HEWLETT PACKARD

EDIT/1000 User's Manual

Software Technology Division
11000 Wolfe Road
Cupertino, CA 95014-9804

Manual Part No. 92074-90001
E1292

Printed in U.S.A. December 1992
Third Edition

NOTICE

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

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THE 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.

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

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARs 252.227.7013.

Printing History

The Printing History below identifies the edition of this manual and any updates that are included. Periodically, update packages are distributed which contain replacement pages to be merged into the manual, including an updated copy of this printing history page. Also, the update may contain write-in instructions.

Each reprinting of this manual will incorporate all past updates; however, no new information will be added. Thus, the reprinted copy will be identical in content to prior printings of the same edition with its user-inserted update information. New editions of this manual will contain new information, as well as all updates.

To determine what manual edition and update is compatible with your current software revision code, refer to the Manual Numbering File or the Computer User's Documentation Index. (The Manual Numbering File is included with your software. It consists of an "M" followed by a five digit product number.)

First Edition	Aug 1980	
Update 1	Oct 1981	Scratch File and other enhancements
Update 2	Jul 1982	RTE-A.1
Update 3	Jun 1983	RTE-A
Update 4	Dec 1983	CI file system added to RTE-6/VM
Update 5	Jan 1985	
Second Edition	Aug 1987	
Update 1	Jan 1988	Software Revision 5.1 (5010)
Update 2	Jul 1990	Software Revision 5.2 (5020)
Third Edition	Dec 1992	Software Revision 6.0 (6000)

Preface

EDIT/1000 is a utility program used to input and edit textual information. It is part of the RTE Operating System (versions A and 6/VM) and runs on the HP 1000 system computers. EDIT is used to create and modify files containing:

- Source code for programs in many languages; for example, BASIC, FORTRAN, Macro, Pascal, and C.
- Text for documents such as memos, letters, program notes, or manuscripts.

This manual provides a starting point for new EDIT/1000 users and a reference for experienced users. A first-time EDIT user should read Chapter 1, work through the tutorial in Chapter 2 and then read Chapter 3 for details on the operation of the EDIT program. Experienced EDIT users will benefit from the quick reference task charts in Chapter 2, the alphabetical command listing in Chapter 4 and the information on Regular Expressions in Chapter 5. The appendix section on error and information messages will aid both new and experienced users. The manual consists of five chapters and appendices:

CHAPTER 1 – EDIT/1000 BASICS

Chapter 1 presents the basic concepts and operation of the EDIT/1000 program.

CHAPTER 2 – GETTING STARTED TUTORIAL

Chapter 2 contains a three-part tutorial designed to instruct the new user on how to access and exit the EDIT program, and to edit a file using the terminal editing keys and line mode commands such as Copy, Move, Undo, and List. The chapter concludes with several quick reference task charts of EDIT commands.

CHAPTER 3 – EDIT/1000 OPERATIONS

Chapter 3 provides detailed information on the operation of the EDIT/1000 program, including command syntax, line range specification, pattern specification for search and exchange commands, EDIT session options, error recovery, batch operation, and error and information messages.

CHAPTER 4 – EDIT/1000 COMMANDS

Chapter 4 is designed as a reference chapter. It contains an alphabetical listing of all the EDIT commands.

CHAPTER 5 – EDIT REGULAR EXPRESSIONS

Chapter 5 presents instructions on using EDIT's Regular Expressions option to match pattern strings in text files and to accomplish file tasks in conjunction with the RTE user interface.

APPENDICES

The Appendices for the EDIT/1000 Manual provide a listing of error and information messages, a glossary, instructions on loading the EDIT/1000 program, and information on using EDIT in a multipoint environment.

Table of Contents

Chapter 1
EDIT/1000 Basics

Chapter 2
Getting Started Tutorial

Chapter 3
EDIT/1000 Operations

Chapter 4
EDIT/1000 Commands

Chapter 5
EDIT/1000 Regular Expressions

Appendices

List of Illustrations

Tables

1

EDIT/1000 Basics

Introduction

This chapter provides descriptions of the basic concepts and structure of the EDIT/1000 program. The contents include

- EDIT/1000 Description and Features
- EDIT/1000 Information Sources
- Working with EDIT
- Overview of EDIT Operations
- Error and Information Messages

The information in this chapter will enable you to

- Learn about EDIT by using the EDIT online reference (help) facility, the RTE-A online Hello tutorial, or this reference manual.
- Access the EDIT program by using the EDIT runstring from the RTE system user interface (either the Command Interpreter, CI or the File Manager, FMGR).
- Exit the EDIT/1000 program.
- Briefly describe what is meant by the terms EDIT Work Area, Line Mode, Screen Mode, and Pending Line.
- Recognize EDIT error and informational messages.

EDIT/1000 Description

EDIT/1000 is a utility program that runs on the HP 1000 system computers, under the RTE-A and RTE-6/VM operating systems. EDIT/1000 is used to create and modify files containing

- Source code for programs in many languages; for example, BASIC, FORTRAN, Macro, Pascal, and Ada.
- Text for documents such as memos, letters, program notes, or manuscripts.

EDIT/1000 is designed to increase your productivity in program development and text preparation or modification. As with any software tool, the command syntax and the program operation must be understood to take full advantage of the capabilities of the program.

EDIT/1000 Features

One of the features of the EDIT/1000 program is that for existing files, EDIT allows you to edit a copy of the file (referred to as your *work file*) so as to preserve your source file. When the work file is edited to your satisfaction, it can be used to update the source file. Additional features of the EDIT/1000 program follow:

- A simple runstring that can be expanded to include EDIT commands as you become more experienced.
- Over 50 one- or two-character commands, most of which are named mnemonically to aid in recall.
- Prompts that require you to confirm the execution of commands that delete or significantly alter data.
- A command (the Undo command) that allows you to reverse the effect of the last command executed.
- A line editing mode that enables you to operate EDIT on a terminal that does not support block mode.
- Special line mode commands that enable you to use the local terminal keys to edit text while in line mode.
- A screen mode that enables you to see and edit an entire screen of text using the local terminal keys and control key combinations.
- A screen mode command that enables you to execute some of the more powerful line mode commands (for example, search, exchange, move, and copy) from screen mode.
- A system of error, informational, and instructional messages that keeps you informed of the status of the program and your files.
- A recovery operation that reconstructs files damaged during an abnormal program termination.
- A command stack feature that allows you to review, modify, and re-execute any of the last 20 EDIT commands entered during the current EDIT session.

Manual Conventions

The following conventions have been used throughout this Manual:

Table 1-1. Manual Conventions

Convention	Description
`text`	EDIT recognizes user entries in both upper- and lowercase. User input is printed in lowercase letters and is highlighted. The only exceptions to this are the commands containing the letter L, and the N command. Commands containing an L are printed in uppercase because they cannot be distinguished from the number one when entered in lowercase. The N command is printed in uppercase to distinguish it from the *n* command, which represents the entry of a line number.

`Opened file SMART::MOI`
`200 lines read.`
	System response is printed as it is displayed by EDIT. EDIT uses both uppercase and lowercase letters.

`UPPERCASE LETTERS`	Used in command syntax display to indicate an EDIT command. For example: CO is used for the Copy command.
`lowercase letters`	Used in command syntax display to indicate variables to be replaced by values as defined in the text. For example, n is often used to indicate the entry of an integer: T n1,n2,n3
`CI>`	The Command Interpreter user interface prompt. The EDIT program can be initiated from this prompt or from the File Manager (FMGR) prompt. The examples in this manual use the CI> prompt.
[ ]	Indicates parameters that are optional.
< >	Indicates information that is to be input by the user. For example: <command> <options> <text>
^x or CTRL-x	The caret (^) symbol or the abbreviation CTRL are used to represent the CONTROL key on your terminal keyboard. Many EDIT commands require you to press and hold the control key and then depress the alphabetic key. For example: ^B or CTRL-S. A screen mode control key command is always followed by a carriage return.

EDIT/1000 Information Sources

There are many ways to learn about EDIT/1000, ranging from reading the manual to experimenting with the program. The most common methods follow:

- Refer to EDIT's online Quick Reference facility.
- Use the RTE-A online tutorial, "Hello".
- Read the first three chapters of this Manual.

Online Quick Reference

Online reference is easily accessed with the EDIT help and information display commands. The *help* (? or H) command displays any of the following:

- A one screen summary of all the EDIT line mode commands and other topics for which help is available.
- A brief explanation for each command, including command syntax, default line range, and options available.
- Abbreviations used by EDIT.
- Pattern and associated special characters.
- Pending line character editing commands.
- Regular expressions and metacharacters (for pattern search).
- Line specifications and associated special characters.

The *information display* commands provide pattern defaults, the default source file name, the current status of options, and other pertinent information. Typical commands of this type are the Show (SH) and Line Length (LE) commands. For a listing of the commands used to access online reference and the specific topics available, refer to the descriptions of the H and ? (Help) commands and the SH (Show) commands in Chapter 4.

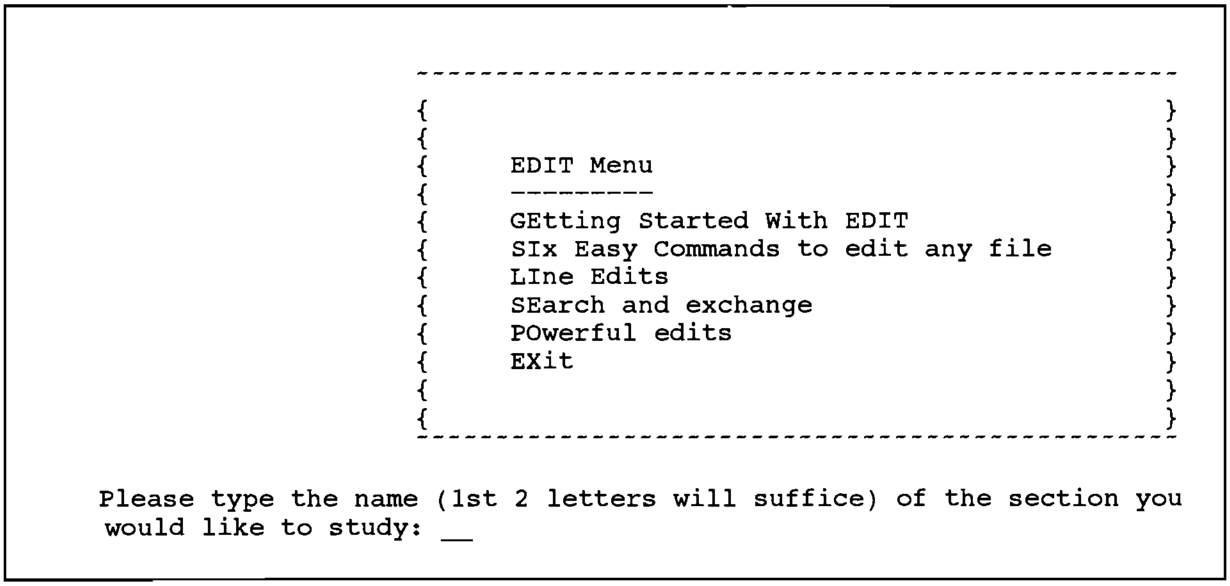
RTE-A "Hello" Program

The RTE-A Operating System primary provides an online tutorial that introduces the RTE-A Operating System, its utilities, file system, and program development and control. A large portion of the "Hello" program is devoted to EDIT/1000. To access the EDIT/1000 portion of the Hello tutorial:

1. Enter `hello` at the CI prompt. (Note that if the Hello program is not loaded on your system, you will receive the message *No such file HELLO*. See your System Manager to have the program loaded.)
2. At the Hello Main Menu, enter `de` to select the topic, DEveloping Programs.

3. At the Program Development Menu, select EDIT by entering ed. (For a general discussion of how to develop a program, first select in, Introduction to Program Development.)

4. The system displays the EDIT menu, illustrated in Figure 1-1:



```
                  -------------------------------------------------
                  {                                               }
                  {                                               }
                  {     EDIT Menu                                 }
                  {     ---------                                 }
                  {     GEtting Started With EDIT                 }
                  {     SIx Easy Commands to edit any file        }
                  {     LIne Edits                                }
                  {     SEarch and exchange                       }
                  {     POwerful edits                            }
                  {     EXit                                      }
                  {                                               }
                  {_______________________________________________}
                  -------------------------------------------------

Please type the name (1st 2 letters will suffice) of the section you
would like to study: __
```

Figure 1-1. RTE–A Hello Program Menu

5. Type the first two letters of the desired section and press the carriage return.

6. To exit the Hello program, continue to enter ex at each menu until the CI> prompt is displayed.

Working with EDIT

Accessing EDIT

The EDIT/1000 program is accessed from either the Command Interpreter (CI) or File Manager (FMGR) RTE user interfaces. The examples in this manual use the CI> prompt.

EDIT Runstring

The EDIT runstring is the command or group of commands used to initiate the EDIT/1000 program. The simplest form of the EDIT runstring consists of the word, EDIT. This form of the command string accesses the EDIT/1000 program but does not access a specific text file for editing. Specific files can be accessed after EDIT has been initiated. All EDIT commands, including the runstring, must be followed by a carriage return. An example of the simplest EDIT runstring follows:

```
CI> edit
```

For more information on EDIT options and commands that can be included in the runstring, refer to the EDIT Session Options section in Chapter 3 and the description of commands in Chapter 4 of this Manual. For instructions on entering the runstring to initiate EDIT sessions, refer to Chapter 2 of this Manual.

EDIT Prompt (/)

Once an editing session has been initiated with the run-string, EDIT commands are entered at the EDIT prompt. The default EDIT prompt character is the slash (/). The EDIT prompt character can be set to any non-alphanumeric character (except a space or a comma) using the Set (SE) command to set the PC (Prompt Character) option. Refer to the description of the SE command in Chapter 4 of this Manual.

Getting Help

EDIT's online quick reference facility provides brief descriptions of the commands, the special characters, and other online options. The H, HE, and ? commands can be used interchangeably from EDIT's line mode. Examples of the commands are provided following, entered at the EDIT prompt:

Use	**To Obtain**
/H or /HE or /?	A summary of online reference.
/? <command>	Information on a specific command.
/? EX	An explanation of EDIT abbreviations.
/? RE	A description of regular expressions (used in pattern matching).
/? LS	A description of line specifications.
/? PA	Pattern descriptions.
/? PL	A description of pending line character editing.
/? RM	A description of the recovery mode.
/? DA	A display of the EDIT date code.
/? AB	A listing of the EDIT abort messages.
/? RO	A description of EDIT's runstring options.

The MORE feature permits help information to be displayed one screen at a time. Pressing the letter "a" aborts the listing, pressing the return key displays the rest of the current help message without pausing. Pressing any other key causes the next screen to be displayed.

Exiting EDIT

There are three commands used to exit an EDIT session. Two of the commands exit the EDIT program, saving the work done during the editing session, and the third command exits EDIT without saving the work done during the editing session.

- EC (Exit and Create), which uses the information in the work file to create a new source file and then purges the work file.
- ER (Exit and Replace), which uses the information in the work file to write over (replace) the original source file from which the work file copy was made. The work file is then purged.
- A (Abort), which aborts the EDIT session and discards the data entered or changes made to the work file.

Note that a fourth command, AS (Abort and Save Work File), is used to *temporarily* save the work file. This command is rarely used and is described under the System Errors section in Chapter 3.

Overview of EDIT Operations

To take full advantage of the capabilities of the EDIT program, it is important to understand

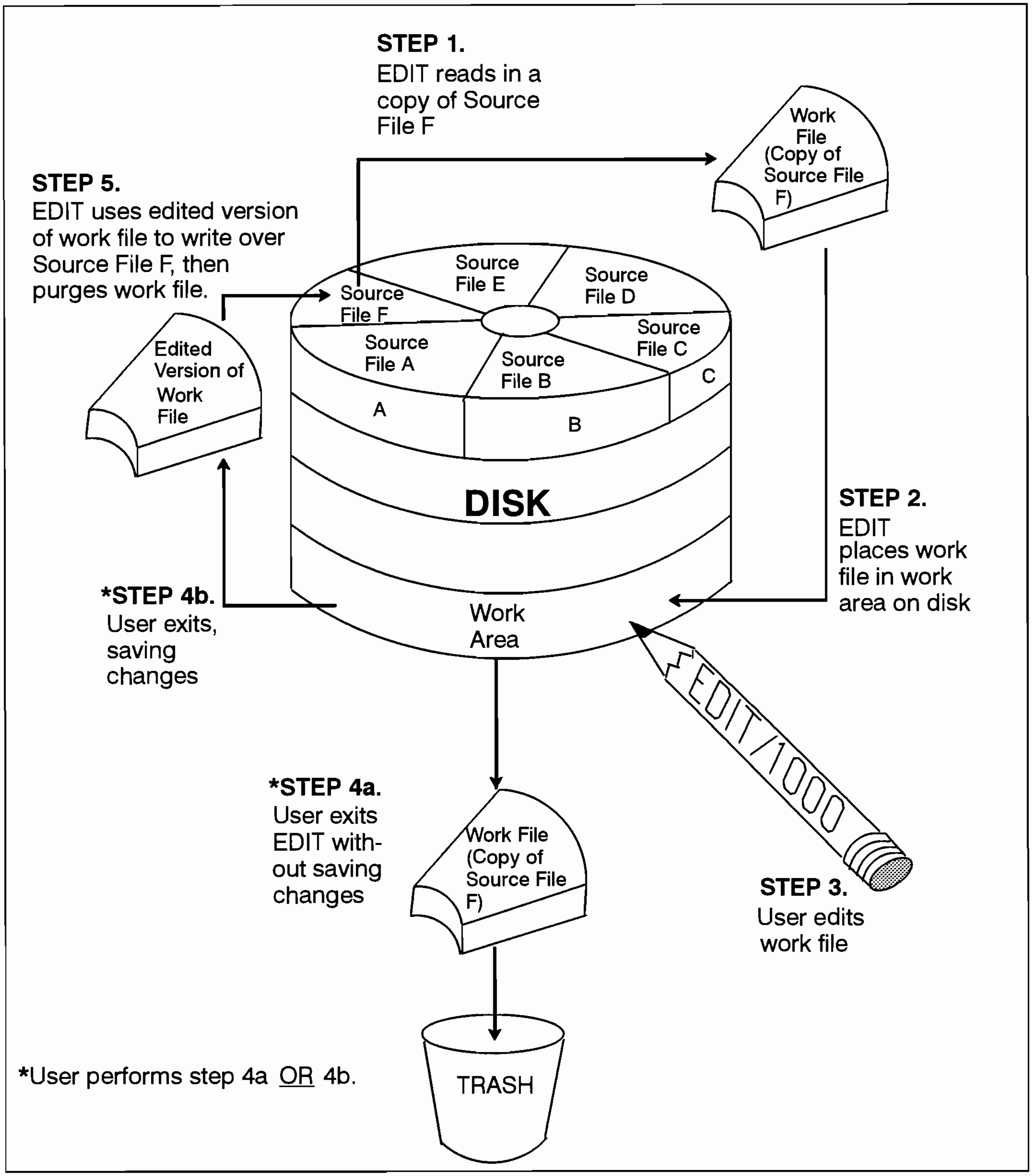
- The EDIT Work File
- How to Use EDIT to Enter Text
- The Line and Screen Editing Modes
- The Pending Line
- EDIT File Size
- File Name Specification
- Dangerous Command Confirmation

The EDIT Work File

When the EDIT/1000 program is run, EDIT reads the source file specified in the runstring and transfers a copy of the file to a work area on disk. This copy is referred to as the work file or *scratch* file. When lines are inserted or deleted, the work file is updated and the line numbering is changed accordingly for each modification.

When editing is completed and EDIT is terminated with one of the exit commands, the content of the work file is used to update (write over) the original source file. If the file originally specified does not exist, EDIT uses the edited work file to create a new source file, assigning the user-specified name. An overview of the EDIT operation for an existing file is illustrated in Figure 1-2.

Under normal termination conditions, the work file is purged after it is used to update or create a file. In the case of a system crash or other abnormal EDIT termination, the work file is not purged and EDIT initiates a recovery operation the next time the EDIT program is run. A new editing session cannot be initiated until the recovery operation is terminated by aborting or exiting EDIT. If the work file is corrupted, it must be purged before another EDIT session can be initiated.



STEP 1. EDIT reads in a copy of Source File F

STEP 2. EDIT places work file in work area on disk

STEP 3. User edits work file

***STEP 4a.** User exits EDIT without saving changes

***STEP 4b.** User exits, saving changes

STEP 5. EDIT uses edited version of work file to write over Source File F, then purges work file.

*User performs step 4a OR 4b.

Figure 1-2. Overview of EDIT Operations for an Existing File

Using EDIT to Enter Text

The EDIT program can be operated in two different ways:

- interactive operation
- batch operation

When the EDIT/1000 program is run interactively, you carry on a dialog with the program, entering text and receiving information and error messages from EDIT, as necessary. When EDIT is run in batch operation, it is driven from a file and is not interactive. Batch operation is discussed in Chapter 3 of this Manual.

The EDIT program is typically run interactively. Interactive operation has two editing modes:

- Line mode – allows you to edit a single line of text at a time and enter line commands that modify a group of lines.
- Screen mode – allows you to view several lines of text at a time and edit using the local terminal keys and control key combinations.

Line mode is the default editing mode. When the EDIT program is initiated, you are placed in line mode, (unless you have indicated that you wish to be placed in screen mode when you start EDIT).

The recommended method for using EDIT is to enter text in the screen mode and move back into line mode to accomplish such activities as: searching and exchanging text strings, deleting several lines of text, displaying quick reference information on command syntax, and merging text from another file. Details on using the screen and line modes are provided in the tutorial in Chapter 2.

Line Mode Editing

In the Line Mode, often referred to as the command line mode, EDIT displays one line of text at a time and the EDIT prompt character is displayed on the following line. The line of text displayed is the text that will be altered by any edits and it is referred to as the current *pending line*. EDIT commands are typed in at the EDIT prompt (/) and are entered into the system by using the carriage return.

When the pending line or text listings are output by EDIT, they are always preceded by two blank spaces. This convention allows room for the EDIT prompt and a single character command, which aligns the new text with that displayed by EDIT. Error and information messages, in contrast, always begin in column 1 so that the difference between an EDIT message and a line of text can be easily determined.

In the line mode example following, the EDIT prompt and the first line of text (the pending line) are displayed when EDIT is first accessed for the file TESTFILE:

```
CI> edit testfile
EDIT : Use ? for help
Opened file TESTFILE::HERMAN:4:96:39
490 lines read.
  The first line of TESTFILE.
/
```

If an error is made when entering a command, you can *erase* it by using the backspace key prior to executing the command with a carriage return. You can also reverse many of the EDIT commands after execution by using the Undo (UN) command.

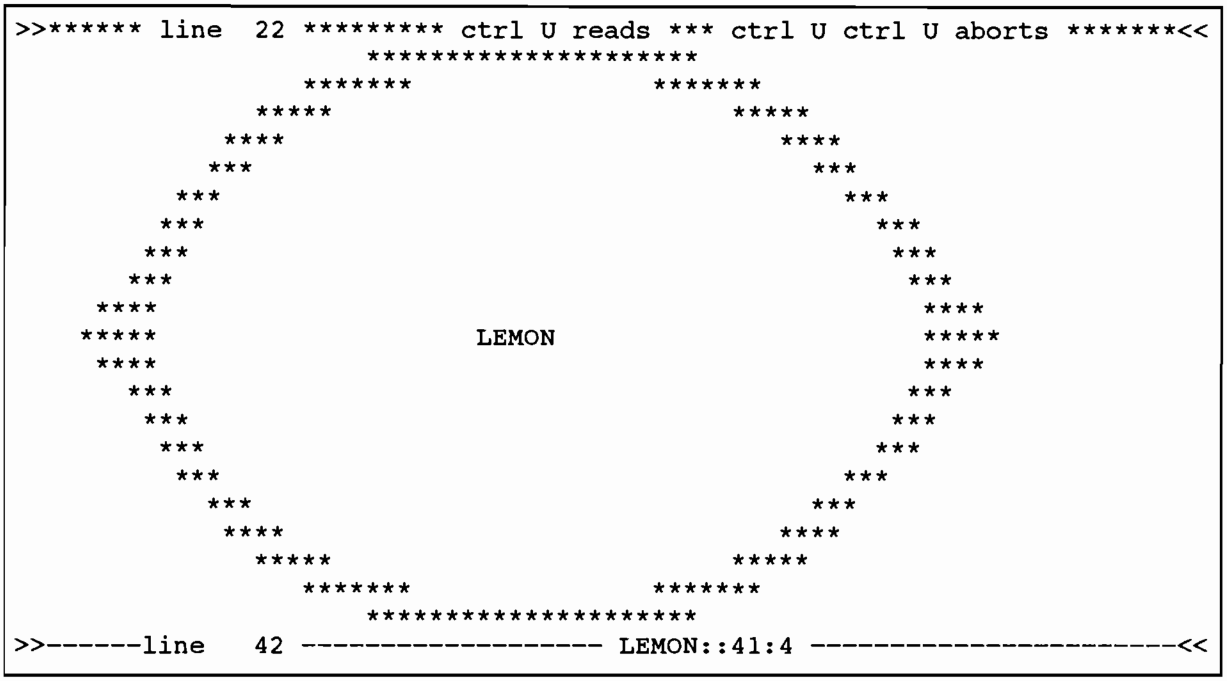
For more on Line Mode Editing, refer to Chapter 3.

Screen Mode Editing

The EDIT screen mode is accessed by entering the Screen (S) command from the EDIT prompt, or by including the S command in the EDIT runstring. In the screen mode, EDIT treats the terminal screen as a window through which you can view several lines of text at a time. Editing is carried out inside this window and when commanded to do so, EDIT reads your screen and places the new edits into the work file. The local terminal keys (cursor arrow keys, insert keys, and delete keys) and control key combinations (CTRL and an alpha key, followed by a carriage return) are used for the editing process.

Many of the control key combinations serve a dual purpose. They perform an action and can be entered once or twice before the carriage return to either save or ignore new edits to the screen. For example, the CTRL-F, carriage return command advances the screen display to the next 20 or so lines of text. If entered as CTRL-F followed by the carriage return, the original screen is read and new edits are saved. If entered as CTRL-F CTRL-F, carriage return the screen is not read by EDIT, new edits are ignored, and the next screen is displayed more quickly.

Screens displayed are bracketed by a beginning line and an ending line that provide the beginning and ending line numbers of the portion of the file being displayed. (See Figure 1-3.) In addition, the beginning line displays the command used to exit the screen mode and return to the line mode (CTRL-U or CTRL-U CTRL-U, followed by a carriage return). The ending line displays the file descriptor of the file being edited, unless you are editing a new file and did not specify a filename when initiating the EDIT program.



```
>>******* line  22 ********* ctrl U reads *** ctrl U ctrl U aborts *******<<
                      ********************
                  *******               *******
               *****                         *****
             ****                               ****
            ***                                   ***
          ***                                       ***
         ***                                          ***
        ***                                            ***
       ***                                              ***
     ****                                                ****
    *****                    LEMON                       *****
     ****                                                ****
       ***                                              ***
        ***                                            ***
         ***                                          ***
          ***                                       ***
            ***                                   ***
             ****                               ****
               *****                         *****
                  *******               *******
                      ********************
>>------line   42 ------------------- LEMON::41:4 ------------------------<<
```

Figure 1-3. Screen Example

To enter text into a new file in screen mode, you must position the cursor on the bottom screen bracket line and use the insert line key on your terminal to insert sufficient lines for text entry. Do not type over the beginning or ending screen brackets or you will receive an error message when you attempt to save the screen edits with a control key combination command (refer to the S command description in Chapter 4).

EDIT typically displays 21 lines of text at a time: 10 lines above the current pending line (or current cursor position), the pending line, and 10 lines below the pending line. A command is available to display the maximum number of lines allowed by the terminal.

Although the screen mode provides an efficient means of editing, the more powerful EDIT commands (such as the Move, Copy, Find, and Exchange commands) must be entered from the line mode. Control key commands are available in screen mode to enable you to escape the screen mode to execute the more powerful line mode commands. All control key commands entered must be followed by a carriage return.

For more information on the screen mode, refer to the tutorial in Chapter 2, Chapter 3, and to the description of the Screen (S) command in Chapter 4.

Pending Line

When EDIT is run on an existing file in the Line Mode, the first line of the work file is displayed. The displayed line is the current line available for editing and is called the *pending line*. This line remains as the pending line until you request a new pending line with an EDIT command. EDIT always maintains a “pending line” pointer into the work file.

A specific line of text can be accessed by entering the line number at the EDIT prompt (/), or by using a search command to locate a line that contains a specific character string. To access the last line of a file, one of the end-of-file characters ($ or >) can be entered at the EDIT prompt.

When a line number is entered, EDIT positions the work file so that the specified line is the pending line. When a search command is entered, EDIT searches the work file until the requested pattern is located. That line then becomes the new “pending line” and is displayed on the terminal.

EDIT File Size

The EDIT/1000 program can be used to edit files that are up to 32,500 lines (records) long. Files that exceed that limit will be truncated when they are read by EDIT.

File Name Specification

In EDIT, files are specified using the standard RTE naming convention (file descriptor or file namr). The file descriptor/namr consists of several user-definable parameters, such as file type, file size, security code (for File Manager files), the directory location of the file, and the system node (location) where the file is stored.

When EDIT is run, certain default parameters are assumed for each file name entered. If the default parameters for the file are acceptable, you enter only the file name when specifying a file. However, if any of the parameters are to be specified in the command, the position of those that are omitted must be indicated by a colon (:).

For more information on file descriptors, refer to the appropriate manual for the RTE system being used.

Dangerous Command Confirmation

EDIT requires execution confirmation for commands that either delete or significantly alter data (that is, *dangerous* commands). This is done with the OK? prompt that is displayed following the entry of any of the following commands: Abort (A), Abort and Save Work File (AS), Delete (D), File Input (FI), Exchange (G & X), Kill (K), Transfer (TR), Unconditional Exchange (U), and Exchange and Search (Y). After editing the file TESTFILE, the command entry and program response for the Abort (A) command would look similar to the following:

```
CI> edit
(editing performed)
/ a
OK? y
EDIT aborted by user
Closed file TESTFILE::HERMAN:4:1
end of edit
CI>
```

The asking feature can be suppressed two different ways. The command entered can be terminated with the EDIT prompt character (for example, a/), as follows:

```
/ a/
EDIT  aborted by user
end of edit
CI>
```

or the Set (SE) command can be used to set the asking option (AS) to OFF, as follows:

```
/ se as off
  Asking...............  AS =off
```

The first method, terminating commands with the EDIT prompt character, is the recommended method because leaving the AS option set to ON provides a safeguard against key stroke mistakes.

Error and Information Messages

Two types of error or information messages are displayed during operation of the EDIT/1000 program: EDIT program errors and RTE system errors. All EDIT/1000 errors and the most common RTE system errors are described in detail in Appendix A, Error and Information Messages.

When EDIT receives a command it does not understand, it reprints the entry followed on the next line by a question mark in column one. EDIT places a caret (^) under the questionable field. Error and informational messages are displayed starting in column one to differentiate them from listings of the file text.

In the following example, a Find (F) command was entered to locate the word *bicycle*. The range of lines specified (line 2 to line 1) was illegal because the start of the range (line 2) was larger than the end of the range (line 1). EDIT repeated the erroneous command, printed a question mark and a caret to indicate the location of the error and displayed an error message (Start > stop) in column 1.

```
/ 2 1 f/bicycle/
/2 1 f/bicycle/
?  ^
Start > stop
/
```

Mistakes can be easily corrected while editing because EDIT's online quick reference is always available with information on the command syntax, the purpose of each command, and the command defaults. In addition, EDIT supplies a command stack for modifying and re-executing commands and an undo command (UN) that reverses the effects of the command previously executed. Finally, EDIT requires execution confirmation (in the form of the OK? prompt) for all commands that delete or significantly alter data.

Error and informational messages are further described in Chapter 3 of this Manual. A detailed listing of the messages with descriptions and suggested action is included in Appendix A of this Manual.

2

Getting Started Tutorial

Introduction

This chapter provides a tutorial for the beginning EDIT/1000 user. The tutorial is divided into three parts that can be read separately or all in one sitting, as your schedule permits:

- Part 1 – Screen Mode Editing
- Part 2 – Line Mode Commands
- Part 3 – Efficient Editing

Figure 2-1 illustrates the topics covered under each part of the tutorial.

This chapter also includes a set of quick reference task charts that provide a collection of commands used to accomplish specific tasks, such as Displaying Information and Moving Around the Editing Screen.

The information in this chapter will enable you to

- Run the EDIT/1000 program (EDIT command) and enter the name of an existing file to be edited (FI, File Input command).
- Use the Abort (A), Exit and Replace (ER), and Exit and Create (EC) commands to exit the EDIT program.
- Create a sample file using the screen mode and selected line mode commands.
- Use the Screen (S) command to enter EDIT/1000's screen mode and use the control key commands to move between screens, display a larger screen, and exit the screen mode.
- Use the local terminal keys to insert and delete text.
- Use the CTRL-C (^ C) key combination to temporarily exit the screen mode to execute one line mode command.
- Use the H or ? (Help) command to access EDIT's online reference, and the SH (Show) command to display the current settings for EDIT/1000 session options.

- Use the N (Line Number) command to find out where you are, and the n (Specific Line Number) to move to a specific line.
- Use the L (List) and W (List Window) commands to display a portion of a file.
- Use the Kx (Mark Line) command to mark a line or block of lines.
- Save or create a file before making major changes using the WR (Write and Replace) or WC (Write and Create) commands.
- Copy text using the CO (Copy) command, move text using the MO (Move) command, delete text using the K (Kill) command, and find and replace text using the search (F, B) and exchange (X, G) commands.
- Reverse EDIT commands using the UN (undo) command.
- Merge all or a portion of one file into another using the M (Merge) command.
- Repeat a previously-executed command without retyping by using the command stack review (/) feature.

Prerequisites

This tutorial assumes that you have read Chapter 1 EDIT/ 1000 Basics and that you know how to use the editing keys on your terminal (cursor arrows, insert and delete keys). You can work through the tutorial without reading Chapter 1, but your understanding of the capabilities of EDIT/1000 may not be as complete. Finally, this tutorial assumes that you are logged on to the RTE CI user interface and are ready to run the EDIT/1000 program.

Before you start, make sure that the EDIT/1000 program has been loaded onto your RTE system and that it is ready to be run. If it is not, ask your System Manager to load it for you.

The procedures and examples provided in this chapter represent common means of accomplishing the various tasks illustrated. However, there are alternate methods to accomplish the tasks and you are encouraged to try different methods to develop your skills in using EDIT. Figure 2-1 illustrates the contents of the three-part tutorial.

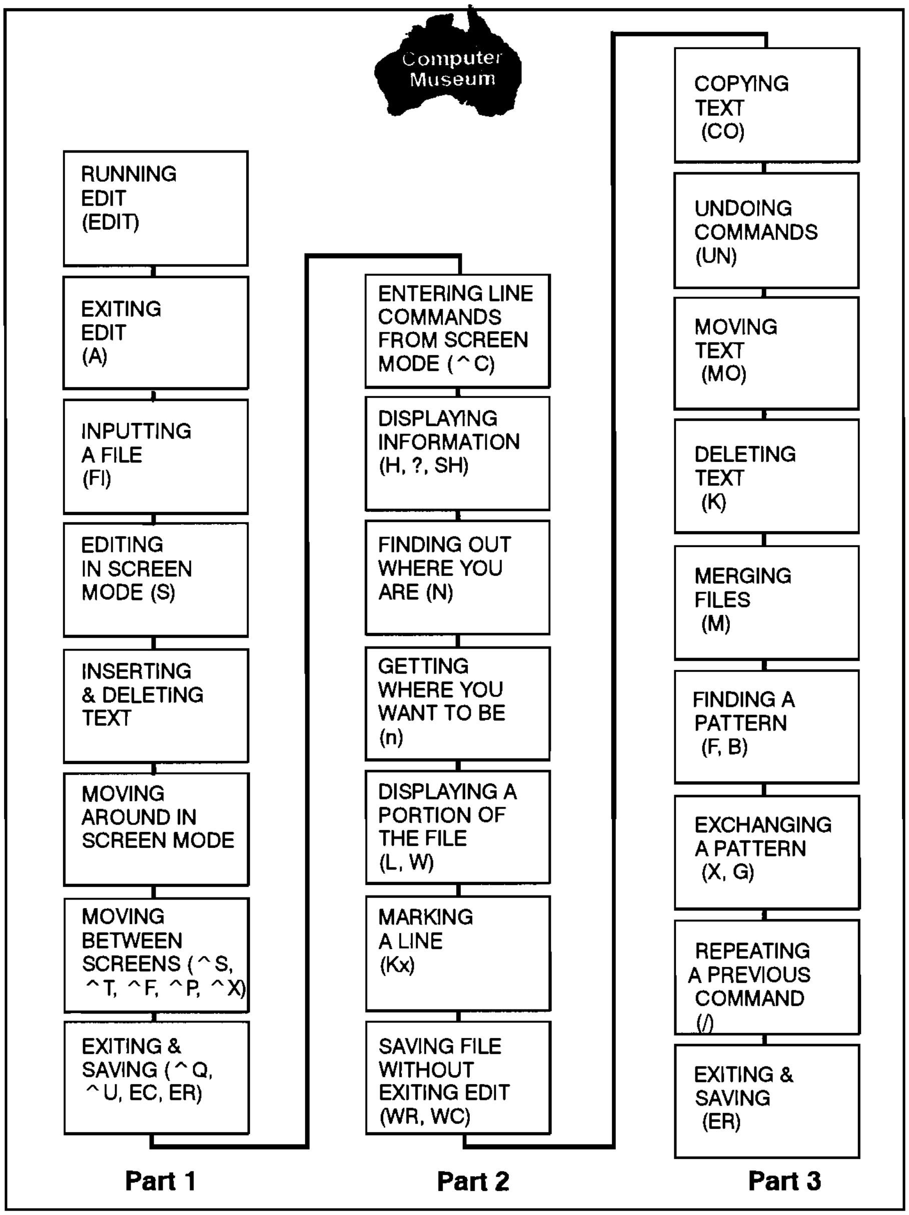

RUNNING EDIT (EDIT)
EXITING EDIT (A)
INPUTTING A FILE (FI)
EDITING IN SCREEN MODE (S)
INSERTING & DELETING TEXT
MOVING AROUND IN SCREEN MODE
MOVING BETWEEN SCREENS (^S, ^T, ^F, ^P, ^X)
EXITING & SAVING (^Q, ^U, EC, ER)
ENTERING LINE COMMANDS FROM SCREEN MODE (^C)
DISPLAYING INFORMATION (H, ?, SH)
FINDING OUT WHERE YOU ARE (N)
GETTING WHERE YOU WANT TO BE (n)
DISPLAYING A PORTION OF THE FILE (L, W)
MARKING A LINE (Kx)
SAVING FILE WITHOUT EXITING EDIT (WR, WC)
COPYING TEXT (CO)
UNDOING COMMANDS (UN)
MOVING TEXT (MO)
DELETING TEXT (K)
MERGING FILES (M)
FINDING A PATTERN (F, B)
EXCHANGING A PATTERN (X, G)
REPEATING A PREVIOUS COMMAND (/)
EXITING & SAVING (ER)
Part 1
Part 2
Part 3

Figure 2-1. EDIT/1000 Tutorial

Tutorial Part 1 – Screen Mode Editing

Contents and Sample File

This portion of the tutorial provides instructions on using EDIT/1000's screen mode to edit text. The flow chart in Figure 2-2 illustrates the contents of Part 1 of the tutorial.

Throughout this tutorial, a pointing hand symbol and italics are used to highlight directions on how to create a sample file and use EDIT commands to manipulate it.

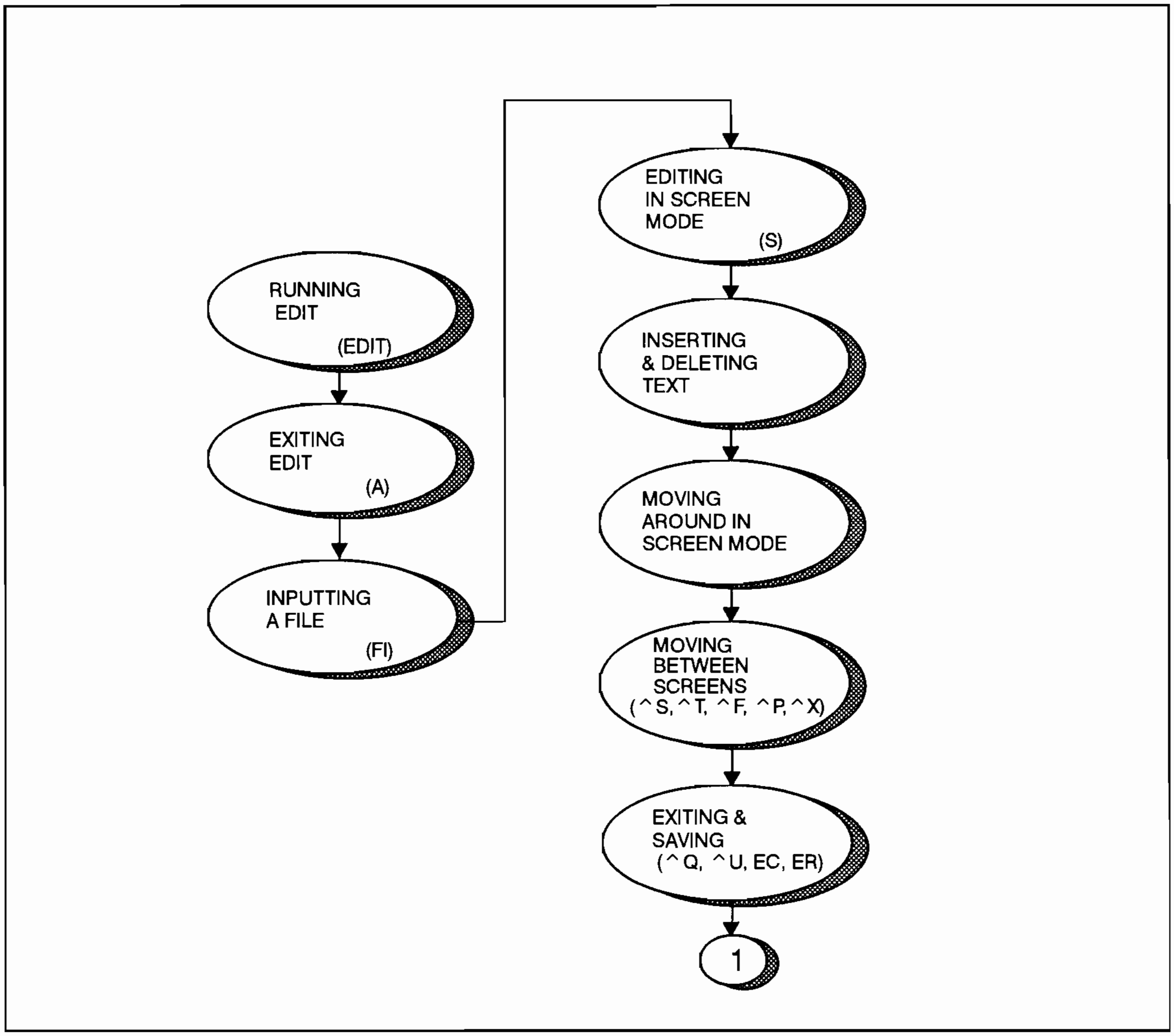
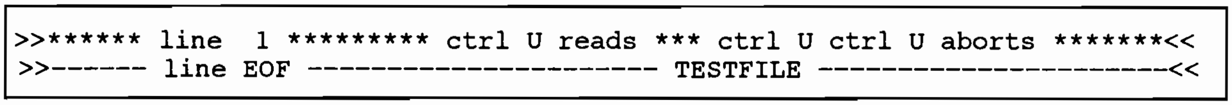

RUNNING EDIT (EDIT)
EXITING EDIT (A)
INPUTTING A FILE (FI)
EDITING IN SCREEN MODE (S)
INSERTING & DELETING TEXT
MOVING AROUND IN SCREEN MODE
MOVING BETWEEN SCREENS (^S, ^T, ^F, ^P, ^X)
EXITING & SAVING (^Q, ^U, EC, ER)
1

Figure 2-2. Tutorial Map, Part 1 – Screen Mode Editing

Quick Start Exercise

This two-page quick start exercise provides a brief introduction to working with EDIT/1000, and should give you a feel for using the EDIT program to create a text file. Table 2-1 illustrates the tasks involved in creating a text file. Specific instructions are listed to the right of each task. The tutorial which follows this exercise will provide more detail on text entry and manipulation using the EDIT/1000 program.

