, could be specified. In addition, the value returned (logical 0,1,2,or 3) determines if none, one, or both of the other two procedures (USERCOMMAND and USERADD) are to be invoked.

This procedure is called each time the user has initiated an EDIT/3000 process by using the MPE command

```
:RUN EDITOR.PUB.SYS;PARM=16
```

The parameters for this procedure are as described in paragraph 6-1. STRING contains the characters of a command line, which may consist of only one command or of multiple commands separated by semicolons. The first character of the array STRING must be a semicolon and the array must be terminated by an ASCII carriage return (%15).

A logical 3 (or TRUE) should be returned if both USERCOMMAND and USERADD are to be invoked. Returning logical 2 results in invoking only USERADD, while returning a logical 1 invokes only USERCOMMAND. If logical 0 (or FALSE) is returned, calls to these other procedures will not be executed. It is the procedure's responsibility to inform the user of which commands are executed when EDIT/3000 is begun and whether the other procedures are invoked or not.

See the example in paragraph 6-5.

6-3. **HP32201'USERCOMMAND PROCEDURE.** USERCOMMAND will allow for a user written procedure to scan EDIT/3000 commands after they are entered. For example, the procedure could screen the syntax of EDIT/3000 commands, or could interpret a user defined command as a standard EDIT/3000 command.

This procedure is invoked if logical 1 or 3 (or TRUE) is returned from USERINIT when an EDIT/3000 process is initiated. If this is the case, then USERCOMMAND is called each time the user has entered an EDIT/3000 command line, which may consist of a single command or of multiple commands separated by semicolons.

The parameters for this procedure are as described in paragraph 6-1. As for USERINIT, STRING contains the characters of a command line. If the user has continued the command line with an ampersand (&), then the procedure received the whole input in the STRING array; that is, line continuation has been processed. In addition, if Z:: has been entered (see the Z::= command explained in Section III), then it is replaced by the string value of Z::. The array is terminated by an ASCII carriage return (%15). Note that if the user defined procedure changes the length of the command line, then it must also change the value of LENGTH accordingly and must insure that the terminating carriage return is still effective.

If the procedure returns the value FALSE, the command line will be rejected and the user will be prompted with a slash (/) to reenter the line. It is the procedure's responsibility to inform the user that the input was not accepted and to issue any diagnostic messages. The command line is accepted if TRUE is returned.

See the example in paragraph 6-5.

6-4. **HP32201'USERADD PROCEDURE.** USERADD will allow for a user written procedure to scan lines entered using the EDIT/3000 ADD command (see Section III for a description of the ADD command).

This procedure is invoked if logical 2 or 3 (or TRUE) has been returned from USERINIT when an EDIT/3000 process is initiated. If this is the case, then USERADD is called after the final carriage return (that is, one which does not follow an ampersand continuation character) each time a line of text has been entered using the ADD command.)

The parameters for this procedure are as described in paragraph 6-1. STRING contains the characters of a line entered in the ADD mode. As for USERCOMMAND, line continuation (&) has been processed. If tabs are in use, they are not yet expanded into spaces; that is, the tab character has not been replaced by spaces in the STRING array. Note that if the user written procedure changes the length of the line, then it must also change the value of LENGTH accordingly.

If FALSE is returned, the line of text will be rejected and the user will be prompted with the line number unless ADDQ was used. It is the procedure's responsibility to inform the user that the input was not accepted and to issue any diagnostic messages. The line is accepted if TRUE is returned.

See the example in paragraph 6-5.