To create a file, enter the EDIT runstring from the RTE user interface prompt (in this example, CI>). Once in the EDIT line mode, enter the S (Screen Mode) command to start screen mode. You know you are in screen mode when you see the top and bottom screen brackets illustrated in Figure 2-3.



```
>>******* line  1 ********* ctrl U reads *** ctrl U ctrl U aborts *******<<
>>------ line EOF ---------------------- TESTFILE -----------------------<<
```

Figure 2-3. Example of Screen Brackets

Before entering text in the screen mode, you must insert blank lines between the screen brackets, using the insert line (INS LINE) key on your terminal keyboard. As you type in text, use a carriage return after each line and do not type over or delete either the top or bottom screen bracket.

After entering text you must exit the screen mode. To create the new file, you use a line mode command to save the new file and exit the EDIT program.

Table 2-1. EDIT/1000 Quick Start Exercise

TASK	INSTRUCTIONS
1. Start the EDIT program from the RTE-A CI user interface (prompt = CI>).	1. At the CI> prompt, enter the *EDIT* command, a space, and the file name *NEWFILE.* Then press the carriage return: `CI> edit newfile`
2. Enter the EDIT screen mode.	2. At the EDIT prompt (/), enter the S (Screen) command, followed by a carriage return: `/s`
3. Type your name 23 times, each time on a different line.	3. Using the cursor arrow keys on your terminal, position the cursor on the bottom screen bracket. Use the INS LINE key on your terminal to insert 25 blank lines between the top and bottom screen brackets. Type your name, followed by a carriage return. Repeat 22 times.
4. Go back to the first line of the text.	4. Enter the Control–P command to go to the previous screen, as follows: Press and hold the CTRL key on your terminal keyboard, then press the P key. Release both keys and press the carriage return.
5. Exit the screen mode, then exit EDIT and save your new file.	5. Exit the screen mode using the Control–U command, as follows: Press and hold the CTRL key on your terminal keyboard, then press the U key. Release both keys and press the carriage return. Exit EDIT using the ER (Exit and Replace) command at the line mode prompt, followed by a carriage return: `/er`

Running EDIT (EDIT)

The EDIT/1000 program can be accessed from either the RTE Command Interpreter (CI) or the RTE File Manager (FMGR) user interface. The examples provided in this tutorial use the CI prompt, CI>. Note that on some systems, the CI prompt includes a decimal point and a number. This is your CI program number and it is based on your session number.

The runstring, *EDIT,* initiates the EDIT/1000 program. There are many variations of the EDIT runstring that you can use to execute commands and set options. The most common variation is to include a filename in the EDIT runstring to access a specific text file. A space or a comma must be used between the EDIT command and the filename. (If you are using FMGR, you must use a comma.) For more on the EDIT runstring and its capabilities, refer to Chapter 3 EDIT/1000 Operations.

The following example illustrates the use of the EDIT runstring from the Command Interpreter (CI), and the typical system response:

```
CI> edit
EDIT : Use ? for help
FI,<file name> specifies file to edit.
EOF
/
```

The prompt character for the EDIT/1000 line mode is the slash mark (/). This character can be changed using the EDIT SE (Set) command. For instructions on changing the prompt character, refer to the description of the SE command in Chapter 4 EDIT/1000 Commands.

The following example illustrates the entry of the EDIT runstring with a specific filename:

```
CI> edit first
EDIT : Use ? for help
Opened file FIRST::HERMAN:4:4:36
28 lines read.
  This is line one of the file FIRST.
/
```

If the filename entered does not exist, EDIT allows you to enter the editing session but informs you that the file does not exist and that you can enter text and create the file by exiting the session with an ER (Exit and Replace) command. (This command is described in detail at the end of Part 1 of the tutorial.) The response displayed by EDIT in this situation follows:

```
CI> edit new
EDIT : Use ? for help
No such file NEW
An ER or the first WR will create it.
EOF
/
```

If you are logged on and have a CI> prompt, enter the command EDIT TESTFILE *and press the carriage return.*

Exiting EDIT without Saving (A)

If you change your mind about editing, or have made changes that you don't want to save, you can use the A (Abort) command to exit the EDIT program. The A command returns you to the RTE user interface (CI or FMGR) and ignores any changes made to the file you accessed. If editing a new file, the A command exits EDIT without creating the file.

The A command is considered a dangerous command because it has the potential to delete data that you may want to save. For this reason, EDIT displays an OK? prompt that requires you to confirm the abort by typing in *yes* or a *y*. EDIT displays the OK? prompt only if changes have been made to the work file.

Enter the Abort command (A) at the EDIT prompt (/). Because you did not make changes to your file, EDIT will not prompt you for confirmation of the abort and the editing session will be terminated without creating the file TESTFILE. (You will create it under the section on Inserting and Deleting Text, later in Part 1 of the tutorial.)

The following example illustrates the use of the abort command from a new file to which no changes were made:

```
CI> edit new
EDIT : Use ? for help
No such file NEW
An ER or the first WR will create it.
/ a
EDIT aborted by user
end of edit
CI>
```

The following example illustrates the use of the abort command from an existing file to which no changes were made:

```
CI> edit first
EDIT : Use ? for help
Opened file FIRST::HERMAN:4:4:36
28 lines read.
  This is line one of the file FIRST.
/ a
EDIT aborted by user
Closed file FIRST::HERMAN:4:4:36
end of edit
CI>
```

The example which follows illustrates the system response when the abort command is used from an existing file to which changes were made:

```
CI> edit first
EDIT : Use ? for help
Opened file FIRST::HERMAN:4:4:36
28 lines read.
  This is line one of the file FIRST.
(changes made)
/ a
OK? y
EDIT aborted by user
Closed file FIRST::HERMAN:4:4:36
end of edit
CI>
```

The abort (A) command is only one of three commands used to exit the EDIT/1000 program. The EC (Exit and Create) and ER (Exit and Replace) commands enable you to exit edit and save the changes made to the new or existing file. The EC and ER commands are described at the end of this portion of the tutorial. A fourth command, AS (Abort and Save Work File), is not normally used and is described in Chapter 4 of this manual.

Inputting a File (FI)

If the EDIT runstring is entered without designating a specific file to be edited, you have the option to use the FI (File Input) command to edit an existing file or a new file. The FI command is entered at the line mode prompt (default = /) and is followed by a carriage return.

If you choose to edit an existing file, use the FI command followed by the filename. For example:

```
CI> edit
EDIT : Use ? for help
FI,<file name> specifies file to edit.
EOF
/ fi first
Opened file FIRST::HERMAN:4:4:36
28 lines read.
  This is line one of the file FIRST.
/
```

The FI command accesses the desired file and positions you on the first line of the file, ready for editing.

There are two ways to edit a new file. You can designate a new filename when you start the EDIT session (either in the EDIT runstring or after the program is initiated, via the FI command) and then exit EDIT with the ER (Exit and Replace) or WR (Write and Replace) command. The second way to edit a new file is to run EDIT without specifying a filename, type in text, and create the file by terminating EDIT with either the EC (Exit and Create) or WC (Write and Create) command, plus a filename.

The recommended method of editing a new file is to specifying the filename when you start EDIT. This enables you to exit EDIT entering only the ER command, without a filename. The commands EC, WC, ER, and WR are described in this tutorial.

☞ *Enter the* EDIT *command at the RTE user interface prompt. Then enter the command* FI TESTFILE *at the EDIT prompt. You should receive the message*

```
No such file TESTFILE
An ER or the first WR will create it.
EOF
```

followed by the EDIT prompt. You are now ready to access the screen mode and enter a file.

Editing in Screen Mode (S)

As described in Chapter 1 EDIT/1000 Basics, EDIT has two editing modes, line mode and screen mode. Line mode is used to enter line commands such as the screen mode command, and the commands to search for and exchange text, copy and move text, and merge in text from another file. You know you are in line mode when you see the EDIT prompt (default = /). Unless you specify the screen mode in the EDIT runstring, you will be placed in the EDIT line mode.

You will typically use the screen mode to enter text, since it speeds up the editing process and allows you to use the local terminal keys (cursor arrow keys and insert and delete keys) to move about the screen and insert and delete lines and characters.

The S (Screen) command is used to access EDIT's screen mode. You know you are in screen mode when you see the top and bottom screen brackets illustrated in Figure 2-4. The S can be entered for an existing file as a part of the EDIT runstring:

```
CI> edit existingfile,s
```

or it can be entered at the first EDIT prompt:

```
CI> edit

EDIT : Use ? for help
FI, <filename> specifies file to edit.
EOF
/ fi existingfile
Opened file EXISTINGFILE::HERMAN:4:4:36
28 lines read.
  This is the first line of EXISTINGFILE.
/ s
```

To access the screen mode without specifying a filename, enter the command

```
CI> edit,,s
```

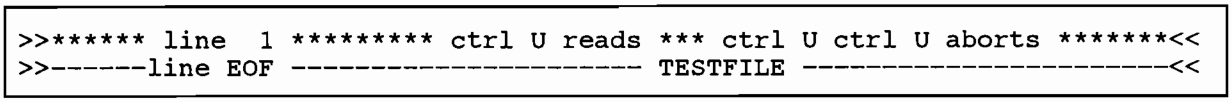
EDIT's screen mode treats the terminal screen as a window through which you can view and make changes to a section of the file text. When you enter a command to move to another screen or to exit screen mode, EDIT reads the screen and updates the work file. The screen is read unless you use a command that specifies that changes made since the last screen read are to be ignored.

EDIT typically displays 21 lines of text at a time: 10 lines above the current pending line, the current pending line, and 10 lines below the current pending line. The EDIT option that controls how many lines are displayed is SD (Screen Default), and this option can be changed by using the SE (Set) command. (Refer to the description of the SE command in Chapter 4 EDIT/1000 Commands.)

Your terminal memory limits the number of lines that can be displayed at a time. The terminal limit can be viewed by using the SH (Show) command to display the SL (Maximum Screen Mode Lines) option.

To access the screen mode, enter the S (Screen) command at the EDIT prompt, followed by a carriage return. A beginning and ending screen bracket should be displayed at the top of the screen.

The EDIT text screen is bracketed by a beginning line and an ending line which show the line numbers of the first and last lines of text displayed on the screen. The beginning line also displays the command that must be used to exit screen mode and return to line mode (CTRL-U to save the new screen edits or CTRL-U CTRL-U to ignore new screen edits, both followed by a carriage return). Text is entered *between* the beginning and ending screen brackets. Refer to Figure 2-4 for an example of the screen brackets, and to the next section for instructions on entering text.



```
>>****** line  1 ********* ctrl U reads *** ctrl U ctrl U aborts *******<<
>>------line EOF ---------------------- TESTFILE -----------------------<<
```

Figure 2-4. Example of Screen Brackets

Note that in Figure 2-4, the bottom screen bracket indicates that you are at the end of the file (EOF). The bottom screen bracket also provides the filename for the file being edited (TESTFILE, in this example). If you are editing an existing file, the bottom bracket will also include the name of your RTE working directory and several numbers that indicate characteristics of your file (such as file size and type).

Inserting and Deleting Text

The EDIT/1000 screen mode enables you to use the local terminal editing keys to enter text. Keys marked as INS LINE and INS CHAR are used to insert lines and characters, respectively. On most terminals, pressing the INS CHAR key allows you to insert characters at the location of the cursor until the key is pressed again. Keys marked as DEL LINE and DEL CHAR are used to delete lines and characters, respectively. On some terminals, lines and characters are inserted and deleted using the soft (function) keys rather than marked keys.

Before typing in lines of new text, you must insert several blank lines. To do this, position the cursor on the bottom screen bracket and use the INS LINE key/function key. Insert enough lines so that you do not type over the beginning or ending screen brackets. If the screen brackets are deleted or typed over, EDIT is unable to recognize the text boundaries and displays the following error message when you attempt to save the screen edits:

```
Start and/or Stop line not found
O saves original text written to screen
S saves text just read from screen
B saves both (inserts screen text before original text)
What should be saved?
```

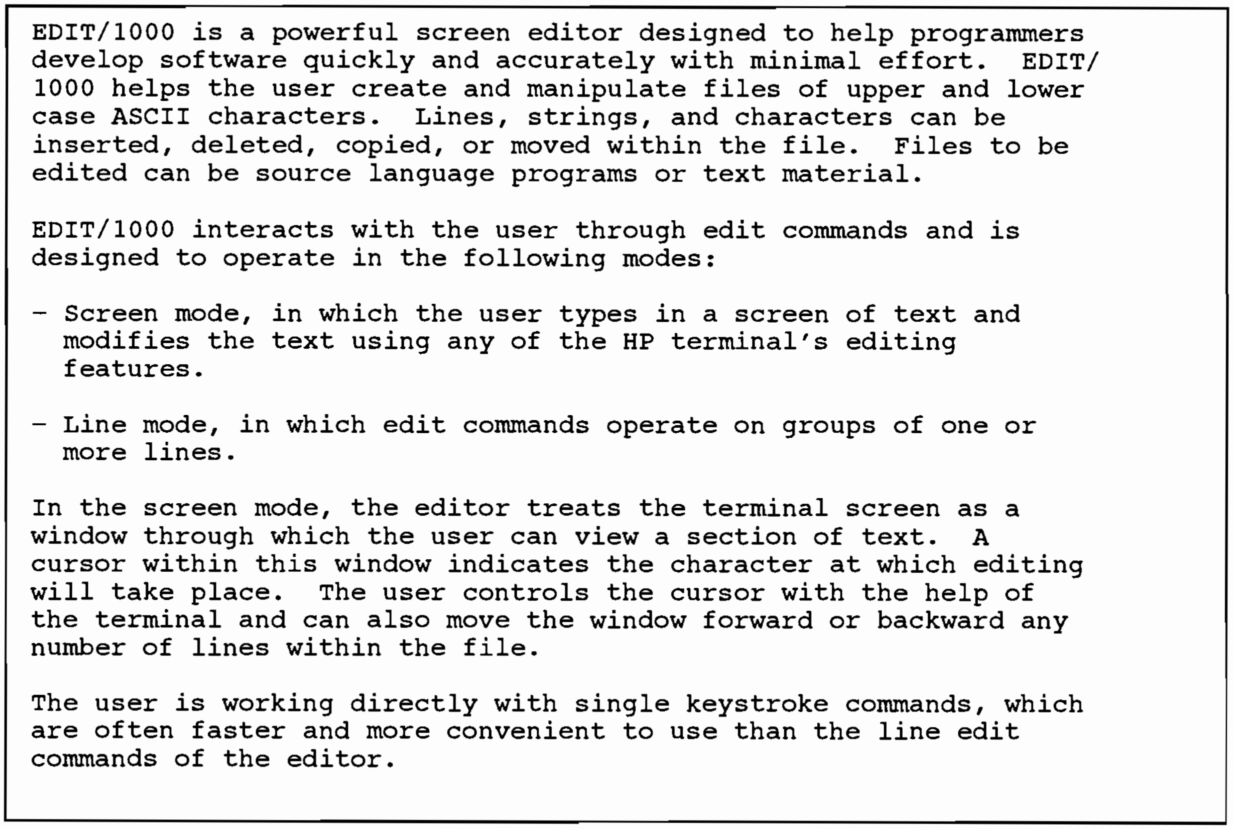
The safest response is to enter a B if you are not sure about the status of your text. This saves both the original text and the text that was currently being displayed. You can then delete whatever text is erroneous or redundant.

EDIT does not provide a wrap-around facility. You must therefore use a carriage return at the end of each line. If you continue typing, most terminals will wrap the text. However, the text is wrapped after the 78th character and not necessarily between words.

Note

You can continue to insert blank lines for text entry even if the beginning screen bracket is rolled off the top of your terminal screen, as long as you do not have more than 24 consecutive blank lines when the screen is read. The screen is read by EDIT when you enter a control key command to either move around the text file or exit the screen mode. (See following sections.) If you must have more than 24 blank lines in your text, enter a blank space on each line.

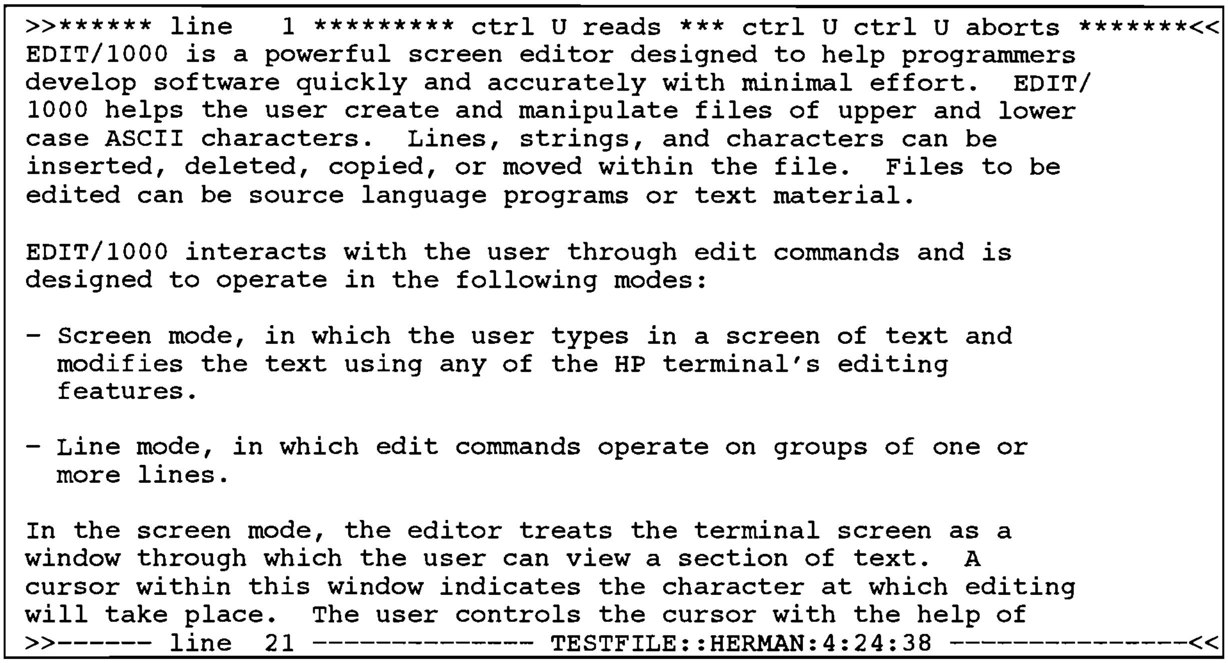
Type in the sample text illustrated in Figure 2-5. Practice using the insert line key to add lines before typing text, the cursor arrow keys to move around the screen, the insert character key, and the delete character and line keys. Remember to use a carriage return at the end of each line, and do not type over the beginning or ending screen brackets.



```
EDIT/1000 is a powerful screen editor designed to help programmers
develop software quickly and accurately with minimal effort.  EDIT/
1000 helps the user create and manipulate files of upper and lower
case ASCII characters.  Lines, strings, and characters can be
inserted, deleted, copied, or moved within the file.  Files to be
edited can be source language programs or text material.

EDIT/1000 interacts with the user through edit commands and is
designed to operate in the following modes:

- Screen mode, in which the user types in a screen of text and
  modifies the text using any of the HP terminal's editing
  features.

- Line mode, in which edit commands operate on groups of one or
  more lines.

In the screen mode, the editor treats the terminal screen as a
window through which the user can view a section of text.  A
cursor within this window indicates the character at which editing
will take place.  The user controls the cursor with the help of
the terminal and can also move the window forward or backward any
number of lines within the file.

The user is working directly with single keystroke commands, which
are often faster and more convenient to use than the line edit
commands of the editor.
```

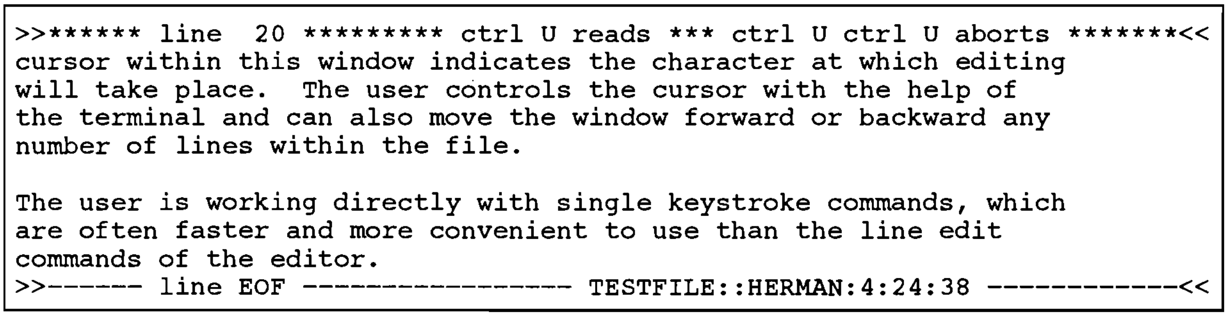
Figure 2-5. TESTFILE - Sample Text File for the Tutorial - Part 1

Because the sample text is more than 21 lines long, EDIT displays it in two screens. The first and second screens of the sample text are displayed in Figure 2-6 and Figure 2-7:



```
>>******* line   1 ********* ctrl U reads *** ctrl U ctrl U aborts *******<<
EDIT/1000 is a powerful screen editor designed to help programmers
develop software quickly and accurately with minimal effort.  EDIT/
1000 helps the user create and manipulate files of upper and lower
case ASCII characters.  Lines, strings, and characters can be
inserted, deleted, copied, or moved within the file.  Files to be
edited can be source language programs or text material.

EDIT/1000 interacts with the user through edit commands and is
designed to operate in the following modes:

- Screen mode, in which the user types in a screen of text and
  modifies the text using any of the HP terminal's editing
  features.

- Line mode, in which edit commands operate on groups of one or
  more lines.

In the screen mode, the editor treats the terminal screen as a
window through which the user can view a section of text.  A
cursor within this window indicates the character at which editing
will take place.  The user controls the cursor with the help of
>>------ line  21 -------------- TESTFILE::HERMAN:4:24:38 ---------------<<
```

Figure 2-6. TESTFILE Screen Display - First Screen



```
>>****** line  20 ********* ctrl U reads *** ctrl U ctrl U aborts *******<<
cursor within this window indicates the character at which editing
will take place.  The user controls the cursor with the help of
the terminal and can also move the window forward or backward any
number of lines within the file.

The user is working directly with single keystroke commands, which
are often faster and more convenient to use than the line edit
commands of the editor.
>>------ line EOF ----------------- TESTFILE::HERMAN:4:24:38 ------------<<
```

Figure 2-7. TESTFILE Screen Display - Second Screen

Note the two line overlap between screens. EDIT typically displays two lines from the previously displayed screen. (In this case, EDIT starts the display of the second screen with line 20). The command used to go from the first to the second screen, and vice versa, is presented under the section on Moving Between Screens, which follows.

Moving Around in Screen Mode

The EDIT/1000 screen mode provides control key combinations that enable you to move around in the screen mode. In this manual, the CONTROL key is typically abbreviated with the up caret symbol (^) or the letters CTRL.

Note

To execute control key commands, hold down the CTRL key and press the appropriate alpha key, then release both keys. *Follow this by a carriage return.* Be sure to release the CTRL key before pressing the carriage return.

Many control key combinations serve a double purpose. A single execution of a control key combination reads the current screen of text, saving all new edits. For example, a single execution of the command to move forward to the next screen of text, (^F followed by a carriage return), saves the screen as it appears just prior to execution of the command.

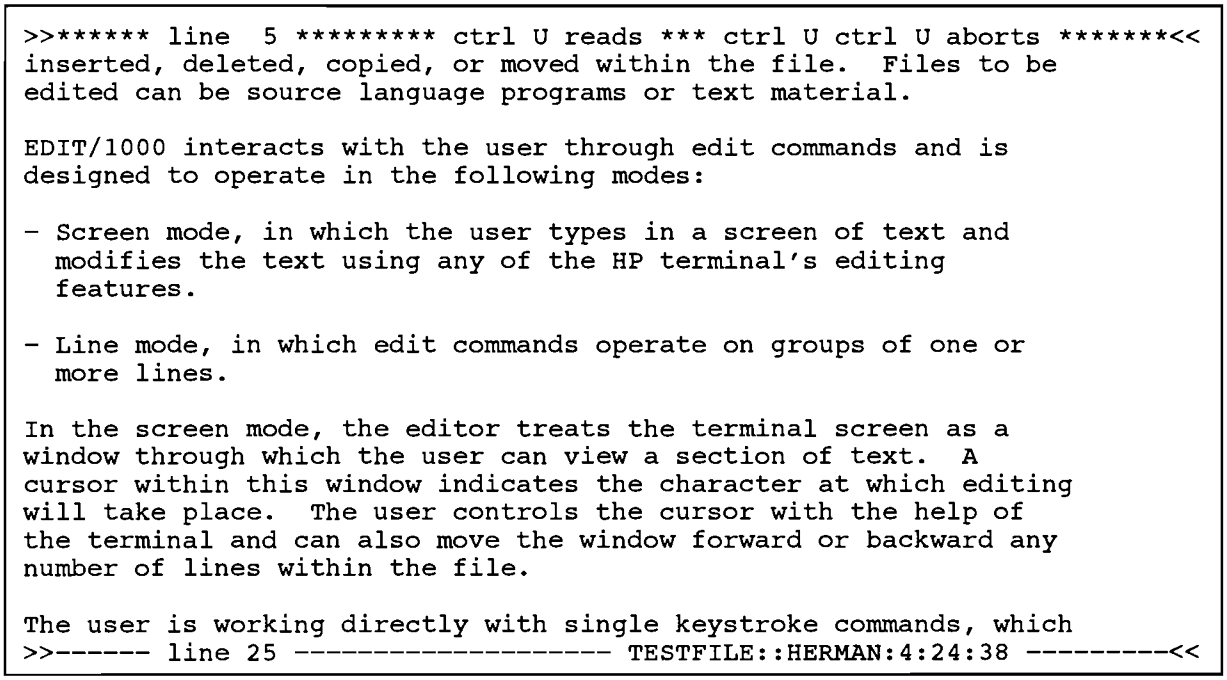
A double execution of the control command, (for example, ^F^F followed by a carriage return), executes the command and ignores edits made to the screen. When you do not want the screen read, use the double control command to reduce the time required to move around in the text file.

This tutorial describes the use of the following control key combination commands: ^F, ^P, ^X, ^S, ^T, ^Q, and ^U. Several additional control key commands are available. For more information, refer to the description of the Screen (S) command in Chapter 4.

Moving between Screens (^F, ^P, ^S, ^T, ^X)

EDIT allows you to save or ignore text entered in the work file during the editing process. This is done when you advance a screen or go back to a previous screen. To move forward one screen, use the ^F (for forward) key combination. This displays the next 21 lines (including a two line overlap) and saves the changes made to the original screen. To move back one screen, use the ^P (for previous) key combination. This displays the 21 lines of text (including a two line overlap) preceding the current screen and saves the changes made to the current screen.

The ^T key combination displays 21 lines of text from the line that the cursor is positioned on at the time of the command. For example, if the cursor is positioned on line five of the sample text and the ^T combination is used, the screen displayed begins with line five, as illustrated in Figure 2-8.



```
>>****** line  5 ********* ctrl U reads *** ctrl U ctrl U aborts *******<<
inserted, deleted, copied, or moved within the file.  Files to be
edited can be source language programs or text material.

EDIT/1000 interacts with the user through edit commands and is
designed to operate in the following modes:

- Screen mode, in which the user types in a screen of text and
  modifies the text using any of the HP terminal's editing
  features.

- Line mode, in which edit commands operate on groups of one or
  more lines.

In the screen mode, the editor treats the terminal screen as a
window through which the user can view a section of text.  A
cursor within this window indicates the character at which editing
will take place.  The user controls the cursor with the help of
the terminal and can also move the window forward or backward any
number of lines within the file.

The user is working directly with single keystroke commands, which
>>------ line 25 -------------------- TESTFILE::HERMAN:4:24:38 ---------<<
```

Figure 2-8. TESTFILE Screen Display After Use of ^S

The ^S key combination can be used instead of the ^T on terminals that do not use Xon/Xoff handshake protocol. Because ^S can cause problems with Xon/Xoff handshake protocol, the examples in this tutorial use ^T.

EDIT/1000 typically displays a 21-line screen. Your terminal probably has the capacity to display more lines, and you may therefore wish to use the ^X key combination to display an extra large screen.

Refer to the description of the S command in Chapter 4 EDIT/1000 Commands for more control key combinations.

Use the control key combinations to move around your new file. Use single and double executions to save or ignore edits, respectively. Practice moving from screen to screen. Your editing efficiency will increase as you increase your skill in moving around your text file.

Exiting & Saving (^Q, ^U, EC, ER)

When you are finished editing and are ready to save your work, use the ^U key combination to quit the screen mode. As with the other control key combinations, a single execution of ^U saves changes made to the current screen, while a double execution ignores new changes. The ^Q key combination can be used instead of the ^U key combination on terminals that do not use Xon/Xoff handshake protocol. Because ^Q can cause problems with Xon/Xoff handshake protocol, the examples in this tutorial use ^U.

The slash (/) prompt indicates that the screen mode has been successfully exited and that you are now in line mode. At this point, your work can be saved as a new file, or the original file can be updated with the new edits. A new file is created using the EC (Exit and Create) command, while the ER (Exit and Replace) command updates the original text file.

EDIT protects your original text file by applying the changes made during the editing session to a work file, which begins as a copy of your text file. When the editing session is terminated, the work file is cleared.

Terminate your editing session with the ER (Exit and Replace) command. Because you entered a filename when you first started the EDIT session, EDIT will allow you to use the ER command instead of the EC command, and will not require you to enter the filename again when exiting the session.

Both the EC and ER commands return you to the RTE user interface. The following is an example of the EC command (note that EDIT does not accept the EC command without a filename):

```
/ ec testfile
Created file TESTFILE::HERMAN:4:24
Closed file TESTFILE::HERMAN:4:24
end of edit
CI>
```

Following is an example of the ER command:

```
/ er
Closed file TESTFILE::HERMAN:4:24:38
end of edit
CI>
```

Warning

EDIT limits the number of records/lines in a file to 32,500. If you attempt to read in a larger file, EDIT displays a warning that the read was stopped before the end of the file was found. If you then execute an ER command, the file is truncated without warning. If you receive the *Line limit reached* message while reading in a file, restart the EDIT session and use the Merge (M) command to create several smaller files.

Part 2 – Line Mode Commands

Contents and Sample File

This portion of the tutorial provides instructions on using EDIT/1000 line mode commands. The flow chart in Figure 2-9 illustrates the contents of Part 2 of the tutorial. The sample file illustrated in Figure 2-10 has been used for the examples in this section.

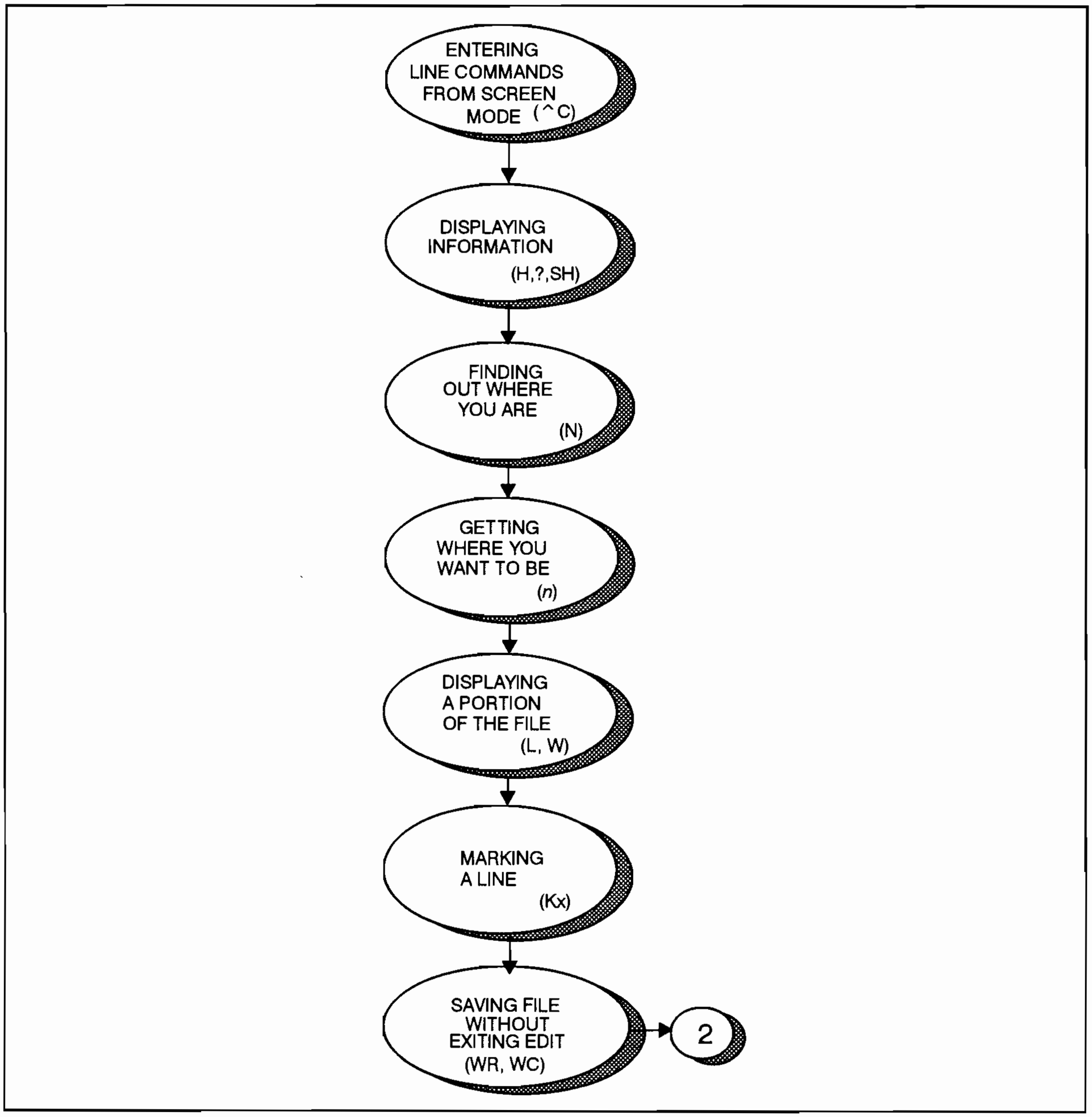

ENTERING LINE COMMANDS FROM SCREEN MODE (^C)
DISPLAYING INFORMATION (H,?,SH)
FINDING OUT WHERE YOU ARE (N)
GETTING WHERE YOU WANT TO BE (*n*)
DISPLAYING A PORTION OF THE FILE (L, W)
MARKING A LINE (Kx)
SAVING FILE WITHOUT EXITING EDIT (WR, WC)
2

Figure 2-9. Tutorial Map, Part 2 - Line Commands From Screen Mode

If you would like to try the exercises described in this tutorial, the sample file used is the same as the sample text file entered in Part 1 of the tutorial. If you did not enter the sample text file in Part 1 of the tutorial, use the following command from the RTE CI user interface to enter EDIT:

```
CI> edit
```

Next, enter the S (Screen) command at the EDIT prompt to access the screen mode. With the cursor positioned on the bottom screen bracket line, use the INS LINE key on your terminal to insert enough blank lines to type in the sample text, illustrated in Figure 2-10. Use the INS CHAR, DEL CHAR, INS LINE, and DEL LINE keys, as necessary, until the text is complete.

```
EDIT/1000 is a powerful screen editor designed to help programmers
develop software quickly and accurately with minimal effort.  EDIT/
1000 helps the user create and manipulate files of upper and lower
case ASCII characters.  Lines, strings, and characters can be
inserted, deleted, copied, or moved within the file.  Files to be
edited can be source language programs or text material.

EDIT/1000 interacts with the user through edit commands and is
designed to operate in the following modes:

- Screen mode, in which the user types in a screen of text and
  modifies the text using any of the HP terminal's editing
  features.

- Line mode, in which edit commands operate on groups of one or
  more lines.

In the screen mode, the editor treats the terminal screen as a
window through which the user can view a section of text.  A
cursor within this window indicates the character at which editing
will take place.  The user controls the cursor with the help of
the terminal and can also move the window forward or backward any
number of lines within the file.

The user is working directly with single keystroke commands, which
are often faster and more convenient to use than the line edit
commands of the editor.
```

Figure 2-10. TESTFILE - Sample Text File for the Tutorial - Part 2

Entering Line Commands from Screen Mode (^C)

The EDIT/1000 screen mode enables you to see several lines of text in relation to each other and allows you to use the local terminal keys to add and delete text. But what about moving text, copying text, searching for specific strings, and displaying system information? These editing functions are accomplished using the powerful EDIT/1000 line mode commands.

EDIT provides a control key combination that is used from screen mode to execute line mode commands. The ^C key combination enables you to escape the screen mode to execute one line mode command. After execution of the command, you are returned to the pending line.

To try the ^C command, use the EDIT runstring from the RTE user interface (CI or FMGR) to start a new edit session with your file, TESTFILE. Use the S (Screen) command to access the screen mode.

Now use the ^C command, followed by a carriage return. Note that the line mode prompt (a slash) is displayed on the line following the current cursor location.

The following sections of this tutorial describe some of the line mode commands that you may wish to execute from screen mode. For now, use the carriage return to return to the screen mode.

The ^C command is recommended when you wish to execute one line mode command that does not display information that you need to read. If you want to enter more than one line mode command, or if you want to display information, use the ^U command.

If you use the ^C^C command, EDIT does not read the screen and the line mode pending line is the line your cursor was on in screen mode, unless there are wrap around lines (extra long lines). If you use the ^C command, the screen is read and the pending line is the line your cursor was on when in screen mode.

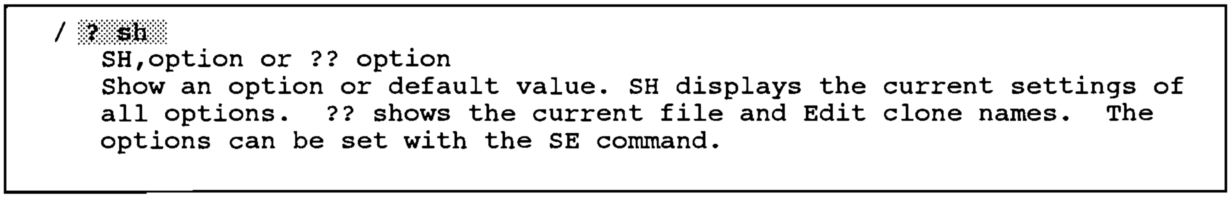
If you use the ^U (or ^Q) command, EDIT reads the screen and the pending line is the line after the last line shown for the previous screen. For example, if the last line of the screen is line 21, use of the ^U command makes line 22 the new pending line. If you use the ^U^U (or ^Q^Q) command, EDIT does not read the screen and the new pending line is the first line of the screen from which the ^U^U was executed.

Displaying Information (H, ?, SH)

EDIT contains a wealth of information that is useful to view during the editing session. This information must be accessed through the line mode. EDIT's online reference facility is accessed with the Help (H or ?) command. Information on various EDIT options and defaults is displayed using the SH (Show) command.

Online reference provides brief descriptions of all the EDIT commands and many of the options. The ? (Help) command used by itself displays a menu of all items for which online reference is available. The ? command can also be used in conjunction with any of the Help menu items to display the online reference information available for that menu item.

To try the Help command, use the ^U key combination to escape screen mode and use the Help command to request help on the SH (Show) command. (You use ^U instead of ^C when displaying information because ^C immediately returns you to the screen mode, allowing you about two seconds to read the information displayed.) Your entries and the system response should look similar to Figure 2-11, following:



```
/ ? sh
  SH,option or ?? option
  Show an option or default value. SH displays the current settings of
  all options.  ?? shows the current file and Edit clone names.  The
  options can be set with the SE command.
```

Figure 2-11. Example of the Online Reference Display for the SH Command

To try the SH (Show) command, enter the command to show the default for the EDIT prompt character. Your entries and the system response should look similar to the following:

```
/ sh pc
 Prompt character.....  PC =/
/
```

To display all EDIT options, use the command SH, or SH ALL. Try this command now. For more information on the Help and Show commands, refer to the command descriptions in Chapter 4 EDIT/1000 Commands.

Finding Out Where You Are (N)

Many EDIT commands are executed at the current pending line. For this reason, you may need to know the line number of the current pending line. The N (Line Number) command is used while in line mode to display the line number of the current pending line. If using the screen mode to enter text, exit the screen mode using ^U followed by a carriage return. If you use ^C to exit the screen mode, the line number is not displayed long enough for you to read it.

In screen mode, the first and last line numbers are displayed in the screen bracket lines.

Getting Where You Want to Be (n)

One way of getting from line one to line 798 of your file is to use the ^F key to page through the screens of text until line 798 is located. EDIT provides a shortcut for this time-consuming activity. The n (Specific Line Number) command enables you to go to any line in the file by entering the desired line number or a $ to go to the end of the file.

To try this out, use ^C from the screen mode, followed by a carriage return. At the prompt, enter any line number or a $. The line for which a number was entered is the current pending line on the new screen displayed. The cursor is located on the pending line, which is at the center of the screen. If a $ was entered, the cursor is located on the current pending line, which is the line above the end of file line.

Displaying a Portion of the File (L, W)

EDIT provides two line mode commands that enable you to display blocks of text from the line mode: L (List) and W (List Window). If you are using the screen mode, you may wish to include line specifications with these commands to display specific portions of the file, to visually scan the entire file, or to list the file or portion of the file to a printer while editing.

The L (List) command allows you to display either the entire file or a block of lines from the text file. Line specifications are optional. If no line specifications are included in the command string, the L command defaults to display 20 lines of text, starting with the current pending line. The last line displayed becomes the new pending line.

Suppose you are editing and your current location is at the end of the file, at about line 100. Before you exit the editing session, you would like to view the first 20 lines of text to see if you entered a comment to describe the file. To do this, you would use ^U to escape the screen mode (you are displaying information and will need more time to view the data than a ^C would provide), and then enter the command

```
/ 1 20 L
```

EDIT would display the first 20 lines of text, followed by the prompt. To return to the end of the file, you would enter a $ at the prompt. If you decided to enter a comment, you would enter an S (Screen) to enter screen mode.

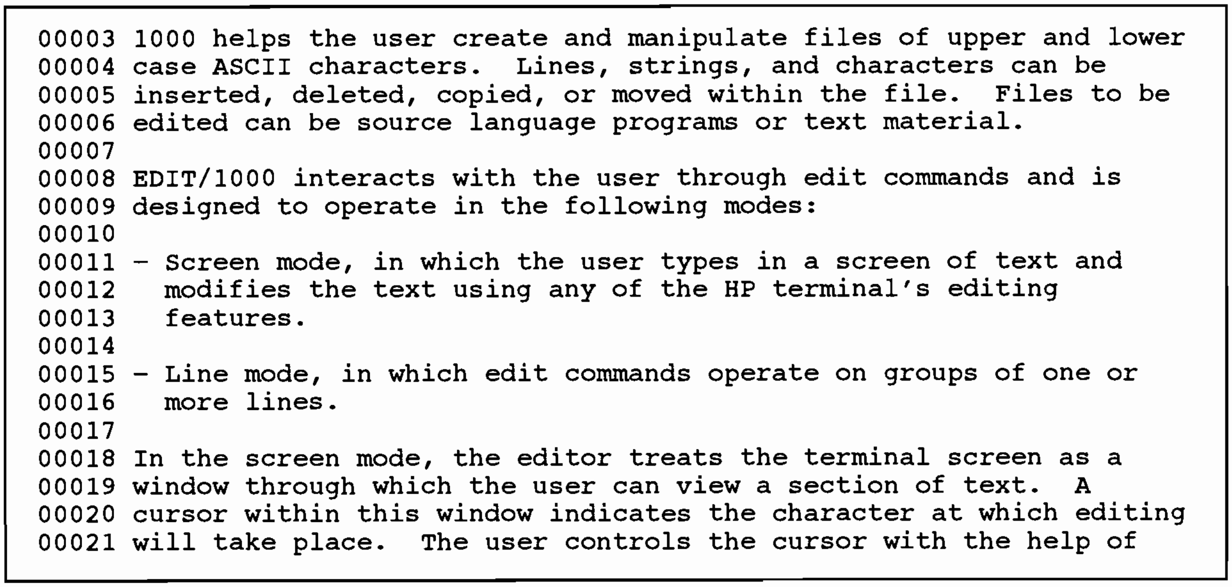
There are many ways to use the L command. For example, you can enter /L 14 to list the next 14 lines of the file, including the pending line; enter /18 L to list 20 lines beginning with line 18; enter /10,14 L 4 to list lines 10 through 14; and enter /LN to list 20 lines with the line numbers. It is also possible to direct the output from the L command to a file or device (for example, a printer). For details, refer to the task charts in this chapter.

The W (List Window) command displays a vertical (lines above and below) window with the applicable line numbers to the left of the text. EDIT indicates the current pending line by placing an arrow to the left of the line number. The window command does *not* change the pending line.

The window command also allows you to specify a line range, and defaults to display 21 lines of text. If you used ^U to exit the screen mode and entered the command

```
/ 3 21 w
```

on the sample text file, the text would be displayed as illustrated in Figure 2-12, which follows:



```
00003 1000 helps the user create and manipulate files of upper and lower
00004 case ASCII characters.  Lines, strings, and characters can be
00005 inserted, deleted, copied, or moved within the file.  Files to be
00006 edited can be source language programs or text material.
00007
00008 EDIT/1000 interacts with the user through edit commands and is
00009 designed to operate in the following modes:
00010
00011 - Screen mode, in which the user types in a screen of text and
00012   modifies the text using any of the HP terminal's editing
00013   features.
00014
00015 - Line mode, in which edit commands operate on groups of one or
00016   more lines.
00017
00018 In the screen mode, the editor treats the terminal screen as a
00019 window through which the user can view a section of text.  A
00020 cursor within this window indicates the character at which editing
00021 will take place.  The user controls the cursor with the help of
```

Figure 2-12. Example of Window (W) Command Output

If you want to display the text without line numbers, use the command WU (List Window Unnumbered).

Marking a Line (Kx)

The Kx (Mark Line) command is used to mark a line for future reference. During the editing process, you will find it helpful to mark certain lines. Marked lines can be specified in find, copy, and move commands, making those tasks more efficient and easier to perform. Since line numbers change frequently as a result of inserting and deleting text, specifying a line mark in find, copy, and move commands is often more reliable than specifying a line number.

The command to mark a line in line mode is Kx, where x represents any single alphabetic character (a–z). EDIT accepts only alpha characters. EDIT indicates an error if punctuation characters are used, and deletes lines if numeric characters are used (this is because the K, or Kill command, is entered with a number to delete a specific number of lines).

Evidence that a line is marked is seen in screen mode, where a colon and the alpha character mark are displayed to the right of the marked line, outside of the text entry area. Use the Show Mark (SH MA) command from the line mode to show all active marks and their current line numbers. As editing continues, EDIT keeps the marks current by adjusting the line numbers associated with each marked line.

In line mode, a line is marked by entering Kx, where x is an alpha character. This marks the current pending line. If desired, a single line specification can be entered prior to the Kx command to mark a specific line other than the pending line. For example, the command

```
/ 4 kb
```

would mark line four with a b.

To mark a line from screen mode, you can either use ^ C to access the line mode and mark the desired line as described above, or you can use the control key combination ^ K. To use the latter method, position your cursor on the line to be marked, then enter ^ K followed by a carriage return. EDIT places a colon in the right-hand margin of the line and positions your cursor to the right of the colon. Enter the desired alpha character.

Note

Line marks entered in screen mode must be saved by reading the screen (single execution of one of the control commands). If you mark more than one line with the same alpha character, EDIT applies the mark to the last line found during the screen read.

Saving the File without Exiting EDIT (WC, WR)

The WC (Write and Create) and WR (Write and Replace) commands are used to save a file without exiting the editing session. These commands can be used to save a file prior to major changes so that if these changes are not satisfactory, you can revert to the version of the file saved prior to the changes without losing the rest of your file. WC and WR are also used when you wish to save a file but continue editing on another file.

Suppose you have just entered about 200 lines of commands in a file that is approximately 1000 lines long. You need to do several global changes (covered in the next portion of the tutorial) to change (for example) a file descriptor used throughout your file. If the change is not done correctly, you want to be able to abort the file so that the changes are ignored by EDIT. But what about the 200 lines of text you just entered? If you aborted EDIT, they would be ignored, too. The solution to this problem requires that you save the file before executing the global changes. Using an EC (Exit and Create) or an ER (Exit and Replace) would exit the editing session. You can speed up the process by using either the WC or WR command.

As with the EC command, EDIT requires a filename when creating a file using the WC command. After executing a WC or WR command, continue editing the same file, or use the FI (File Input) command to close the current file and access a different file. (The FI command is described in Part 1 of this tutorial.) Suppose you are editing the file TESTFILE and want to save it under a different filename. You would then like to check the newly created file to ensure that it was copied correctly. After using ^ U to escape the screen mode, your entries and the system response would look like this:

```
  Current pending line when ^U was executed.
/ wc newtestfile
Created file NEWTESTFILE::HERMAN:4:18
Closed file NEWTESTFILE::HERMAN:4:18
  Current pending line.
/ fi newtestfile
Closed file TESTFILE::HERMAN:4:24:38
Opened file NEWTESTFILE::HERMAN:4:18:38
89 lines read.
  First line of text.
/
```

Note that the source file is not closed until a new file (NEWTESTFILE) is input.

You have reached the end of Part 2 of the tutorial. If you were trying out the exercises as they were discussed, what command should you use at this point to exit and save your file? If you said ER (Exit and Replace), you are right!

Tutorial Part 3 – Efficient Editing

Contents and Sample File

This portion of the tutorial provides instructions on useful EDIT/1000 commands that enable you to copy, move, and delete text, undo (reverse) commands, merge EDIT files, search and exchange character strings, and use the EDIT command stack to speed up the editing process. The flow chart in Figure 2-13 illustrates the contents of Part 3 of the tutorial. The sample file illustrated in Figure 2-14 has been used for the examples in this section.

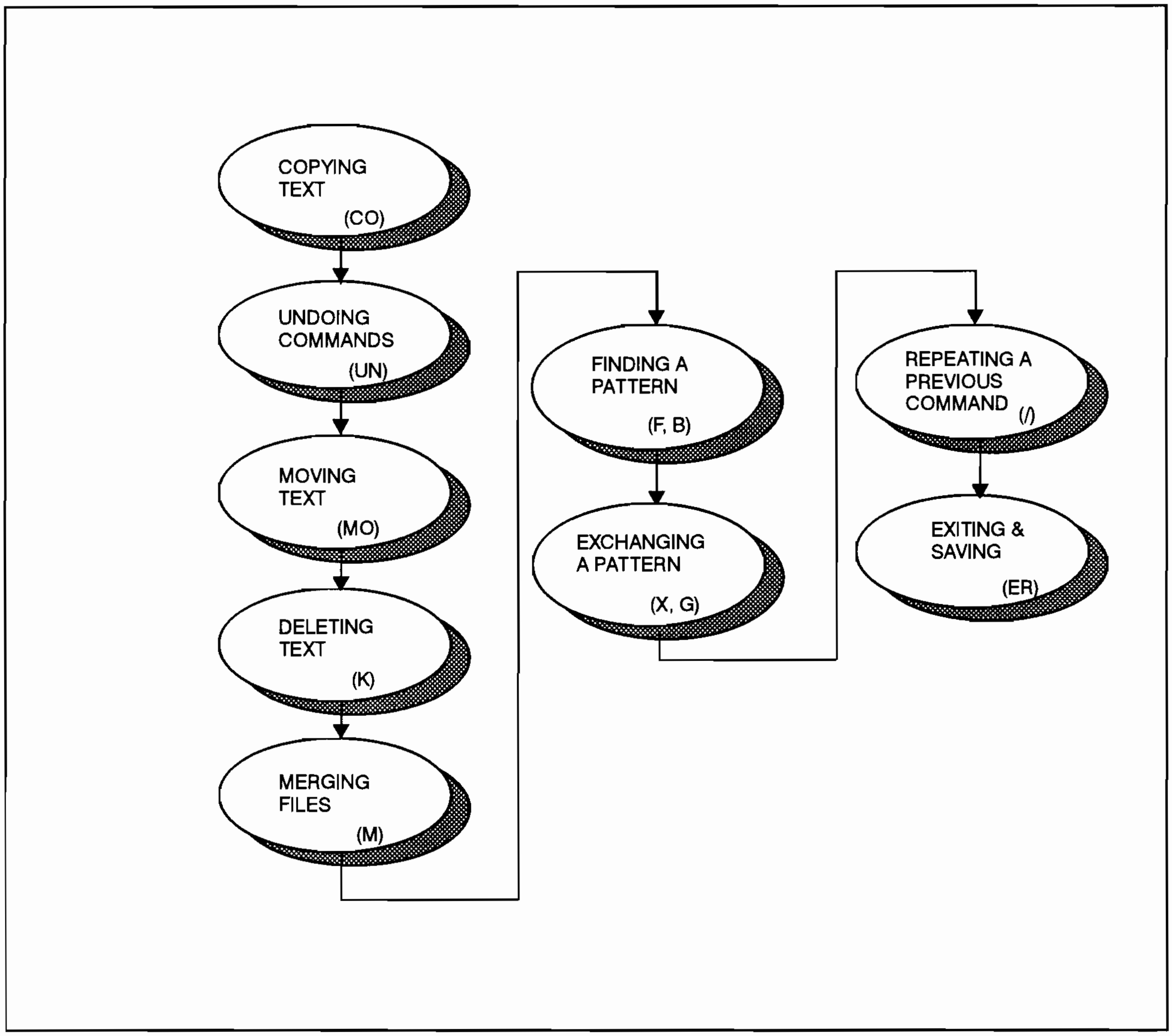
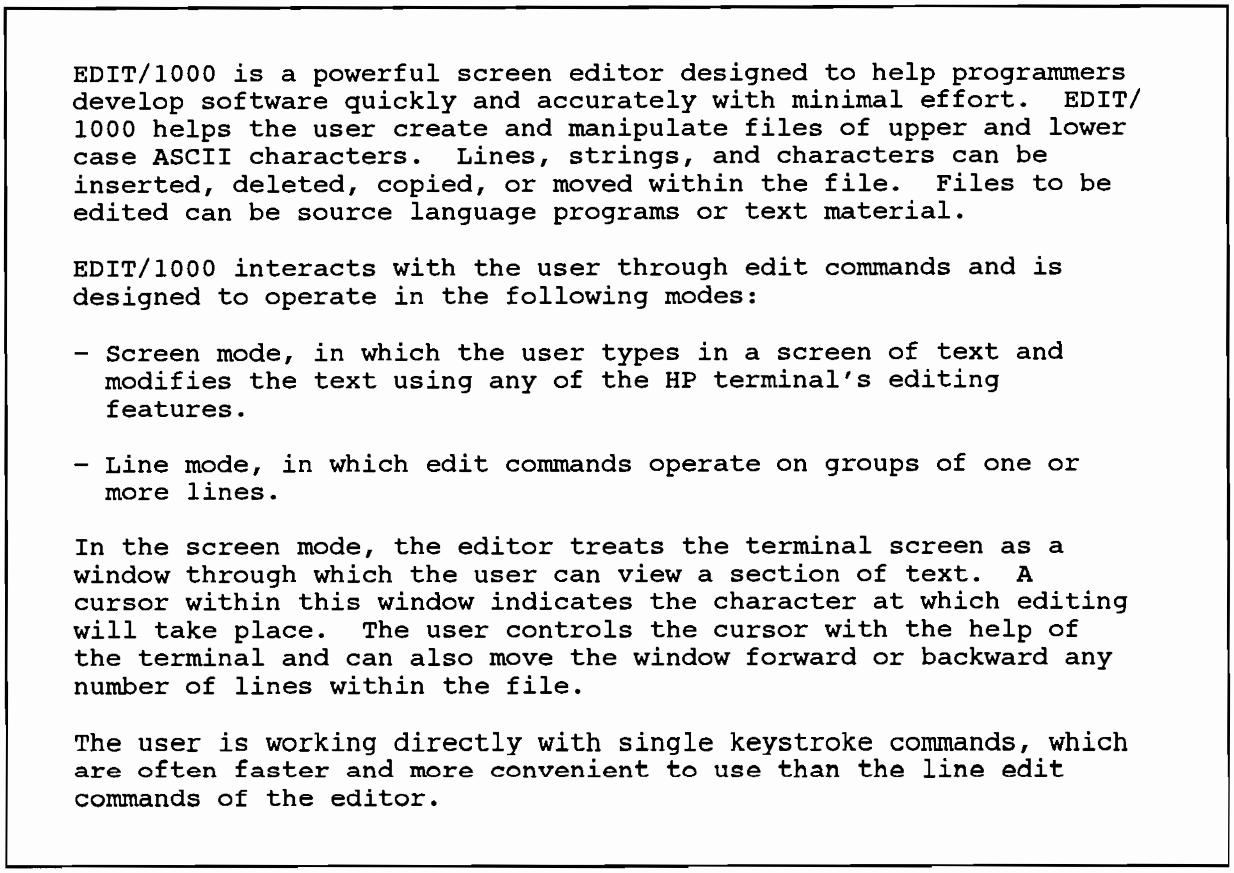


Figure 2-13. Tutorial Map, Part 3 – Efficient Editing

If you would like to try the exercises described in this tutorial, the sample file used is the same as the sample text file entered in Part 1 of the tutorial. If you did not enter the sample text file in Part 1 of the tutorial, use the following command from the RTE CI user interface to enter EDIT:

```
CI> edit
```

Next, enter the S (Screen) command at the EDIT prompt to access the screen mode. With the cursor positioned on the bottom screen bracket line, use the INS LINE key on your terminal to insert enough blank lines to type in the sample text, illustrated in Figure 2-14. Use the INS CHAR, DEL CHAR, INS LINE, and DEL LINE keys, as necessary, until the text is complete.



```
EDIT/1000 is a powerful screen editor designed to help programmers
develop software quickly and accurately with minimal effort.  EDIT/
1000 helps the user create and manipulate files of upper and lower
case ASCII characters.  Lines, strings, and characters can be
inserted, deleted, copied, or moved within the file.  Files to be
edited can be source language programs or text material.

EDIT/1000 interacts with the user through edit commands and is
designed to operate in the following modes:

- Screen mode, in which the user types in a screen of text and
  modifies the text using any of the HP terminal's editing
  features.

- Line mode, in which edit commands operate on groups of one or
  more lines.

In the screen mode, the editor treats the terminal screen as a
window through which the user can view a section of text.  A
cursor within this window indicates the character at which editing
will take place.  The user controls the cursor with the help of
the terminal and can also move the window forward or backward any
number of lines within the file.

The user is working directly with single keystroke commands, which
are often faster and more convenient to use than the line edit
commands of the editor.
```

Figure 2-14. TESTFILE – Sample Text File for the Tutorial – Part 3

Copying Text (CO)

The CO (Copy) command is used from the line mode to copy one or more lines of text. The text to be copied is specified in the command either with a range of line numbers, or with line markers. Use of the line markers is easier from the screen mode.

To copy a range of lines by specifying line numbers, determine the line numbers of the first and last lines of the text block to be copied. Do this by using the N (Line Number) or W (Window) command from the line mode (remember to use ^U to exit screen mode). You may need to press the carriage return a few times (this prints out successive lines of the text) or enter + or −n (where n is the number of lines to go forward or back in the text) to locate the desired lines of text.

Once the desired line is located, use the N command to print the line number. The line numbers are then used with the CO command to copy the text. The text is copied and inserted below the current pending line, so you must set the pending line to the desired location prior to using the copy command. This is accomplished from the line mode by entering the line number of the line after which the copied text is to be inserted.

For example, to copy lines one through five of the sample text to the end of the file, enter ^U to escape the screen mode, enter a dollar sign ($) at the prompt to move the current pending line to the end of the file, and enter the copy command:

```
/$
/1 5 co
```

To copy text by marking lines, locate and mark the desired lines of text using ^K while in screen mode. (Assume that you used a and b to mark the beginning and ending lines of the text block to be copied.) Locate the line after which the copied text is to be inserted. Position your cursor on that line and use ^C to open up the screen for a line mode command. Now use the copy command with the line markers as follows:

```
/ :a :b co
```

☞ *Now use the line marker method to copy lines one through five of TESTFILE and insert them at the end of the file.*

By now it should be clear that for screen mode editing, the line marker method of copying lines is faster and easier than the line range specification method.

Undoing Commands (UN)

EDIT provides a command that allows you to reverse almost all the commands entered that alter or delete data, (with the exception of the A, AS, EC, ER, FI, WC, WR, and UY commands). The UN (Undo) command enables you to undo the last command executed. So, if you just copied the wrong block of text, you can un-copy it by entering the UN command from the line mode. To try the command, use ^U to escape the screen mode, then enter

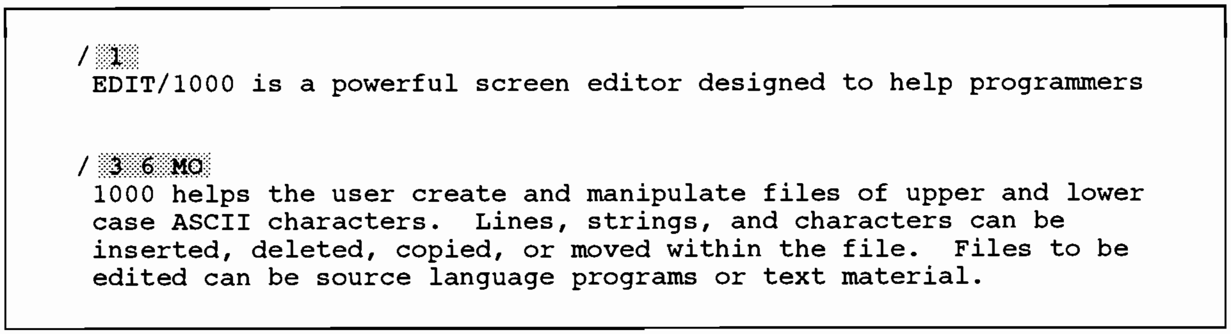
```
/un
```

Use the UN command to reverse the copy command you just used in TESTFILE.

Moving Text (MO)

The MO (Move) command moves one or more lines of text from their original position to after the pending line. The command is entered proceeded by a beginning and an ending line number. If only one line is to be moved you do not have to indicate an ending line number.

For example, assume you wanted to move lines three through six of TESTFILE to after the first line of the file. After using ^U to escape the screen mode, your entries and the system response would look similar to Figure 2-15:



```
/1
 EDIT/1000 is a powerful screen editor designed to help programmers


/3 6 MO
 1000 helps the user create and manipulate files of upper and lower
 case ASCII characters.  Lines, strings, and characters can be
 inserted, deleted, copied, or moved within the file.  Files to be
 edited can be source language programs or text material.
```

Figure 2-15. Example of Move (MO) Command

Use the MO command example described above, and then use the UN (Undo) command to reverse the move.

Deleting Text (K)

The command to delete lines of text is appropriately named the Kill (K) command. The K command is executed from the line mode using either a line range specification prior to the K in the command string, or a number of lines after the K. For example, to delete the current pending line, enter

```
/k
```

To delete the entire file, enter

```
/1 $ k
```

To delete ten lines, beginning at the current pending line, enter

```
 /k 10
```

To delete lines 40 through 45, enter

```
 /40 45 k
```

When the K command is used and more than one line is to be deleted, EDIT displays the message OK?, requiring you to enter a y (yes) or n (no) to confirm or cancel the delete command. If you were to delete lines 12 through 18 of TESTFILE and then use the UN (Undo) command to reverse the delete command, your entries and the system response would be displayed as follows:

```
/12 18 k
OK? y
 window through which the user can view a section
of
/un
/
```

Note that the original line 19 (line 12 after execution of the K command) is listed as the current pending line following the delete.

Merging Files (M)

Suppose you have 200 lines of code that you would like to copy from file JUNQUE to file REVISEDJUNQUE. EDIT's Merge (M) command allows you to copy text from one file to another file in one simple step.

The M command is used from the line mode to copy all or part of another file into the current work file, inserting it after the current pending line. So in this example, you'll use the M command from the file REVISEDJUNQUE.

Before using the M command, use EDIT to access JUNQUE (the copy-from file). JUNQUE must be accessed so that you can obtain the line numbers for the first and last lines of the block of text to be merged into REVISEDJUNQUE. This is accomplished by using the RU (Run Program) command to run another copy of EDIT. After determining the line numbers (assume the first line is line 82 and the last line is 281), exit JUNQUE using the A command (no changes were made).

Now locate your current pending line at the line where you want to merge the text from JUNQUE. You want to merge the text at the end of your file REVISEDJUNQUE, so enter $ at the EDIT prompt to go to the end of the file. Now enter the merge command. Your entries from the line mode of REVISEDJUNQUE and the system response follow:

```
/m junque 82:281
Opened file JUNQUE::HERMAN:4:664:38
200 lines read.
Closed file JUNQUE::HERMAN:4:664:38
  System lists last line (line 281) merged.
/
```

To try the Merge command, create a file NEWFILE using the same commands used to create TESTFILE. Type in about 30 lines of text. Open the file TESTFILE and merge in lines 10 through 20 of file NEWFILE at line 15 of TESTFILE. Use the command

```
/15 m newfile 10:20
```

Finding a Pattern (F, B)

EDIT supplies two commands that can be used to search your file for character strings. The F and B commands are both referred to as *Find a Pattern* commands. A line specification can be used with either command to specify the range for the pattern search. The only difference between the commands is the default range. If no line specification is indicated for the B command, the search operation begins at line one (the beginning) of the file. If no line specification is used for the F command, the search (Find) operation begins after the current pending line.

The search example in this tutorial uses the F (Find) command, because it is mnemonic and is easier to remember. With the F command you can locate just the first occurrence or all occurrences of a given character string within a given line range. This is done by using the A (All) option in the F command string.

The character string to be located is specified between any matching set of punctuation marks except commas and blanks (which are used to separate elements of the command string). The most common set of *delimiters* used are slashes (/). For example, to find the first occurrence of the word COLOUR, you would enter the command

```
/ 1 f/colour/
```

(Note that if you aren't at line one, you must enter a one with the Find command to start at the beginning of the file.) However, if you wanted to find the first occurrence of the string Xon/Xoff, you would use a punctuation mark other than the slash for a delimiter. The following command would work in this instance:

```
/ 1 f!xon/xoff!
```

Finding All Occurrences of a Pattern (A Option)

With the F command you can locate just the first occurrence of a string or all occurrences of the string within a given line range. This is done by using the A (All) option in the F command string. For example, to find all occurrences of the word COLOUR, enter the command

```
/ 1 f/colour/a
```

Suppose you had entered this command with the current pending line at line 50. If you had not specified the line range, this command would have located all occurrences of COLOUR from line 51 to the end of the file, missing those located in the text between line 1 and line 50.

Enter commands to find the first occurrence of the word EDIT, all occurrences of the word EDIT, and all occurrences of the word EDIT between line 11 and line 26. Note how the matched lines are listed after execution of the command.

Distinguishing between Uppercase and Lowercase

When the F command is used with the A option, EDIT finds all occurrences of the specified string, disregarding whether the string occurs in uppercase and/or lowercase. To distinguish between uppercase and lowercase alpha characters, set the Case Folding option to OFF using the SE (Set) command, prior to entering the F command. For example, to find all occurrences of the word COLOUR, (in all uppercase letters), enter the commands illustrated in Figure 2-16:

```
/ se cf off
 Case Folding..............  CF =off
/ 1 f/COLOUR/a
 01177 of the word COLOUR, you would enter the command:
 01179 /1 $ F/COLOUR
 01192 of the word COLOUR, use ^U to escape screen m...
 01195 /1 $ F/COLOUR/A
 01199 this command would have located all ...COLOUR
 01215 word COLOUR, (in all upper case letters), enter
 01220 /1 $ F/COLOUR/A

 EOF 7 matches
```

Figure 2-16. Example of Find (F) Command with Case Folding = OFF

After listing all matched lines, EDIT indicates that it has reached the end of the file (or prints the word LIMIT to indicate that it reached the end of the specified line range) and indicates the number of *lines* that contain one or more matches.

Use the same search exercises as those listed in Figure 2-16 to find occurrences of the word EDIT in TESTFILE. Then do the exercise again with the Case Folding option set to ON to see the difference in the results.

Search Metacharacters

The previous search exercises are examples of *literal* pattern matching. That is, the character string to be found is typed in exactly as it occurs in the text. EDIT provides two special characters, or metacharacters, that can be used to match a more generalized pattern. The @ character, or *wildcard* character is used in patterns to indicate zero or more occurrences of any character.

For example, to find all lines that contain the letter E followed by the letter X, regardless of the characters in-between, enter the command

```
/ 1 f/e@x/a
```

Figure 2-17 illustrates the lines matched in the TESTFILE sample text by this command:

```
00006 edited can be source language programs or text material.
00011 - Screen mode, in which the user types in a screen of text and
00012   modifies the text using any of the HP terminal's editing
00019 window through which the user can view a section of text.  A
```

Figure 2-17. Example of Output for f/e@x/a Command

The second metacharacter provided by EDIT is the caret (^). The caret is referred to as the *anchor* character and is used at the beginning of a pattern to indicate that EDIT is to match only those lines where the pattern occurs at the beginning of the line. For example, to find all occurrences of the word THE in TESTFILE that occur at the start of a line, enter the command

```
/ 1 $ f/^the/a
```

Figure 2-18 illustrates the lines matched by this command in the TESTFILE sample text:

```
00022 the terminal and can also move the window forward or backward any
00025 The user is working directly with single keystroke commands, which
```

Figure 2-18. Example of Output for f/ ^ the/a Command

EDIT provides several other metacharacters that can be used for more sophisticated pattern matching in both find and exchange commands. These metacharacters can be used when the EDIT

Regular Expressions option (RE) is set to ON. This type of pattern matching is quite involved and powerful. For instructions on using the metacharacters available with Regular Expressions ON, refer to Chapter 5 EDIT Regular Expressions.

Exchanging a Pattern (X, G)

EDIT supplies two commands that enable you to exchange one or more occurrences of a character string with a substitute string. The X and G commands are both referred to as Exchange commands and a line specification can be used with either command to indicate the range in which they are to be executed. Both commands find *all* occurrences of the pattern within the range specified.

The X and G commands produce similar results except that the X command provides a listing of the lines affected by the exchange and the G command does not. This tutorial uses the X command in the sample exercises because it is easier to remember (eXchange) and because the listing of lines affected by the command allows you to immediately check your exchange so that you can undo (UN) the command, if necessary.

The syntax for the exchange command requires that you enter the pattern to be matched and a substitute pattern. The substitute pattern is required. Because you may want to replace the matched character string with a null string (with nothing), the substitute field can be empty. The match pattern and the substitute pattern are delimited with matching punctuation marks, as with the Find command. For example, to change all occurrences of the word TEH to THE, enter the command

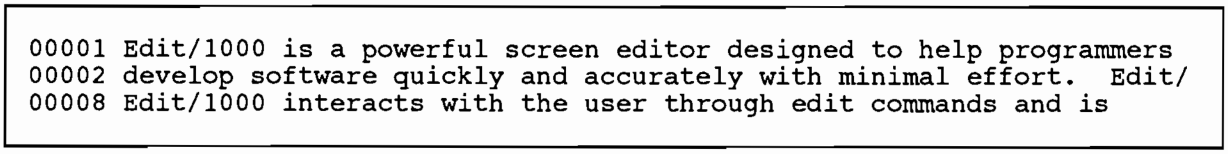
```
/ 1 $ x/teh/the/
```

If you tried this exchange on the sample text TESTFILE, use the UN (Undo) command to reverse the exchange.

As with the Find commands, the Case Folding option can be set to OFF to differentiate between upper and lower case in the pattern match. For example, use the Set command to turn CF OFF, then enter the command

```
/ 1 $ x/EDIT/Edit/
```

Figure 2-19 illustrates that this command changes the following lines in the sample text TESTFILE:



```
00001 Edit/1000 is a powerful screen editor designed to help programmers
00002 develop software quickly and accurately with minimal effort.  Edit/
00008 Edit/1000 interacts with the user through edit commands and is
```

Figure 2-19. Example of Output for x/EDIT/Edit/a Command

If you tried this command out on TESTFILE, be sure to Set Case Folding to ON before continuing.

If you are sure of the change and do not need to see a listing of the affected lines following the exchange, you can either use the G command, or use the Q (Quiet) option with the X command. For example, to change all occurrences of Hewlet to Hewlett without a listing of the affected lines (a pretty safe bet), enter the command

```
/ 1 $ x/Hewlet/Hewlett/q
```

Repeating a Previous Command (/)

At this point, you are ready for a shortcut. EDIT provides a command stack feature that enables you to repeat a previously executed command without retyping. The command stack command lists the last 20 commands used during the EDIT session.

To use the command, enter the slash mark (/) from the line mode and press the carriage return. The command stack is listed. Locate the cursor under the desired command and press the carriage return. The command is executed. You can also use this feature to execute a *version* of a previously-executed command. Do this by editing the command with the local terminal keys prior to pressing the carriage return.

For example, if you used ^U to escape the screen mode and entered the command stack command, your entries and the system response might look like this:

```
 //

---Commands---
 SE CF OFF
 S
 SE CF ON
```

Try using the command stack with the sample text TESTFILE. Change one of your exchange commands and re-execute.

Exiting & Saving (ER)

Congratulations, you've finished the tutorial! To exit and save a file that you're finished editing, which command do you use? (If you need help, the answer is printed in the left-hand margin!) The ER (Exit and Replace) command saves your work and returns you to the RTE user interface.

For more practice in using EDIT, refer to Chapter 5, EDIT Regular Expressions.

Quick Reference Task Charts

Table 2-2. Task Chart for Moving around the Editing Screen

Moving around the Editing Screen			
Task	*** Command**	**Screen or Line Mode**	**Command Description/ Remarks**
To go to the end of a line...	^Z	S	Press and hold CTRL key, then press the Z key. Release both keys and press carriage return.
To go to the beginning of a line...	^A	S	Press and hold CTRL key, then press A key. Release both keys and press carriage return.
To go to the next screen, saving edits on the current screen...	^F	S	Press and hold CTRL key, then press F key. Release both keys and press carriage return.
To go to next screen, without saving edits...	^F^F	S	Press and hold CTRL, then press F key and release both keys; repeat and press carriage return.
To go to previous screen, saving edits on current screen...	^P	S	Press and hold CTRL, then press P key. Release both keys and press carriage return.
To go to previous screen, without saving edits...	^P^P	S	Press and hold CTRL, then press P key and release both keys; repeat and press carriage return.
To start a new screen from cursor position, saving edits on current screen...	^T or ^S	S	Press and hold CTRL, then press T or S key. Release both keys and press carriage return.
To start a new screen from cursor position, without saving edits...	^T^T or ^S^S	S	Press and hold CTRL, then press T or S key and release both keys; repeat and press carriage return.
To display an extended screen, saving edits on current screen...	^X	S	Press and hold CTRL, then press X key. Release both keys and press carriage return.
To display an extended screen, without saving edits...	^X^X	S	Press and hold CTRL, then press X key and release both keys; repeat and press carriage return.

* The ^ character represents the control (CTRL) key on your terminal.

Table 2-3. Task Chart for EDIT Screen Mode Commands

Editing in Screen Mode

Task/Description	Command
To start screen edit beginning at 10 lines above the pending line.	S
To set default for screen size; initially 10 lines above to 10 lines below the pending line, with a 2-line, screen-to-screen overlap.	SE SD
To set maximum screen size (in number of lines); default depends on the type of terminal used.	SE SL
To set the control-D line editing graphics submode.	SE CD
Screen Mode Control Key Combination Commands	**Command**
To quit screen mode.	CTRL-U
To quit screen mode; won't work on terminals using Xon/Xoff handshake protocol.	CTRL-Q
To go back to previous screen.	CTRL-P
To go back to previous screen; note that CTRL-P does not work on X.25 pad terminals.	CTRL-R
To go forward to next screen.	CTRL-F
To start next screen at cursor position.	CTRL-T
To start next screen at cursor position; won't work on terminals using Xon/Xoff handshake protocol.	CTRL-S
To start extra large screen at cursor position.	CTRL-X
To temporarily escape screen mode to execute one EDIT line mode command and return to current screen.	CTRL-C
To copy line indicated by cursor to below that line. Reset margins.	CTRL-O
To position cursor to set a line marker.	CTRL-K
To move cursor to first character on line. Reset margins.	CTRL-A
To move cursor to last character on line. Reset margins.	CTRL-Z
To join current line to following line. Reset margins.	CTRL-J
To break current line into two lines at the current cursor position.	ESC-4 CTRL-B
To enter the line editing graphics submode while in screen mode.	CTRL-D

NOTE: Single execution of the screen mode control key commands reads and saves the edits on the current screen. Double execution of the commands causes EDIT to ignore any new changes to the screen.

Table 2-4. Task Chart for Line EDIT Commands

Editing in Line Mode	
Task/Description	**Command**
To insert a line after the pending line.	<space>
To copy a block (rectangle) of text from one position to another.	BC
To move a block (rectangle) of text from one position to another.	BM
To edit the pending line and advance to the next line.	C
To copy line(s) to after pending line.	CO
To exchange a pattern on the pending line.	G
To insert a line before the pending line.	I
To pull the line below the current pending line up and join it to the end of the pending line.	J
To delete line(s).	K
To move a line to after the pending line.	MO
To show the pending line number.	N
To copy and edit the pending line.	O
To edit the pending line.	P
To edit the pending line locally using the terminal edit keys.	Q
To replace the pending line with new text.	R
To add time and date information to a line and display line.	TI
To unconditionally exchange a field.	U
To exchange patterns.	X
To exchange patterns on the pending line and advance to the next occurrence of the match pattern.	Y
Control Key Combination Commands Used for Pending Line Edits	
To break line at cursor position.	CTRL-B
To delete characters.	CTRL-C
To replace characters.	CTRL-R
To insert characters.	CTRL-S
To truncate line at cursor position.	CTRL-T
To extend line.	CTRL-X

Table 2-5. Task Chart for Commands that Display Information

Displaying Information	
Task	**Command**
To display a summary of commands in alphabetical order.	?(or H)
To display information about command specified.	?<cmd> *or* H <cmd>
To display EDIT abort messages.	?AB *or* H AB
To display explanation of abbreviations.	?EX *or* H EX
To display pattern explanation.	?PA *or* H PA
To display pending line character mode editing information.	?PL *or* H PL
To display line specification information.	?LS *or* H LS
To display regular expression explanation.	?RE *or* H RE
To display EDIT runstring options.	?RO *or* H RO
To display recovery mode messages.	?RM *or* H RM
To display header lines to mark column numbers.	HL
To display header lines to pending lines.	HLP
To display all option setting and default parameters.	SH [ALL]
To display setting for option specified.	SH <opt>
To display text of lines before modification and the commands used to undo the change.	SH UN
To display 20 lines plus next pending line.	L
To display line length in characters.	LE
To display file size in number of lines.	LI
To display 20 lines with line numbers plus next pending line.	LN
To display lines without line numbers; turns off LN command.	LU
To display pending line number.	N
To display approximate file size in 16-bit words.	SZ
To display 20 lines with numbers and pointer to pending line.	W
To display 20 lines with or without line numbers (WN, WU) and pointer to pending line.	WN or WU
To display command stack to allow selection of any command for execution.	/
To display EDIT program name and current source file.	??

Table 2-6. Task Chart for Option Setting Commands

Setting EDIT Session Options	
Task	**Command**
To set anchor character, initially ^.	SE AC
To set escape character, initially \.	SE EC
To set indefinite character, initially @.	SE IC
To set prompt character, initially /.	SE PC
To set command separator character, initially |.	SE CS
To set tab character, initially TAB key or CTRL-I.	SE TC
To specify tab columns for ASMB (7 and 21).	TA
To specify tab columns for FTN (7 and every 4 columns).	TF
To specify tab columns for Macro programs (10,26,40,44,48).	TM
To specify tab columns for Pascal (every 3 columns).	TP
To specify tab columns for HP–UX (every 8 columns).	TU
To set tab stops local to terminal as defined by T for screen mode.	TS
To offset tab stops 2 columns for line mode edits.	TL
To set window columns, initially 1 and 256.	SE WC
To set default screen size, initially 10 lines above and 10 lines below pending line with a 2-line screen-to-screen overlap.	SE SD
To set maximum screen mode size; default depends on terminal type.	SE SL
To set vertical window size, initially 10 lines above and 10 lines below pending line (21 lines total).	SE VW
To set line length, initially at the maximum of 256.	SE LE
To set prompt for dangerous commands on or off, initially on.	SE AS
To set case folding on or off; initially on.	SE CF
To set regular expressions on or off, initially off.	SE RE
To set screen mode display functions on or off, initially on.	SE DF
To set no-match return to pending line (on) or to lower range limit (off); initially on.	SE RT
To set automatic time-stamp updating on or off, initially on.	SE TS
To set BELL with prompt on/off, initially off.	SE BE

Commands that may corrupt your file if executed inadvertently prompt you for confirmation. If the OK? prompt is displayed, only a y(es) answer executes the command. Except for the Abort (A) and Transfer (TR) commands, even if you enter "y" by mistake, you can still recover by entering the Undo (UN) command. The OK? prompt can be suppressed by concluding the command string with a slash (technically, the current EDIT prompt character). The following chart summarizes the commands that require confirmation.

Table 2-7. Task Chart for Commands Requiring Confirmation for Execution

Commands Requiring Confirmation for Execution		
Command	**Mnemonic**	**Description**
Abort	A	Confirmation prompt issued only if file has been modified.
Abort and save scratch file	AS	Confirmation prompt is always issued.
Delete until match	D	Default range is delete only the pending line regardless of pattern specified. Confirmation prompt always issued.
Kill lines	K	confirmation prompt issued if more than one line is to be deleted. Confirmation is not required if a new list file is created or an existing file is appended.
Transfer input	TR	Confirmation prompt is always issued. While executing the TR command file, no confirmation prompts are displayed.
Unconditional replace	U	Confirmation prompt is not issued if only one line is to be changed.
Exchange	X	Confirmation prompt is not issued if only one line is to be changed.
List	L,,filename	Confirmation prompt issued if listing to a file and the existing file is in danger of being overwritten.

Table 2-8. Task Chart for Search Commands

Searching for Patterns		
Command	**Mnemonic**	**Description**
To find a pattern	B	Default range is from beginning to end of file. Range must be specified for successive searches.
	F	Default range is from line after pending line to last line. Useful in successive searches.
	’pattern’	Searches forward for pattern specified. If pattern is omitted, the last pattern specified in B, F, ’ ’, or ‘ ‘ command is used.
	‘pattern‘	Searches backward for pattern specified. Defaults are the same as ’ ’ (forward search) command.

Table 2-9. Task Chart for Exchange Commands

Exchanging Patterns	
Task/Description	**Command**
To exchange characters on pending line.	G
To exchange pattern on pending line and search for next occurrence of match pattern. If found, makes that line the pending line.	Y
To exchange all occurrences of match pattern with substitute pattern. Default is exchange only pending line. Specify range as "1 $" for the entire file.	X
To unconditionally replace the specified number of characters with the substitute pattern, starting at the left window column. Default range is pending line only. To affect entire file, specify range as "1 $".	U

Table 2-10. Task Chart for Delete Commands

Deleting Text	
Task/Description	**Command**
To delete lines, use the Kill command.	K
To delete lines up to a line that contains the pattern specified in the command, use the Delete command. If used with the V (reverse) option, the D command can be used to delete lines that contain the pattern specified.	D

Table 2-11. Task Chart for Miscellaneous EDIT Commands

Miscellaneous EDIT Commands		
Task	**Command**	**Description**
Delete Trailing Blanks	BK	Kill trailing blanks for all lines. Truncates lines longer than the line length limit.
Delete Tabs	TK	Kill tabs; insert blanks according to current tab stops.
Run a Program	RU	Run a program and return to EDIT.
Screen Copy	SC	Copy terminal display memory to EDIT work area.
Transfer to Command File	TR	Reads and executes EDIT commands from a command file.
Mark a Line	Kx	Mark a line with label x (letter A through Z). ":x" is used in line specification to point to the line marked.
Add Line Sequence Numbers	#	Add a 3-character ID and sequence numbers on columns 73 through 80.
Add Time & Date Information	TI	Add a 30-character field with time and date information.
Add Time Stamp		A time stamp may be created on any line in the form: <YYMMDD.HHMM>. The right angle bracket must be either the last or the next to the last non-blank character on that line. The information is in two-digit form, showing the year, month, and day followed by the hour and minute. This field is automatically updated when EDIT writes a file.
Terminate EDIT	EC	Exit and create new file.
	ER	Exit and replace source file.
	A	Abort EDIT session without replacing source file.
	AS	Same as A but save scratch file.
File Input/Output	WR	Replace source file and remain in EDIT session.
	WC	Create new file and remain in EDIT session.
	FI	Input file to replace current source file.
	M	Merge file specified to after pending line.
	L	Create or open list file specified.
	K	Create or open list file specified.
	FCS	Close source file.
	FCL	Close list file.
Reversing An Edit	UN	Undo a modification.
	SH UN	Show undo list.
	UY	Recover line(s) from the undo list.