6-5. **EXAMPLES**

Three user written SPL/3000 procedures, which satisfy the requirements described above, are shown here. They are each compiled into a User Subprogram Library named USERUSL by using the *MPE/3000* SPL command. (See the *MPE Commands Reference Manual* for a discussion of the SPL command.)

```
:SPL INIT.USERUSL

PAGE 0001    HP32100A.06.05 [4W] (C) HEWLETT-PACKARD COMPANY 1976


00001000  00000 0    $CONTROL SUBPROGRAM
00002000  00000 0    $CONTROL SEGMENT=INIT'SEG
00003000  00000 0    BEGIN
00004000  00000 1    LOGICAL PROCEDURE HP32201'USERINIT(STRING,LEN,
                     USERSPACE,PROCSPACE);
00005000  00000 1    BYTE ARRAY STRING;
00006000  00000 1    INTEGER LEN;
00007000  00000 1    ARRAY USERSPACE;
00008000  00000 1    ARRAY PROCSPACE;
00009000  00000 1
00010000  00000 1    BEGIN
00011000  00000 2     MOVE STRING:="";0;USERINIT 1.0;; 0;/S SHORT;; S SHORT;";
00012000  00032 2     LEN:=39;
00013000  00034 2     STRING(LEN):=%15;  <<CR>>
00014000  00037 2     HP32201'USERINIT:=TRUE;
00015000  00041 2    END;
00016000  00000 1    END.
 PRIMARY DB STORAGE=%000;    SECONDARY DB STORAGE=%00000
 NO. ERRORS=0000;            NO. WARNINGS=0000
 PROCESSOR TIME=0:00:00;     ELAPSED TIME=0:00:17
```

```
:SPL COMMAND,USERUSL

PAGE 0001    HP32100A.06.05 [4W] (C) HEWLETT-PACKARD COMPANY 1976


00001000  00000 0    $CONTROL SUBPROGRAM
00002000  00000 0    $CONTROL SEGMENT=COMMAND'SEG
00003000  00000 0    BEGIN
00004000  00000 1
00005000  00000 1    LOGICAL PROCEDURE HP32201'USERCOMMAND(STRING,L
                     EN,USERSPACE,PROCSPACE);
00006000  00000 1    BYTE ARRAY STRING;
00007000  00000 1    INTEGER LEN;
00008000  00000 1    ARRAY USERSPACE;
00009000  00000 1    ARRAY PROCSPACE;
00010000  00000 1    BEGIN
00011000  00000 2    LOGICAL POINTER WORDPTR;
00012000  00000 2    LOGICAL ARRAY L(0:10);
00013000  00000 2    BYTE ARRAY B(*)=L;
00014000  00000 2    INTRINSIC PRINT,FWRITE;
00015000  00000 2
00016000  00000 2    HP32201'USERCOMMAND:=TRUE;
00017000  00011 2    MOVE B:="USERCOMMAND ENTERED!";
00018000  00031 2    FWRITE(USERSPACE(1),L,-20,0);
00019000  00037 2    @WORDPTR:=@STRING&LSR(1);
00020000  00042 2    PRINT(WORDPTR,-LEN,0);
00021000  00046 2    <<
00022000  00046 2    <<   IMPLEMENT THE COMMAND "QUIT"
00023000  00046 2    <<   CHANGE "QUIT" TO "KEEP" AND INSERT ";END" AT THE END
00024000  00046 2    <<   OF THE STRING
00025000  00046 2    >>
00026000  00046 2    IF STRING="QUIT" THEN
00027000  00057 2    BEGIN
00028000  00057 3      MOVE STRING:="KEEP";
00029000  00067 3      MOVE STRING(LEN):=";END";
00030000  00100 3      STRING(LEN:=LEN+4):=13;  <<CR>>
00031000  00107 3    END;
00032000  00107 2    END;
00033000  00000 1    END.
 PRIMARY DB STORAGE=%000;   SECONDARY DB STORAGE=%00000
 NO. ERRORS=0000;           NO. WARNINGS=0000
 PROCESSOR TIME=0:00:01;    ELAPSED TIME=0:00:26
```

```
:SPL ADD,USERUSL

PAGE 0001   HP32100A.06.05 [4W] (C) HEWLETT-PACKARD COMPANY 1976


00001100  00000 0    $CONTROL SUBPROGRAM
00002000  00000 0    $CONTROL SEGMENT=ADD'SEG
00003000  00000 0    BEGIN
00004000  00000 1    LOGICAL PROCEDURE HP32201'USERADD(STRING,LEN,U
                     SERSPACE,PROCSPACE);
00005000  00000 1    BYTE ARRAY STRING;
00006000  00000 1    INTEGER LEN;
00007000  00000 1    ARRAY USERSPACE;
00008000  00000 1    ARRAY PROCSPACE;
00009000  00000 1    BEGIN
00010000  00000 2    LOGICAL POINTER WORDPTR;
00011000  00000 2    LOGICAL ARRAY L(0:10);
00012000  00000 2    BYTE ARRAY B(*)=L;
00013000  00000 2    BYTE ARRAY SPACES(0:9);
00014000  00000 2    BYTE ARRAY INPUT(0:255);
00015000  00000 2    BYTE POINTER SRC'PTR;
00016000  00000 2    BYTE POINTER DEST'PTR;
00017000  00000 2    DEFINE TAB'CHAR="*";
00018000  00000 2    DEFINE INDENTATION=10-((@DEST'PTR-@STRING) MOD 10);
00019000  00000 2    EQUATE CARRIAGE'RETURN=13;
00020000  00000 2    INTRINSIC PRINT,FWRITE;
00021000  00000 2
00022000  00000 2    HP32201'USERADD:=TOS;
00023000  00021 2    MOVE B:="USERADD ENTERED!";
00024000  00037 2    FWRITE(USERSPACE(1),L,-16,0);
00025000  00045 2    @WORDPTR:=@STRING&LSP(1);
00026000  00050 2    PRINT(WORDPTR,-LEN,0);
00027000  00054 2    <<
00028000  00054 2    <<  SEARCH FOR *'S--WHEN FOUND, INSERT SPACES
00029000  00054 2    <<  THE TEXT IS "TABBED" OVER TO THE NEXT COLUMN THAT IS
00030000  00054 2    <<  A MULTIPLE OF TEN
00031000  00054 2    >>
00032000  00054 2     MOVE INPUT:=STRING,(LEN);
00033000  00060 2     INPUT(LEN):=CARRIAGE'RETURN;
00034000  00063 2     MOVE SPACES:="          ";
00035000  00076 2
00036000  00076 2     TOS:=@STRING;
00037000  00077 2     TOS:=@INPUT;
00038000  00100 2
00039000  00100 2    TAB:
00040000  00100 2     MOVE *:=* WHILE AN,0;
00041000  00101 2     @SRC'PTR:=TOS;
00042000  00102 2     @DEST'PTR:=TOS;
00043000  00103 2
00044000  00103 2     IF SRC'PTR=TAB'CHAR THEN
00045000  00106 2     BEGIN
00046000  00106 3       MOVE DEST'PTR:=SPACES,(INDENTATION),2;
00047000  00117 3       LEN:=LEN+INDENTATION-1;
00048000  00127 3       TOS:=@SRC'PTR+1;
00049000  00131 3       GO TO TAB;
00050000  00132 3     END;
00051000  00132 2
00052000  00132 2     IF SRC'PTR=CARRIAGE'RETURN THEN RETURN;
00053000  00136 2
00054000  00136 2     DEST'PTR:=SRC'PTR;
00055000  00140 2     TOS:=@DEST'PTR+1;
00056000  00142 2     TOS:=@SRC'PTR+1;
00057000  00144 2     GO TO TAB;
00058000  00145 2    END;
00059000  00000 1    END.
 PRIMARY DB STORAGE=%000;    SECONDARY DB STORAGE=%00000
 NO. ERRORS=0000;            NO. WARNINGS=0000
 PROCESSOR TIME=0:00:01;     ELAPSED TIME=0:00:16
```