Table 2-12. Task Chart for Commands with Common Defaults

Commands with Common Defaults		
Command	**Mnemonic**	**Common Default**
Search for a pattern	F	Last search pattern specified.
	B	Last search pattern specified.
...forward	’ ’	Last search pattern specified.
...backward	‘ ‘	Last search pattern specified.
Exchange	G	Last match/substitute string specified.
	X	Last match/substitute string specified.
...& Find	Y	Last match/substitute string specified.
Replace source file	ER	Source file.
	WR	Source file.
Create a list file	L	List file.
	K	List file.
Unconditional exchange	U	Last match/substitute string specified.
Delete	D	Last search pattern specified.
Fill text	FL	Current line

3

EDIT/1000 Operations

Introduction

This chapter provides details on the operation of the EDIT/1000 program. The contents include

- Working with EDIT
- Line Specification
- Pattern Specification
- Search and Exchange Command Options
- EDIT Session Options
- Line Mode Editing
- Screen Mode Editing
- Error and Information Messages
- Recovering from Errors
- Batch Operation

The information in this chapter will enable you to

- expand the EDIT runstring to include EDIT commands
- understand how EDIT text files are created, edited, saved, and deleted
- date files when they are created or edited
- list EDIT/1000 output to a line printer

- understand the EDIT/1000 command syntax and how to enter EDIT commands
- effectively use line specifications, pattern specifications, and command options in EDIT commands
- edit using the line mode
- access and exit the screen mode, and edit in the screen mode using local terminal keys and control key combinations
- execute line commands from the screen mode
- recognize EDIT error and informational messages
- recover from user and system errors
- understand the basics of batch operation
- run a program from EDIT.

Working with EDIT

This section covers details on working with EDIT. It provides information on

- the EDIT runstring
- naming files
- creating or accessing files
- closing files
- deleting files
- EDIT's file naming defaults
- EDIT's text display format
- EDIT's command syntax
- line length
- file size limit

The EDIT Runstring

Chapter 1 describes how to use the simplest form of the EDIT runstring to initiate the EDIT/1000 program. The simplest form of the EDIT runstring consists merely of the word, EDIT. The runstring can be expanded to include the specific file to be edited. It can be further expanded to include runstring options and EDIT commands.

Runstring options start with a minus sign (−) and are entered after the EDIT command and before the filename. EDIT commands follow the filename, and are separated from the filename by a space or a comma, and from each other by command separators (default = |). The following options can be entered before the filename in the EDIT runstring:

−B The Batch option is used to run EDIT without any user interaction (for example, to run EDIT from a command file). If an error occurs (for example, if a file cannot be merged or the command file does not end with an exit command) EDIT will abort. The Batch option turns on the quiet option.

−L:n Set line length. "n" is an integer number of characters that becomes the maximum line length. Lines longer than the maximum line length (default = 256 characters) will be truncated without warning when the file is read in by EDIT.

−Q The Quiet option is used to indicate that nothing should be listed to the terminal during the EDIT operation. If an error occurs (for example, if a file is missing) the quiet operation is turned off. This option can also be turned off by using the Set Quiet Off (SE QU OF) command. The quiet option also implies an affirmative response to all dangerous command prompts (that is, *OK?*).

−R The Recover option instructs EDIT to begin a recovery operation on the work file of the filename entered in the runstring. (See section on Recovering From Errors in this chapter.)

−S::directory The S option specifies the work file directory to access.

In the following example, the Batch option (−B) initiates a batch editing operation on the file WARM.TXT and turns on the quiet option so that nothing is listed to the terminal during the operation. The Transfer (TR) command instructs EDIT to read commands from the file COOL.CMD.

```
CI> edit -b warm.txt tr cool.cmd
```

More on the EDIT Runstring

EDIT commands can be entered in the runstring after the filename. When more than one command is specified in the EDIT runstring, they must be separated by command separators. The default character for the command separator is the vertical bar (|), although it can be set to any other non -alphanumeric character (except a space or a comma). The number of commands allowed is limited to the line length. For example, the command used to edit the existing file TESTFILE in the screen mode with the Case Folding option set to OFF would be entered as

```
CI> edit testfile,se cf off|s
```

Note that a space, a comma, or a command separator (default = |) is used to separate the filename from the EDIT run command (EDIT) and from the command string. Commands entered with the runstring have the same format as those entered interactively.

If you enter the name of a file that does not exist (for example, if TESTFILE is a new file) and include the screen mode command (S) in the runstring, EDIT displays the message

```
No such file TESTFILE
An ER or the first WR will create it.
```

and places you in screen mode. The screen displayed contains only the start and stop brackets, indicating that the file did not previously exist. (To check if the file will be created under the name entered in the command string, return to line mode and enter the ?? status command.)

If a command included in the runstring cannot be executed by EDIT, EDIT ignores the rest of the commands and immediately places you in line mode.

If commands that issue the OK? prompt are included, they must be suppressed with a slash (/), otherwise, EDIT places you in line mode to solicit a response to the prompt and will ignore the rest of the commands.

For more information on EDIT options, refer to the EDIT Session Options section later in this chapter. For instructions on entering the runstring to initiate EDIT sessions, refer to the Tutorial in Chapter 2 of this manual.

Note

Back quotes can be used in the EDIT runstring to delimit characters for which CI normally performs special processing. Use of the single back quotes will prevent this special processing. Refer to the CI User's Manual for a description of the characters used to indicate special processing.

Naming a File

In EDIT, files are named using the standard RTE naming convention (referred to in CI as a file descriptor or in FMGR as a file namr), which consists of several user-definable parameters. The CI file descriptor formats supported include:

```
   filename.type-extension::dir:type:size[user]>node

                         -OR-

/dir/subdir/filename.type-ext:::type:size[user]>node

                         -OR-

   filename:security-code:cartridge:type:size
```

The last format listed above is the format supported for the FMGR file namr. For more information, refer to the User's Manual for your RTE system.

Creating or Accessing a File

New EDIT text files are created through a two-step process:

1. accessing EDIT and inserting text
2. exiting EDIT by using one of the save commands; or by listing lines of text from an existing file to a new file

When the EDIT runstring is used without specifying a filename, EDIT displays the following message:

```
CI> edit
EDIT : Use ? for help
FI,<filename> specifies file to edit.
EOF
/
```

When the EDIT runstring is entered with a filename that does not exist, EDIT displays the following message:

```
CI> edit smart
EDIT : Use ? for help
No such file SMART
An ER or the first WR will create it.
EOF
/
```

Note that in each case the message displayed is followed by the EDIT prompt (/). This indicates that if text is entered in either the line or screen mode, it will be written into the new work file. A new source file is then created with the EC (Exit and Create) command and a filename. The EC command terminates the editing session and displays the following message:

```
/ ec smart
Created file SMART::MIRIAM:4:1
Closed file SMART::MIRIAM:4:1
end of edit
CI>
```

In the situation where the EDIT session is initiated with a filename that does not exist, the session can also be terminated with the ER (Exit and Replace) command, as follows:

```
/ er
Created file SMART::MIRIAM:4:1
Closed file SMART
end of edit
CI>
```

Files can also be created without leaving the edit session, by using the WC (Write and Create), the WR (Write and Replace), the L (List), and the K (Kill, or delete) commands.

The WC command works in the same manner as the EC command except that it does not terminate EDIT, but returns to the current pending line after creating the file specified. The WR command works similar to the ER command, so that when the use of an ER creates a file, the use of a WR will also create a file. The L and K commands can create a new file by copying (L) or deleting (K) lines from an existing file and listing them to a new file. (Note that the L and K commands can also be used to append text to an existing file by entering a plus (+) sign before the filename.) Examples of the WC, L, and K commands follow.

To create a new file named EDSIL, enter the command

```
/ wc edsil::41:18:4
Created file EDSIL::41:18:4
Closed file EDSIL::41:18:4
/
```

To create the file TRYLIST, consisting of lines one through 40 of the current file (the double comma defaults the *maximum number of lines* field to the line range specified), enter the command

```
/ 1 40 L ,, trylist
Created file TRYLIST::MIRIAM:4:24
Posted file TRYLIST::MIRIAM:4:24
  Pending line.
/
```

The following examples illustrate two attempts to create the 50-line file, NAMES. In the first attempt, the user chose not to execute the command and entered *no* to the request for confirmation. In the second attempt, the command was executed and the existing file NAMES was overwritten with the 50 new lines.

```
/ L 50 names
File already exists NAMES:::4:24
Opened file NAMES::41:4
OK?  no
Command not executed.
Closed file NAMES::41:4:1
/

/ L 50 names
File already exists NAMES:::4:24
Opened file NAMES::41:4
OK?  yes
Posted file NAMES::41:4               (File is overwritten)
EOF
/
```

In the next example, the entire contents of file NAMES is deleted and used to create the file NEWFIL.

```
/ 1 $ L ,, newfil
Closed file NAMES::41:4:5            (NEWFIL is still open)
Created file NEWFIL::41:4:24
Posted file NEWFIL::41:4:24
EOF
/
```

Note that the list file is posted in the second, fourth, and fifth examples above. This indicates that the file has been opened and written to, but has not been closed. The command used to close a list file (FCL) is discussed in the following section.

When the L or K command is entered with a filename, EDIT attempts to create a file. If the create fails because the file entered already exists, EDIT opens the file and then displays the

OK? prompt. If the OK? prompt is answered with an affirmative entry, the data in the existing file is overwritten with the new information.

For the EC and WC commands, the file must not exist before the command is used. Use the ER or WR command if the file exists.

After a file has been created, it can be accessed from within an EDIT session by using the File Input (FI) command at the EDIT prompt. The FI command is entered with a filename (either an existing or new file). If EDIT's current work file has been modified, at the time the FI command is used EDIT displays the dangerous command confirmation prompt (OK?) to indicate that data may be lost because recent changes to the current file have not been saved:

```
/ fi bunko
OK? y
Closed file SMART::HERMAN:4:1:2
Opened file BUNKO::HERMAN:4:560:56
155 lines read.
  First line of file.
/
```

After a file has been accessed, it is edited using the line mode, the screen mode, or both editing modes. A detailed description of entering commands from both modes is provided under the sections on Line and Screen Mode Editing, later in this chapter.

Dating Files

As you create or modify your files, you may want to date them. This can be done either manually with the TI command or automatically by creating a time stamp field on any line in your file.

The TI command adds a 30-character field to the current pending line starting at the column specified. The default starting column is column 1. Text outside the 30-character field is not affected. Note the example in Figure 3-1:

```
  Current pending line.
/ti 40
  Current pending line.                 8:00 AM  TUE.,  26  MAY, 1987
```

Figure 3-1. TI Command Example

EDIT does not update the time and date information. It can only be done with another TI 40 command on the same line.
A time-stamp field can be created on any line. This field is updated by the system time each time EDIT replaces the file (that is, EC, WC, ER or WR). Enter the following field at the location in your file where the time stamp is to be made:

```
<YYMMDD.HHMM>
```

You can enter any 10 digits between the angle brackets, but you must enter the brackets and the period in the position illustrated. When EDIT updates the time stamp the date is displayed in the preceding format, where the year, month, day, hour, and minute are each indicated with two digits. For example, two o'clock in the afternoon, July 4, 1987 is represented as <870704.1400>.

In Figure 3-2, the file is dated by creating a dummy field (in file TEST, using the line mode P command and CTRL-X), then using the WR command to write the change to the file while still remaining in EDIT, and using the FI command to re-edit the file and check the time stamp.

```
    This is line one.
/P<CTRL-X><222222.2222>
     This is line one.<222222.2222>
/wr
Time stamping output file.
Posted file TEST::41:4:1:30
/fi,test
Closed file TEST::41:4
Opened file TEST::41:4
Time stamp found.
1 line read.
    This is line one.<870905.0850>
/
```

Figure 3-2. Time Stamp Example

The time stamp updating option is initially on. Updating can be suppressed by including an option setting command to turn time stamp update off. For example:

```
CI> edit,specs,se ts of
```

In this case, the *Time stamp found* message is displayed but the update is not made.

EDIT does not create the time stamp. If not included in your file, the time stamp feature is ignored. Upon file input, EDIT displays a *Time stamp found* message if the time-stamp field is found in your file.

The time stamp feature should not be confused with the TI command which adds a field of 30 characters with the time and date information. This 30-character field is not altered except with another TI command on the same line.

Listing Lines to a Printer

If you need a printed copy of a number of lines or of the entire file, use the L command and enter the LU number (Logical Unit number) of a list device (printer) in place of a file name.

Note

This procedure should be used with caution. On a multi-user system, the FCL (File Close, List) command should be used immediately after execution of the list to free up the printer. If you do not use the FCL command, other system users will be unable to use the printer because it is locked to your EDIT session.

If you want a record of the lines deleted with the K command, use the K command in the same manner as with the L command. In the example below, 6 refers to the LU of the printer; 1$ specifies the entire file; 200 restricts the listing to a maximum of 200 lines.

```
/1 $ L,200,6
```

The following examples illustrate how to list lines 1 to 150 to the printer and how to produce a listing of lines deleted.

```
/1 150 L,,6
```

```
/1 $ k,,6
```

Suppose you wish to print the text file on a line printer while you continue to edit. Use the L (List) command as follows:

```
/1 $ l,,6
```

In the command above, the line specification 1 $ indicates the entire file, from line 1 to the end. The List command requires you to enter the maximum number of lines to list. The double commas are entered to default the maximum to the line specification entered (in this case, the number of lines contained in the entire file). The 6 is the printer designation. If you were listing the file or a portion of it to another file instead of printing it, you would replace the printer number with a filename.

Closing Files

EDIT's source file is closed when the EDIT session is terminated using any of the exit commands (EC, Exit and Create, ER, Exit and Replace, or A, Abort), or when the FI (File Input) command is used to access another file.

The FCS (File Close, Source) command closes the current source file while allowing you continued access to the work file. This is convenient for those situations when the file is being edited but must also be accessed by another program, or when changes made to the work file are not to be saved and you wish to ensure that entering an ER or WR will not alter the original source file.

The FCL (File Close, List) command is used to close a list file that has been created by using the L (List) or K (Kill) commands. When the L or K commands are used, the list file that is created or appended remains open until another L or K command is used with a specified filename, or the EDIT session is terminated, or the FCL command is used. If you wish to access the list file (to check the list operation, or to edit the list file) the list file must first be closed and then re-opened. The FCL command allows the list file to be closed without leaving the EDIT session.

Note

If the list file is a device (such as a line printer), use the FCL command to close and release the file as soon as you have completed the listing. The device will be unavailable to other system users while EDIT has it open for a list file.

Deleting Files

You may delete the contents of a file while in EDIT, but the file itself cannot be deleted from within the EDIT session. Files must be deleted from the RTE user interface (CI or FMGR) by using the Purge (PU) command. (Note that you can use the EDIT Run (RU) command to run a copy of CI or FMGR while in EDIT.)

File Naming Defaults

EDIT retains the source file name that is entered in the runstring or via a File Input (FI) command, as the default file. Anytime you use a command that accesses a file (EC, ER, FI, K, L, M, TR, WC, WR) you can choose to take advantage of this filename defaulting, or you can enter a new filename. There are several ways to default all or part of the filename.

For example, if a "–" (dash) is specified in the filename, the character in the same position in the default name is placed into the new name. The at (@) sign can also be entered to default either that filename or the file type extension. If a type extension but no name is entered, EDIT uses the default filename. If no type extension is entered, EDIT uses an *empty* type extension. If both the filename and extension are omitted but at least one character is present in the filename field, both the filename and extension are defaulted. For example:

Default File Descriptor	File Descriptor Specified in Command	File Descriptor Used
`/DIR/TEST.TXT`	`new.@`	`NEW.TXT`
`/DIR/TEST.TXT`	`-a-k.mac`	`TASK.MAC`
`/DIR/TEST.TXT`	`.tmp`	`TEST.TMP`
`/DIR/TEST.TXT`	`/scratch/.tmp`	`/SCRATCH/TEST.TMP`
`/DIR/TEST.TXT`	`text`	`TEXT`
`/DIR/TEST.TXT`	`:`	`TEST.TXT`

If a security code is entered and the cartridge is not, the cartridge is taken from the source file name. For example, if the source file name is INPUT::XX and you enter the command *ER :SC*, EDIT writes to file INPUT:SC:XX, and does not look on any other cartridges. This is the only defaulting carried out for file directories.

Note

For RTE-A and RTE-6/VM, EDIT cannot access files that do not conform to CI filename syntax. For example, if you have a FMGR filename that starts or ends with a period, or contains more than one period, you must rename the file with the FMGR RN command before you can access it with the EDIT program.

For more information on the file naming conventions, refer to the appropriate manual for your RTE system.

Text Display Formats

When EDIT displays the pending line or text listings from the work file on your terminal, the line or listing is always preceded by two blanks. This convention allows room for the EDIT prompt and a single character command, which aligns the new text with that displayed by EDIT. Error and information messages always begin in column 1 so that the difference between an EDIT message and a line of text can be easily determined.

In the line mode example below, the EDIT prompt and the first line of text (the pending line) are displayed when EDIT is first accessed for the file TESTFILE:

```
CI> edit testfile
EDIT : Use ? for help
Opened file TESTFILE::HERMAN:4:96:39
490 lines read.
  The first line of TESTFILE.
/
```

EDIT displays messages that provide information on the file opened and the number of lines in the file. It then prints the pending line, offset by two spaces, followed by the prompt character.

If an invalid command is entered at this point, an error indication (? ^) is displayed starting in column 1. An error message may also be displayed, as follows:

```
CI> edit testfile
EDIT : Use ? for help
Opened file TESTFILE::HERMAN:4:96:39
490 lines read.
  The first line of TESTFILE.
/ 2 1 f/bicycle
/2 1 f/bicycle/
?  ^
Start > stop
/
```

In this example, a Find (F) command was entered to locate the word *bicycle*. The range of lines that the command was to affect (line 2 to line 1) was illegal because the start of the range (line 2) was larger than the end of the range (line 1). EDIT repeats the invalid command, prints a question mark and a caret to indicate the location of the error and displays an error message (Start > stop) starting in column 1.

Error messages are covered in more detail later in this chapter.

EDIT Command Syntax

To take full advantage of EDIT's capabilities, you must understand its command syntax. EDIT/1000 commands have the following basic form:

```
[<line specification>]<command>[<parameters>]
```

EDIT read commands from left to right, separating the individual units of the command (line specifications, command mnemonic, and parameters) for translating.

Almost all EDIT commands begin with an optional line specification, followed by a command mnemonic or character, which is followed by one or more optional parameters.

The syntax used for the command parameters is dependent on the specific command entered. Chapter 4 provides the parameter syntax for each EDIT command. The syntax for line specifications is not dependent on the command entered. However, not all commands permit the use of line specifications and some commands allow the entry of only one line specification. The syntax for line specifications is covered later in this chapter.

Note

An asterisk (*) in the first column of a command is interpreted as a comment and EDIT ignores the rest of the text on that line.

Line Length

EDIT has a default line length of 256 characters. If your output device supports a shorter maximum line length, you can shorten the EDIT line length to prevent possible line truncations when the file is used. For instructions on how to set the line length after EDIT is initiated, refer to the EDIT Session Options section of this chapter or to the description of the Set (SE) command in Chapter 4. For instructions on increasing or decreasing the line length via the runstring, refer to the description of the runstring in this chapter. Regular Expression operators are limited in length to 256 characters (see the Regular Expression Length section in Chapter 5 for more information).

File Size Limit

The EDIT program can be used to edit files that are up to 32,500 lines/records long. When EDIT's work file exceeds 32,500 lines, a read operation for any file input is terminated and additional editing is prohibited. If you attempt to edit a file that exceeds 32,500 lines, EDIT will read in the first 32,500 lines and report that the line limit has been exceeded. *Do not* execute an ER (Exit and Replace) command at this point or the source file will be truncated to 32,500 lines. Instead, break the over-sized file into several smaller files, edit them individually and then re-merge them.

First, use the *FI, <smallerfilename>* command to clear EDIT's work space and place you in a new file without truncating the original, over-sized file. Create several of these smaller files, merging in portions of the larger file using the EDIT Merge (M) command with a line range. Exit each of the smaller files using the WR (Write and Replace) command to create a new file. Edit the new files individually and then merge them together again using the RTE CI MERGE utility.

Line Specification

The line specification indicates the line or line range affected by the command. The line specification consists of zero, one, or two *linespecs*. If it is unclear where the first linespec stops and the second begins, the linespecs must be separated by a space or a comma. The semantics for each command indicates how many, if any, linespecs are applicable and whether they are required or optional.

This manual uses the following syntax to represent line specifications:

```
   [linespec 1] [linespec 2]
```

Both linespec 1 and linespec 2 consist of a base line specification, and an optional line specification offset:

```
<base line specification> [<line specification offset>]
```

The base line specification indicates the number of lines from the beginning of the file. The base line may be a number, such as 42, or one of the following special characters:

. The pending line.

$ The last line in the file.

> Same as $; last line in the file.

:x A marked line, where the mark (indicated by x) are upper- or lowercase alphabetic characters (A-Z).

’pattern’ Search forward to line containing specified pattern.

‘pattern‘ Search backward to line containing specified pattern.

***** Used for linespec 2 to indicate that its base line value is the value of linespec 1, including the linespec 1 offset.

For example, the following line specification consists of two linespecs:

```
   .-10 *+20
```

In the first linespec, the period (.) is the base line specification and the −10 is the offset. In the second linespec, the asterisk (*) is the base line specification and the +20 is the offset. A command that includes this line specification affects text from 10 lines above the pending line (.) to 20 lines beyond the beginning point of the first linespec, for a total of 21 lines.

If the base line specification is omitted but an offset is entered, EDIT uses the current pending line as the default base line specification.

The optional [<line specification offset>] is the number of lines relative to the base line specification. When written as a number relative to the base line, the following syntax is used, where "n" refers to an integer:

−[n] Negative n lines relative to the base line

^[n] Same as −[n]; negative n lines relative to base line

+[n] Positive n lines relative to the base line

If only a + or − is used, EDIT uses a default of one line. (That is, + = +1.)

For some commands, the first specification indicates where in the file a change is to take place. For example, in the syntax for the Merge (M) command, you first enter the line where text is to be inserted (merged) from another file, then enters the M command and the name of the file from which text is to be merged, and then enters the line number of the first line of text to be merged followed by the number of lines to be merged. The command syntax is abbreviated like this:

```
[<linespec 1>]M <file descriptor> [<start line>] [<#  lines>]
```

(Note that the Merge command has two versions of the command syntax; refer to the description of the M command in Chapter 4 of this manual.)

Because the line specification is optional, each EDIT command has a default that is used if the command is entered without a line specification. EDIT closely monitors the line specification and reports an error if an illegal line range or specification is entered. For example, if the first line number entered for a Find (F) command is greater than the second line number, an error message similar to the following would be displayed:

```
/20 15 F/smurf/
?    ^
Start > stop
```

The table below provides examples of line specifications. Valid line specifications for each command are provided in Chapter 4 of this manual.

Table 3-1. Line Specification Examples

Search (F) entire file for the first occurrence of the string OK:	/1 F/OK/
Search (F) from pending line to last line for the string TEH:	/. F/TEH/
Delete (K) lines 10 through 59:	/10,59 K
List (L) from pending line to 10 lines above the last line:	/.$−10 L
Exchange (X) AUG with SEPT from line 1 to 10 lines below the pending line:	/1.+10 X/AUG/SEPT
Go back (up) 10 lines and make it the new pending line:	/−10 or /^10
Go forward 10 lines and make it the new pending line:	/+10
Go back 40 lines and show vertical window (EDIT returns to pending line):	/−40 W
Search forward for first occurrence of CAKE:	/’CAKE’
Search backward for first occurrence of CAKE:	/‘CAKE‘
Search backward for first occurrence of CAKE (when used after a search for CAKE):	/‘ ‘
Move (MO) 6 lines (starting from 20 lines below the pending line):	/+20,*+5 MO
Go to last line (make it new pending line):	/$
Go to first line (make it new pending line):	/1
Edit line 40:	/40 P/////1980
Edit 5 lines above the pending line:	/−5 P/////Aug
Move lines marked with x through y to after the pending line:	/:x :y MO
Mark line 10 with an A:	/10KA

Pattern Specification

EDIT provides several commands for which patterns can be specified. The search (F and B) and exchange (G, X, and Y) commands all use patterns to locate specific character strings in text files.

A pattern is a string of characters to be searched for, matched, or substituted. There are two methods of pattern matching possible with EDIT/1000: standard pattern matching and pattern matching that uses the Regular Expressions option to search for more involved, or generalized patterns. Standard pattern matching is described in this section and is assumed throughout the majority of this manual, as it is the default matching for the EDIT program. Regular Expressions pattern matching is described in Chapter 5.

A *match* occurs when a line is found containing the pattern specified in the search or exchange command. Depending on the command, EDIT indicates the match by either making the line located the pending line or, (as in a multiple search), by listing the lines matched.

A match occurs if a specified pattern is found anywhere within the horizontal search window. The search window is applied during searches. However, a match outside the window can occur if the N (No Window) option is specified in the search or exchange command.

During multiple searches, or when the search for a particular pattern is unsuccessful, EDIT returns to the original pending line. If desired, the Return (RT) option can be turned off using the Set command to force EDIT to remain at the end of the file (EOF) or at the search line range limit after an unsuccessful search.

In a match pattern there are two types of characters: literal characters and metacharacters. Literal characters represent themselves (for example, the character *a* represents the letter *a*). Metacharacters represent patterns other than their literal value. For example, the at sign character (@) represents zero or more occurrences of any character in the work file. Most of the EDIT metacharacters available are used in the Regular Expressions method of pattern matching. However, there are three metacharacters used in standard pattern matching:

^ The anchor character. Use at the beginning of a pattern string to match only those lines where the pattern occurs at the beginning of the search window (usually column 1).

@ The indefinite, or *wildcard* character. Use to match any pattern string, making the shortest match possible in the line. For example, if the pattern is a@a, the string ababbabbba would contain two matches. The shortest match on this line is the first aba. (For more on shortest possible matching, refer to Chapter 5 EDIT Regular Expressions.)

\ The default escape character, used before a special character to make the character represent itself, rather than its special character meaning. For example, \@ would represent the literal at sign.

All patterns used in a command must be offset by a delimiter. The delimiter used to identify the start and stop of a pattern can be any punctuation character except commas or spaces. The first non-comma, non-space character encountered in a pattern sets the value for the pattern <delimiter>. All pattern delimiters used in a command must match. In this manual, the slash (/) character is used as the pattern <delimiter>.

The EDIT Case Folding (CF) option is normally set to ON during an editing session. To differentiate between upper- and lowercase letters in a match pattern, the CF option must be set to OFF.

The search commands F and B are used to find one or more lines that contain the pattern specified. The only difference between the commands is that the start line default for the B command search is line one, and the start line default for the F command is the current pending line. The pattern syntax for a search (F or B) command is:

```
<delimiter><match pattern>[<delimiter>]
```

Note that the second delimiter is optional for a search command if there are no additional options to be added after the match pattern. If used with the All (A) option, the F and B commands find and produce a listing of all occurrences of the specified pattern. For example, the command f/and/ would locate the first occurrence of the word AND; and the command f/and/a would locate all occurrences of AND. The A option is described further in the section Search and Exchange Command Options in this chapter. For more information on the F and B commands refer to the command descriptions in Chapter 4 of this manual.

Note that the command separator (default = |) is a literal character inside a pattern. Therefore, if another command is to be included in the command string, the find pattern must be terminated with a matching delimiter.

The exchange commands, G, X, and Y, are used to exchange all occurrences of a match pattern with a substitute string within the line range specified in the command. The pattern syntax for an exchange command is:

```
/<match pattern>/<substitute pattern>[/]
```

The G (Character Exchange on Pending Line) and X (Exchange) commands locate a pattern and replace it with the substitute string specified. The only difference between the G and X commands is that X command provides a summary report of the lines exchanged and the G command does not. The following exchange command would change all occurrences of the word *funky* with *distasteful:*

```
   1 $ X/funky/distasteful/
```

The U (Unconditional Exchange) command has the same pattern syntax as the exchange commands, but behaves differently. It deletes the specified number of characters on each line in the line range, regardless of the type of character, and replaces the characters with the substitute. For example, the command

```
   1 $ U/xx/line/
```

would delete the first two characters of every line in the file, and insert the four character string *line.*

The Y (Exchange and Search) command performs one exchange on the line specified in the command and then searches for the next occurrence of the pattern. The command syntax for the Y command calls for only one optional line specification:

```
   [linespec 1] Y/pattern/substitute/
```

If a line specification is not entered, the exchange is performed on the current pending line. This command is useful when there is a need to check the next match before executing the exchange. After initiating the exchange process, you indicate whether the next exchange is to be performed by entering a *y* to exchange the string or an *f* to advance to the next line with a matching pattern.

Pattern Defaults

EDIT maintains default match patterns that are shared by the find and exchange (B, F, G, U, X, and Y) commands. If one of these commands is used without a specified pattern, a default pattern is used. Each time a pattern is specified, it becomes the default pattern for the next execution of the command.

There are two pattern default buffers (or pattern holders): one used for search commands (B, F, D, ’ ’, and ‘ ‘), and one that is used for exchange commands (G, U, X, and Y). The default buffer for exchange commands also has a substitute default string.

When a find command (B, F, D, ’ ’, or ‘ ‘) is entered, the pattern default is used if the pattern was not specified (for example, *F* or *F,A*) or if the pattern specified is null (for example, *F//*).

When an exchange command (G, U, X, or Y) is entered, the default match pattern and its associated substitute string is used if this parameter is omitted (for example, *X* or *1$X,,Q*). If an exchange command is entered with a null (zero length) match pattern, the default Find pattern is used as the match pattern (for example, *X//XZZ/*). The match pattern used (either defaulted or entered) and the new substitute specified then become the Exchange match pattern and substitute default. Table 3-2 provides a summary of the pattern defaults with examples.

The Y command is an exchange command that sets both the default Exchange pattern parameter (match pattern and substitute) and the default Find pattern.

The default match patterns can be copied between the Find and Exchange buffers. To use the default Find pattern when executing an exchange, specify a null match pattern and the appropriate substitute. To copy the default Exchange pattern into the Find buffer, use the Y command.

When an empty pattern is specified with a U command, it is taken as a zero length delete field specification, not a default Find pattern specification. For example, use of the command *U//x/* inserts an x at the start of the line (or search window).

Table 3-2. Default Pattern Summary

Command	Match Pattern?	Substitute Pattern?	Example	EDIT Action
F, B, D, ’ ’, and ‘ ‘	None or null	N/A	F F//A	EDIT uses default find pattern.
G, X, and Y	None specified		X 1$X,,Q	EDIT uses default exchange pattern and associated substitute pattern. Y also sets the Find default.
G, X, and Y	Null specified	Null specified	X///	EDIT uses default find pattern and a null substitute pattern. This becomes new default pair for exchange commands. Y also sets the Find default.
G, X, and Y	Null specified	Specified	X//XYZ/	EDIT uses default Find pattern and specified substitute pattern. This becomes new default pair for exchange commands. Y also sets the Find default.
U	None specified		U 1$U,,Q	EDIT uses default exchange pattern and associated substitute pattern.
U	Null specified	Null specified	U///	EDIT interprets this as a zero length delete and substitute field. It replaces nothing with nothing (but blank pads to left search window).
U	Null specified	Specified	U//XYZ/	EDIT interprets this as a zero length delete field and a substitute field.

Command Parsing

EDIT reads commands from left to right, separating the individual units of the command (line specifications, command mnemonic, options, etc.) for translating. This process is referred to as *parsing*.

This process determines how a pattern included in a search or exchange command is interpreted. It is necessary to understand the process before effective patterns can be created. The order used by EDIT to translate search and exchange commands follows:

1. EDIT notes the line range, if specified, then recognizes that it is a command that contains a pattern (commands B, F, ‘ ‘,’ ’, G, U, X, and Y).

2. EDIT identifies the pattern by noting the first non-blank, non-alphanumeric character after the command mnemonic, and uses that character as the pattern delimiter.

3. EDIT reads the pattern from left to right to the next delimiter (this delimiter must match the first delimiter used and must not be preceded by the escape character).

4. EDIT examines the pattern to see if it is legal. If the pattern is defaulted and no pattern exists in the default pattern buffer, EDIT displays the command, placing an arrow under the end of the illegal pattern. For example:

```
/1$ X//the/
?      ^
```

5. If the pattern is legal, EDIT scans and parses the rest of the command for the substitute string (if it is an exchange command) and options.

Note that the command separator (|) is interpreted as a literal character when within pattern delimiters. If you want to include another command on the line, the trailing pattern delimiter is required.

Regular Expressions Pattern Specification

Regular expressions is an EDIT/1000 option that allows pattern searches using regular character strings and *metacharacters* to define matched strings. Metacharacters are a set of characters used to specify patterns. They can be turned ON using the *SE RE on* (Set Regular Expressions ON) command. The metacharacters are

. Matches any character except the end of a line.

^ Anchors search to beginning of the window or line.

$ End of the line or window, whichever is shorter.

[xyz] A class of characters (find any one of these characters).

[^wxy] Negated class (find all but these characters).

* Match zero or more occurrences of the previous pattern.

+ Match one or more occurrences of the previous pattern.

<n> Match exactly n occurrences of the previous pattern.

@ Match zero or more characters (shorthand for .*).

: Delimiter used between alphanumerics and other characters.

{xyz} Indicates a tagged field; recalled by &n in the substitute string.

&n Recalls the n'th tagged field; used only in the substitute string (1 <= n <= 9; '&' by itself is the substitute field).

>n Same as &n, but shift substitute string to uppercase.

<n Same as &n, but shift substitute string to lowercase.

<$> Break line at position of metacharacter.

A dash between alphanumerics in a character class specifies a range, such as [0-9], [A-Z] or [A-Z0-9]. Character classes may also include escaped metacharacters such as [\^\\], which is the class containing ^ or \. Matches containing *, +, or @ extend to the longest possible match.

EDIT normally exchanges all occurrences of the character string specified in a pattern. This may not always be desirable. For example, a search for all occurrences of the word *and* would also locate the words *band*, *command*, and *sand*. Regular expressions allows you to indicate that only the string specified is to be located or exchanged and not incidences of the string occurring within other strings. This is done by delimiting the pattern string with two colons (:). For example, to exchange all occurrences of *and* with *but*, excluding incidences of the string *and* within other character strings, enter the exchange command

```
X/:and:/but/
```

For more information and examples for Regular Expressions, refer to Chapter 5.

Pattern Specification Shortcuts

The following shortcuts can be used to locate a line with the specified pattern and execute a single command, if they are used from the line where the search is to begin:

```
'pattern'<cmmd>

`pattern`<cmmd>
```

The first example finds the specified pattern by searching forward from the current pending line to the end of the file. The second example locates the pattern by searching backwards from the current pending line to the beginning of the file.

For more information on the search and exchange commands and the pattern specifications and options available with them, refer to the tutorial in Chapter 2 and the command descriptions in Chapter 4 of this manual. For more information on advanced pattern specifications, refer to Chapter 5 EDIT Regular Expressions.

Search and Exchange Command Options

EDIT provides a wide variety of find and exchange command options that you can use to alter the effect of a command. The applicable commands include Find (B and F), Delete (D), and Exchange (G, X, and Y). The parameters valid for each command are included in the command syntax provided in Chapter 4 of this manual. They include

A The All option is entered to apply the command to all lines where the specified pattern was matched. This option is valid for the B, F, and D commands.

V The Reverse pattern option is entered to apply the search command to all lines that do *not* contain the specified pattern. This option is valid for the B, F, and D commands.

Q The Quiet option is entered to suppress a listing of the lines affected by the command. This option is valid for all search and exchange commands.

N The No Window option is entered to disregard the horizontal window columns setting and allow a match to the specified pattern anywhere on a line. This option is valid on all search and exchange commands.

R The Remove option is entered to remove lines that are blank (zero length) after an exchange. Valid for the G, X, and Y commands.

S The Single Exchange option is entered with an exchange command to allow only one exchange per line, at the left-most match. Valid for the G, X, and Y commands.

EDIT Session Options

EDIT/1000 provides several options that can be set (using the SE command) for a single editing session. The options are returned to their default value when a new editing session is initiated. However, if the write (WC and WR) or File Input (FI) commands are used to manipulate files without exiting EDIT, the options set for the initial EDIT session are maintained. The options available for editing sessions are summarized in the tables that follow.

Table 3-3. EDIT Session Options

Option	Default	Description
AC	^	Anchor character used in pattern specification. Anchors the pattern to the start of search window.
AS	on	Asking for verification of dangerous commands (that is, A, AS, D, FI, K, TR, U, X, and Y). Set to OFF to suppress OK? prompt. During execution of a transfer file, asking is always off. For D, K, U, X, and Y commands, prompt is displayed only if command modifies more than one line. For the A and FI commands, OK? is displayed only if the file has been modified. Asking can be suppressed by terminating a command with the prompt character (default prompt character is "/").
BE	off	Bell. Set to ON to enable bell with EDIT prompt.
CF	on	Case folding. Must be set to OFF to differentiate between upper- and lowercase in pattern searches. This option does not affect the EDIT command syntax. EDIT accepts commands in either upper- or lowercase, even if CF = off.
CS	|	Command separator. Used to separate commands in a command or runstring. The number of commands is limited by the number of characters in a line. However, only 79 characters are saved in the command stack. The command separator does not apply inside a find or exchange pattern.

Table 3-3. EDIT Session Options (continued)

Option	Default	Description
DF	on	Display functions. Applies to the screen mode display of display enhancements such as blinking and inverse video, and control characters. When set to ON, EDIT displays graphics for control characters. OFF allows the terminal to interpret the control codes. The control characters null, line feed, carriage return, inquire, acknowledge, and delete are always removed in screen mode.
EC	\	Escape character. Used inside of patterns and line character edits to remove any special meaning from the next character. For example, \\ is interpreted as a backslash, not the escape character.
IC	@	Indefinite match, or *wildcard* character. Used in patterns to specify zero or more occurrences of any character. With Regular Expressions option OFF, it matches the shortest possible match in a line. With Regular Expressions ON, it matches the longest possible match.
LE	256	Maximum number of characters allowed per line. Changing the value of LE does not affect existing lines until you modify them. You can truncate lines to a maximum length by using the Set Length (SE LE) and Kill Trailing Blanks (BK) commands. See also, −L runstring option in this chapter.
PC	/	Prompt character. Note that "/" is also the default character used to suppress asking (OK?) for dangerous commands, and the place-saver character used in line mode character edits.
QU	off	Quiet option. Set to ON to run EDIT without listing any output to the terminal. (When EDIT prompts for input, quiet mode automatically turns off. If you interactively set quiet mode on, it looks as if nothing happened because EDIT will prompt for the next command and turn off quiet mode.)
RE	off	Regular expressions. Set to ON to allow the use of Regular Expressions mode metacharacters in special pattern searches/ matches.
RT	on	Find return. ON causes an unsuccessful search or an "All" option search command (F or B) to return to the original pending line after reaching the second line specification. OFF causes EDIT to leave the pending line at one line beyond the second line specification entered.

Table 3-3. EDIT Session Options (continued)

Option	Default	Description
SD	10,10,2	Screen size defaults. Numbers indicate screen format in the form: number of lines displayed above pending line, number of lines displayed below pending line, and number of lines overlapped. Overlapped lines are those displayed from the previous screen as a result of CTRL-P or CTRL-F commands. Screen size is equal to the sum of the first and second number, plus one. Parameters are separated by blanks or commas.
SL		Screen mode line limit. Sets the maximum number of lines that are displayed for a screen edit. If not set by a SE SL command, EDIT picks a default value based on the number of bytes of memory returned by the terminal in response to primary status request. EDIT automatically issues this request when the first S, Q, or command stack command is issued. If an SE SL 0 command is issued, EDIT assumes the terminal does not support screen mode and issues an error message in response to an S command. EDIT accepts any positive number as the line limit, but a value that exceeds the terminal's capability results in data loss during screen mode edits.
TC	TAB, CTRL-I	Line character edit tab character. Note that if the tab character is set to something other than CTRL-I, CTRL-I becomes an alternative *insert* command for line character edits.
TS	on	Time stamp. ON causes EDIT to update all time stamps when the work file is written to with an ER or WR command. Time stamp format is <YYMMDD.HHMM>, where "<" and ">" are the characters *less than* and *greater than* and they enclose the date information. The date information is numeric, where YY = year, MM = month, DD = day, HH = hours, and MM = minutes. A time stamp must end as the last or next to last non-blank character on a work file line.
VW	10,10	Vertical window. Use to set the default line range for the W (Window) and L (List) commands. The first number is the default number of lines above the pending line to start display of a window. The second number is the default number of lines below the pending line to end the display. The sum of the numbers, plus one is the default size of the window displayed for the W or L commands when only the first line specification is entered.
WC	1,256	Window columns. Specifies the left and right columns that are used as the horizontal search window for pattern searches.