Next, the Segmenter subsystem is accessed with the MPE/3000 SEGMENTER command. The USL Segmenter command is used to identify the USL file (USERUSL) which contains the function procedures to be added to the SL file, and the BUILDSL command is used to create an SL file with 20 records maximum and one disc extent. The segment containing each procedure is added to the SL file using the ADDSL command, and by specifying the segment name for each procedure (see the $CONTROL SEGMENT= statement at the beginning of each SPL/3000 procedure listed above). The LISTSL command shows the segments added and the names of entry points for each segment (in this case, there is only one entry point for each). Assuming we are logged-on in the PUB group of the SCC account, note that the SL file name is SL.PUB.SCC, so that the segmented library of the PUB group in the SCC account will have to be searched for these procedures. EXIT is typed to terminate Segmenter operation. (See the *MPE Segmenter Reference Manual* for a more detailed explanation of the Segmenter and segmented libraries).

Note that if the procedures are to be installed in the system SL, all three procedures should be put in the same segment, if possible, rather than in separate segments — as was done here. This saves space in the CST (Code Segment Table), which is shared by everyone on the system and keeps track of the segments in the system SL.

```
:SEGMENTER
HP32050A.00.00 SEGMENTER/3000  (C) HEWLETT-PACKARD CO. 1978
-USL USERUSL
-BUILDSL SL,20,1
-ADDSL INIT'SEG
-ADDSL COMMAND'SEG
-ADDSL ADD'SEG
-LISTSL

SL FILE SL.PUB.SCC

SEGMENT   0 INIT'SEG        LENGTH    44

   ENTRY POINTS     CHECK CAL STT  ADR
   HP32201'USERINI    0    C    1     0

   EXTERNALS        CHECK STT SEG

100

SEGMENT   1 COMMAND'SEG     LENGTH   114

   ENTRY POINTS     CHECK CAL STT  ADR
   HP32201'USERCOM    0    C    1     0

   EXTERNALS        CHECK STT SEG
   PRINT              0     3   ?
   FWRITE             0     2   ?

010

SEGMENT   2 ADD'SEG         LENGTH   154

   ENTRY POINTS     CHECK CAL STT  ADR
   HP32201'USERADD    0    C    1     0

   EXTERNALS        CHECK STT SEG
   PRINT              0     3   ?
   FWRITE             0     2   ?

001

USED                 3000          AVAILABLE           2000

-EXIT

END OF SUBSYSTEM
```

EDIT/3000 is invoked using the MPE RUN command. The PARM=16 parameter is used since the segmented libraries need to be searched. The USERINIT procedure is called, which prints out "USERINIT 1.0" and "/S SHORT" and executes the EDIT/3000 command S[ET] SHORT. Since this procedure returns TRUE (see the listing above), both of the other two procedures will be invoked.

The T[EXT] AFILE command (in response to EDIT/3000's slash prompt) causes the USERCOMMAND procedure to be called, since it is called whenever a command line has been entered. The USERCOMMAND procedure prints out "USERCOMMAND ENTERED!" and repeats the command line.

When the ADD command is used and a line of text is entered, the USERADD procedure is called which prints out "USERADD ENTERED!" and repeats the line. The line is accepted since USERADD returns the value TRUE (see the listing above). When AFILE is listed, we see that USERADD has also interpreted each asterisk (*) in a line of text as a tab.

The QUIT command is interpreted by the USERCOMMAND procedure as the standard EDIT/3000 commands K[EEP] ;END. In this case, the filename is defaulted; note that it is printed out below the command. (The QUIT command also allows for a filename to be specified if desired; see the listing of the USERCOMMAND procedure.)

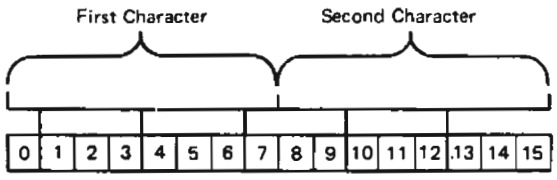
```
:RUN EDITOR.PUB.SYS;PARM=16

HP32201A.7.04 EDIT/3000  WED, JAN 17, 1979,  2:03 PM
(C) HEWLETT-PACKARD CO. 1978
USERINIT 1.0
/S SHORT
/T AFILE
USERCOMMAND ENTERED!
T AFILE
/LIST ALL
USERCOMMAND ENTERED!
LIST ALL
    1      THIS IS LINE ONE.
    2      THIS IS LINE TWO.
/ADD
USERCOMMAND ENTERED!
ADD
    3      *COLUMN TEN,LINE THREE.
USERADD ENTERED!
*COLUMN TEN,LINE THREE.
    4      ABC*123
USERADD ENTERED!
ABC*123
    5      //
...
/LIST ALL
USERCOMMAND ENTERED!
LIST ALL
    1      THIS IS LINE ONE.
    2      THIS IS LINE TWO.
    3              COLUMN TEN,LINE THREE.
    4      ABC     123
/QUIT
USERCOMMAND ENTERED!
QUIT
AFILE
PURGE OLD?Y

END OF PROGRAM
```

ASCII CHARACTER SET

APPENDIX A

ASCII Character	First Character Octal Equivalent	Second Character Octal Equivalent
A	040400	000101
B	041000	000102
C	041400	000103
D	042000	000104
E	042400	000105
F	043000	000106
G	043400	000107
H	044000	000110
I	044400	000111
J	045000	000112
K	045400	000113
L	046000	000114
M	046400	000115
N	047000	000116
O	047400	000117
P	050000	000120
Q	050400	000121
R	051000	000122
S	051400	000123
T	052000	000124
U	052400	000125
V	053000	000126
W	053400	000127
X	054000	000130
Y	054400	000131
Z	055000	000132
a	060400	000141
b	061000	000142
c	061400	000143
d	062000	000144
e	062400	000145
f	063000	000146
g	063400	000147
h	064000	000150
i	064400	000151
j	065000	000152
k	065400	000153
l	066000	000154
m	066400	000155
n	067000	000156
o	067400	000157
p	070000	000160
q	070400	000161
r	071000	000162
s	071400	000163
t	072000	000164
u	072400	000165
v	073000	000166
w	073400	000167
x	074000	000170
y	074400	000171
z	075000	000172
0	030000	000060
1	030400	000061
2	031000	000062
3	031400	000063
4	032000	000064
5	032400	000065
6	033000	000066
7	033400	000067
8	034000	000070
9	034400	000071
NUL	000000	000000
SOH	000400	000001
STX	001000	000002
ETX	001400	000003
EOT	002000	000004
ENQ	002400	000005

ASCII Character	First Character Octal Equivalent	Second Character Octal Equivalent
ACK	003000	000006
BEL	003400	000007
BS	004000	000010
HT	004400	000011
LF	005000	000012
VT	005400	000013
FF	006000	000014
CR	006400	000015
SO	007000	000016
SI	007400	000017
DLE	010000	000020
DC1	010400	000021
DC2	011000	000022
DC3	011400	000023
DC4	012000	000024
NAK	012400	000025
SYN	013000	000026
ETB	013400	000027
CAN	014000	000030
EM	014400	000031
SUB	015000	000032
ESC	015400	000033
FS	016000	000034
GS	016400	000035
RS	017000	000036
US	017400	000037
SPACE	020000	000040
!	020400	000041
"	021000	000042
#	021400	000043
$	022000	000044
%	022400	000045
&	023000	000046
'	023400	000047
(	024000	000050
)	024400	000051
*	025000	000052
+	025400	000053
,	026000	000054
-	026400	000055
.	027000	000056
/	027400	000057
:	035000	000072
;	035400	000073
<	036000	000074
=	036400	000075
>	037000	000076
?	037400	000077
@	040000	000100
[	055400	000133
\	056000	000134
]	056400	000135
Δ	057000	000136
—	057400	000137
`	060000	000140
{	075400	000173
|	076000	000174
}	076400	000175
~	077000	000176
DEL	077400	000177



COMMAND LANGUAGE

APPENDIX B

Table B-1 allows you to determine the precise syntax requirements of a command, or to quickly refresh your knowledge of the command language. The table is helpful to have, particularly during an interactive session, as a general reference tool. If further clarification is needed, the page number of a complete explanation of each command is provided.