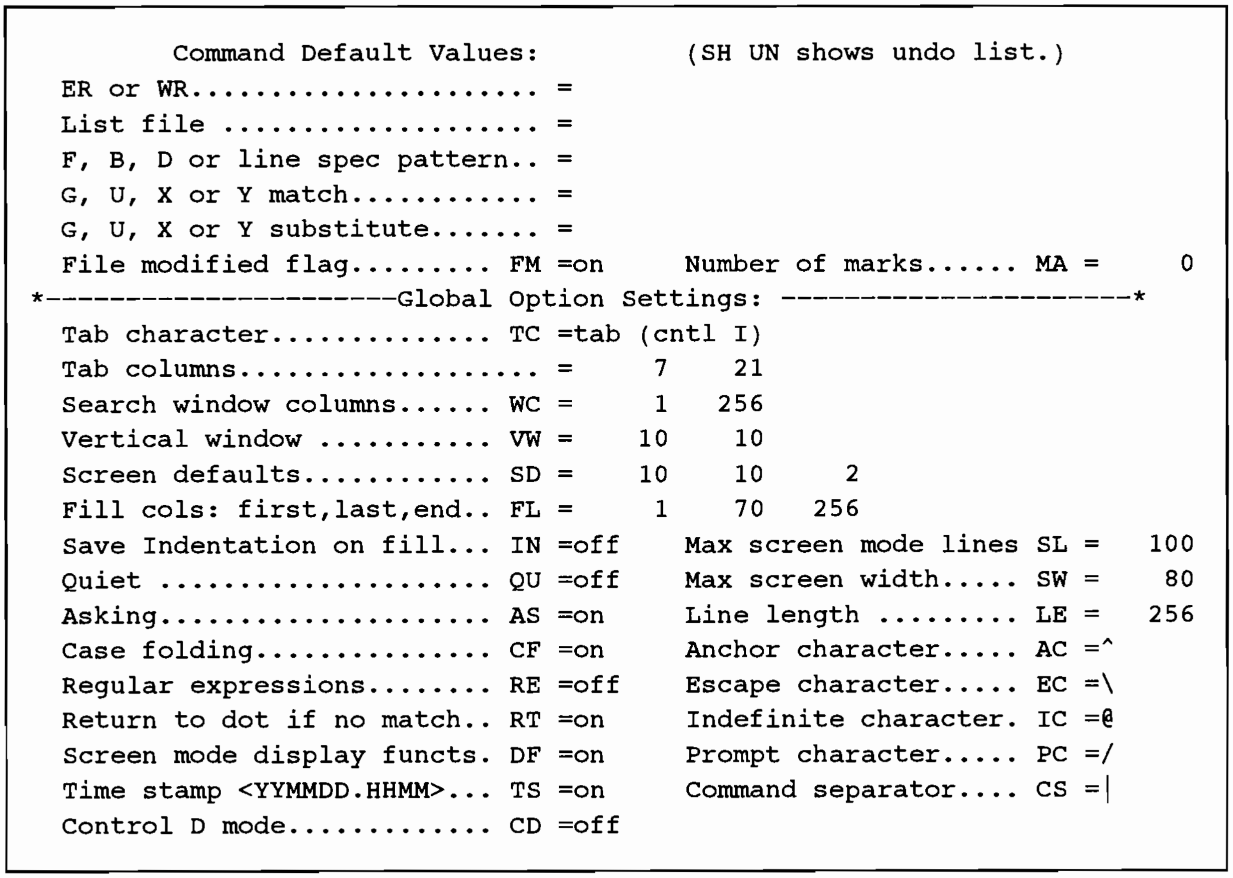
Table 3-3. EDIT Session Options (continued)

Option	Default	Description
IN		Sets the indentation on or off.
FL		Fills text to the margins set with SE FL.
SW		Screen width. Uses the SH command to show the current screen width. This will be zero until either the command stack or a screen mode command is entered, at which time the terminal screen width is sensed.
SW n		Screen width set. Uses the SE command to set screen width to n. n must be greater than −1 and less than 256. If set is 0, EDIT will sense the width and set the sensed value on the next command stack or screen mode command.

Screen width is sensed by doing a cursor left from column 1. The cursor position is then read back from the terminal. For terminals that do vertical scrolling, this will usually return 80 for the screen width. To use the vertical scroll option, the user must use the SE SW n option to set the desired width.

View Options

The defaults for these options can be checked by using the Show (SH) command. EDIT displays the defaults for all session options, as illustrated in Figure 3-3:



```
          Command Default Values:          (SH UN shows undo list.)
   ER or WR..................... =
   List file ................... =
   F, B, D or line spec pattern.. =
   G, U, X or Y match........... =
   G, U, X or Y substitute....... =
   File modified flag......... FM =on     Number of marks...... MA =     0
 *---------------------Global Option Settings: ---------------------*
   Tab character.............. TC =tab (cntl I)
   Tab columns.................. =     7     21
   Search window columns...... WC =     1    256
   Vertical window ........... VW =    10     10
   Screen defaults............ SD =    10     10     2
   Fill cols: first,last,end.. FL =     1     70    256
   Save Indentation on fill... IN =off    Max screen mode lines SL =   100
   Quiet ..................... QU =off    Max screen width..... SW =    80
   Asking..................... AS =on     Line length ......... LE =   256
   Case folding............... CF =on     Anchor character..... AC =^
   Regular expressions........ RE =off    Escape character..... EC =\
   Return to dot if no match.. RT =on     Indefinite character. IC =@
   Screen mode display functs. DF =on     Prompt character..... PC =/
   Time stamp <YYMMDD.HHMM>... TS =on     Command separator.... CS =|
   Control D mode............. CD =off
```

Figure 3-3. Example of Display for Show (SH) Command

You may also check the setting for a specific option by entering the SH command for that option. For example

```
/ sh re
   Regular expressions........ RE =off
```

Set Options

The SE (Set) command is used to set the options. Those options that are characters (as opposed to ON/OFF settings and numeric entries) can be set to any non-alpha, non-numeric printing character other than a space or a comma. The listing below provides some examples (note that the default value for each option being changed is included in parentheses). If a Set command is entered without an option value, EDIT applies the default value. For ON/OFF options, the default value is the opposite of the current value (that is, the value is toggled each time the Set command is used).

Table 3-4. Option Setting Examples

To Change...	Default	Use
Escape character to ^	\	/se ec ^
Indefinite character to "	@	/se ic "
Prompt character to)	/	/se pc)
Regular expressions option to ON	off	/se re on
Line length to 80 (After this change, lines longer than 80 characters will be truncated as they are edited or when the BK command is used.)	256	/se le 80
Case folding to off	on	/se cf off
Search window to columns 7 through 21	1,256	/se wc 7 21
Screen size default to 41 lines (The number of screen lines is still limited by the SL option)	10,10,2	/se sd 20 20 1
Maximum screen mode size to 30 lines (Be aware of maximum size so that you don't lose the screen overflow protection; Check terminal setting with SH SL command)	Number is terminal dependent	/se sl 30
Vertical window to 31 lines	21	/se vw 20 10
Anchor character to +	^	/se ac +
Tab character to]	TAB key, CTRL-I	/se tc]
Command separator to ;	|	/se cs ;
Suppress verification prompt	on	/se as off
Screen display functions	on	/se df off
To leave pending line at lower limit after unsuccessful or multiple finds	Returns to original pending line	/se rt off

Line Mode Editing

When editing in the command line mode, EDIT displays a single line of text at a time, and the EDIT prompt character is displayed on the following line. The line of text displayed is the *pending line*. This is the text that will be altered by commands entered at the EDIT prompt.

Any of the EDIT commands (including the screen mode, S, command) can be typed in at the prompt and executed by pressing the carriage return. If an error is made when entering a command, it can be *erased* by using the backspace key prior to executing the command with a carriage return. Note that the left cursor arrow key does *not* correct line mode commands.

Many of the EDIT commands can be reversed after execution through use of the Undo (UN) command. The UN command does *not* reverse the A, AS, EC, ER, FCL, FCS, FI, L, RU, S, SE, TR, UY, WC, or WR commands. (Chapter 4 provides descriptions of all EDIT commands.)

Editing in the line mode is inefficient because only one line of text is displayed at a time. However, if you wish to perform a search or an exchange, merge text from another file, display file or help information, set or show EDIT session options, or save the file, these activities must be executed from the line mode.

Line Character Edits

Line character edits are available in the line mode to enable you to do character editing without using any of the terminal editing keys. Line character editing consists of text entry and several control key/alpha key combinations.

Line character editing is possible in conjunction with the following pending line edit commands: C (Edit Pending Line and Advance a Line), O (Duplicate Pending Line and Edit), and P (Pending Line Edit) commands.

Line Character Modes

During line character editing, text entered with a pending line edit command is applied based on the character edit mode currently active: delete mode, replace mode, and insert mode. The character edit mode is determined by the control key command entered, following the pending line command:

CTRL-C Change to Delete characters mode. Enter CTRL-C first, then any non-control character (for example, x) for each character to be deleted.

CTRL-R Change to Replace characters mode. Used after a CTRL-C or CTRL-S. Use the current prompt character for characters to be replaced.

CTRL-S Change to Insert characters mode. (CTRL-I also changes to insert mode if CTRL-I does not perform the terminal tab operation.)

CTRL-P Same function as CTRL-S. (Frees CTRL-S to be used with XOFF protocols.

Additional Character Edits

Additional control characters and commands used in line character edits are described below:

<text> Text to be entered on the pending line. To preserve an existing character on the pending line during replace mode, use the current prompt character (the slash unless changed) for each character to be preserved.

CTRL-B Break the line at this position. Move remaining text to column 1 of the next line.

CTRL-T Truncate line.

CTRL-X Extend line, adding characters to the end of the line.

CTRL- A non-printing escape character. It is the same as "\" but it does not print. It cannot be redefined.

/ Enter the current prompt (default = /) to preserve the character in that position on the pending line.

<tab> Skip to the next tab stop; if in replace mode (CTRL-R), leave the skipped-over characters unmodified. Note that <tab> defaults to CTRL-I, and the terminal will probably move the cursor to the next terminal tab stop when it is entered. The terminal tabs may not match EDIT's internal tab stops, and therefore the display text may not match the internal text. For more information, refer to the information on the TL and TS commands, under the description of the T command in Chapter 4.

For the R (Replace Pending Line with Text) command and the space ( ) command, the tab operation inserts blanks up to the next tab stop. During replace mode, the tab character can be used to preserve existing characters up to the next tab stop.

The command separator character (default is "|") terminates a line character edit command. To use a literal command separator in a line character edit, use the escape character (\) to nullify the special meaning of any command.

Line Character Edit Examples

The following examples illustrate line mode edits that can be used to delete, replace, and insert characters, and edits that can be used to truncate, extend, and break text lines. Each example uses the Pending Line Edit (P) line mode command with the appropriate control key combination commands.

```
  Deletion Example for EDIT
/P/////////////////<ctrl C>xxx
  Deletion Example  EDIT
/

  This is a replacement example.
/P<ctrl C>xxxxxxxxxx<ctrl R>R
  Replacement example.
/

  This is anacter example.
/P//////////<ctrl S> insert char
  This is an insert character example.
/

  This illustrates how to truncate lines......
/P///////////////////////////////////////<ctrl T>
  This illustrates how to truncate lines.
/

  An extend line example
/P<ctrl X> is illustrated below.
  An extend line example is illustrated below.
/

  A line break example.
/P//<ctrl B>/////<ctrl B>
  break example.
/-3 .L
 00001 A
 00002 line
 00003 break example.
```

Screen Mode Editing

The EDIT screen mode is accessed by entering the Screen (S) command from the EDIT prompt or by including the S command in the EDIT runstring.

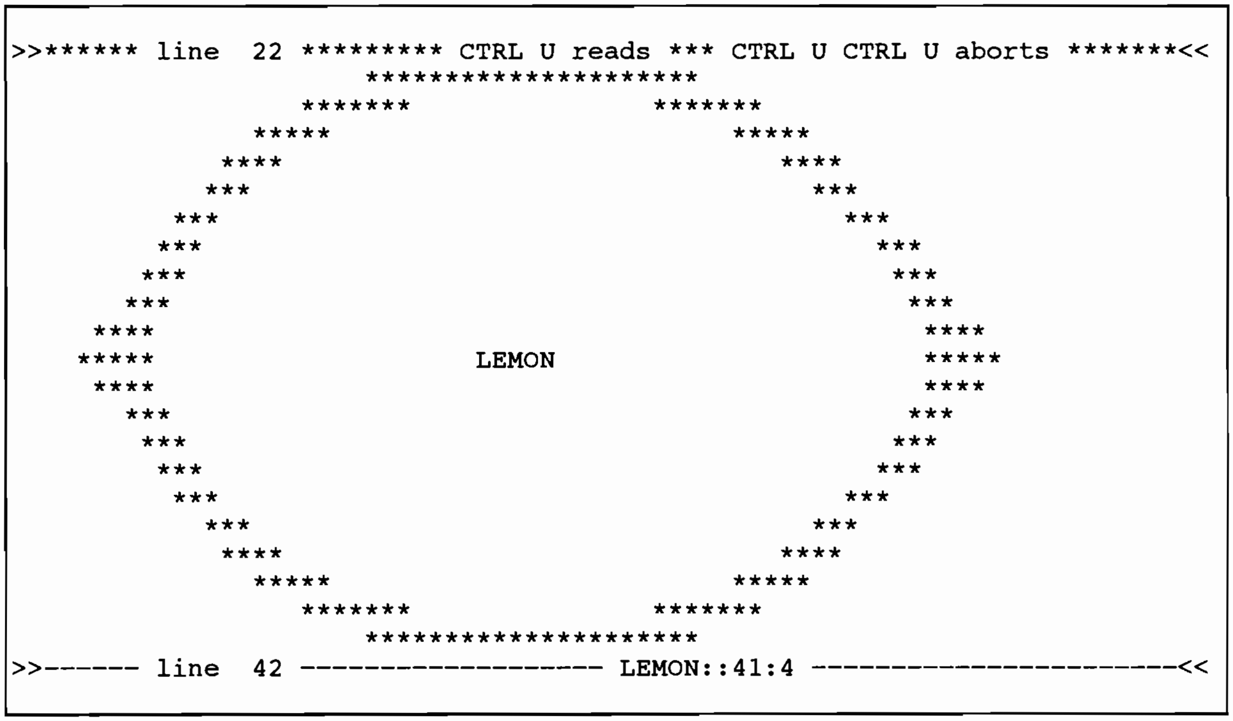
The screen mode displays several lines of text at a time and enables local terminal key edits. Screens displayed are bracketed by a beginning line and an ending line that provide the beginning and ending line numbers of the portion of the file being displayed. The beginning screen bracket line displays the command used to exit the screen mode and the ending line displays the file descriptor of the file being edited. If the file descriptor is too long to display in its entirety, EDIT displays a portion of the file descriptor followed by three dots (...).

The screen mode may be started from the pending line or any line specified. For example

Table 3-5. Examples of Screen Mode Commands

To:	Use:
Start screen around pending line	`s`
Start screen from line 100	`100s`
Start screen 45 lines before pending line	`.-45s`
Run EDIT from CI and start screen from line 1	`edit,<file>,s`
Start a large screen from current pending line	`.$ s`
Start a large screen from ten lines above the current pending line.	`,$ s`

For example, for the file LEMON, the command /22 S would display the screen illustrated in Figure 3-4:



```
>>****** line  22 ********* CTRL U reads *** CTRL U CTRL U aborts *******<<
                      *********************
                  *******               *******
               *****                          *****
             ****                                ****
            ***                                    ***
          ***                                        ***
         ***                                           ***
        ***                                              ***
       ***                                                 ***
     ****                                                   ****
    *****                    LEMON                          *****
     ****                                                   ****
       ***                                                 ***
        ***                                              ***
         ***                                           ***
          ***                                        ***
            ***                                    ***
             ****                                ****
               *****                          *****
                  *******               *******
                      *********************
>>------ line  42 ------------------- LEMON::41:4 -----------------------<<
```

Figure 3-4. Sample Screen Mode Display

Screen Display

EDIT typically displays 21 lines of text at a time (10 lines above the pending line, the pending line, and 10 lines below the pending line). To display a larger screen, use the CTRL-X command (described below). This displays the maximum number of lines allowed by the terminal. The screen size can be checked with the SH SD (Show Screen Default) command and the maximum lines allowed can be checked with the SH SL (Show Screen Limit) command. The maximum number is used for screen overflow protection. If the screen size is changed to a greater number, this protection is lost.

When moving from one screen to another, EDIT overlaps screens, including two lines on the second screen from the previous screen. (The two line overlap is the default that can be reset using the SE SD command described below.) For example, a given screen displays lines 701 to 721. If CTRL-F is used, the second screen displays lines 720 to 740, and CTRL-P displays lines 682 to 702. To redefine the default screen size and overlap, use the SE (Set) command and change the SD (Screen Default) option.

To change the screen size, use the SE SD (Set Screen Default) command. Specify three numbers to indicate the number of lines above and below the pending line and the number of overlap lines. For example, to display a 60-line screen with a one line overlap, enter

```
/se sd 30 30 1
```

Or, to display a 30-line screen with a five line overlap, use

```
/se sd ,,20 5
```

In the example above, the "`,,`" tells EDIT to use the default value for the screen size. The default value for lines above the pending line is 10.

EDIT uses a conservative algorithm to allocate the maximum number of lines it supports in screen mode. This is done because EDIT cannot determine your terminal's model number or type. If your terminal can support more than EDIT's estimate, use the Set Screen Limit (SE SL) command to set a new maximum number of lines for your screen edit.

Saving or Ignoring Screen Edits

You have the option of saving or ignoring your screen edits each time you move from one screen to another or when you exit the screen mode. Control key commands (described in the next section) are used to move between screens and exit the screen mode. All control key commands are followed by a carriage return. A single execution of a control key command (for example, CTRL-F to move forward a screen) will instruct EDIT to read the screen and update the work file with the new edits. Double execution of a control key command (for example, CTRL-F CTRL-F) instructs EDIT to ignore changes to the current screen.

Line Commands from Screen Mode

All the powerful EDIT commands (such as Copy, Move, Find, Exchange, Transfer, etc.) must be executed from the line mode. Line commands can be executed by exiting the screen mode with the CTRL-U or CTRL-Q commands. In addition, the CTRL-C command temporarily exits the screen mode to enable you to execute one EDIT line mode command.

The CTRL-C command in screen mode displays the EDIT prompt "/" at the current cursor location. After entering an EDIT command, the current screen is read before execution of the command. After execution of the command, EDIT returns to screen mode. To avoid reading (and saving) the screen, enter a double CTRL-C command. In this case, the prompt displayed is a backslash (\). If an error occurs during execution of the line command entered, you are removed from screen mode and placed in line mode.

Editing Text

Once in screen mode, you may use terminal keys and control (CTRL) keys to edit text. Although terminal keys vary between terminal models, they usually include cursor arrow keys, and insert and delete keys for characters and lines.

In a new file, blank lines must be inserted between the top and bottom screen brackets prior to entering text. The number of lines inserted is limited only by the size of your terminal memory, since EDIT reads the entire terminal screen memory when the applicable control key command is entered. If you exceed the terminal screen memory, you will lose lines from the bottom of your text screen as well as the bottom screen bracket. When the screen is read, EDIT will display a screen error message indicating that it could not locate the bottom screen bracket.

On most terminals, the memory lock feature prevents you from inserting too many lines, thus exceeding the terminal screen memory and losing the bracket and bottom text. On terminals without this feature, or when you have turned off the memory lock on your terminal, do not insert more lines than your terminal memory supports. (Consult your terminal reference manual for the terminal screen memory size.)

If you insert more than 24 consecutive blank lines, EDIT may display a screen error message indicating that it could not locate the bottom screen bracket when the reading the new screen (see Screen Mode Error Message which follows). If you must have more than 24 consecutive blank lines in your file, insert the character space on each line.

Screen Mode Control Key Commands

Sixteen control key command combinations are available on all terminals that support screen mode: Q, U, P, R, F, S, T, X, C, O, K, A, Z, J, B, and D. To enter these commands, press and hold the CTRL key, press the alphabetic key, release both keys, then press the carriage return.

Most of the control commands save new edits when they are executed. To execute the command but ignore new edits, double the command. For example, CTRL-U quits screen mode, saving new edits; whereas CTRL-U CTRL-U quits screen mode and ignores any edits made since the last screen save.

It is possible to cancel a control key combination that you entered but do not wish executed, if you press some other printing character or non-command key PRIOR to pressing the carriage return.

CTRL-A Moves the cursor to the first character on the line. Also, allows you to reset margins (set through local terminal keys) back to the initial terminal setting.

CTRL-Z Moves cursor to last character on the line. Also resets margins (see CTRL-A). If a line is longer than 78 characters, CTRL-Z moves the cursor to the last character of the line, whether that character is on the same line as the first character or five lines below.

ESC-4 CTRL-B Breaks line into two lines at the current cursor position. Blanks are inserted at the front of the second line so that it lines up with the first non-blank character of the first line. ESC-4 sets a new left margin and CTRL-B breaks the line at the new margin. Margins are then reset by the command.

CTRL-J Joins current line to following line. Also resets margins (see CTRL-A).

CTRL-C Allows execution of *one* line mode command. After carriage return, EDIT prompt is displayed (/ if ^C is used, and \ if ^C^C is used). Enter line mode command(s). When multiple commands are entered on the same line, separate the commands with a vertical bar (|). Press the carriage return. After command execution, EDIT returns to screen mode with cursor positioned at the current pending line.

Note

Do not use the cursor arrow keys to move the cursor from the line provided for line mode entry after use of CTRL-C. When the carriage return is used, the line containing the command to be executed is deleted. If the cursor is on another line when the carriage return is pressed, a line of text is deleted and the command becomes part of your text.

CTRL-F Saves new screen edits and advances to next screen. Double to ignore new edits.

CTRL-P Saves new screen edits and goes back to previous screen. Double to ignore new edits. CTRL-P does not work on X.25 pad terminals.

CTRL-R Saves new screen edits and goes back to previous screen. Use for X.25 pad terminals.

CTRL-K Marks line indicated by the current cursor location. Following the carriage return, places a colon in column 79 of the screen and requires entry of an alphabetic character.

CTRL-O Copies the line indicated by current cursor location.

CTRL-Q Saves screen edits and quits screen mode. Double to ignore new edits. Should not be used if your terminal uses Xon/Xoff handshake protocol.

CTRL-S Saves new edits and starts next screen at line indicated by the current cursor position. Double to ignore new edits. Should not be used if your terminal uses Xon/Xoff handshake protocol.

CTRL-T Same as CTRL-S. Use for terminals with Xon/Xoff handshake protocol.

CTRL-U Same as CTRL-Q. Use for terminals with Xon/Xoff handshake protocol.

CTRL-X Same as CTRL-S, except a large screen is provided. Size of screen is determined by amount of terminal memory available. Check size available with the SH SL command.

Note

If your terminal follows Xon/Xoff handshake protocol, CTRL-Q and CTRL-S are not interpreted as EDIT commands. Use CTRL-U in place of CTRL-Q and CTRL-T instead of CTRL-S.

CTRL-D

Control-D (CTRL-D) is a sub-mode of screen mode that adds line editing graphic commands to screen mode. CTRL-D mode can be enabled by typing either CTRL-D plus carriage return while in screen mode, or by typing the SE CD (set control-D) command. In either case, display functions will be turned off. Once CTRL-D mode is enabled, it stays enabled until it is disabled with another SE CD command. While CTRL-D mode is enabled, screen mode back spaces are destructive, and DEL or rub-outs echo \ followed by a carriage return linefeed. The edit display functions flag will be set to OFF when CTRL-D mode is enabled and should be left off while CTRL-D mode is being used.

To use the line drawing features of CTRL-D mode, your terminal must have the line drawing character set ROM installed and enabled (usually, it is enabled with the sequence "ESC, right parenthesis, uppercase B"). If your terminal does not have the line drawing character set installed, you can still use the Move and Copy features of the CTRL-D mode. Printing can be done on any printer that supports the HP line drawing sets.

When you enter a CTRL-D during screen mode with CTRL-D sub-mode enabled, EDIT responds by turning on an inverse video mark around the current cursor position. It also saves the screen location of this point in an internal list of points. Entering additional CTRL-Ds displays additional inverse video marks on the screen and adds points to the points list. These points become the arguments of the CTRL-D mode commands.

Normally these points define a closed polygon around the area to be used by the CTRL-D command. When two consecutive points, having different row and column addresses, are used to define a line or an edge, a horizontal line or edge is defined first followed by a vertical line or edge to reach the second point. (Note that lines and edges must be horizontal or vertical.)

Whenever there is an inverse video point displayed, you are in CTRL-D mode. While in CTRL-D mode, characters are not echoed as they are entered. Rather, they are interpreted as CTRL-D mode commands (see the following list) when carriage return is pressed. If the entered characters do not form a valid CTRL-D command after pressing carriage return, the terminal bell will sound, and the cursor is placed at the last point in the list.

Note that systems without the new serial drivers will echo characters due to an incompatibility in the MUX driver. To enter a command on these systems, type CTRL-C followed by a carriage return; EDIT will open a line on the screen and display the following prompt:

```
CTRL-D mode>
```

At this point, you should enter a CTRL-D mode command. The prompt and the command will be executed and then erased after you press carriage return.

To erase the last point in the CTRL-D list, position the cursor to the display point and enter another CTRL-D. This will erase the inverse video mark and the point for the list. To remove all points, enter the CTRL-D command “Q”.

CTRL-D mode commands are defined below. They are all single character commands sometimes followed by options. The characters inside the angle brackets are options and are defined later in the list.

A<B E P N> Arrow. Draw a path (see the “P” command for path definition) through the defined points, and put an arrow head at the end of the path. Arrow heads may not display correctly on most terminals; however, LaserJets with the line drawing character set will display them.

B<B E P N> Box. Draw a closed polygon through the defined points.

C Copy. The last point is saved as a locator point and then removed from the list. The area defined by the closed polygon through the remaining points is then copied so that the first marked point is on top of the locator point.

E Erase path. The path (see “P” command) through the points is erased. (This is the same as the P E command.)

L<B E P N> Lines. Take the point list as pairs of points and draw lines between the pairs

M Move. Same as copy, but erase the original area before copying.

P <B E P N> Path. Start at the first point in the list and draw a line through each consecutive point until the end of the list has been reached. As always, if two points are not aligned horizontally or vertically, draw the horizontal line first followed by a vertical line.

Q Quit. CTRL-D mode. That is, erase all marks and return to normal screen mode.

R Re-mark. Restore the marks of the previous CTRL-D mode command and reenter CTRL-D mode. If the previous command was Copy or Move, the locator points are not restored.

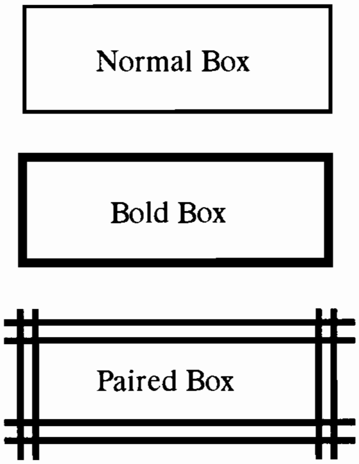
U Undo. Undo the last CTRL-D mode command.

Options:

N No read. Normally the screen will be read before the CTRL-D mode command is executed. If there have not been any screen mode changes to the area to be affected by the CTRL-D mode command that is going to be executed, the screen read time can be saved by entering an "N" with the command.

Line Style Options: B=Bold, P=Paired, E=Erase.

Example Boxes:



Screen Mode Error Message

If the beginning or ending screen bracket lines are deleted, EDIT displays the following message:

```
Stop and/or Start Line(s) Not Found.
O  Saves Original Text Written To Screen
S  Saves Text Just Read From Screen
B  Saves Both(Inserts Screen Text Before Original Text)
What Should Be Saved?
```

Enter O to save the original screen contents, S to save all text that EDIT read in as it rolled it across the screen, or B to save both the original screen contents and the new text read in. If you are unsure of what should be saved, B is the safest action, as it is easier to delete duplicated text than it is to re-enter lost text.

Line Length Limit

In screen mode editing, the screen line limit is 78 characters. To enter lines longer that 78 characters, use the cursor arrow keys to place two periods at columns 79 and 80 of the screen, then continue to enter text for the same line on the next line of the screen.

Exiting Screen Mode

Exit the screen mode with either the CTRL-U or CTRL-U CTRL-U command. The CTRL-U command returns EDIT to the line after the last line in the screen. The CTRL-U CTRL-U command returns EDIT to the first line of the screen.

For more information on the screen mode, refer to the tutorial in Chapter 2 and to the description of the S command in Chapter 4.

Filling in Text

The Fill command (FL) is used to fill text to the margins set with the SE FL command and displayed with the SH FL command. The <first> column is the left fill margin, the <last> column is the right fill margin, and the <end> column is the last column from which to gather text. Over the range of lines specified, this command attempts to fill each line between the fill margins. Lines that are too long will have their ending words moved to the next line. Lines that are too short will be made longer, if possible, by moving up words from the following lines. Paragraph breaks are indicated by lines that are blank between the <first> column and the <end> column inclusive. The first and last line of the line specification are also treated as a paragraph break.

The first word of a paragraph may be indented. Setting fill indentation ON (see the SH IN command) keeps the Fill command from moving the first word of a paragraph. It starts the fill after the first blank following the first non-blank character in the paragraph. Indentation may be either positive or negative.

The Fill command will put one blank between words except in the case where sentence-ending punctuation ‘ ! ?) : . ’ is followed by a capital letter, in which case two blanks are inserted. The punctuation and capital letter will be ‘seen’ even if hidden by single or double quotes. However, the Fill command will not put in two blanks where only one is currently in the line; for example, ‘Dr. Foo’ will not be changed.

Fill command format:

```
[.][*] FL[Q][R][I]
```

Options:

Q Quiet operation (don’t list changed lines).

R Remove indentation (overrides current SE IN value)

I Leave indentation (overrides current SE IN value).

Advanced feature:

Fill allows you to fill text in boxes. The boxes may be either standard character set boxes, or alternate character set boxes. In order to do this, Fill counts escape and control sequences, and attempts to use column numbers as they appear on the screen. In the case of line-drawing boxes, the ON and OFF control should be immediately before and after each edge; otherwise, the Fill command may assume that they are part of the text to be filled. Note that extra control characters may confuse the Fill command.

Example of a box before using Fill:

```
  --------------------------------------------
*1|     This is text of a      box.            |*1
*2|                A test to see if the fill   |*2
*3|code will actually work.  These lines need  |*3
*4|to be filled.  We also have numbers to keep |*4
*5|track of the lines.                         |*5
*6|                                            |*6
  |--------------------------------------------|
''''/''''1''''/''''2''''/''''3''''/''''4''''/''''5''''
```

Fill instruction:

SE FL 5 50 50 Sets up the columns.
:a :b FL Fills the box.

Example of a box after using Fill:

```
  --------------------------------------------
*1|This is text of a box.  A test to see if the |*1
*2|fill code will actually work.  These lines   |*2
*3|need to be filled.  We also have numbers to  |*3
*4|keep track of the lines.                     |*4
*5|                                             |*5
*6|                                             |*6
  |---------------------------------------------|
''''/''''1''''/''''2''''/''''3''''/''''4''''/''''5''''
```

If more lines are needed in a paragraph, they are created by using the break line prior to the <start> column and after the <end> column.

If lines are left over after the fill, they are purged only if they are totally blank.

Error and Information Messages

Two types of error or information messages are displayed during operation of the EDIT/1000 program: user input errors and RTE file system errors. All EDIT/1000 error messages and the most common RTE file system errors are described in detail in Appendix A, Error and Information Messages.

EDIT reprints user input errors followed on the next line by a question mark in column one. EDIT places a caret (^) under the erroneous field. Error and informational messages are displayed starting in column one to differentiate them from listings of the file text. For example:

```
/125 % f
?    ^            (Illegal line range; user
/                  entered % instead of $)
```

Figure 3-5 provides examples of informational messages.

```
/sh zebra
Not an option.  Type SH to show all options and their current set-
ting.
/

/ec,lemon
File already exists LEMON:::4:1

/m,xyzww
No such file XYZWW
```

Figure 3-5. Examples of EDIT Informational Messages

When it is displayed, the RTE break message requires user input. This message is displayed as a result of user entry when the system is not ready for input. In other words, you may have inadvertently interrupted the EDIT session, or you may have purposefully interrupted the session to cancel an EDIT command or listing (see following section on Break Mode). The RTE-A and RTE-6/VM break messages are displayed below:

```
CM> _                                  (RTE-A)

S=<LUnumber> COMMAND ? _               (RTE-6/VM)
```

Press the carriage return to ignore the message and EDIT will continue the current operation. Enter *BR* to terminate the current EDIT operation. EDIT stops the current operation, prints the message *Operator Break*, and you can proceed to enter EDIT commands.

Recovering from Errors

The most common errors occurring during an EDIT session are user input errors. Occasionally, however, the RTE system may crash or the EDIT program may abort itself due to an internal error in the middle of an editing session.

User Errors

Typing errors made when entering a command at the EDIT prompt can be *erased* by using the terminal backspace key prior to entering the command with the carriage return. Almost all of the EDIT commands can also be reversed after entry by using the Undo (UN) command. Simply enter UN at the EDIT prompt to reverse the last work file change. EDIT is able to undo the last work file change because it maintains an *undo list* of the last changes made as a result of an EDIT command. To view this list prior to using the UN command, enter the SH UN (Show Undo List) command at the EDIT prompt.

If changes are made to a file that cannot be reversed using the UN command, you should abort (A) EDIT, to avoid saving the changes, or start re-editing the original text file using the FI: command. (The colon defaults to the most recently used filename.)

Break Mode

The RTE break commands can be used at any time to terminate the execution of an EDIT command or command listing. To interrupt the command, press any key. The system displays one of the following messages:

```
S=<LUnumber> COMMAND ? _              (RTE-6/VM)

CM> _                                 (RTE-A)
```

Enter *BR*. EDIT stops the current operation, prints the message *Operator Break*, and you can proceed to enter EDIT commands. If EDIT was executing an exchange during the break, the exchange is partially completed when EDIT resumes. The UN command can be used to undo the partial change.

System Errors

When the EDIT program is abnormally terminated due to a system crash, or an internal EDIT program error, the work file is left in the work area of the storage diskor cartridge. EDIT runs a recovery operation (referred to as recovery mode) the next time it is run to allow any editing not saved before the abnormal termination to be written to a permanent file.

During the recovery mode, an attempt is made to reconstruct the internal tables from information contained in the work file. This allows the data to be checked with the List or Find commands or to be written to a file with the EC (Exit and Create) or ER (Exit and Replace) command. If unable to reconstruct any of the internal tables, EDIT asks you if it is OK to purge the work file. You should answer *yes*. If EDIT is able to reconstruct the work file, you must save it (with an EC or ER command) or purge it (with an A command) before editing can be done. If you wish to leave the work file unchanged, you can used the AS (Abort and Save Work File) command.

If EDIT finds inconsistencies in the work file, it attempts to correct them. Depending on the nature of the inconsistencies, EDIT may or may not be successful. In any event, do not write over the original file with the recovered version without making a comparison between the two files.

The recovery mode can be terminated only by exiting EDIT. While in the recovery mode, the work file cannot be changed. If the EDIT session is aborted in the recovery mode, the work file is purged. If the work file must be saved, or you happened to enter the recover mode for someone else's file and do not know if it is OK to delete the work file, abort using the AS (Abort and Save Work File) command. This may allow time to consult with the System Manager to determine what can be done to save the file. However, unless the –S runstring option is used, EDIT continues to enter the recovery mode each time it is run until the work file is purged or recovered (with EC or ER).

The only way to exit EDIT and save the work file without purging it is to use the AS (Abort and Save Work File) command. If the AS command is used, EDIT displays the dangerous command confirmation prompt (OK?). The next time an EDIT session is initiated, EDIT runs a recovery operation to reconstruct the work file, as in Figure 3-6.

```
/as
OK? y
EDIT aborted by user
Closed file SMART::HERMAN:4:1:2
end of edit
CI> edit
File already exists EDIT+0000000000.EDIT::SCRATCH:1:26:128
Work file EDIT+0000000000.EDIT::SCRATCH:1:512:128
EDIT : Use ? for help
Entering recover mode.
      6 lines recovered.
/
```

Figure 3-6. Example of Use of the Abort and Save (AS) Command

To recover a work file resulting from an aborted EDIT session on another terminal, enter the runstring

```
CI> edit,-r <work file descriptor>
```

A brief explanation of the recovery mode can be obtained by entering the ? RM (Help on Recovery Mode) command. In addition, the recovery mode can be circumvented by using the runstring option that places the system scratch file on another disk or directory:

```
CI> edit,-s::<directory>
```

EDIT responds to this command by displaying the directory that the file will be placed in, followed by the full file descriptor.

The following message indicates that EDIT has run out of space for the work file:

```
OUT OF DISK WORK SPACE
```

When this message is displayed, no additional changes are allowed to your work file. If the message is displayed while your file is being read, you should abort EDIT. Otherwise, use an ER or EC to save your work and exit EDIT.

If you run out of work space and have sufficient space in a directory other than the default directory, you can place the work file on another storage disk or directory. For example, to place the work file on directory EK, enter:

```
CI> edit,-s::ek, <filename>
```

In this example, upon any abnormal EDIT termination, the work file remains on directory EK. The next time EDIT is run, it goes to the default work area disk to open a work file and, because the file is on a different disk, does not enter the recovery mode.

Security

If the system crashes or EDIT is abnormally terminated, the work file is accessible to any user on the system the next time an editing session is started. If you are editing a file for which security is imperative, use the −S option in the EDIT runstring to specify a private directory for residence of the work-file. If the system crashes during an editing process, this practice ensures that you will have sole access to the work file and will be present to monitor the recovery operation.

Recovery Mode Messages

The following error/informational messages may be displayed during the recovery operation:

```
Disk read error on first recover read.

Frame size words do not match in recover file.

Recover frame size words not correct for this
Edit.

Both head and tail missing in recover file.
```

Any of the above messages indicate that the work file is corrupted. Examine the work file carefully to determine if any portion of the file can be salvaged.

Batch Operation

EDIT can be run by batch or interactively. When running EDIT, several methods are available to accomplish the task without user interaction. One means is to create a file consisting of EDIT commands. This command file is then transferred to from an editing session or included in the EDIT program runstring. A second method involves including editing commands in the EDIT program runstring. EDIT also provides batch and quiet options that control the amount of information displayed on your screen and EDIT's action when an error is encountered.

In the command file method, you instruct EDIT to read its commands from a prepared file instead of typing them into your terminal. This is done by using EDIT's Transfer (TR) command. This transfer can be entered during an EDIT session. When the end of the command file is reached, EDIT reverts back to interactive operation, prompting you for the next command. If there is an Abort (A) or Exit (ER or EC) command in the command file, EDIT executes the command and terminates the session.

In the runstring method, you enter all the commands required to perform your editing tasks on the line you type in at the CI prompt. Obviously, if you choose to use this method, the number of required EDIT commands must be small enough to fit on one line. You are limited to a total of 252 characters, including the EDIT program name and the source file name.

You can use the command file method without ever using EDIT interactively. This is done by using the runstring method to cause EDIT to execute a transfer (TR) command to your EDIT command file. Note the following example:

```
CI> edit -q test.txt tr change.edc
```

This command causes EDIT to run in quiet operation, use the source file TEST.TXT, and read its commands from the file CHANGE.EDC.

−Q and −B Runstring Options

EDIT provides two runstring options that are useful during batch editing. The Quiet (−Q) and Batch (−B) options must be entered before the source filename in the runstring. Both cause EDIT to suppress the listing of text and information to the screen. The difference between the options is in EDIT's response to command or data errors. If the −Q option is used, reaching the end of the command file or encountering an error causes EDIT to become interactive and prompt the user for input. For the −B option, reaching the end of the command file or encountering an error causes EDIT to abort itself without updating the source file.

EDIT Session Status

EDIT sets a status of 0 or −1 in the CI variable $RETURN1 to indicate how the editing session (interactive or batch) was terminated. The value 0 indicates that EDIT was terminated with an Exit (ER or EC) command. The value −1 indicates that EDIT was terminated with the Abort (A) command, or that EDIT aborted itself instead of asking for user input because the −B option was included in the runstring. This returned status can be tested by using the CI *IF* command.

Entering Multiple Commands on One Line

Multiple EDIT commands can be entered on one line in the runstring in an EDIT command file or during interactive operation. Each command is separated from the next by a special character called the command separator. The default for this character is the vertical bar (|), and it can be changed to any punctuation character other than a space or a comma by using the *SE CS* command. Multiple commands on one line are then executed in order, from left to right.

When multiple commands are entered on one line during an interactive editing session, the entire command string (up to 79 characters) is saved in the command stack. In a command string with multiple EDIT commands, a failed command (such as a no-match in a find command) causes EDIT to ignore the next command and to return to interactive operation.

Figure 3-7 provides examples of including commands in the EDIT runstring.

```
CI> edit,lemon,1$ x/jsm/jsb//|bk|se cf off|1$ f/jsb/|-1|msubrtn|er

CI> edit,lemon,1$ ‘u//   /q/‘|er
```

Figure 3-7. Including Commands in the EDIT Runstring

The first example runs EDIT for the file LEMON; performs an exchange (X) on the entire file to change all occurrences of JSM to JSB (suppressing the OK? confirmation prompt with a slash); deletes all trailing blanks (BK); sets (SE) the Case Folding option to off; performs a search (F) to locate the first occurrence of the string JSB; goes back (up) one line in the text and merges (M) in the file SUBRTN; and finally, exits the file and replaces the source file with the work file.

The second example runs EDIT on the file LEMON and performs an unconditional exchange (U) to place three blanks at the start of each line, uses the quiet (Q) option to suppress a listing of the exchange, and exits, replacing the source file. Note the use of back quotes in the runstring to suppress the special processing performed by CI to change all blanks to commas.

Running a Program from EDIT

To run a program from EDIT, enter RU followed by the runstring. If you run another EDIT for any reason, do not use the EDIT command separator (|) in the runstring, or it will terminate the run command.

The first example in Figure 3-8 illustrates an attempt to run the help program for additional explanation on the error message *Illegal file name.* In the second example, another copy of EDIT is run on the file TEST. The second copy of EDIT is then exited with an ER command and the original editing session is resumed.

```
/ec123
Illegal file name 123:::4:4
/ru,help
FMGR-015
ILLEGAL NAME
THE FILE NAME DOES NOT CONFORM TO THE SYNTAX RULES. CORRECT THE NAME
AND RE-ENTER THE COMMAND.

Resume EDIT on CHAP1::41:4
/

/ru,edit,test
EDIT : user ? for help
no such file TEST
An ER or the first WR will create it.
EOF
(File is edited)
/er
Closed file TEST::41:4
end of edit
Resume EDIT on CHAP1::41:4
/
```

Figure 3-8. Examples of Running a Program from EDIT

4

EDIT/1000 Commands

Introduction

This chapter provides detailed descriptions of the EDIT/1000 commands and options. The contents include

- descriptions of the EDIT/1000 alphabetic commands
- descriptions of the EDIT/1000 non-alphanumeric commands

For each command, the description includes the uses, command syntax, remarks, and examples. For most commands, a summary of the command default is also included.

The information in this chapter will enable you to

- understand the operation of each EDIT/1000 command
- select the EDIT/1000 commands needed to accomplish your tasks

A (Abort Edit Session)

Uses

Use to abort EDIT/1000 session without saving changes.

Command Syntax

A[/]

/ Suppresses the OK? prompt displayed when changes have been made.

Remarks

If changes have been made to the work file, this is considered a dangerous command because it has the potential to delete data that you meant to save. To prevent this from happening, EDIT displays an OK? prompt after entry of the A command. If desired, the OK? prompt can be suppressed.

Examples

Figure 4-1 illustrates the use of the A command with the OK? prompt. The third example shows use of the A command with a slash to suppress the OK? prompt.

```
EOF
/a
OK? n
Command not executed.
/

/a
OK? y
EDIT aborted by user
Closed file TESTFILE::HERMAN:4:1
end of edit
CI>

/a/
EDIT aborted by user
Closed file TESTFILE::HERMAN:4:1
end of edit
CI>
```

Figure 4-1. A (Abort) Command Examples

AS (Abort and Save Work File)

Uses

Use to abort the EDIT session but save the work file. The next time the EDIT program is executed, EDIT enters the recover mode to recover the changes made prior to your use of the AS command. (Refer to Chapter 3 for an explanation of recovery mode.)

Command Syntax

AS[/]

/ Suppresses the OK? prompt.