Table B-1. Command Language Reference Chart

Command	Syntax	Use	Example	Remarks	Page Reference
ADD	ADD[Q] [*linenumber*] [,HOLD[Q] [,NOW]]	To enter lines of text into the WORK file from an input file or from the HOLD file.	ADD 60, HOLD	The contents of the HOLD file are entered into the WORK file beginning with the line number 60.	3-6
BEGIN	BEGIN[Q]	Used as the first expression in a matching BEGIN-END pair.	WHILE FINDQ"this"(+3) BEGINQ FINDQ"program"/*(+20) LIST* END	This is a blocked set of commands within a BEGIN-END pair.	4-7
CHANGE	CHANGE[Q] { *oldstring* / *startcolumn*[/*startcolumn*] } TO *newstring* [IN *rangelist*]	To replace old text with new text.	CHANGE "RECORD" TO "LINE" IN 40/70	All occurrences of RECORD in lines 40 through 70 will be changed to LINE.	3-10
COPY	COPY[Q] *range* TO *linenumber* [BY *increment*]	To copy text from one location into another in the WORK file.	COPY 2/2.15 to 8.01 BY .01	Lines 2 through 2.15 will be copied to begin at line 8.01 with lines numbered in increments of .01.	3-14
DELETE	DELETE[Q] [*rangelist*]	To delete characters and lines from the WORK file.	DELETE 50/75, 150(3)/155(7)	Lines 50 through 75 and characters in the range beginning with the third column in line 150 through the seventh column in line 155 will be deleted.	3-17
END	END	To terminate execution of EDIT/3000 or, when used with a matching BEGIN command, to terminate an iterative operation.	END	Control is returned to MPE/3000.	3-20
FIND	FIND[Q][* / *range*]	To locate a point in the WORK file.	FIND "PROGRAM"	The pointer will move to the next occurrence of the string PROGRAM.	3-21
GATHER	GATHER[Q] *range* TO *linenumber* [BY *increment*]	To move portions of text from one location to another in the WORK file and renumber the lines.	GATHER 50/55 TO 5.1 BY .1	Lines 50 through 55 will be moved to begin at line 5.1 and will be renumbered in increments of .1.	3-23
HOLD	HOLD[Q] [*range*] [,APPEND]	To copy text from the WORK file into the HOLD file.	HOLD 100/150, APPEND	Lines 100 through 150 are copied into the HOLD file and are appendad to tha text that already exists in the HOLD file.	3-27
INSERT	INSERT[Q] *position* [BY *increment*] [,HOLD[Q] [,NOW]]	To insert text into the WORK file from an input file or the HOLD file.	INSERT 25(4)	Line 25 is printed with the pointer indicated directly under column 4. Any text may then be typed in. The remainder of line 25 is saved and printed following the insertion.	3-29
JOIN	JOIN[Q] *file name* [*(linenumber[/linenumber])* / *(#recnum/#recnum)*] [TO *linenumber*] [BY *increment*]	To add all or part of a JOIN file to the WORK file.	JOIN EDIT02 TO 1000 BY .1	The text in EDIT02 is added to the WORK file beginning at line 1000. Lines are numbered in increments of .1.	3-35

Table B-1. Command Language Reference Chart *(Continued)*

Command	Syntax	Use	Example	Remarks	Page Reference
KEEP	KEEP [[Q] *file name* [(*range*)] [,UNNUMBERED]]	To save all or part of the WORK file in a user file.	KEEP EDIT02	The contents of the WORK file are saved in EDIT02, a user file, and the line numbers are stored in ASCII format at the front or rear of each line.	3-39
LIST	LIST[Q] [*range*] [,UNN[UMBERED]] [,OFFLINE] [,TRANSLATE] [,NOTEXT]	To print out any portion or all of the WORK file.	LIST ALL UNNUMBERED, OFFLINE	The contents of the WORK file are printed on the file printer or at the device specified in the system log-on command. Line numbers are not printed.	3-47
MODIFY	MODIFY[Q] *range list*	To modify text in the WORK file using three operations: delete (D), insert (I), and replace (R).	MODIFY 50/100	Lines 50 through 100 are printed line by line to allow a modification templet to be entered for each line in the range.	3-51
NOT	NOT	Causes setting of FLAG to be reversed after the next following expression is interpreted.	WHILE FINDQ,"this"(+3) BEGINQ NOT FINDQ"program"/*(+20) LIST* END	Occurrences of "this" that are *not* followed by "program" are listed.	4-12
OR	OR	Resets FLAG to true if it is false, or skips OR command and command following if FLAG is true.	BEGINQ FINDQ"this"/*(+40) OR FINDQ"program"/*(+20) END	If "this" is found within 40 nonblank characters of the pointer, the second FIND command is skipped over. If "this" is not found, the FLAG is set to true, and the second FIND is executed.	4-16
PROCEDURE	PROCEDURE *[pname,* [G | P | S] *[range list]]*	Calls logical procedure stored in a library outside the subsystem.	PROCEDURE CARDTEST, G, 100/300	Logical procedure CARDTEST is called from the group library. Records 100 through 300 of the WORK file are affected by the procedure.	4-24
Q	Q*character string*	To print a message at the terminal.	Q "HELLO. PLEASE ENTER YOUR IDENTITY CODE."	When this command is executed, the delimited message will be printed at the terminal.	3-54
REPLACE	REPLACE[Q] *rangelist* [,HOLD[Q][,NOW]]	To replace lines in the WORK file.	REPLACE 10/20	Each line in the range from 10 through 20 is listed followed by the line number to allow you to enter a text replacement.	3-55

Table B-1. Command Language Reference Chart *(Continued)*

Command	Syntax	Use	Example	Remarks	Page Reference
SET	SET[FROM=*linenumber*] [,DELTA=*increment*] [,LEFT=*colnum*] [,RIGHT=*colnum*] [,LENGTH=*colnum*] [,{QUIET / DISPLAY}] [,{SHORT / LONG}] [,{BATCH / POLL}] [,DEPTH=*limit*] [,TIME=*limit*] [,SIZE=*limit*] [,{FRONT / REAR}] [,{FIXED / VARIABLE}] [,FORMAT={*COBOL* / *DEFAULT*}] [,{TABS[=(*colnum*[,*colnum*]...)] / NOTABS}] [,TABCHAR[=*string*]]	To alter options that are normally set by the subsystem and that govern editing operations. Note: Default conditions are shown underlined.	SET FROM=100 DELTA=10, QUIET, SHORT	Lines in the WORK file will begin with line number 100 and will be numbered in increments of 10. The position of the pointer will not be displayed each time a location in the WORK file is found, and commands that take "Q" operate as if Q had been specified. Commands can be entered using only their initial character. Otherwise, the first and last column position read in a line of the WORK file will be 1 and 72, respectively.	3-59
TEXT	TEXT[*file name* [{*(line number/line number)* / *(#recnum/#recnum)*}] [,UNNUMBERED]	To copy contents of a user file into the WORK file to be edited.	TEXT EDIT02	Contents of EDIT02 are copied into the WORK file and numbered using the line numbers stored in the front or rear bytes of each line.	3-67
USE	USE[*filename*]	To instruct EDIT/3000 to read commands from a user file but to send messages to and, normally, to expect text from the terminal.	USE EDFIL	EDIT/3000 will go to the beginning of EDFIL and execute commands from that file until it finds end-of-file mark.	3-75
VERIFY	VERIFY[{ALL / *option list*}]	To obtain a reminder of the values of options declared in the SET command.	VERIFY DELTA, FROM	The values set up for DELTA and FROM when EDIT/3000 was initialized are listed.	3-79
WHILE	WHILE[FLAG]	Causes EDIT/3000 to iterate the next two edit expressions (or one, if FLAG is declared) until the first one fails.	WHILE FIND "this" LIST *	Each occurrence of "this" following the pointer is found and listed until the end of the WORK file is found before another "this" is found. The FLAG is then set to false.	4-2
XPLAIN	XPLAIN[{*com. name list* / ALL}] [,OFFLINE]	To obtain an explanation of the commands.	XPLAIN A,S	Explanations of the ADD and SET commands will be printed at the terminal.	3-81
YES	YES	Sets the FLAG to true			4-20
Z : :=	Z : := *object record*	To assign the value of the character string Z : :=.	Z : := "DOUBLE PRECISION"	The string "DOUBLE PRECISION" will be substituted for Z : := whenever Z : : occurs in a command outside of a string.	3-83

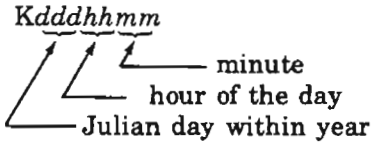
Table B-1. Command Language Reference Chart (Continued)

Command	Syntax	Use	Example	Remarks	Page Reference
:	:*MPE* command	To enter system commands from within EDIT/3000	:PURGE AFILE	The *MPE* PURGE command is executed	3-86

WORK FILE RECOVERY

APPENDIX C

EDIT/3000 creates a WORK file whenever an ADD, TEXT, or JOIN command is entered. This WORK file is a permanent file and is named by EDIT/3000 as follows:



Note: EDIT/3000 logic prevents duplicate names within a group.

To recover a WORK file (for example, after a system crash), proceed as follows:

1. Enter an MPE/3000 :LISTF command to determine the K number under which the WORK file was saved by EDIT/3000.

2. Access EDIT/3000 and enter these commands:

   /TEXT K*dddhhmm*
   /KEEP *filename* (any new file name)
   /TEXT *filename*

INDEX

Absolute column position, 3-4
Absolute record position. 3-4
Accessing MPE/3000, 3-86
Account password, 2-2
acctname, 2-2
ADD command, 3-6
— ADDSL command, 4-26, 6-6
Ampersand, 2-12
Ampersand, use of, 3-53
acctpass, 2-2
APPEND, 3-28
ASCII character set, A-1
Asterisk, 3-3