Remarks

The next time EDIT/1000 is scheduled from the same session number, the program enters the recover mode and recovers the file you were working on when you executed the AS command. Note that until recover is run, the affected file may be inadvertently deleted by any user on the system (for example, during clean-up of the scratch directory).

This is considered a dangerous command because it has the potential to delete data that you meant to save. To prevent this from happening, EDIT displays an OK? prompt after entry of the AS command. If desired, the OK? prompt can be suppressed.

Examples

Figure 4-2 illustrates the use of the AS command to abort EDIT, saving the work file. The ER (Exit and Replace) command is used to save the file following the recovery operation.

```
/as
OK? y

EDIT aborted by user
Closed file TESTFILE::HERMAN:4:1:28
end of edit
CI>

CI> edit
File already exists EDInn+0000000000.EDIT::SCRATCH:1:256:128
Work file EDInn+0000000000.EDIT::SCRATCH:1:512:128

EDIT: Use ? for help
Entering recover mode.
    4 lines recovered.
/er testfile
Opened file TESTFILE::HERMAN:4:1:28
Closed file TESTFILE::HERMAN:4:1:28
end of edit
CI>
```

Figure 4-2. AS (Abort and Save Work File) Command Example

B (Find a Pattern)

Uses

- Use to search your work file from the beginning of the file for a line containing the pattern specified. Letter options may be used to specify a multiple search to locate more than one occurrence of the pattern.
- Use with Set option RE (Regular Expressions) set to ON to search using the regular expression metacharacters.
- Use with Set option RT (Return) set to ON (default) to return to pending line after search if no match found or if the All (A) search option is specified.
- Use with RT set to OFF to remain at the line below the line range specified if no match is found or the All (A) search option is specified.
- Use with Set option CF (Case Folding) set to OFF to differentiate between upper and lower case letters during search.
- Use with the horizontal search window (set the WC option) to restrict the search to within specified columns.

Command Default

1 $ B/last find pattern/

Command Syntax

[linespec 1] [linespec 2] B/pattern/ [A V Q N]

linespec 1
linespec 2 Optional line range. If no line range is specified, the search begins at line 1 and continues to the end of the file. (See the section on line specifications in Chapter 3 for a complete list of possible line specifications.)

/pattern/ Pattern to be found. The suggested delimiter for a pattern string is the slash (/). Any punctuation mark can be used except comma or spaces. First and last delimiter must match. If no options are used with the command, the last delimiter may be omitted.

A Optional all flag. Enter to find and print all occurrences of a pattern.

V Optional reverse flag. Enter to find a line that does NOT contain the pattern specified.

Q Optional quiet flag. Enter the Quiet option with the A option to indicate the total number of matched lines, without listing the lines.

N Optional no window flag. Enter to instruct EDIT to disregard the search window columns during the search.

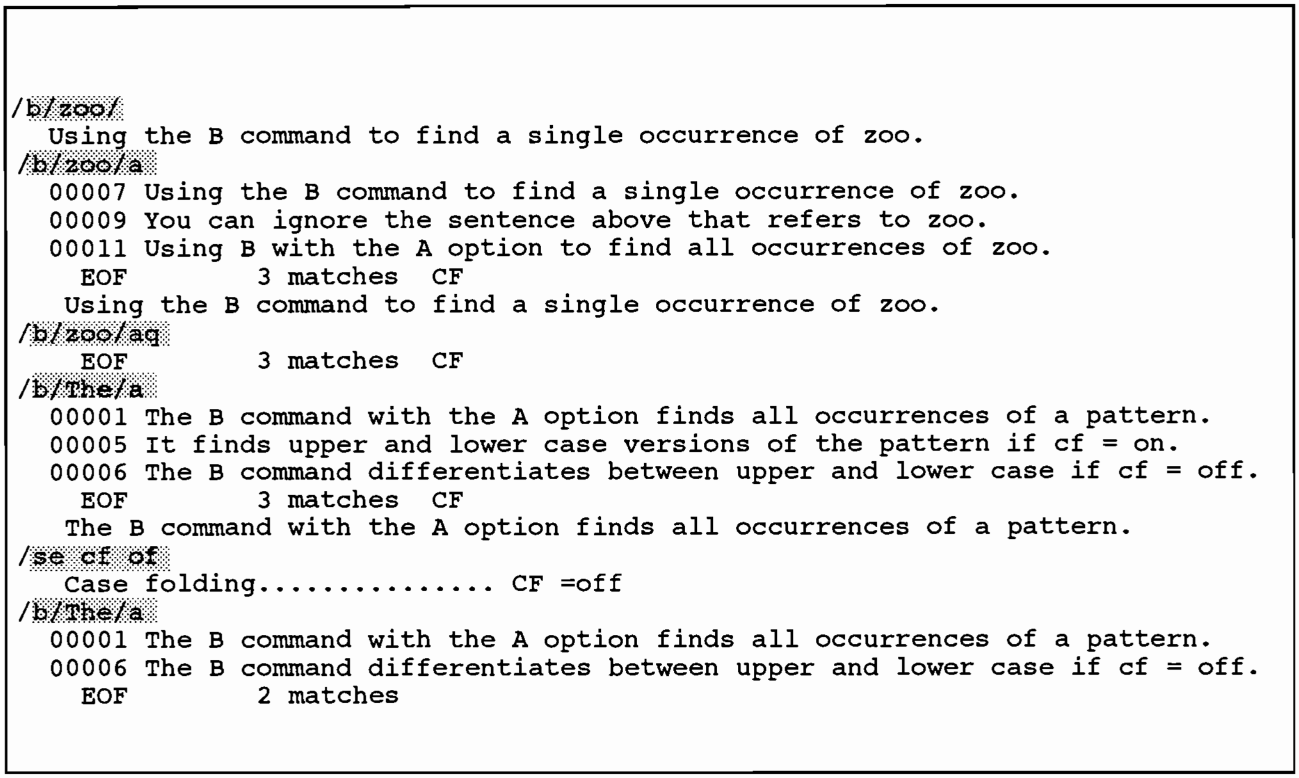
Remarks

The B command is similar to the F command, except that the start search default is line 1, whereas the start search default for the F command is the pending line.

The B command uses and sets the default Find match pattern.

Examples

Figure 4-3 illustrates the use of the B command to find a pattern, to find all occurrences of a pattern, to find all occurrences without listing the matched lines, and to find all occurrences of a pattern with the Case Folding option off to differentiate between upper- and lowercase letters.



```
/b/zoo/
  Using the B command to find a single occurrence of zoo.
/b/zoo/a
  00007 Using the B command to find a single occurrence of zoo.
  00009 You can ignore the sentence above that refers to zoo.
  00011 Using B with the A option to find all occurrences of zoo.
    EOF        3 matches  CF
   Using the B command to find a single occurrence of zoo.
/b/zoo/aq
    EOF        3 matches  CF
/b/The/a
  00001 The B command with the A option finds all occurrences of a pattern.
  00005 It finds upper and lower case versions of the pattern if cf = on.
  00006 The B command differentiates between upper and lower case if cf = off.
    EOF        3 matches  CF
   The B command with the A option finds all occurrences of a pattern.
/se cf of
   Case folding.............. CF =off
/b/The/a
  00001 The B command with the A option finds all occurrences of a pattern.
  00006 The B command differentiates between upper and lower case if cf = off.
    EOF        2 matches
```

Figure 4-3. B (Find a Pattern) Command Examples

BC (Block Copy)

Uses

Use to copy a rectangle of text from one area to another area in the work file. This command enables cut and paste editing.

Command Default

.–1 * BC 1:256 1

Command Syntax

[linespec 1] [linespec 2] BC startcol [:stopcol] [destcol] [Q]

linespec 1
linespec 2 Optional line range that indicates height of rectangle to be copied. If no line range is specified, BC copies only the line above the current pending. (See the section on line specifications in Chapter 3 for a complete list of possible line specifications.)

startcol:stopcol Starting and stopping column numbers for the block copy. Indicates width of rectangle to be copied. Stop column is optional. If omitted, it defaults to last column (determined by length of the line on a line by line basis). If both start and stop columns are included, they must be separated by a colon (:).

destcol Optional destination column for placement of the copied block. Along with the pending line, this parameter defines the upper left corner of the destination. If omitted, text is copied to the end of the destination line (current pending line).

Q Optional. Enter the Quiet command to suppress a listing of the copied block.

Remarks

The text from the source area overwrites the text at the destination.

The BC command should be executed from the line where you wish the first line of the copied text to overwrite current text.

If at the end of the file, this command appends the copied text.

The Undo (UN) command reverses the BC command.

Refer also to the BM (Block Move) command.

Examples

Figure 4-4 illustrates the use of the BC command to copy columns three and four of lines one through three to line three, columns four and five. Note the use of the LN (List Numbered Lines) command to list the lines before and after execution of the BC command.

```
  AAxxAAA
/1 LN 4
 00001 AAxxAAA
 00002 BByyBBB
 00003 CCzzCCC
 00004 DDDDDDD
/3
  CCzzCCC
/1,3 bc 3:4 4
  CCzxxCC
  DDDyyDD
     zz

  CCzxxCC                                                  (Pending line is line 3.)
/1 LN
 00001 AAxxAAA
 00002 BByyBBB
 00003 CCzxxCC
 00004 DDDyyDD
 00005    zz
/
```

Figure 4-4. BC (Block Copy) Command Example

BK (Kill Trailing Blanks & Truncate Lines)

Uses

- Use to delete all trailing blanks on every line of the file.
- Use to truncate lines longer than a maximum line length set with the line length option (SE LE).

Command Default

1 $ BK

Command Syntax

[linespec 1] [linespec 2] BK

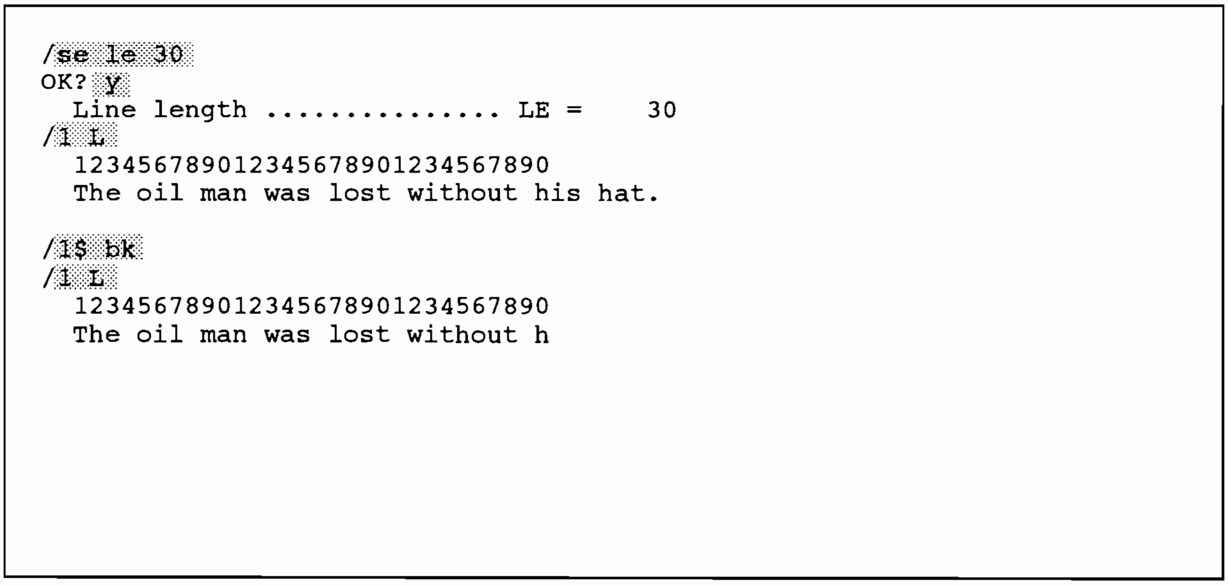
linespec 1
linespec 2 Optional line range. If no line range is specified, the operation begins at line 1 and continues to the end of the file. (See the section on line specifications in Chapter 3 for a complete list of possible line specifications.)

Remarks

To check the line length prior to executing the BK command, use SH LE (Show Line Length). Use *SE LE n* to set line length.

Examples

Figure 4-5 illustrates the use of the SE (Set) command to set the line length (LE) to 30. The BK command is then used to delete all characters beyond column 30. Finally, the L (List) command is used to display the change.



```
/se le 30
OK? y
  Line length ............... LE =    30
/1 L
  123456789012345678901234567890
  The oil man was lost without his hat.

/1$ bk
/1 L
  123456789012345678901234567890
  The oil man was lost without h
```

Figure 4-5. BK (Kill Trailing Blanks and Truncate Lines) Command Example

BM (Block Move)

Uses

Use to move a rectangle of text from one area to another area in the work file. This command enables cut and paste editing.

Command Default

.−1 * BM 1:256 1

Command Syntax

[linespec 1] [linespec 2] BM startcol [:stopcol] [destcol] [Q]

linespec 1
linespec 2 Optional line range that indicates height of text rectangle to be moved. If no line range is specified, BM moves only the line above the current pending. (See the section on line specifications in Chapter 3 for a complete list of possible line specifications.)

startcol:stopcol Starting and stopping column numbers for the block move. Indicates width of rectangle to be moved. Stop column is optional. If omitted, it defaults to last column (determined by length of the line on a line by line basis). If both start and stop columns are included, they must be separated by a colon (:).

destcol Optional destination column for placement of the moved text block. Along with the pending line, this parameter defines the upper left corner of the destination. If omitted, text is moved to the end of the destination line (current pending line).

Q Optional quiet flag. Enter the Quiet option to suppress a listing of the text moved.

Remarks

The text from the source area overwrites the text at the destination. The source area is replaced with blanks.

The BM command should be executed from the line where you wish the first line of the moved text to overwrite current text.

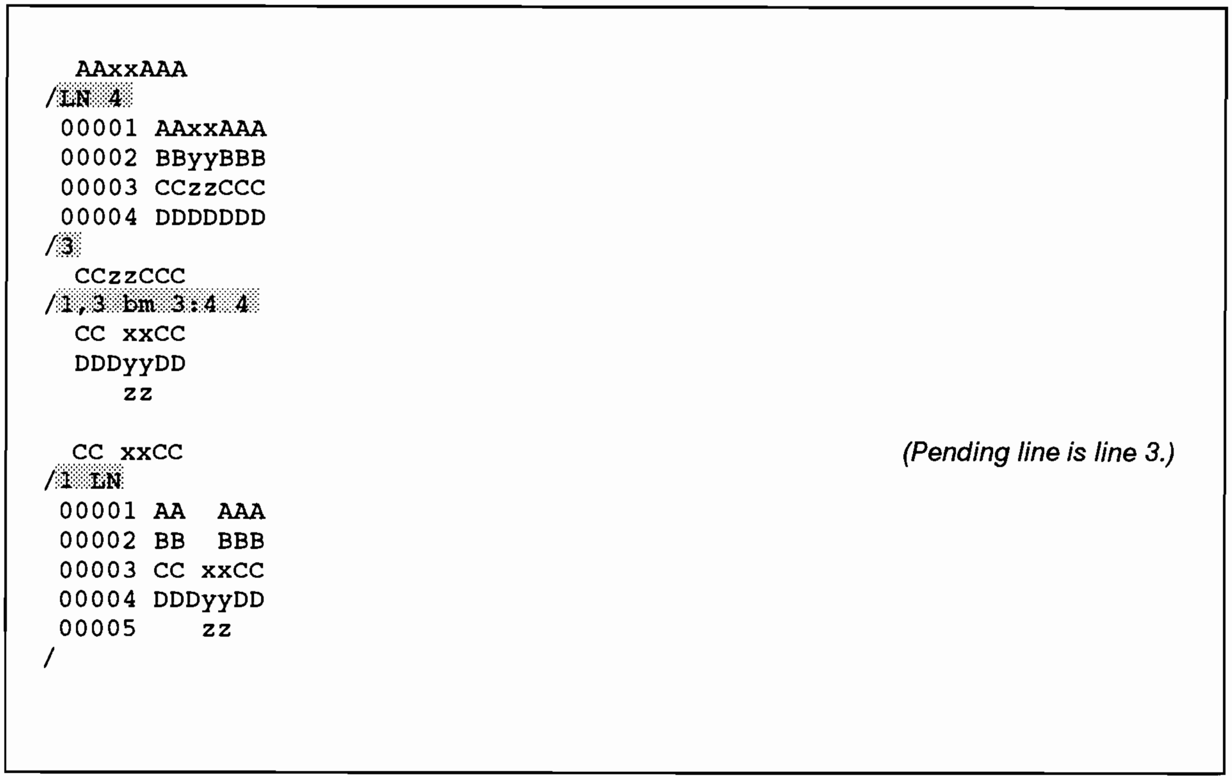
If at the end of the file, this command appends the moved text.

The Undo (UN) command reverses the BM command.

Refer also to the BC (Block Copy) command.

Examples

Figure 4-6 illustrates the use of the BM command to move columns three and four of lines one through three to line three, columns four and five. Note the use of the LN (List Numbered Lines) command to list the lines before and after execution of the BM command.



```
  AAxxAAA
/LN 4
 00001 AAxxAAA
 00002 BByyBBB
 00003 CCzzCCC
 00004 DDDDDDD
/3
  CCzzCCC
/1,3 bm 3:4 4
  CC  xxCC
  DDDyyDD
     zz

  CC  xxCC                                          (Pending line is line 3.)
/1 LN
 00001 AA  AAA
 00002 BB  BBB
 00003 CC  xxCC
 00004 DDDyyDD
 00005     zz
/
```

Figure 4-6. BM (Block Move) Command Example

C (Edit Pending Line & Advance Line)

Uses

Use in line character mode to edit the pending line and then go to the next line.

Command Syntax

[linespec 1] C [line character edits]

linespec 1 Optional line specification. Use to edit a specific line. If omitted, the command edits the current pending line. (See the section on line specifications in Chapter 3 for a complete list of possible line specifications.)

line character edits This command enables the use of line character edits, including text entry and use of control (CTRL) keys. The line character edit options are briefly described following. For more information, refer to the Line Mode Character Editing section in Chapter 3.

CTRL-B Break the line at this position. Move remaining text to next line.

CTRL-C Delete characters.

CTRL-R Replace characters.

CTRL-S Insert character.

CTRL-P Same function as CTRL-S. (Frees CTRL-S to be used with Xoff protocols.)

CTRL-T Truncate line.

CTRL-X Extend line, adding characters to the end of the line.

CTRL- A non-printing escape character. It is the same as "\" but it does not print. It cannot be redefined.

/ Current prompt (default = /) saves corresponding character.

<text> Replacement text to be entered on the pending line.

Remarks

The Undo (UN) command reverses the C command.

If the first character of the line edit text is "O", it must be preceded by the escape character (default = \) so EDIT does not interpret the entry as a CO (Copy) command.

Examples

Figure 4-7 shows the use of the C command. For more examples, see the Line Mode Editing section in Chapter 3.

```
 Illustrating the breaking of a line into two.
/c/////////////<CTRL-B>
 New pending line is listed.  Go back 2 and list 2.
/-2 L2
 Illustrating
 the breaking of a line into two.
/
```

Figure 4-7. C (Edit Pending Line and Advance Line) Command Example

CO (Copy Lines)

Uses

- Use to copy a range of lines to be inserted below the current pending line.
- Use with the Kx command to mark and copy sections of text.

Command Default

. * CO

Command Syntax

[linespec 1] [linespec 2] CO [Q]

linespec 1
linespec 2 Range of lines to be copied. An ending range specification is optional. If ending range is omitted, only one line is copied. (See the section on line specifications in Chapter 3 for a complete list of possible line specifications.)

Q Optional. Enter the Q option to suppress display of lines copied.

Remarks

The Undo (UN) command reverses the CO command.

Examples

Figure 4-8 illustrates the use of the CO command to copy three lines of text. Note the use of the LN (List Numbered Lines) command to display the text before and after execution of the CO command.

```
/1 LN
 00001 GET 18
 00002 GO TO 20
 00003 PRINT 21
EOF

/1 3 co
 00001 GET 18
 00002 GO TO 20
 00003 PRINT 21

/1 LN
 00001 GET 18
 00002 GO TO 20
 00003 PRINT 21
 00004 GET 18
 00005 GO TO 20
 00006 PRINT 21
EOF
/
```

Figure 4-8. CO (Copy Lines) Command Example

D (Delete Lines)

Uses

- Use to delete lines up to, but not including the line in which the pattern specified occurs. If no pattern is located, or if the A (all) option is used, EDIT deletes to the end of the line range specified.
- Use with RE (regular expressions) ON to use metacharacters. (Use the *SE RE ON* command to turn on regular expressions; refer to Chapter 5 for more information on metacharacters.)
- Use with the V option to delete lines containing the pattern specified.
- Use with CF (case folding) OFF to differentiate between upper and lower case letters. (Use *SE CF OFF* command to turn off case folding.)
- Use with the horizontal search window (set WC option) to restrict the search to within the specified columns.

Command Default

.+1 $ D /last find pattern/

Command Syntax

[linespec 1] [linespec 2] D /pattern/ [A V Q N]

linespec 1
linespec 2 Optional line range. May be absolute line numbers or line specification characters, with or without offset. If the first range parameter is omitted, the delete begins at the current pending line and deletes that line whether or not a match to the pattern is found. The pattern search then starts on the next line. If second range parameter is omitted, pattern search continues to the end of the file. (See the section on line specifications in Chapter 3.)

/pattern/ Pattern to match. Use any matching punctuation marks (except commas and spaces) as delimiters. If no options are used, the second delimiter may be omitted. If no pattern is specified, search defaults to the last pattern specified for either the F, D, or B commands. Use the Show (SH) command to check the default.

A Optional all flag. Use to delete all lines (in the line range) that do NOT match the pattern. Displays a tilde (~) for all lines deleted. Displays the total number of lines matched (that is, saved), followed by CF (case folding) and RE (regular expressions) options, if they are ON. New pending line is the line below the ending range specified.

V Optional reverse flag. Use the reverse (V) to delete a line containing the pattern specified. When the A and V options are used together, all lines in the line range that match the pattern are deleted.

Q Optional. Enter the Q (Quiet) option to suppress display of lines deleted and lines matched.

N Optional. Enter the N (No window) option to allow a match to the pattern anywhere on a line. Otherwise, pattern is matched only within the current horizontal window columns. (Use *SH WC* for the current window column search setting.)

Remarks

The Undo (UN) command reverses the D command.

Examples

Figure 4-9 shows the use of the D command to delete the lines in the file up to the first line that contains the letter G, to delete all lines that do *not* contain the letter G (using a slash to suppress asking), and to delete *all* lines that contain the letter G. The LN (List Numbered) command displays the file prior to the delete.

```
/1 5 LN
 00001 START
 00002 PRINT 25
 00003 GO TO 20
 00004 PRINT 21
 00005 GET 18

/1$ d/g/
OK? y
~ START
~ PRINT 25
  00001 GO TO 20
/un

/1$ d/g/a/
~ START
~ PRINT 25
 00001 GO TO 20
~ PRINT 21
 00002 GET 18

   EOF          3 matches CF

/1$ d/g/av
OK? y
 00001 START
 00002 PRINT 25
~ GO TO 20
 00003 PRINT 21
~ GET 18

   EOF          2 matches CF
```

Figure 4-9. D (Delete Lines) Command Examples

EC (Exit and Create File)

Uses

Use to exit EDIT session and create a new file.

Command Syntax

EC <filedescriptor>

<filedescriptor> Enter the filename descriptor to be assigned to the new file. (Refer to the section in Chapter 3 on filename specifications.)

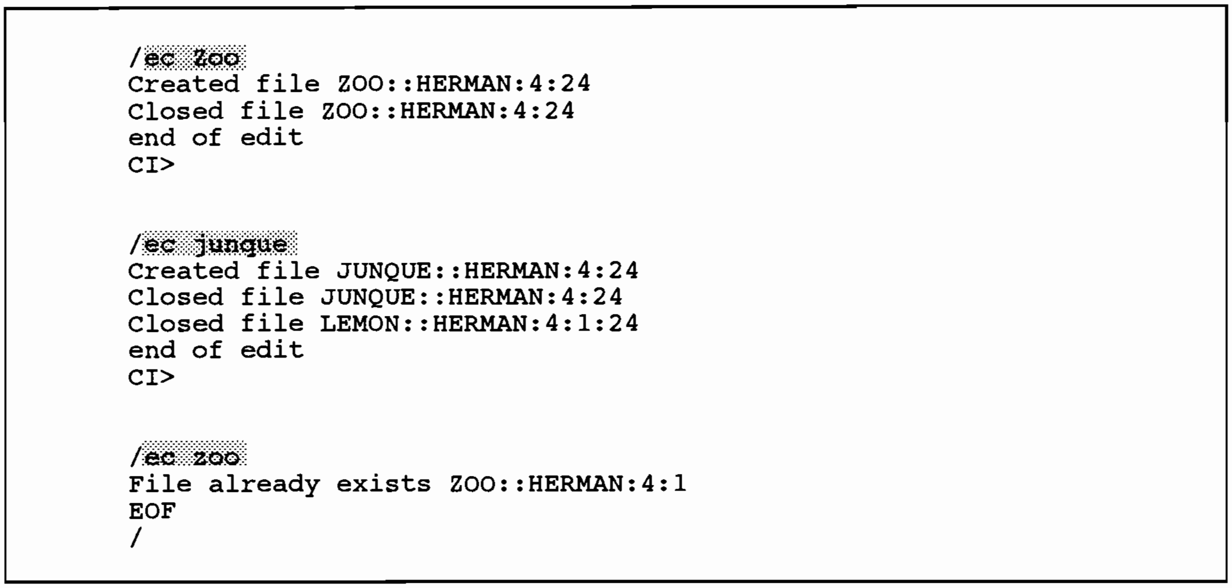
Remarks

If you do not specify a file size, EDIT determines the number of blocks required to hold your current work file.

See the ER command description for remarks concerning file errors.

Examples

Figure 4-10 illustrates the use of the EC command to create a new file through EDIT, to create a file by copying another file (the file LEMON was copied in this example to create JUNQUE), and an attempt to create a file using an existing filename.



```
/ec Zoo
Created file ZOO::HERMAN:4:24
Closed file ZOO::HERMAN:4:24
end of edit
CI>


/ec junque
Created file JUNQUE::HERMAN:4:24
Closed file JUNQUE::HERMAN:4:24
Closed file LEMON::HERMAN:4:1:24
end of edit
CI>


/ec zoo
File already exists ZOO::HERMAN:4:1
EOF
/
```

Figure 4-10. EC (Exit and Create File) Command Examples

ER (Exit and Replace File)

Uses

Use to exit EDIT session and replace a file with the current work file.

Command Default

ER <current source file>

Command Syntax

ER [<filedescriptor>]

<filedescriptor> Enter the filename descriptor of the file to be replaced. (Refer to section in Chapter 3 on filename specification.)

Remarks

Using the ER command overwrites a file until it reaches the END OF FILE or disk space becomes full. If EDIT overwrites an original file with a newly edited file that is larger than the original, then a file extent may be created. Should a file extend beyond the storage capacity of the disk, EDIT displays the following error message:

```
RAN OUT OF DISK SPACE <filename>
```

This message signals that you have corrupted the original file. Do not exit EDIT or your work file will be purged and the original lost. To recover:

1. Use the FCS command to close the source file.
2. Use the EDIT RU command to run CI or FMGR.
3. Free up disk space by deleting or moving data so that you can replace a file.
4. Exit from CI or FMGR.
5. Use ER to re-write the file.

Examples

Figure 4-11 illustrates the use of the ER command to replace a source file, to replace another file leaving the original source file (LEMON) unchanged, an unsuccessful attempt to replace a non-existent file, an unsuccessful attempt to replace a file that has a security code (note that security codes apply to FMGR cartridges only), and replacement of a file using a security code in the file descriptor.

```
/er
Closed file LEMON::HERMAN:4:1:4
end of edit
CI>

/er zoo
Opened file ZOO::HERMAN:4:1:18
Closed file ZOO::HERMAN:4:1:18
Closed file LEMON::HERMAN:4:1:4
end of edit
CI>

/er blimp
No such file BLIMP
/

/er edtmp
Opened file EDTMP::41:4
Illegal access EDTMP::41:4
Closed file EDTMP::41:4
/
/er edtmp:yl:41
Opened file EDTMP:YL:41:4
Closed file EDTMP:YL:41:4
end of edit
CI>
```

Figure 4-11. ER (Exit and Replace File) Command Examples

F (Find a Pattern)

Uses

- Use to search your work file for a line containing the pattern specified in the command string and make the matching line the new pending line. May specify a multiple search to locate more than one occurrence of the pattern.
- Use with set option RE (regular expressions) set to ON to search using the regular expression metacharacters.
- Use with the Return (RT) option set to ON (default) to return to pending line after an unsuccessful search or a multiple search.
- Use with RT set to OFF to remain at the line below the line range after an unsuccessful search or a multiple search.
- Use with set option CF (case folding set to OFF to differentiate between upper and lower case letters during search.
- Use with the horizontal search window (set WC option) to restrict the search to specified columns.

Command Default

.+1 $ F /last find pattern/

Command Syntax

[linespec 1] [linespec 2] F /pattern/ [A V Q N]

linespec 1
linespec 2 Optional line range. A line range may be specified for this command using either absolute line numbers separated by a space or comma, or line specification characters. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.) If no range is specified, the search begins at line below the current pending line, and goes to the end of the file.

/pattern/ Pattern to be found. Use any matching punctuation marks (except spaces or commas) as delimiters. If no options are used with the command, the last delimiter may be omitted. The default pattern is the last one specified in a B, D, F, ’ ’, or ‘ ‘ command. Use *SH<B, D, or F>* command to check default pattern.

A Optional all flag. Enter to find and list all occurrences of the pattern. A message provides the total number of matching lines found and the options affecting the search (CF and/or RE).

V Optional reverse flag. Enter to find line(s) that do NOT contain the pattern specified.

Q Optional quiet flag. Enter the Q (Quiet) option with the A option to indicate the total number of matches located, without listing the matched lines.

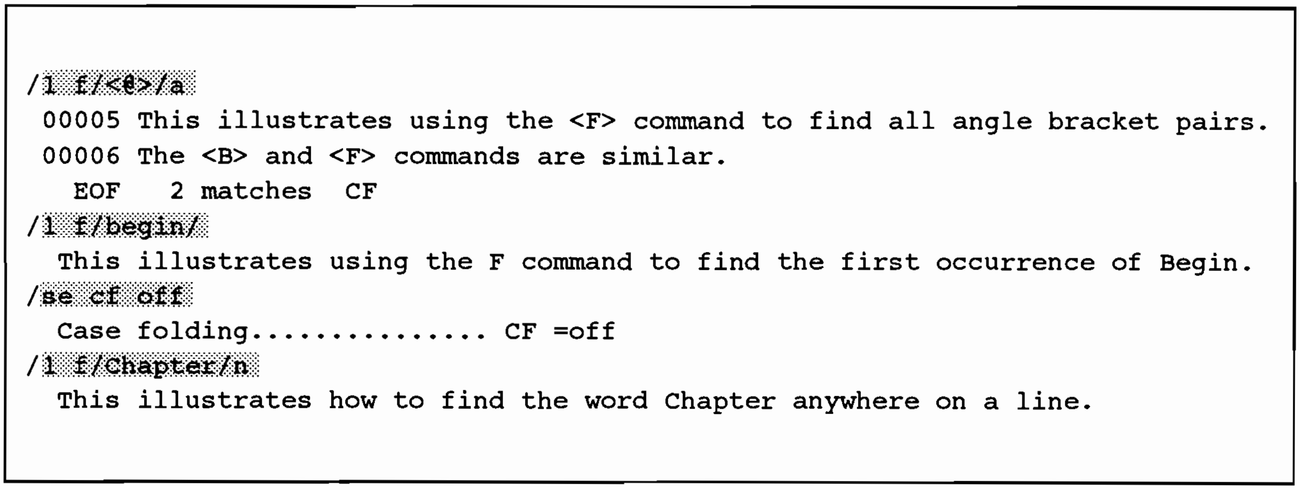
N Optional. Enter the N (No Window) option to disregard the horizontal window columns setting and allow a match anywhere on a line. (To view setting for window columns, use *SH WC* command.)

Remarks

The F command is similar to the B command except that the start search default is the line after the pending line, whereas the start search default for the B command is line 1.

Examples

Figure 4-12 illustrates the use of the F command to find all occurrences of angle bracket pairs, to find the first occurrence of the string *begin*, and to find the string *Chapter* with the Case Folding option off to differentiate between upper- and lowercase letters.



```
/1 f/<@>/a
 00005 This illustrates using the <F> command to find all angle bracket pairs.
 00006 The <B> and <F> commands are similar.
   EOF   2 matches  CF
/1 f/begin/
  This illustrates using the F command to find the first occurrence of Begin.
/se cf off
  Case folding.............. CF =off
/1 f/Chapter/n
  This illustrates how to find the word Chapter anywhere on a line.
```

Figure 4-12. F (Find a Pattern) Command Examples

FCL (File Close, List)

Uses

- Use to close the list file opened or created with the L or K command. Use of this command allows access to a list file without having to leave the editing session.

- Use to close a list file listed to a printer. This releases (unlocks) the printer for use by other users.

Command Syntax

FCL

Remarks

If the list file is already closed, no message is displayed.

When a list file is closed, EDIT performs a page ejection for printers, writes an EOF (End Of File) mark for tape drives, and displays two blank lines for terminals.

If list file is still open when EDIT is exited, EDIT automatically closes the file.

Examples

Figure 4-13 illustrates the use of the FCL command to close a list file after it is created using the L command.

```
/l 200 wow::41
Created file WOW::41:4
Posted file WOW

/fcl
Closed file WOW::41:4
```

Figure 4-13. FCL (File Close, List) Command Example

FCS (File Close, Source)

Uses

- Use to close the current source file so that it can be accessed by other programs.
- Use to close the current source file without changes, allowing you to continue to access the work file to carry out tasks such as disk packing.
- Use to prevent accidentally overwriting with the ER or WR commands while working with a file when you wish no further changes to the source file.

Command Syntax

FCS

Remarks

Note that this command separates the work file from the source file, so that there is no default filename for the ER and WR commands. If these commands are used, a filename must be specified. If the filename for the source file closed is specified, the source file is re-opened.

The default source filename continues to be displayed and is available, even if the source file has been closed using the FCS command.

Examples

Figure 4-14 illustrates the use of the FCS command to close the source file, LEMON. The ?? (Display Source File Status) command is used to display the status of the file LEMON.

```
/fcs
Closed file LEMON::HERMAN:4:1:8
EOF
/??
  EDIT on LEMON::HERMAN:4:1:8, closed
/
```

Figure 4-14. FCS (File Close, Source) Command Example

FI (File Input)

Uses

Use to replace the current EDIT work file with another file, allowing you to edit a new file without aborting or exiting the editor.

Command Syntax

FI <filedescriptor> [/]

<filedescriptor> Enter the filename descriptor to be assigned to the new file. (Refer to section in Chapter 1 on filename specification.)

/ Enter the current prompt character (default = /) to suppress the OK? prompt that is displayed if the current file (that is, the file that will be closed as a result of the FI command) has been modified since the last save.

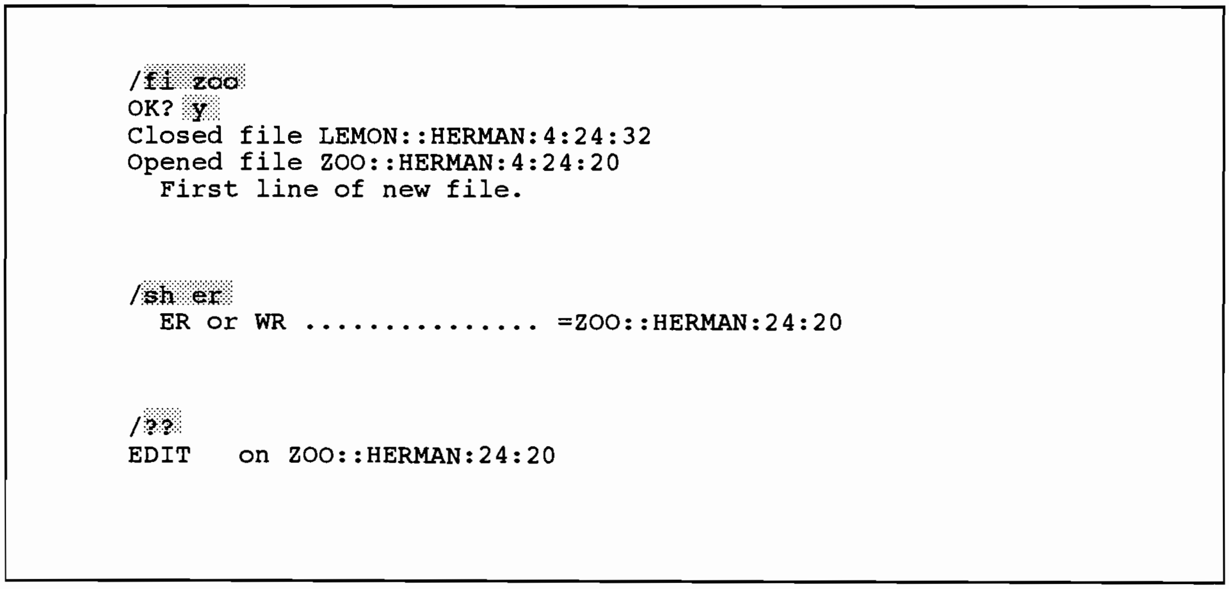
Remarks

FI does *not* update the current EDIT source file. If previous edits are to be saved, be sure to use the WR or WC command prior to using FI. The OK? prompt is displayed if changes have been made to the current work file.

The FI command sets the default filename.

Examples

Figure 4-15 illustrates the use of the FI command to input the file ZOO (after making modifications to the original source file, LEMON), the SH (Show) command to check the default source file, and the ?? (Display Source File Status) command to find out what is currently being edited.



```
/fi zoo
OK? y
Closed file LEMON::HERMAN:4:24:32
Opened file ZOO::HERMAN:4:24:20
  First line of new file.


/sh er
  ER or WR ............... =ZOO::HERMAN:24:20


/??
EDIT   on ZOO::HERMAN:24:20
```

Figure 4-15. FI (File Input) Command Example

FL (Fill)

Uses

The Fill command (FL) is used to fill text to the margins set with the SE FL command, and displayed with the SH FL command. The <first> column is the left fill margin, the <last> column is the right fill margin, and the <end> column is the last column from which to gather text. Over the range of lines specified, this command attempts to fill each line between the fill margins. Lines that are too long will have their ending words moved to the next line. Lines that are too short will be made longer, if possible, by moving up words from the following lines. Paragraph breaks are indicated by lines that are blank between the <first> column and the <end> column inclusive. The first and last line of the line specification are also treated as a paragraph break.

The first word of a paragraph may be indented. Setting fill indentation on (see the SH IN command) keeps the Fill command from moving the first word of a paragraph. It starts the fill after the first blank following the first non–blank character in the paragraph. Indentation may be either positive or negative.

The Fill command will put one blank between words except in the case where sentence-ending punctuation ' ! ?) : . ' is followed by a capital letter, in which case two blanks are inserted. The punctuation and capital letter will be 'seen' even if hidden by single or double quotes. However, the Fill command will not put in two blanks where only one is currently in the line; for example, 'Dr. Foo' will not be changed.

Fill is designed to allow filling of text in boxes. The boxes may be standard character set boxes, or alternate character set boxes. In order to do this, Fill counts escape and control sequences and attempts to use column numbers as they appear on the screen. In the case of line-drawing boxes, the ON and OFF control should be immediately before and after each edge; otherwise, the fill command may assume that they are part of the text to be filled. Further, extra control characters may confuse the Fill command.

Command Syntax

[.] [*] FL [Q] [R] [I]

Options:

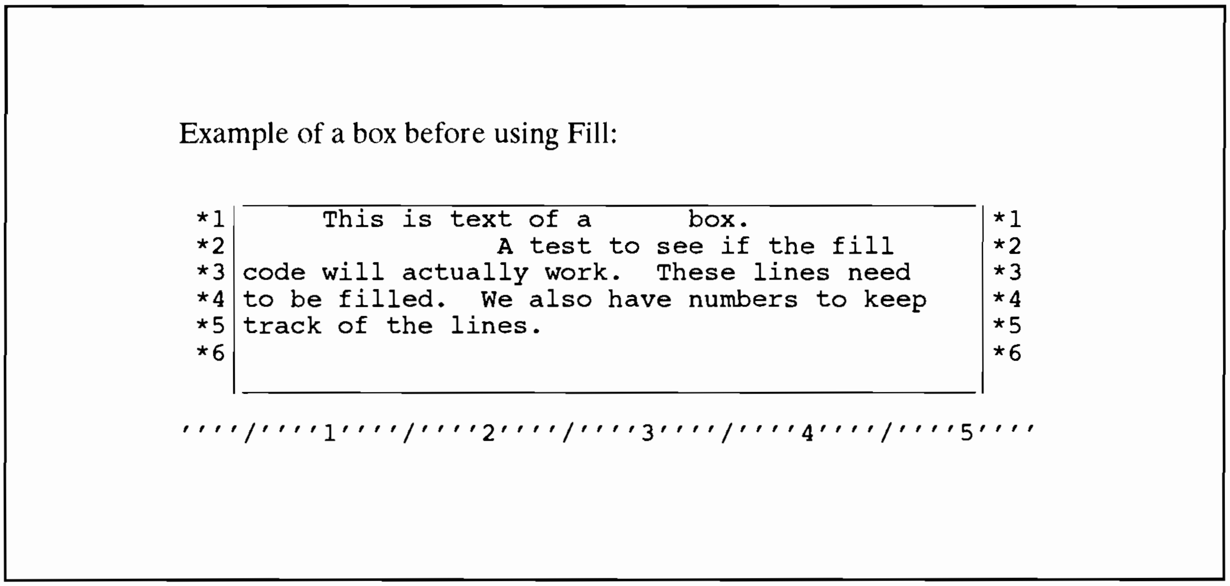
Q	Quiet operation (don't list changed lines).
R	Remove indentation (overrides current SE IN value)
I	Leave indentation (overrides current SE IN value).

Remarks

If more lines are needed in a paragraph, they are created by using the break line prior to the <start> column and after the <end> column.

If lines are left over after the fill, they are purged only if they are totally blank.