BASICENTRY entry point to EDIT/3000, 2-13
Batch
  initiating a batch job, 2-4
  running EDIT/3000 in batch mode, 2-7
  SET BATCH command, 3-59
  terminating a batch job, 2-6
BEGIN command, 4-8
Blank characters, 3-2
BREAK, 2-9, 2-10
BS, 2-3, 2-4
— BUILDSL command, 4-26, 6-6
:BYE command, 2-3

Card reader, using as an input device, 2-7
CHANGE command, 3-10
Character position, 3-5
COBOL, 3-60
colnum, 3-3
Colon (:) command, 3-86
Column, 3-2
Column position 3-4
Commands
  ADD, 3-6
  —ADDSL, 4-26, 6-6
  BEGIN, 4-8
  —BUILDSL, 4-26, 6-6
  :BYE, 2-3
  CHANGE, 3-10
  colon (:), 3-86
  COPY , 3-14
  definition of terms, 3-1
  DELETE, 3-17
  descriptions, 3-2
  :EDITOR, 2-6
  END, 3-20
  :EOJ, 2-6
  :FILE, 3-49, 2-13
  FIND, 3-21
  :FORTRAN, 4-26
  GATHER, 3-23
  :HELLO, 2-1
  HOLD, 3-27
  INSERT, 3-29
  :JOB, 2-4
  JOIN, 3-35
  KEEP, 3-39
  language, B-2
  LIST, 3-47
  —LISTSL, 4-29, 6-6
  MODIFY, 3-51
  multiple, 2-12
  NOT, 4-12
  optional parameters, 3-2
  OR, 4-16
  PROCEDURE, 4-24
  Q, 3-54
  REPLACE, 3-55
  required parameters, 3-2
  :RUN, 2-13, 6-1
  SET, 3-59
  summary, 1-4
  TEXT, 3-67
  USE, 3-75
  —USL, 4-26, 6-6
  VERIFY, 3-79
  WHILE, 4-2
  XPLAIN, 3-81
  YES, 4-20
  Z::=, 3-83
  :, 3-86
Command string maximum length, 2-12
Command summary, 1-4
Comment in command strings, 2-12, 3-75
Comments in USE file, 3-75
Continuing a command, 2-12
CONTROL H, 2-9, 3-64
CONTROL I, 3-62
Controls, 2-9, 2-11
CONTROL X, 2-9
CONTROL Y, 2-11, 3-13, 3-85
COPY command, 3-14
Copying a TEXT file from cards, 3-72
cpusecs, 2-3, 2-4
Creating a two-column format, 3-5
CS, 2-3, 2-4
customizing EDIT/3000, 6-1

Data protection, 1-2
Default EDIT/3000 operating conditions, 3-60
Default size of WORK file, 1-2
Definition of terms, 3-1
DELETE command, 3-17
Delimiters, 3-2
DELTA, 3-59
DEPTH, 3-60
device, 2-5
DISPLAY, 3-61
DS, 2-3, 2-4

INDEX

Editing modes, 1-2
:EDITOR command, 2-6
EDIT/3000
  advanced commands, 4-1
  commands (see Commands)
  command summary, 1-4
  default operating conditions, 3-60
  description, 1-1
  editing modes, 1-2
  error messages, 5-2
  features, 1-1
  file definitions, 1-2
  pointer, 3-3
  special controls, 2-9
EDIT/3000 special entry point, 2-13
EDITIN file, 1-2
EDITOUT file, 1-2
EDTLIST file, 3-49
END command, 3-20
:EOJ command, 2-6
Error messages, 5-2
Error recovery procedures, 5-2
ES, 2-3, 2-4
Escape 1, 3-64
Extra data segments, 3-69

:FILE command, 3-49
File definitions, 1-2
Files
  definitions, 1-2
  :FILE command, 3-49
  HOLD file, 1-3
  INPUT file, 1-2
  JOIN file, 1-3
  OUTPUT file, 1-2
  saving, 3-39
  TEXT file, 1-3
  USE file, 1-3
  WORK file, 1-2
  WORK file size, 1-2, 3-62
  WORK file recovery, C-1
filename, 3-3
FIND command, 3-21
FIXED, 3-60
Fixed-length records, 3-60
FLAG, 4-2
FORMAT, 3-60
:FORTRAN command, 4-26
FORTRAN/3000 sample procedure, 4-26
FROM, 3-61
FRONT, 3-61

GATHER command, 3-23
grouppass, 2-2
Group library, 4-24
groupname, 2-2
Group password, 2-2

Hard errors, 5-1
:HELLO command, 2-1
HIPRI, 2-3, 2-5
HOLD command, 3-27
HOLD file, 1-3
Horizontal tab, 3-62

increment, 3-3
Initiating a batch job, 2-4
Initiating an interactive session, 2-1
INPUT file, 1-2
inputpriority, 2-3, 2-4
INSERT command, 3-29
integer, 3-3
Interactive
  initiating a session, 2-1
  running EDIT/3000 in an interactive session, 2-6
  terminating a session, 2-3

:JOB command, 2-4
jobname, 2-4
JOIN command, 3-35
JOIN file, 1-3

KEEP command, 3-39
Keeping an unnumbered file, 3-42
Keeping the WORK file contents on a card
  punch device, 3-45

LEFT, 3-61
LENGTH, 3-61
Library
  group, 4-24
  public, 4-24
  segmented, 4-24
  system, 4-24
  user subprogram, 4-26
limit, 3-3
Line, 3-1
Line feed, 2-9
Line length, 3-61
Line number intervals, 3-60
linenumber, 3-3
Line printer listings, 3-49
LIST command, 3-47
Listing the WORK file contents offline, 3-49
-LISTSL command, 4-26
Logical record numbers, 3-5
LONG, 3-61

Margins in the WORK file, 3-61
Messages, 5-2
Mode of operation, 1-2
MODIFY command, 3-51
MPE/3000 special controls, 2-9

newstring, 3-3
Nonprinting characters, 3-2, 3-5
NOT command, 4-12
NOTABS parameter, 3-59, 3-62
NOTEXT, 3-47
Null string, 3-2
numcopies, 2-5

Offline list file, 3-49
oldstring, 3-3
Optional parameters, 2-1
OR command, 4-16
OUTPUT file, 1-2
outputpriority, 2-5

Parameters
  descriptions, 3-3
  optional, 2-1
  required, 3-2
  summary, 3-88
Password, 2-2
Pointer, 3-3
POLL, 3-59
Position, 3-1
position parameter, 3-4
PROCEDURE command, 4-24
Procedures
  HP32201'USERINIT, 6-2
  HP32201'USERCOMMAND, 6-2
  HP32201'USERADD, 6-3
Prompt character, 2-7
Public library, 4-24

Q command, 3-54
QUIET, 3-61

Range, 3-1
range parameter, 3-4
Range list, 3-1
rangelist parameter, 3-5
Reading commands and text from a USE file, 3-77
Reading commands from a USE file, 3-76
REAR, 3-61
recnum, 3-5
Record, 3-1
Record size, 3-61
Relative column position, 3-4
Renumbering the WORK file, 3-24
REPLACE command, 3-55
Required parameters, 2-1
RESTART, 2-5
RIGHT, 3-61
Running EDIT/3000 in an interactive session, 2-6
Running EDIT/3000 in batch mode, 2-7

Saving files, 3-39
Segmented library, 4-24, 6-1, 6-6
Segmenter, 4-26, 6-6
Semicolon, 2-12
Sequencing information, 3-43
sessionname, 2-1
SET BATCH, 3-59
SET command, 3-59
SET DELTA, 3-59
SET DEPTH, 3-60
SET DISPLAY, 3-61, 3-66
SET FIXED, 3-41, 3-60
SET FORMAT, 3-60
SET FROM, 3-61
SET FRONT, 3-61
SET LEFT, 3-61
SET LENGTH, 3-61
SET LONG, 3-61
SET POLL, 3-59
SET QUIET, 3-61, 3-66
SET REAR, 3-61
SET RIGHT, 3-61
SET SHORT, 3-61
SET SIZE, 3-62
SET VARIABLE, 3-41, 3-60
SHORT, 3-61
SIZE, 3-62
Soft errors, 5-1
Special characters, 3-2
Special controls
  EDIT/3000, 2-11
  MPE/3000, 2-9
Special entry point to EDIT/3000, 2-13
Splitting a line into two lines, 3-32
SPL/3000 sample procedure, 4-24, 6-1, 6-3
startcolumn, 3-5
startline, 3-5
stopcolumn, 3-5
stopline, 3-5
String, 3-2
String delimiters, 3-2
string parameter, 3-5
String searches, 3-21
Summary of commands, 1-4
System library, 4-24

TABCHAR parameter, 3-59, 3-62, 3-79
TABS parameter, 3-59, 3-62, 3-79
Tabset (Escape 1), 3-64
Terminal types, 2-2
termtype, 2-2
Terminating a batch job, 2-5
Terminating an interactive session, 2-3
Terminating EDIT/3000
  in batch mode, 2-6
  in interactive mode, 2-3
TEXT command, 3-67
TEXT file, 1-3
Two-column format, 3-58

UNNUMBERED, 3-42, 3-67, 3-70
USE command, 3-75
USE file, 1-3
  Comments, 3-75
Use of ampersand, 3-53
User Interface procedure, 6-1
User labels, 3-46
username, 2-2
userpass, 2-2
User password, 2-2
User subprogram library, 4-26
Using a card reader as an input device, 2-7
Using EDIT/3000 special controls semicolon and ampersand, 2-12
USL, 4-26
-USL command, 4-26

VARIABLE, 3-60
Variable-length records, 3-60
VERIFY command, 3-79

WHILE command, 4-2
WHILE loop, 3-85
WORK file, 1-2
WORK file recovery, C-1
WORK file size
  default, 1-2
  specified with SET SIZE, 3-62

XPLAIN command, 3-81

YES command, 4-20

Z::= command, 3-83

: command, 3-86