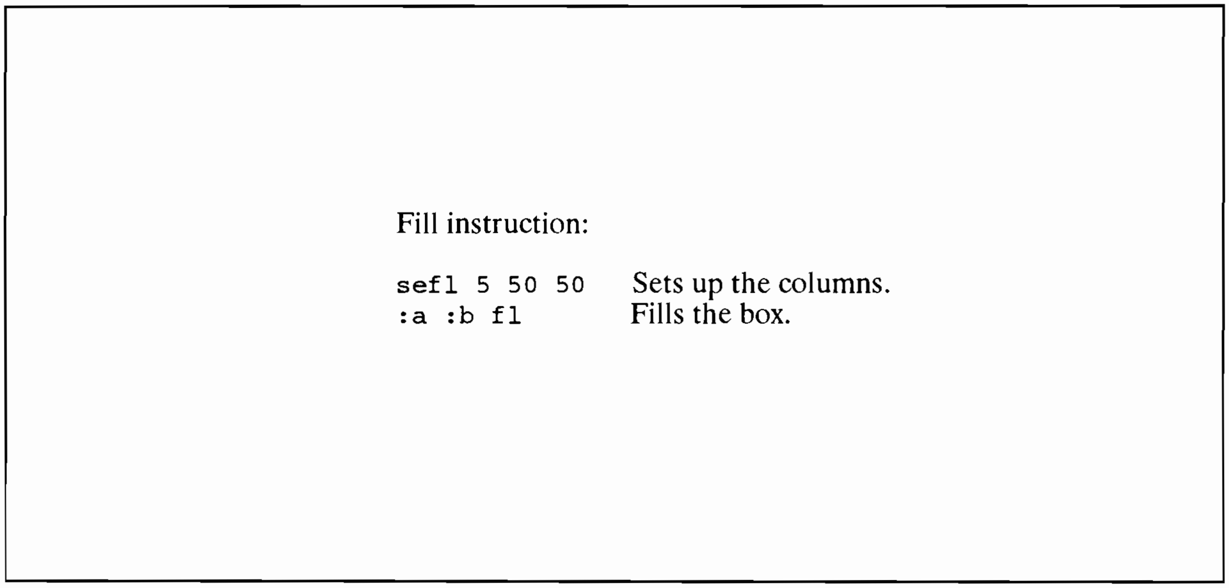
Examples



Example of a box before using Fill:

```
*1|     This is text of a      box.                |*1
*2|                A test to see if the fill       |*2
*3|code will actually work.  These lines need      |*3
*4|to be filled.  We also have numbers to keep     |*4
*5|track of the lines.                             |*5
*6|                                                |*6
  |________________________________________________|
''''/''''1''''/''''2''''/''''3''''/''''4''''/''''5''''
```

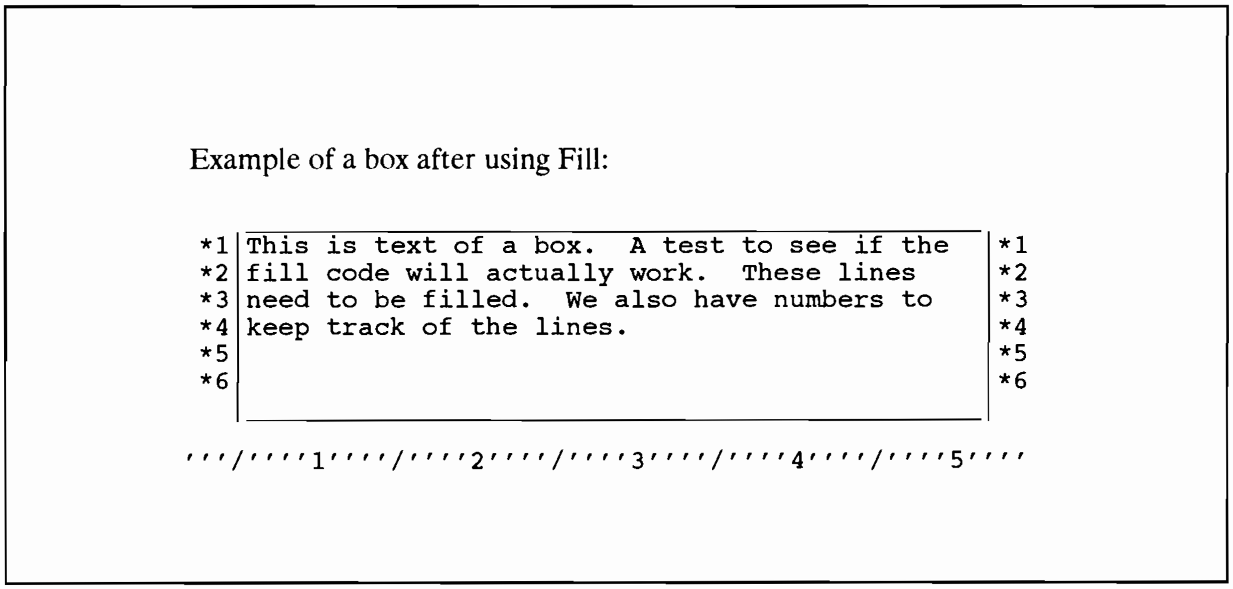
Figure 4-16. Box Before Using Fill



Fill instruction:

```
sefl 5 50 50     Sets up the columns.
:a :b fl         Fills the box.
```

Figure 4-17. Example of Fill Instruction



Example of a box after using Fill:

```
  -----------------------------------------------
*1|This is text of a box.  A test to see if the  |*1
*2|fill code will actually work.  These lines    |*2
*3|need to be filled.  We also have numbers to   |*3
*4|keep track of the lines.                      |*4
*5|                                              |*5
*6|                                              |*6
  |-----------------------------------------------|

'''/''''1''''/''''2''''/''''3''''/''''4''''/''''5''''
```

Figure 4-18. Box After Using Fill

G (Character Exchange on Pending Line)

Uses

- Use to search for a pattern and exchange it with a new pattern within the line range specified in the command string.
- Use with RE (Regular Expressions) set to ON to use the metacharacters for pattern matching. (Refer to Chapter 5 for more information on Regular Expressions.)

Command Default

. * G <last exchange and substitute>

Command Syntax

[linespec 1] [linespec 2] G /pattern/substitute/ [N R S]

linespec 1
linespec 2 Optional. A line range may be specified for this command using either absolute line numbers separated by a space or comma, or line specification characters. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.) If no range is specified, the exchange affects only the pending line.

/ Delimiter for the pattern and its substitute; use matching punctuation marks with the exception of commas and spaces. All three delimiters must be used when any option or OK? prompt suppression is used. If no options are used, the last delimiter may be omitted.

The default pattern and substitute for exchange commands may be used by omitting the pattern, delimiters, and substitute. To view the current pattern and substitute defaults, use the SH command for any of the exchange commands (G, U, X, and Y). To default the pattern and substitute but specify an option, enter *G,,<option>*. For more information, refer to the section on pattern defaults in Chapter 3.

pattern Pattern to be located. (See previous section in Chapter 3 on pattern specifications.) If Regular Expressions is set to ON, the metacharacters can be used for unique pattern matches (refer to Chapter 5 for more on Regular Expressions). If the pattern is omitted, EDIT uses the default pattern for Finds.

substitute Pattern to replace the search pattern specified first in the command string. Substitute pattern can be longer or shorter or shorter than the search pattern. EDIT inserts or deletes characters as necessary.

N Optional no window flag. Enter the N option to disregard the horizontal window columns setting and allow a match anywhere on a line. (To see setting for window columns, use *SH WC* command.)

R Optional remove flag. The R option removes empty (zero length) lines after an exchange occurs.

S Optional single exchange flag. The S option allows, at most, one exchange per line.

Remarks

The Undo (UN) command reverses the G command. The G command is similar to the X command, except that the X command provides a count of the lines affected by the exchange; the G command does not.

Examples

Figure 4-19 illustrates the use of the G command to exchange the number 2 with the string *tu*, to exchange the number 8 with the string *ate*, and to exchange the string *it* with the string followed by a space. The SE (Set) command is used to set the search window (WC) to columns one through eighteen and the G command is then used to exchange a single occurrence of the letter W with the letter C. Finally, the G command is used to exchange the letter X with the slash (/). Note that the pattern separator used is the exclamation point.

```
  Gra2ity
/g/2/tu/
  Gratuity
/
  I8itall
/g/8/ ate /
  I ate itall
/g/it/it /
  I ate it all
/
  Wombat    Wombat     Womcat
/se wc 1 18
/g/w/c/s
  Combat    Wombat     Womcat
/
 (A+B)X(C+D)=2000
/G !X!/!
 (A+B)/(C+D)=2000
/
```

Figure 4-19. G (Character Exchange on Pending Line) Command Examples

H (Help)

Uses

Use to display online quick reference information on EDIT commands, abbreviations, rules for pattern specifications, special control characters for pending line edits, metacharacters used for regular expressions, line specifications, and to list all help messages.

Command Syntax

H [command]   or   ?[command]

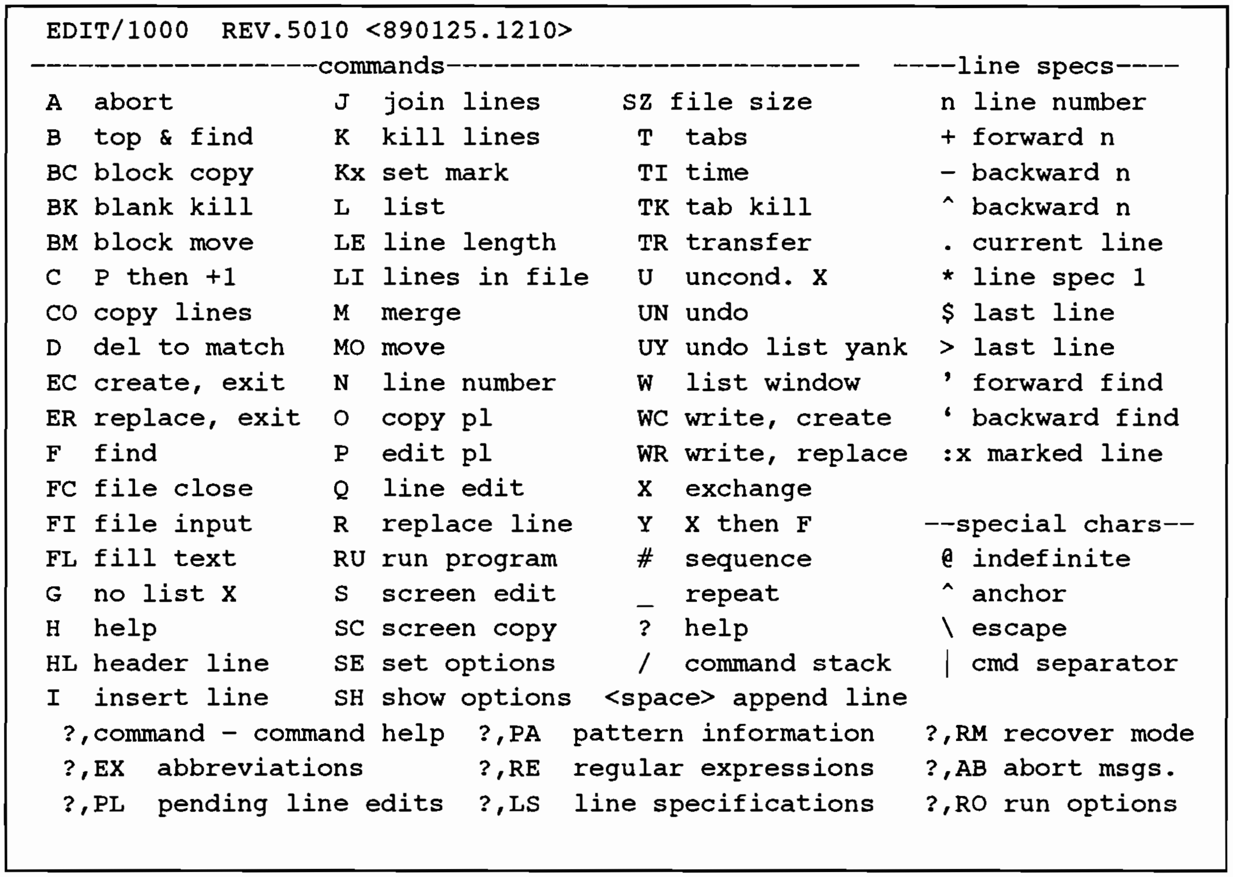
command Information on any EDIT command can be displayed by entering the command mnemonic after the H command.

?command The question mark (?) may be used in place of the H mnemonic. The command mnemonic for which help is desired is entered as above.

Remarks

The MORE feature permits help information to be displayed one screen at a time. Pressing the letter "a" aborts the listing, pressing the return key displays the rest of the current help message without pausing. Pressing any other key causes the next screen to be displayed.

Entry of the H or ? command by itself displays the online reference menu. Figure 4-20 shows the help menu.



```
 EDIT/1000  REV.5010 <890125.1210>
------------------commands--------------------------  ----line specs----
 A  abort          J  join lines     SZ file size        n line number
 B  top & find     K  kill lines      T  tabs            + forward n
 BC block copy     Kx set mark        TI time            - backward n
 BK blank kill     L  list            TK tab kill        ^ backward n
 BM block move     LE line length     TR transfer        . current line
 C  P then +1      LI lines in file   U  uncond. X       * line spec 1
 CO copy lines     M  merge           UN undo            $ last line
 D  del to match   MO move            UY undo list yank  > last line
 EC create, exit   N  line number     W  list window     ' forward find
 ER replace, exit  O  copy pl         WC write, create   ` backward find
 F  find           P  edit pl         WR write, replace  :x marked line
 FC file close     Q  line edit       X  exchange
 FI file input     R  replace line    Y  X then F       --special chars--
 FL fill text      RU run program     #  sequence        @ indefinite
 G  no list X      S  screen edit     _  repeat          ^ anchor
 H  help           SC screen copy     ?  help            \ escape
 HL header line    SE set options     /  command stack   | cmd separator
 I  insert line    SH show options  <space> append line
  ?,command - command help  ?,PA  pattern information   ?,RM recover mode
  ?,EX  abbreviations       ?,RE  regular expressions   ?,AB abort msgs.
  ?,PL  pending line edits  ?,LS  line specifications   ?,RO run options
```

Figure 4-20. EDIT Online Quick Reference Summary Menu

For more information on EDIT/1000's online quick reference, refer to the applicable sections in Chapters 1 and 3.

If the H mnemonic is used, enter a space or comma between the H and the command for which reference is being requested. (For example: use *H,CO* or *H CO*.) If the ? is used, a space or comma is not necessary.

The ?? command displays information about the current EDIT session.

Examples

Figure 4-21 illustrates the use of the H or ? command to display help for the D (Delete) command, the HL (Header Line) command, and the ? (Help) command. The final example is actually the ?? (Display Source File Status) command.

```
/h d
  [.+1][$] D/pattern/ [A][V][Q][N]
   Delete lines until pattern is found.  Deleted lines are flagged with a
   tilde (~), and lines that are not deleted are listed with line numbers.
   If the first line specification is not given, the pending line is
   deleted (regardless of pattern matching).

   To delete all lines that contain a matching pattern, use:

        1 $ D /pattern/ A V

   To delete all lines that do not contain a matching pattern, use:

        1 $ D /pattern/ A
/
/?hl
  HL
   Header line.  Print ruler to indicate line edit mode columns.
/
/? ?
  H [command] or ? [command]
   Help!  H or ? without specifying a command lists all commands.
   ?? gives the current Edit session and source filename.

/??
  EDIT on REFCHAPTER::HERMAN:4:152:53
/
```

Figure 4-21. H (Help) Command Examples

HL (Header Line)

Uses

Use to display a header with tick marks that indicate column positions.

Command Syntax

HL

Remarks

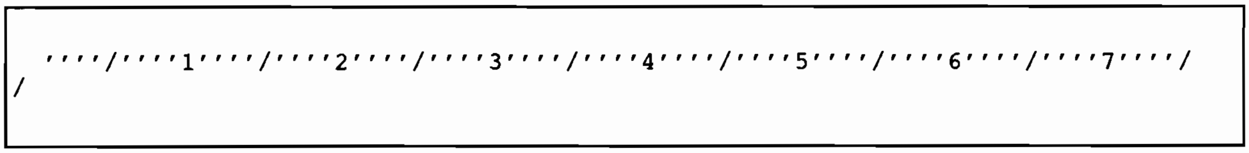
The header line lines up with the line mode listing; that is, two spaces are placed before the tick mark for column one.

If your terminal supports screen mode, the EDIT command prompt and the HL command are overwritten by the header line so that it is on the line just below the last line listed.

The header line will display to the full screen width, and is not limited to 80 characters.

Examples

When the HL command is entered, it displays the following header lines.



```
  ''''/''''1''''/''''2''''/''''3''''/''''4''''/''''5''''/''''6''''/''''7''''/
/
```

Figure 4-22. HL (Header Line) Command Example

HLP (Header Line Pending)

Uses

Use to extend the header line.

Command

HLP

Remarks

The header line is extended, or shortened, to be the same length as the pending line.

J (Join Lines)

Uses

Use in line mode to combine two lines. The current pending line (or the line specified) and the following line are combined into one line and displayed as the current pending line.

Command Default

. J

Command Syntax

[linespec 1] J

linespec 1 Optional line specification. Indicates which line is to be joined to the following line. Default joins the current pending line to the following line. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.)

Remarks

The joined line is the new pending line after execution of this command.

If the new joined line is longer than the line length, it is truncated to the line length without a message.

The UN (Undo) command can be used to reverse the J command.

Examples

Figure 4-24 illustrates the use of the J command to join three lines of text. The LN (List Numbered) command is used before the change to display the text.

```
/1 LN
 00001 Mission
 00002 Impossible
 00003 on today
EOF
/1
 Mission
/j
 MissionImpossible
/j
 MissionImpossibleon today
/
```

Figure 4-24. J (Join Lines) Command Example

I (Insert Line)

Uses

Use in line mode to insert a line of text above the current pending line. Inserts a line above any line specified in the command string.

Command Default

. I<line character edits>

Command Syntax

[linespec 1] I<line character edits>

linespec 1

Optional line specification. Line is inserted above the line specified. Default inserts a line above the current pending line. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.)

<line character edits>

Line mode character edit text to be inserted. If text is not entered, command inserts a zero length (blank) line. For more information, refer to the Line Character Editing section of Chapter 3.

Remarks

After the insert, EDIT displays the current pending line, which is the line you were positioned on when the insert was executed.

The I command allows you to insert text before line one of the work file.

The UN (Undo) command can be used to reverse the I command.

Examples

Figure 4-23 illustrates the use of the I command to insert a line of text above the pending line (line 2). The LN (List Numbered) command is used before and after the change to display the text.

```
/1 LN
 00001 Mission
 00002 on today
EOF
/
/2 i Impossible
/1 LN
 00001 Mission
 00002  Impossible
 00003 on today
EOF
/
```

Figure 4-23. I (Insert Line) Command Example

M (Merge)

Uses

Use from the line mode to copy all or part of another file into the current work file, inserting it after the current pending line.

Command Default

. M <filedescriptor>

Command Syntax

[linespec 1] M <filedescriptor> [start line] [# of lines]

or

[linespec 1] M <filedescriptor> [start line]:[stop line]

linespec 1 Optional line number or line specification to indicate where the named file is to be inserted in the work file. If omitted, the file is merged in after the pending line. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.)

<filedescriptor> File descriptor of file to be merged into current work file. (Refer to previous section in Chapter 3 on filename descriptors.)

start line Optional. If only part of the file is to be merged, specify the first line of text to be copied into the work file. The file can be larger than the 32,500 line file limit. If omitted, the default is the first line of the file named.

# of lines Optional. Indicates the number of lines to be copied from the file named into the current work file. This is NOT a line number. If omitted, default is the end of the file named.

start line:stop line Optional. Use instead of start line/number of lines to indicate beginning and ending line numbers for the merge. Start and stop line must be separated by a colon (:). The numbers entered may be larger than 32,500, but the total number of lines merged must be less than 32,500.

Remarks

The Undo (UN) command reverses the M command.

If the M command fails to open the file, EDIT remains at the pending line.

Once the file has been merged, the pending line is the last line read.

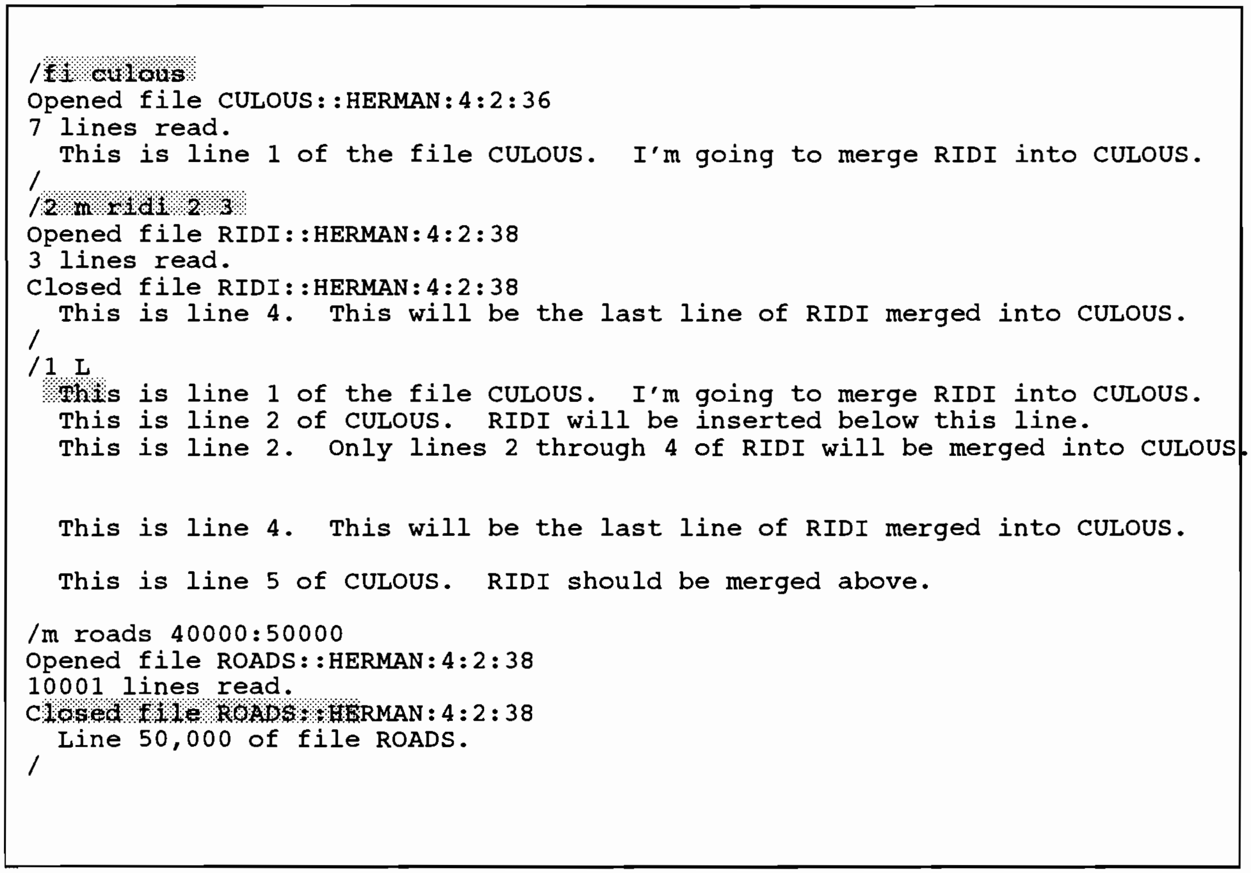
The merge file is opened non-exclusively; that is, another user can access the file while EDIT is reading it for the merge operation.

EDIT cannot merge the file it currently has exclusively opened as the source file. If you need to merge the source file, use the FCS (file close source) command to close it. Then the merge will be successful.

A space or comma is required between the M and the filename if the filename begins with an "O". This prevents EDIT from interpreting it as a Move (MO) command. In all other cases, a delimiter is optional.

Examples

Figure 4-30 illustrates the use of the M command to merge file RIDI into file CULOUS, and to merge a file using a line range.



```
/fi culous
Opened file CULOUS::HERMAN:4:2:36
7 lines read.
  This is line 1 of the file CULOUS.  I'm going to merge RIDI into CULOUS.
/
/2 m ridi 2 3
Opened file RIDI::HERMAN:4:2:38
3 lines read.
Closed file RIDI::HERMAN:4:2:38
  This is line 4.  This will be the last line of RIDI merged into CULOUS.
/
/1 L
  This is line 1 of the file CULOUS.  I'm going to merge RIDI into CULOUS.
  This is line 2 of CULOUS.  RIDI will be inserted below this line.
  This is line 2.  Only lines 2 through 4 of RIDI will be merged into CULOUS.


  This is line 4.  This will be the last line of RIDI merged into CULOUS.

  This is line 5 of CULOUS.  RIDI should be merged above.

/m roads 40000:50000
Opened file ROADS::HERMAN:4:2:38
10001 lines read.
Closed file ROADS::HERMAN:4:2:38
  Line 50,000 of file ROADS.
/
```

Figure 4-30. M (Merge) Command Example

MO (Move)

Uses

- Use from the line mode to move a number of lines to below the pending line. The lines moved are deleted from their previous location.
- Use in conjunction with the Kx (Mark Line) command to move blocks of text.

Command Default

.–1,* MO

Command Syntax

linespec 1 [linespec 2] MO [Q]

linespec 1 Required line number or line specification of the first line to be moved. If omitted, it defaults to the line above the pending line. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.)

linespec 2 Optional line number of the last line to be moved. If omitted, defaults to the line number specified as the first line, and only that line is moved.

Q Optional. Enter the Q (Quiet) option to suppress display of the lines moved.

Remarks

EDIT will not move a block of lines that includes the current pending line. For example, if the current pending line is 25, an attempt to move lines 10 through 50 results in an *Overlap* error message.

Lines moved are displayed unless the Q option is specified. If Q is specified, EDIT displays the new pending line only.

The Undo (UN) command reverses the MO command.

Examples

Figure 4-31 illustrates the use of the Kx and MO commands to mark and move lines eight through ten to below line 12. The LN (List Numbered) command is used to display the text prior to the change and the LU (List Unnumbered) command is used to display the text after the change.

```
/6 LN
 00006 This is line 6.  This is an example of how the Move command works.
 00007 I'm going to move the original lines 8-10 to below the original line 1.
 00008 The answer is:                                              (Original line 8)
 00009 "Nein, W."                                                  (Original line 9)
 00010 Signed, Karnak                                              (Original line 10)
 00011 This is the original line 11.
 00012 "Do you spell your name with a "V", Mrs. Vagner?"           (Original line 12)
EOF
/

/8 ka
a00008 The answer is:
/10 kb
b00010 Signed, Karnak

/12
  "Do you spell your name with a "V", Mrs. Vagner?"              (Original line 12)
/

/:a:b mo
  The answer is:                                                 (Original line 8)
  "Nein, W."                                                     (Original line 9)
  Signed, Karnak                                                 (Original line 10)
/

/7 LU
  I'm going to move the original lines 8-10 to below the original line 12.
  This is the original line 11.
  "Do you spell your name with a "V", Mrs. Vagner?"              (Original line 12)
  The answer is:                                                 (Original line 8)
  "Nein, W."                                                     (Original line 9)
  Signed, Karnak                                                 (Original line 10)

EOF
/
```

Figure 4-31. MO (Move) Command Example

N (Display Line Number)

Uses

Use to display the line number of the current pending line.

Command Syntax

N

Examples

```
/N
  316
```

Figure 4-32. N (Display Line Number) Command Example

n (Specific Line Number)

Uses

Enter a line number or linespec to display a specific line and make it the current pending line.

Command Syntax

linespec

linespec The linespec number (n) entered is a string of numbers or some other line specification character. (Refer to Chapter 3 for more information on line specifications.)

Remarks

When specifying a line number, remember that lines are re-numbered each time lines are added, deleted, or moved within the EDIT session.

If two line specifications are entered, the line range is listed.

Examples

Figure 4-33 illustrates the use of the LN (List Numbered) command to list the file before the *n* command is used several times to list specific lines.

```
  Original pending line.
/1 LN5
 00001 Line 1.
 00002 Line 2.
 00003 Line 3.
 00004 Line 4.
 00005 Last line.  Line 5.
/4
  Line 4.
/-3
  Line 1.
/$
  Last line.  Line 5.
/^3
  Line 2.
/2 4                                          (Go to line 2, then list to line 4.)
 00002 Line 2.
 00003 Line 3.
 00004 Line 4.
```

Figure 4-33. n (Specific Line Number) Command Example

O (Duplicate Pending Line & Edit)

Uses

Use in line mode to copy the pending line and allow character editing of the duplicated line. The duplicated line is then the new pending line.

Command Default

.O

Command Syntax

[linespec 1] O [line character edits]

linespec 1 Optional line specification. Use to go to and duplicate a line other than the current pending line. If omitted, the command edits and copies the current pending line. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.)

line character edits This command enables use of the line character edits, including text entry, and use of control (CTRL) keys. The line character edit options are briefly described following. For more information, refer to the Line Mode Character Editing section in Chapter 3. If no edits are entered, the duplicated line is unchanged.

CTRL-B	Break line at cursor; move remaining text to next line.
CTRL-C	Delete characters.
CTRL-R	Replace characters.
CTRL-S	Insert character.
CTRL-P	Same function as CTRL-S. (Frees CTRL-S to be used with Xoff protocols.)
CTRL-T	Truncate line.
CTRL-X	Extend line, adding characters to the end of the line.
CTRL-	A non-printing escape character. It is the same as "\\" but it does not print. It cannot be redefined.
/	Current prompt (default = /) preserves corresponding character.
<text>	Replacement text to be entered on the new pending line.

Remarks

The Undo (UN) command reverses the O command.

The duplicated line is displayed as the new pending line after completion of the command.

Examples

For examples, see the section on Line Mode Editing in Chapter 3.

P (Pending Line Edit)

Uses

Use in line mode to edit the current pending line and display the edited line as the new pending line.

Command Default

. P

Command Syntax

[linespec 1] P [line character edits]

linespec 1 Optional line specification. Indicates the line to be edited. If omitted, the command edits the current pending line. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.)

line character edits This command enables use of the line character edits, including text entry, and use of control (CTRL) keys. The line character edit options are briefly described following. For more information, refer to the Line Mode Character Editing section in Chapter 3.

CTRL-B	Break line at cursor; move remaining text to next line.
CTRL-C	Delete characters.
CTRL-R	Replace characters.
CTRL-S	Insert character.
CTRL-P	Same function as CTRL-S. (Frees CTRL-S to be used with Xoff protocols.)
CTRL-T	Truncate line.
CTRL-X	Extend line, adding characters to the end of the line.
CTRL-	A non-printing escape character. It is the same as "\" but it does not print. It cannot be redefined.
/	Current prompt (default = /) preserves corresponding character.

Remarks

The Undo (UN) command reverses the P command.

After execution of the command, EDIT re-displays the edited line as the current pending line.

Examples

For examples, see the section on Line Mode Editing in Chapter 3.

Q (Single Line Screen Mode Edit)

Uses

Use from line mode to allow the use of the local terminal keys to edit the current pending line. Displays the pending line with the screen cursor positioned at that line. Referred to as the *single line screen mode* command, because the cursor arrow keys, delete and insert keys can be used to edit the pending line. The edited line is placed in the work file when the carriage return is pressed.

Command Default

.Q

Command Syntax

[linespec 1] Q

linespec 1 Optional line specification. Indicates the line to be edited. If omitted, the command edits the current pending line. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.)

Remarks

The Undo (UN) command can be used to reverse Q command edits.

If changes made to the line are to be disregarded by EDIT, position the cursor on a blank (zero length) line prior to pressing the carriage return. (On many terminals, the CTRL-HOME or SHIFT-HOME key moves the cursor to a blank line.)

When the Q command is used but changes are disregarded, EDIT displays and keeps the original line. If changes are made, EDIT does NOT re-display the line.

If line being edited is marked (Kx command), the mark is displayed in the right margin outside of the screen work area.

For more on screen mode editing, refer to Chapter 3.

Long Line Editing

If editing a line that is more than 78 characters long, EDIT displays two dots (..) in the right margin outside of the work area. This indicates that the line has been wrapped. For such lines, EDIT displays the following message:

```
Put cursor on first line when edit is complete.
```

The first line(s) of text is deleted unless the cursor is positioned on the first line of the wrapped line, prior to pressing the carriage return.

If editing a line with less than 78 characters and characters are added to make the line longer than 78 characters, the message is not displayed and you must manually insert the dots in the last two line positions. If the terminal supports margins, the terminal automatically advances the cursor a line when reaching the last line position of the original line. Use the cursor arrow keys to find the last two line positions and insert the dots. After editing the line, position cursor on the first line and press the carriage return.

R (Replace Pending Line with Text)

Uses

Use in line mode to replace the text of the current pending line with text entered with the *R* command.

Command Default

. R

Command Syntax

[linespec 1] R<text>

linespec 1 Optional line specification. Use to replace text on a specific line. If omitted, the command replaces text on the current pending line. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.)

<text> Replacement text to be entered on the pending line.

Remarks

The Undo (UN) command reverses the R command.

If the first character of the replacement text is "U", precede it with the escape character (default = \) to ensure that EDIT does not interpret the entry as a Run (RU) command.

RU (Run Program)

Uses

Use in line mode to run another program, such as another version of EDIT or the Command Interpreter (CI).

Command Syntax

RU <program><runstring>

<program>
<runstring>

Enter name of program followed by the proper runstring, if necessary.

Remarks

If the current source file is to be used by the program being run, it must first be restored to the disk file with the WR command and closed with the FCS command. It then can be used with another program including another version of EDIT. The filename must be specified for future WR or ER commands after FCS closed the file.

The runstring is case folded and blanks are replaced with commas.

Examples

Figure 4-34 illustrates the use of the RU command to run another version of the EDIT program from EDIT. The A (Abort) command is then used to terminate the second copy of EDIT. The RU command is then used to run a copy of CI from EDIT and the CI EX (Exit) command is used to terminate CI.

```
/ru,edit

ED53A : Use ? for help
FI,<filename> specifies file to edit.
EOF
/a
ED53A aborted by user
end of edit
  Resume EDIT on REFCHAPTER::HERMAN:4:111:41

/ru ci
Using stack file CI.STK::HERMAN
CI> ex
Finished
  Resume EDIT on REFCHAPTER::HERMAN:4:111:41
/
```

Figure 4-34. RU (Run Program) Command Examples

S (Screen Mode Edit)

Uses

Use to edit file in the screen mode. Command displays several lines of text at a time and then allows the use of terminal edit keys. Screens displayed are bracketed by a beginning line and an ending line that provide the beginning and ending line numbers of the portion of the file displayed. (See example of screen display, following.) In addition, the beginning line displays the command used to exit the screen mode and the ending line displays the file descriptor of the file being edited. If the file descriptor is too long to display in its entirety, EDIT displays a portion of the file descriptor followed by three dots (...).

Command Default

.−10 *+20 S

Command Syntax

[linespec 1] [linespec 2] S

linespec 1 Optional first line of screen. If omitted, default for first line of screen is 10 lines above the pending line. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.)

linespec 2 Optional line range that defines the last line of the screen. If omitted, default for last line displayed is 20 lines below the first line displayed.

Screen Mode Commands

Once in screen mode, you may use terminal keys and control (CTRL) keys to edit text. Although terminal keys vary between terminal models, they usually include cursor arrow keys, and insert and delete keys for characters and lines.

Sixteen control key combinations are available on all terminals that support screen mode: Q, U, P, R, F, S, T, X, C, O, K, A, Z, J, B, and D. The CTRL key and alphabetic key are pressed simultaneously and are followed by a carriage return. Many of the control commands save new edits when they are executed. To execute the command but ignore new edits, double the command. For example, CTRL-U quits screen mode, saving new edits; whereas CTRL-U CTRL-U quits screen mode and ignores any edits made since the last screen save.

To cancel a control key combination that you enter but do not wish to execute, press some other non-command key PRIOR to pressing the carriage return.

A brief description of the control keys follow. For more information on the screen mode, refer to the Remarks section for this command and to the section on Screen Mode Editing in Chapter 3 EDIT/1000 Operations.

CTRL-A Moves cursor to the first character on the line. Also resets margins set with local terminal keys back to the setting determined by the terminal.

CTRL-Z Moves cursor to the last character on the line. Also resets margins (see CTRL-A). If a line is longer than 78 characters, CTRL-Z moves the cursor to the last character of the line, whether that character is on the same line as the first character or five lines below.

ESC-4 CTRL-B Breaks line into two lines at the current cursor position. Blanks are inserted at the front of the second line so that it lines up with the first non-blank character of the first line. ESC-4 sets a new left margin and CTRL-B breaks the line at the new margin. Margins are then reset by the command.

CTRL-J Joins current line to following line. Also resets margins (see CTRL-A).

CTRL-C Allows execution of *one* line mode command. After carriage return, EDIT prompt is displayed (/ if ^C is used, and \ if ^C^C is used). Enter line mode command(s). When multiple commands are entered on the same line, separate the commands with a vertical bar (|). Press the carriage return. After command execution, EDIT returns to screen mode with cursor positioned at the current pending line.

Note

Do not use the cursor arrow keys to move the cursor off of the line provided for line mode entry after use of CTRL-C. When the carriage return is used, the line containing the command to be executed is deleted. If the cursor is on another line when the carriage return is pressed, a line of text is deleted and the command becomes part of your text.

CTRL-F Saves new screen edits and advances to next screen. Double to ignore new edits.

CTRL-P Saves new screen edits and goes back to previous screen. Double to ignore new edits. CTRL-P does not work on X.25 pad terminals.

CTRL-R Saves new screen edits and goes back to previous screen. Use for X.25 pad terminals.

CTRL-K Marks line indicated by the current cursor location. Following the carriage return, places a colon in column 79 of the screen and requires entry of an alphabetic character.

CTRL-O Copies the line indicated by current cursor location.

CTRL-S Saves new edits and starts next screen at line indicated by the current cursor position. Double to ignore new edits. Should not be used if your terminal uses Xon/Xoff handshake protocol.

CTRL-T Same as CTRL-S. Use for terminals with Xon/Xoff handshake protocol.

CTRL-X Same as CTRL-S, except a large screen is provided. Size of screen is determined by amount of terminal memory available. Check size available with the SH SL command.

CTRL-U Same as CTRL-Q. Use for terminals with Xon/Xoff handshake protocol.

CTRL-Q Saves screen edits and quits screen mode. Double to ignore new edits. Should not be used if your terminal uses Xon/Xoff handshake protocol.

Note

If your terminal follows Xon/Xoff handshake protocol, CTRL-Q and CTRL-S are not interpreted as EDIT commands. Use CTRL-U in place of CTRL-Q and CTRL-T instead of CTRL-S.

CTRL-D The control-D (CTRL-D) mode is a submode of screen mode editing that adds some line graphic editing commands in to screen mode. CTRL-D mode can be enabled by typing either CTRL-D plus carriage return while in screen mode or, by typing the SE CD (set control-D) command. In either case, display functions will be turned off. Once CTRL-D is enabled, it stays enabled until it is disabled with another SE CD command. When CTRL-D mode is enabled, screen mode back spaces are destructive and DEL or rub-outs echo \ followed by a carriage return linefeed. The edit display functions flag will be set to OFF when control-D mode is enabled, and should be left off while CTRL-D mode is being used.

To use the line drawing features of CTRL-D mode, your terminal must have the line drawing character set ROM installed and enabled (usually, it is enabled with the sequence "escape,right parenthesis,uppercase B"). If your terminal does not have the line drawing character set installed, you can still use the move and copy features of the CTRL-D mode. Printing can be done on any printer that supports the HP line drawing sets.

When you enter a CTRL-D during screen mode with CTRL-D submode enabled, edit responds by turning on an inverse video mark around the current cursor position. It also saves the screen location of this point in an internal list of points. Entering additional CTRL-Ds displays additional inverse video marks on the screen and adds a point to the points list. These points become the arguments to the CTRL-D mode commands.

Normally these points define a closed polygon around the area to be used by the CTRL-D command. When two consecutive points, having different row and column addresses, are used to define a line or an edge, a horizontal line or edge is defined first followed by a vertical line or edge to reach the second point. (Note that lines and edges must be horizontal or vertical.)

Whenever there is an inverse video point displayed, you are in CTRL-D mode. While in CTRL-D mode, characters are not echoed as they are entered. Rather, they are interpreted as CTRL-D mode commands (see the following list) when carriage return is pressed. If the entered characters do not form a valid CTRL-D command after pressing carriage return, the terminal bell will sound and the cursor is placed at the last point in the list.

Note that systems without the new serial drivers will echo characters due to an incompatibility in the MUX driver. To enter a command on these systems, type CTRL-C followed by a carriage return; edit will open a line on the screen and display the following prompt:

```
CTRL-D mode>
```

At this point, you should enter a CTRL-D mode command. The prompt and the command will be executed and erased when you press carriage return.

To erase the last point in the CTRL-D list, position the cursor to the display point and enter another CTRL-D. This will erase the inverse video mark and the point for the list. To remove all points, enter the CTRL-D command "Q".

CTRL-D mode commands are defined below. They are all single character commands sometimes followed by options. The characters inside the angle brackets are options and are defined later in the list.

A <B E P N>   Arrow.   Draw a path (see the "P" command for path definition) through the defined points, and put an arrow head at the end of the path. Arrow heads may not display correctly on most terminals; however, LaserJets with the line drawing character set will display them.

B <B E P N> Box. Draw a closed polygon through the defined points.

C Copy. The last point is saved as a locator point and then removed from the list. The area defined by the closed polygon through the remaining points is then copied so that the first marked point is on top of the locator point.

E Erase path. The path (see "P" command) through the points is erased. (This is the same as the P E command.)

L <B E P N> Lines. Take the point list as pairs of points and draw lines between the pairs

M Move. Same as copy, but erase the original area before copying.

P <B E P N> Path. Start at the first point in the list and draw a line through each consecutive point until the end of the list has been reached. As always, if two points are not aligned horizontally or vertically, draw the horizontal line first followed by a vertical line.

Q Quit. CTRL-D mode. Erase all marks and return to normal screen mode.

R Re-mark. Restore the marks of the previous CTRL-D mode command and reenter CTRL-D mode. If the previous command was copy or move, the locator points are not restored.

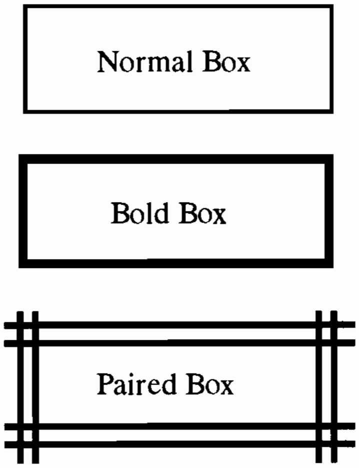
U Undo. Undo the last CTRL-D mode command.

Options:

N No-read. Normally the screen will be read before the CTRL-D mode command is executed. If there has not been any screen mode changes to the area to be affected by the CTRL-D mode command that is going to be executed, the screen read time can be saved by entering an "N" with the command.

Line Style Options: B=Bold, P=Paired, E=Erase.

Example Boxes:



Caution

If the beginning or ending screen bracket lines are deleted, EDIT displays the following message:

```
Stop/start line(s) not found.

O Saves original text written to screen
S Saves text just read from screen
B Saves both (inserts screen text before original text)
What should be saved?
```

Enter O to save the work file as originally written to the screen, S to replace the work file content originally written to the screen with the text that EDIT read in as it rolled it across the screen, or B to save both the original work file contents and insert the new text read in. If unsure of what should be saved, B is the safest action.

Remarks

The Undo (UN) command *cannot* be used to reverse the S command. You must exit the screen mode by using one of the applicable control key combinations.

While in screen mode, EDIT locks the terminal LU and disables interrupt processing from the terminal. If you are using EDIT on the system console, it is not locked and any system messages overwrite your text. While EDIT is reading the screen, it locks your terminal's keyboard, so that any keys that you strike are ignored.

Note that the HP 2621 terminal does not support keyboard locking, so be careful not to strike any keys while the screen is being written to or read.

Use *n S*, (where n is any line number or line specification character), from line mode to start a screen at a specific line.

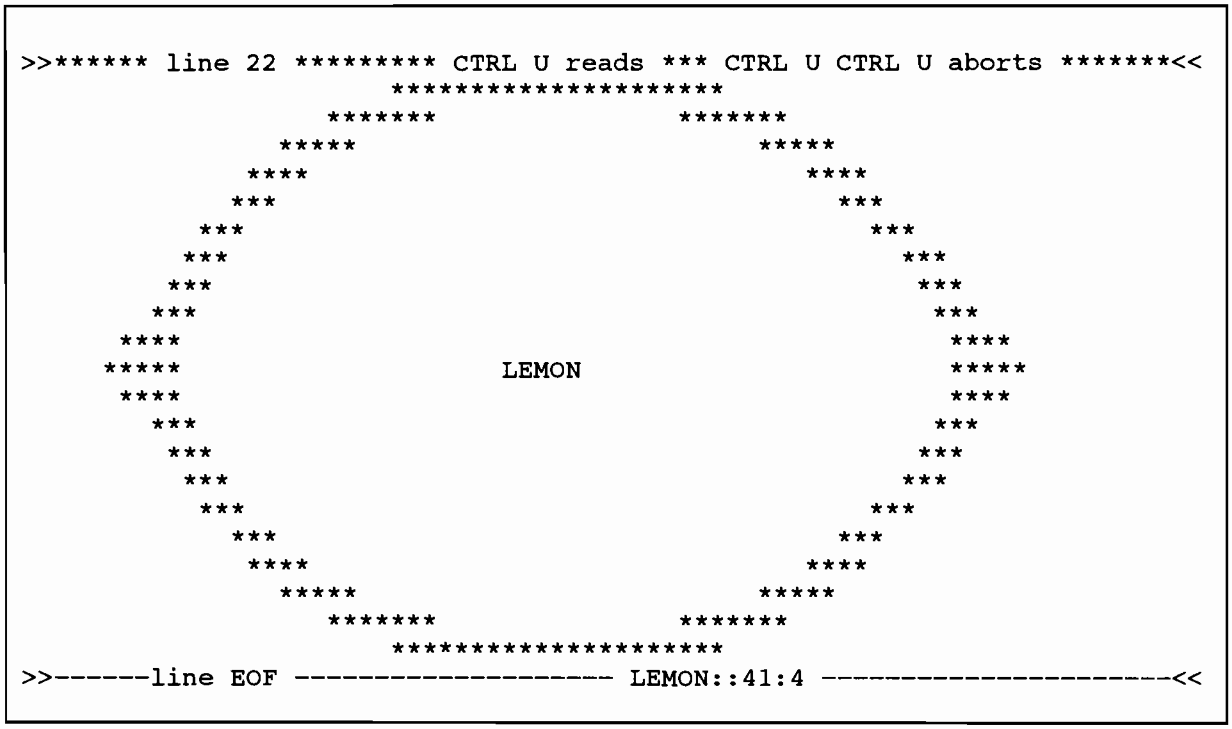
Screen Mode Defaults

To view screen mode defaults, enter *SH SD* (Show Screen Defaults). This command provides the defaults for how lines are displayed on the screen. The same information can be changed using the *SE SD <x> <y> <z>* (Set Screen Defaults) command, where x = number of lines displayed above the pending line, y = number of lines displayed below the pending line, and z = number of overlap lines.

When moving from one screen to another, EDIT overlaps screens, including two lines on the second screen from the previous screen. For example, the first screen displays lines 701 to 721. If CTRL-F is used, the second screen displays lines 720 to 740, and CTRL-P displays lines 682 to 702. To redefine the default screen size and overlap, use the SE (Set) command to change the SD (Screen Default) option.

Examples

Figure 4-35 illustrates the result of the command /22 s, or go to line 22 and start screen mode.



```
>>****** line 22 ********* CTRL U reads *** CTRL U CTRL U aborts *******<<
                       *********************
                   *******               *******
                *****                         *****
              ****                               ****
             ***                                   ***
           ***                                       ***
          ***                                          ***
         ***                                            ***
        ***                                              ***
      ****                                                ****
     *****                    LEMON                       *****
      ****                                                ****
        ***                                              ***
         ***                                            ***
          ***                                          ***
           ***                                       ***
             ***                                   ***
              ****                               ****
                *****                         *****
                   *******               *******
                       *********************
>>------line EOF -------------------- LEMON::41:4 ----------------------<<
```

Figure 4-35. S (Screen Mode Edit) Command Example

SC (Screen Copy)

Uses

Use in line mode to copy everything in the terminal's screen memory and insert it after the current pending line.

Command Syntax

SC

Remarks

The Undo (UN) command reverses the SC command.

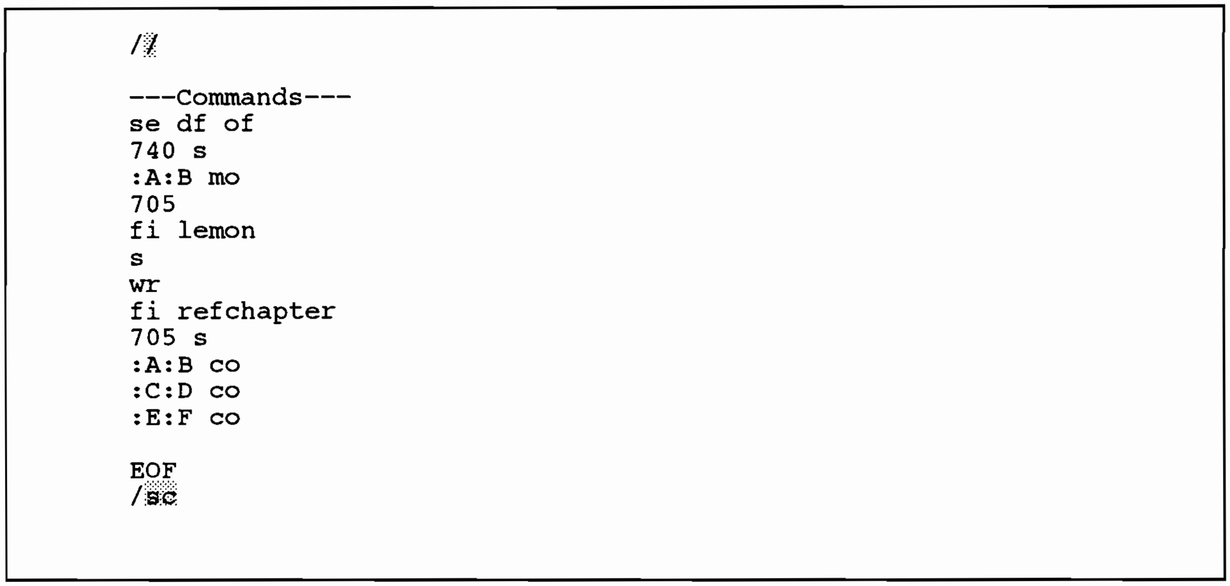
Everything is copied, including the SC command. EDIT stops when 24 or more null (blank) lines are encountered. Any text after the 24th null line is not copied.

On some terminals, the screen memory may be edited prior to executing the SC command by using the SCROLL and DELETE LINE keys.

The SC command can be used in a command file to be accessed via the Transfer (TR) command. This results in a screen copy in the output file, even if the Quiet option (Q) is used with the TR command.

Examples

Figure 4-36 illustrates the result of the use of the screen copy after a command stack (/) command.



```
//

---Commands---
se df of
740 s
:A:B mo
705
fi lemon
s
wr
fi refchapter
705 s
:A:B co
:C:D co
:E:F co

EOF
/sc
```

Figure 4-36. SC (Screen Copy) Command Example

SE (Set Option)

Uses

Use in line mode to set various EDIT options and defaults.

Command Syntax

SE <option> <value>

<option> Enter option to be set. A list of possible options follows. The standard defaults are indicated in the column to the right of the option name. For a more detailed description, refer to the section on EDIT/1000 options in Chapter 3.

<value> When required, enter numeric or on/off value for the option specified. If no value is entered and a numeric value is expected, EDIT uses the default value for that option. Options with numeric values cannot be set to zero (0). If a zero is entered, EDIT uses the default value. If no value is entered and an ON or OFF value is expected, EDIT toggles the current value.

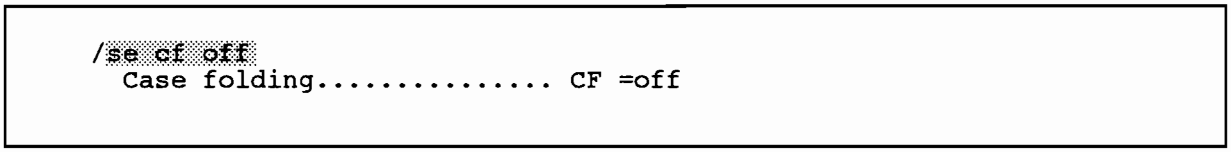
Remarks

To view a current option setting, use the *SH <option>* (Show Option) command. To view all options and the current settings, use the SH ALL (Show All) command.

A brief description of the EDIT options is provided in the following tables. For more details on the options, refer to the section on EDIT Session Options in Chapter 3.

Examples

Figure 4-37 illustrates the use of the SE command to set the Case Folding option to differentiate between upper- and lowercase letters.



```
/se cf off
  Case folding............... CF =off
```

Figure 4-37. SE (Set Option) Command Example

Table 4-1. EDIT Session Options

Option	Default	Description
AC	^	Anchor character used in pattern specification. Anchors the pattern to the start of search window.
AS	on	Asking for verification of dangerous commands, (that is, A, AS, D, FI, K, TR, U, X, and Y). Set to OFF to suppress OK? prompt.
BE	off	Bell. Set to ON to enable bell with EDIT prompt.
CD	off	To set line editing graphics sub-mode within screen mode.
CF	on	Case folding. Must be set to OFF to differentiate between upper and lower case in pattern searches.
CS	|	Command separator. Used to separate commands in a command or runstring.
DF	on	Display functions. Applies to the screen mode display of display enhancements such as blinking and inverse video, and control characters.
EC	\	Escape character. Used inside of patterns and line character edits to remove any special meaning from the next character.
IC	@	Indefinite match, or *wildcard* character. Used in patterns to specify zero or more occurrences of any character.
LE	256	Maximum number of characters allowed per line.
PC	/	EDIT Prompt character.
QU	off	Quiet mode. Set to ON to run EDIT without listing any output to the terminal. (When EDIT prompts for input, quiet mode automatically turns off. If you interactively set quiet mode on, it looks as if nothing happened since EDIT will prompt for the next command and turn off quiet mode.)
RE	off	Regular expressions. Set to ON to allow the use of Regular Expressions mode metacharacters in special pattern searches /matches.
RT	on	Find return. ON causes a search command (F or B) to return to the original pending line after reaching the second line specification. OFF causes EDIT to leave the pending line at one line beyond the second line specification entered.

Table 4-1. EDIT Session Options (continued)

Option	Default	Description
SD	10,10,2	Screen size defaults. Numbers indicate screen format. Lists number of lines displayed above pending line, number of lines displayed below pending line, and the number of lines overlapped (displayed from the previous screen) from one screen to the next as a result of CTRL-P or CTRL-F commands. Screen size is equal to the sum of the first and second number, plus one.
SL		Screen mode line limit. Sets the maximum number of lines that are displayed for a screen edit. If not set by a SE SL command, EDIT picks a default value based on the number of bytes of memory returned by the terminal in response to the primary status request resulting from a screen mode command (S).
TC	TAB, CTRL-I	Line character edit tab character. Note that if the tab character is set to something other than CTRL-I, CTRL-I becomes an alternate *insert* command for line character edits.
TS	on	Time stamp. ON causes EDIT to update all time stamps when the work file is written to with an ER or WR command. Time stamp format is <YYMMDD.HHMM>, where "<" and ">" are the characters *less than* and *greater than* and they enclose the date information. The date information is numeric, where YY = year, MM = month, DD = day, HH = hours, and MM = minutes.
VW	10,10	Vertical window. Use to set the default line range for the W(Window) and L (List) commands. The first number is the default number of lines above the pending line to start display of a window. The second number is the default number of lines below the pending line to end the display.
WC	1,256	Window columns. Specifies the left and right columns that are used as the horizontal search window for pattern searches.
IN		Sets the indentation on or off.
FL		Fills text to the margins set with SE FL.
SW		Screen width. Uses the SH command to show the current screen width. This will be zero until either the command stack or a screen mode command is entered, at which time the terminal screen width is sensed.
SW n		Screen width set. Uses the SE command to set screen width to n. n must be greater than −1 and less than 256. If set is 0, EDIT and set the sensed value on the next command stack or screen mode command.

SH (Show Options)

Uses

Use in line mode to view various EDIT options and defaults.

Command Syntax

SH <option>

<option> Enter option to be displayed. Possible options that can be displayed are illustrated in the example following. To view all possible options, enter SH or SH ALL. (Refer to example, following.) Other EDIT settings that can be viewed using the SH command include

FM Shows whether any changes have been made. Set by a work file modification, and turned off by WC, WR (Write Create and Write Replace), or FI (File Input) commands.

MA Displays the current text marks and their corresponding line numbers.

SW Show Width. Shows the current screen width. This will be zero until either command stack or a screen mode command is entered, at which time the terminal is sensed.

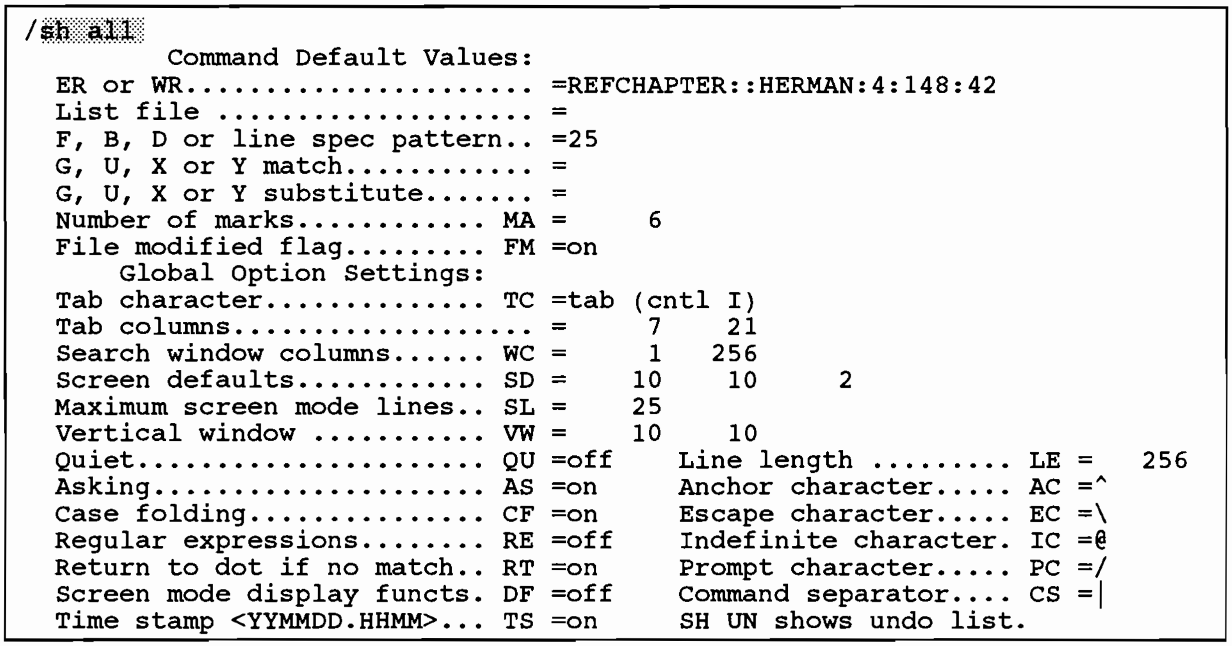
UN Shows the Undo List maintained by EDIT. This list is used when the UN (Undo) command is executed, and it is updated for each text modification.

Remarks

For more information on EDIT options and defaults, refer to the section on the SE (Set Option) command and the section on EDIT/1000 options in Chapter 3.

Examples

Figure 4-38 illustrates the display resulting from the use of the *SH ALL* command.



```
/sh all
         Command Default Values:
  ER or WR...................... =REFCHAPTER::HERMAN:4:148:42
  List file .................... =
  F, B, D or line spec pattern.. =25
  G, U, X or Y match............ =
  G, U, X or Y substitute....... =
  Number of marks............ MA =     6
  File modified flag......... FM =on
      Global Option Settings:
  Tab character.............. TC =tab (cntl I)
  Tab columns................... =     7    21
  Search window columns...... WC =     1   256
  Screen defaults............ SD =    10    10      2
  Maximum screen mode lines.. SL =    25
  Vertical window ........... VW =    10    10
  Quiet...................... QU =off    Line length ......... LE =   256
  Asking..................... AS =on     Anchor character..... AC =^
  Case folding............... CF =on     Escape character..... EC =\
  Regular expressions........ RE =off    Indefinite character. IC =@
  Return to dot if no match.. RT =on     Prompt character..... PC =/
  Screen mode display functs. DF =off    Command separator.... CS =|
  Time stamp <YYMMDD.HHMM>... TS =on     SH UN shows undo list.
```

Figure 4-38. SH (Show Options) Command Example

SZ (Check File Size)

Uses

Use in line mode to display the number of words in the file from line one of the file to the current pending line. Note that record length words (those used by EDIT to format the text file) are not included in the size reported.

Command Syntax

[linespec 1] SZ

linespec 1 Optional line number for end of count. Enter a line number or $ (for end of file) to determine where the count stops. For example, *20 SZ* counts words from line one to line 20. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.)

Remarks

To obtain the TOTAL number of words in the file, the last line number for the file or a $ must be specified in the command string, or the current pending line must be at the EOF.

After executing the count, EDIT returns to the current pending line.

Examples

Figure 4-39 illustrates the use of the SZ command to list file size.

```
/300
  This is line 300 of the file.
/sz
  5992
  This is line 300 of the file.
/$                                                         (Go to end of file)
/
EOF
/300 sz                                    (Count words from line 1 to line 300)
  5992
EOF
/sz                                        (Count words from line 1 to end of file)
  19328
/
```

Figure 4-39. SZ (Check File Size) Command Examples

T (Set Tabs)

Uses

- Use in line mode to set tab columns.
- Use special tab commands (TA, TF, TM, TP, and TU) to set tab columns as they are commonly set for such programming languages as Assembler, FORTRAN, Macro, Pascal, and HP-UX.
- Use TL version of command to set terminal tab stops to line up with line mode tabs.
- Use TS version of command to set terminal tab stops to line up with screen mode tabs.
- Use TU to set stops for the full screen width.

Command Syntax

T [tc1,tc2,...tc30]

tc1,tc2...tc30 Enter up to 30 tab column numbers to indicate the location of the tab stops.

Remarks

The T command is used to specify the column numbers for up to 30 tab stops. Once set, these stops can be displayed and adjusted local to the terminal set with either the TL or TS command. The pre-set tab stops (TA, TF, TM, TP, and TU) can also be used for program development. If no tabs are set, EDIT defaults tab stops to the TA tab setting. Note that the terminal defaults to no tab stops.

To enter a tab stop during text entry, use the current tab character. The default tab character is the TAB key on the terminal (or CTRL-I). It may be changed to another character with the SE TC (Set Tab Character) command. To view the current default for the tab character, use the SH TC (Show Tab Character) command.

The tab column settings for the special tab commands can be viewed online by entering *? T.* The special tab column settings follow:

TA Assembler (ASMB) tab settings, in columns 7, 21, and every column after 21.

TF FORTRAN (FTN) tab settings, in column 7, every 4 columns until the tenth stop, and every column after the tenth stop.

TM Macro tab settings, in columns 10, 26, 40, 44, 48, and every column after column 48.

TP Pascal tab settings, which are every 3 columns until the tenth stop, and every column after the tenth stop, starting in column 4.

TU HP-UX tab settings, in every eighth column, starting in column 9 up to the full screen width.

TL Use the TL command to set the terminal tab stops so that they line up with EDIT's line mode tab stops. (Remember that text displayed in line mode is shifted two characters to the right of column one.)

TS Use the TS command to set the terminal tab stops so that they line up with EDIT's screen mode tab stops.

The terminal's internally-set tab stops and EDIT's tab settings may not always be synchronized. What is displayed on the terminal may be quite different from what is stored in EDIT. Use the TS and TL commands to ensure that the two are synchronized.

Care should be exercised when using the backspace key while in line mode. When backspace is used, the terminal may show that the cursor moved back only one character, but if the last key stroke entered was a tab stop, the tab stop is deleted.

Examples

Figure 4-40 illustrates the use of the T command. Note that the header line has been shifted left to illustrate the column positions set by the tab commands. In actuality, the header line is displayed starting in column 2.

```
/hl
''''/''''1''''/''''2''''/''''3''''/''''4''''/''''5''''/''''6''''/''''7'''
/t10,20,30,40
         T         T         T         TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT

/tl
            T         T         T
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT

/ts
          T         T         T         TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT

/tp
   T  T  T  T  T  T  T  T  T  TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT

/tf
      T   T   T   T   T   T   T   T   T   TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT

/tu
        T   T   T   T   T   T   T   T   T
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
/
```

Figure 4-40. T (Set Tabs) Command Examples

TI (Time)

Uses

Use in line mode to add the time and date to the current pending line.

Command Default

. TI 1

Command Syntax

[linespec 1] TI n

linespec 1 Optional line specification. Specifies which line is to be stamped with the time data. Default is the pending line. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.)

n Starting column number for the time and date information. Default is column one.

Remarks

The Undo (UN) command reverses the TI command.

The time and date information is 30 characters long.

Examples

Figure 4-41 illustrates the use of the TI command to add the time and date at column 20 on line 1171.

```
/1171 ti 20
  Time Stamp Example:  1:15 PM  MON., 26  JAN., 1987
/
```

Figure 4-41. TI (Time) Command Example

TK (Tab Kill)

Uses

Use to expand tab characters into the appropriate number of spaces.

Command Default

1 $ TK

Command Syntax

[linespec 1] [linespec 2] TK

linespec 1
linespec 2 Optional line specification. Specifies range of lines for tab kill operation. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.)

Remarks

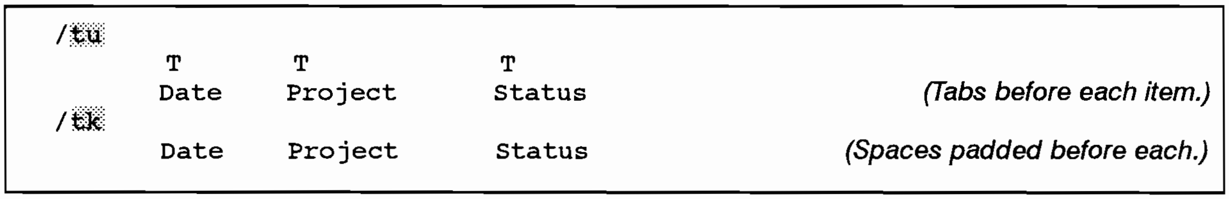
This command deletes the tab characters in the file and adds spaces to shift the next character to the next tab column of EDIT's current tab settings.

The Undo (UN) command reverses the TK command.

Tab characters are not normally processed by HP 1000 software. Files imported from other systems may contain the characters and this command can be used to replace them with the appropriate number of spaces.

Examples

Figure 4-42 illustrates the use of the TK command to pad blanks to place HP-UX tabs (set by the command TU) in the appropriate columns.



```
/tu
      T      T         T
      Date   Project   Status                    (Tabs before each item.)
/tk
      Date   Project   Status                    (Spaces padded before each.)
```

Figure 4-42. TK (Tab Kill) Command Example

TR (Transfer)

Uses

- Use in line mode to transfer to a command file.
- Use when running EDIT to execute a batch file.

Command Syntax

TR<filedescriptor> [Q] [/]

<filedescriptor> Enter the filename of the file to which control is to be transferred. (Refer to section in Chapter 3 on filename specification.)

Q Optional. Enter the Quiet option to suppress listing of the transfer file.

/ Optional suppression of asking for dangerous command confirmation. Enter the current prompt character (default = /) to suppress the OK? prompt.

Remarks

This command transfers to a specified file to execute the commands contained in that file. It is important to note that command files do NOT nest. For example, assume that a transfer command is entered to access Command File A. Within Command File A is a transfer command to Command File B. Once EDIT is finished executing the commands in File B, it returns to the terminal and NOT to Command File A.

If the transfer command file contains an ER (Exit and Replace), EC (Exit and Create), or an A (Abort) command, the applicable command is executed and EDIT terminates.

When the TR command is entered within a command string, (for example, the EDIT runstring), it MUST be the last command in the string. Therefore, the TR command cannot be used in conjunction with the _ (Repeat) command.

This is considered a dangerous command because it has the potential to delete data that you meant to save. Once the TR command is executed, there are no requests from EDIT for verification of dangerous commands contained in the transfer file. To safeguard against costly mistakes, EDIT displays an OK? prompt after entry of the TR command. A *"y"* executes the command.

EDIT error conditions occurring during execution of the transfer terminate the transfer file and return control to the terminal.

If EDIT is running in batch (−B runstring option) any error causes EDIT to abort.

The Comment (*) command can be used to enter comments within an EDIT transfer file.

Examples

```
/tr,cmdfil
 Opened file CMDFIL::JL:4
 OK? y
 Closed file CMDFIL::JL:4
 Closed file EDITING::JL:4                        (ER in command file)
 end of edit
CI>
```

Figure 4-43. TR (Transfer) Command Example

U (Unconditional Exchange)

Uses

Use in line mode to replace a specific number of characters with a substitute string. This command is unconditional because it deletes a specific number of characters on each line in the line range, regardless of the type of character, and inserts the characters from the substitute.

Command Default

. * U /last exchange pattern/last substitute/

Command Syntax

[linespec 1] [linespec 2] U /delete field/substitute/ [Q]

linespec 1
linespec 2 Optional line range. May be line numbers or any of the line specification characters. If omitted, the exchange begins and ends with the current pending line. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.)

/ Delimiter for delete field and substitute. Use any matched punctuation mark (except commas and spaces). The last delimiter may be omitted if the Q (Quiet) option is not used.

delete field Unconditional deletion field. Each character causes one character to be deleted. The delete field may be a zero length field. Deletion starts at the left-most column of the window. To view the setting for window search columns, use the SH WC (Show Window Columns) command. Characters may extend beyond the window.

substitute The new string to replace the field deleted. Number of characters need not be the same as the match field. May also be a zero-length field.

Q Optional. Enter the Q (Quiet) option to suppress listing of matches.

Remarks

If no match field and/or substitute are entered, the U command defaults to the pattern and substitute last entered for any of the exchange commands (U, G, X, or Y). To view the default pattern and substitute, use the SH (Show) command for any of the exchange commands.

The line below the last line specified in the range becomes the new pending line.

All lines in the line range that are shorter than the starting window are blank-padded to the starting window column.

Examples

Figure 4-44 illustrates the use of the U command to delete the first eight characters on all lines of a file, suppressing the listing and the prompt. In the second example, the U command is used to replace the first two characters with the string *11111*, suppressing the prompt with a slash.

```
/1 $ u/xxxxxxxx//Q/

/1 $ u/aa/11111//
```

Figure 4-44. U (Unconditional Exchange) Command Examples

UN (Undo)

Uses

Use to reverse the last file modification command executed.

Command Syntax

UN

Remarks

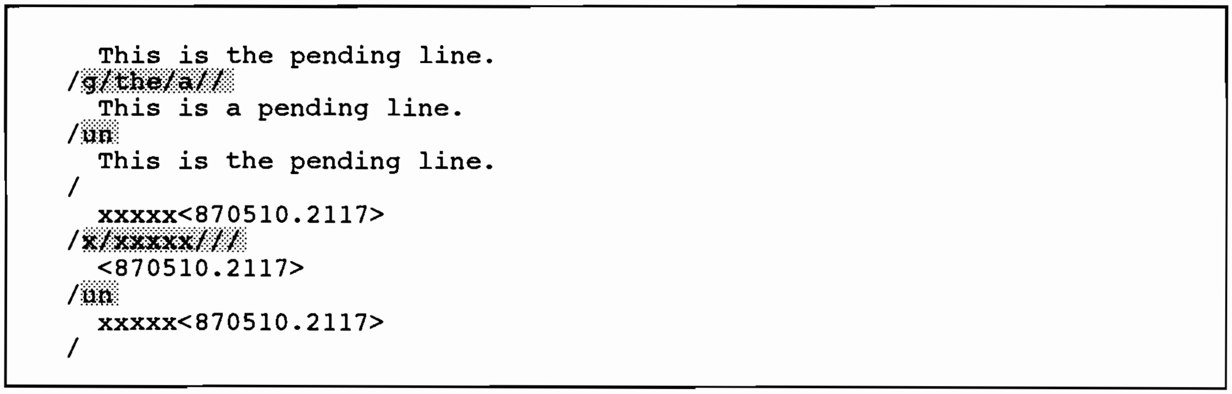
This command can be used after the BC, BK, BM, C, CO, D, G, I, J, K, M, MO, O, P, Q, R, SC, TI, TK, U, X, or Y commands.

An *undo list* of the last change made is maintained by EDIT. The most recent change can be viewed by using the SH UN (Show Undo) command.

The UN command should not be used after the UY (Undo List Yank) command. The UY command rearranges the order of lines in the file but does not alter the Undo list. Therefore, use of the UN command does not restore the file to its original state before use of the UY command or before the command that built the undo list. In some cases, use of the UN command after the UY command may also delete data.

Examples

Figure 4-45 illustrates the use of the UN command to reverse a G (Exchange on pending line) and an X (Exchange) command.



```
  This is the pending line.
/g/the/a//
  This is a pending line.
/un
  This is the pending line.
/
  xxxxx<870510.2117>
/x/xxxxx///
  <870510.2117>
/un
  xxxxx<870510.2117>
/
```

Figure 4-45. UN (Undo) Command Examples

UY (Undo List Yank)

Uses

Use to return all or a portion of the lines contained in the EDIT Undo List to the work file, inserting them after the current pending line. This command modifies the line numbering of the file and thus may invalidate the UN (Undo) command.

Command Default

. UY 1 32767

Command Syntax

[linespec 1] UY [n1] [n2]

linespec 1 Optional line number. Enter the line number after which the recovered lines are to be inserted. Default is the current pending line. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.)

n1 Optional. Enter the first line on the undo list to be recovered. For example, enter 1, 2, or 3 for the first, second, or third line from the top of the list.) Default is the first line of the undo list.

n2 Optional. Enter the number of lines to recover. Default is to recover all lines in the undo list from the start line specified.

Remarks

An *undo list* of the last change made is maintained by EDIT. The most recent change can be viewed by using the SH UN (Show Undo) command.

The UY command rearranges the order of lines in the file but does not alter the undo list. Therefore, use of the UN (Undo) command does not reverse the UY command nor does it restore the file to its original state before use of the command that created the undo list. In some cases, use of the UN command after the UY command may also delete data.

Examples

Figure 4-46 illustrates the use of the UY command to retrieve lines 1 through 20 of the undo list, inserting them at the end of the file.

```
/$ uy 1,20
```

Figure 4-46. UY (Undo List Yank) Command Example

W (List Window)

Uses

- Use from line mode to list a vertical window. Displays the text within the window. This command does not move the pending line.
- Use the WU version of this command to list vertical window without line numbering. Turns off the display of numbers for succeeding W commands.
- Use the WN version of this command to list line numbers. Turns on the display of numbers for succeeding W commands.

Command Default

.–10 *+20 W

Command Syntax

[linespec 1] [linespec 2] W

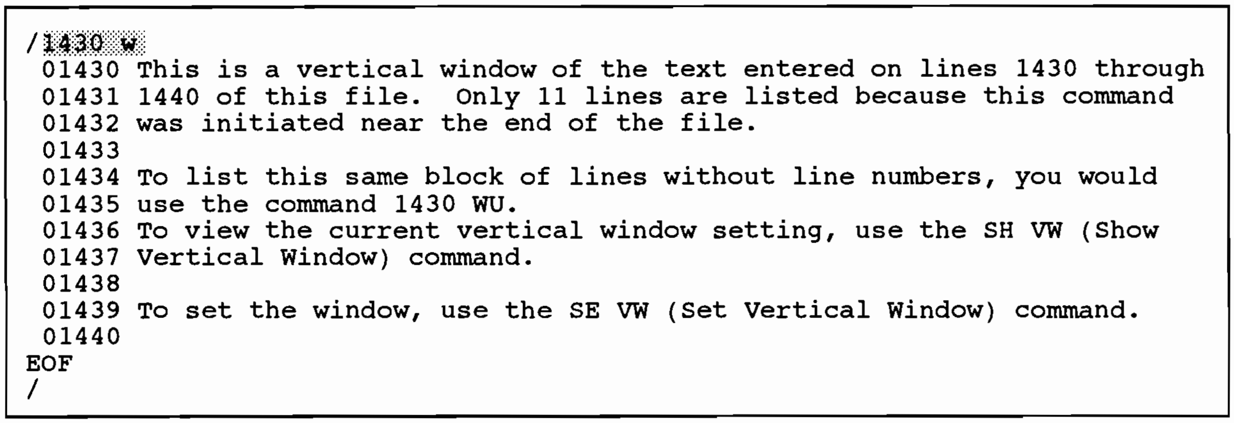
linespec 1
linespec 2 Optional vertical window range. Enter either absolute line numbers separated by a space or comma, or any of the line specification characters. If omitted, linespec 1 defaults to list 10 lines above the pending line and linespec 2 defaults to list the first line specification and the next 20 lines. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.)

Remarks

To view the current vertical window setting, use the *SH VW* (Show Vertical Window) command.

The mark letter for marked lines is displayed in column one. An arrow (>) is displayed to the left of the pending line. If the pending line is also marked, the arrow is displayed instead of the mark.

Examples



```
/1430 w
 01430 This is a vertical window of the text entered on lines 1430 through
 01431 1440 of this file.  Only 11 lines are listed because this command
 01432 was initiated near the end of the file.
 01433
 01434 To list this same block of lines without line numbers, you would
 01435 use the command 1430 WU.
 01436 To view the current vertical window setting, use the SH VW (Show
 01437 Vertical Window) command.
 01438
 01439 To set the window, use the SE VW (Set Vertical Window) command.
 01440
EOF
/
```

Figure 4-47. W (List Window) Command Example

WC (Create File without Exit)

Uses

Use to create a new file without exiting EDIT.

Command Syntax

WC <filedescriptor>

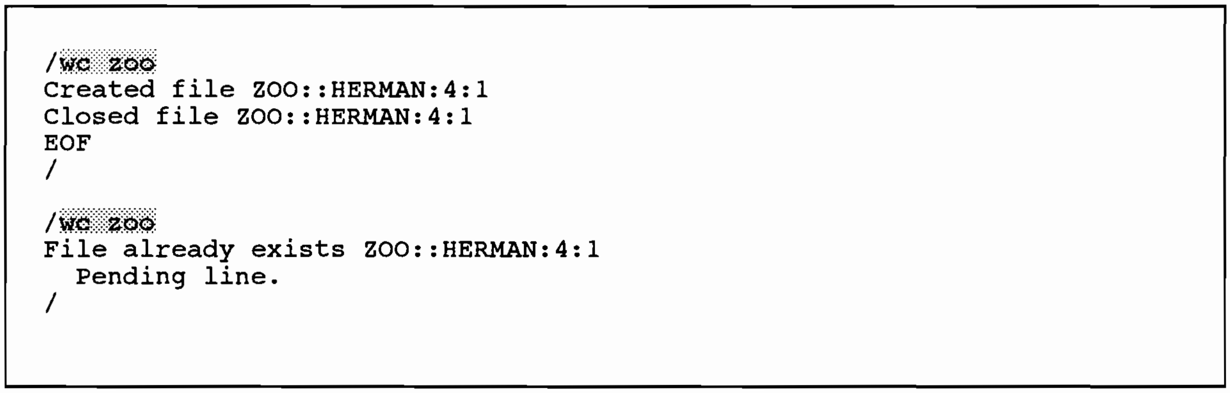
<filedescriptor> Enter the filename descriptor to be assigned to the new file. (Refer to section in Chapter 3 on filename specification.)

Remarks

See the ER (Exit and Replace) command description for remarks regarding file errors.

Examples

Figure 4-48 illustrates the use of the WC command to create a new file, and an attempt to create a file with an existing filename.



```
/wc zoo
Created file ZOO::HERMAN:4:1
Closed file ZOO::HERMAN:4:1
EOF
/

/wc zoo
File already exists ZOO::HERMAN:4:1
  Pending line.
/
```

Figure 4-48. WC (Create File without Exit) Command Examples

WR (Write and Replace)

Uses

Use to replace a file with the new edits without exiting EDIT.

Command Syntax

WR [<filedescriptor>]

<filedescriptor> Enter the filename descriptor of the file to be replaced. The default is the current source file. (Refer to section on filename descriptors in Chapter 3.)

Remarks

See the ER (Exit and Replace) command description for remarks regarding file errors.

If the source file has been closed and is on the working directory, the *WR* command opens the source, writes to it, and closes it again.

Examples

Figure 4-49 illustrates the use of the WR command to replace the current file LEMON, to write to another existing file E2, and to write to the existing file E4, designating the directory and file information.

```
/wr
Posted file LEMON::HERMAN:4:1:4
  First line of the file.
/wr e2
Opened file E2::HERMAN:4:1:4
Closed file E2::HERMAN:4:1:4
  First line of the file.
/wc e4::herman:4:1:4
Created file E4::HERMAN:4:1:4
Closed file E4::HERMAN:4:1:4
  First line of file.
/
```

Figure 4-49. WR (Write and Replace) Command Examples

X (Exchange)

Uses

- Use to search for a pattern and exchange it with a new pattern within the line range specified in the command string. Use with the Q (Quiet) option to suppress listing of matched lines.
- Use with RE (Regular Expressions) set to ON to use the metacharacters for pattern matching. (Refer to Chapter 5 for more information on Regular Expressions.)

Command Default

. * X <last exchange and substitute>

Command Syntax

[linespec 1] [linespec 2] X /pattern/substitute/ [N Q R S] [/]

linespec 1
linespec 2 Optional. A line range may be specified for this command using either absolute line numbers separated by a space or comma, or line specification characters. If no range is specified, the exchange affects only the pending line. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.)

/ Delimiter for pattern and substitute. Use matching punctuation marks with the exception of commas and spaces. All three delimiters must be used when any option or OK? prompt suppression is used. If no options are used, the last delimiter may be omitted.

The default pattern and substitute for exchange commands may be used by omitting the pattern, delimiters, and substitute. To view the current pattern and substitute defaults, use the SH (Show) command for any of the exchange commands (G, U, X, and Y). To default the pattern and substitute but specify an option, enter *X,, <option>*.

pattern Pattern to be located. (See section in Chapter 3 on pattern specifications.) If Regular Expressions is set to ON, the metacharacters can be used for unique pattern matches (refer to Chapter 5 for more on Regular Expressions). If a null pattern is entered, the Find default pattern is used.

substitute Pattern to replace the search pattern specified first in the command string. Substitute pattern can be longer or shorter or shorter than the search pattern. EDIT inserts or deletes characters as necessary to fill the line.

N Optional no window flag. Enter the N option to disregard the horizontal window columns setting and allow a match anywhere on a line. (To see setting for window columns, use SH WC command.)

Q Optional quiet flag. Enter the Q option to indicate the total number of exchanges, without listing the matched lines.

R Optional remove flag. Removes null (blank) lines after an exchange is executed.

S Optional single exchange flag. The S option allows only one exchange per line.

/ Suppresses the OK? prompt displayed when changes have been made.

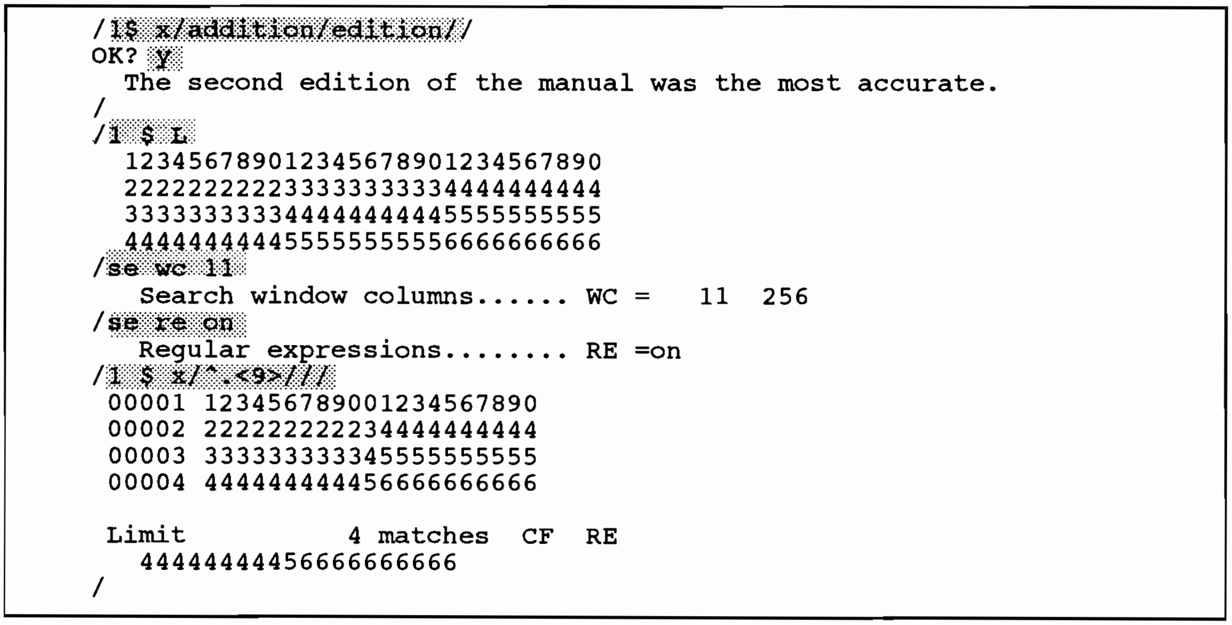
Remarks

The Undo (UN) command reverses the X command.

The X command is similar to the G command, except that the X command provides a summary listing of the lines affected by the exchange; the G command does not.

Examples

Figure 4-50 illustrates the use of the X command to replace the word *addition* with the word *edition*, and to delete columns 11 through 19. The L (List) command was used to display the file prior to the changes and the SE (Set) command was used to set the left column of the search window to column 11 and to turn on Regular Expressions.



```
/1$ x/addition/edition//
OK? y
  The second edition of the manual was the most accurate.
/
/1 $ L
  123456789012345678901234567890
  222222222233333333334444444444
  333333333344444444445555555555
  444444444455555555556666666666
/se wc 11
   Search window columns...... WC =   11  256
/se re on
   Regular expressions........ RE =on
/1 $ x/^.<9>///
 00001 123456789001234567890
 00002 222222222234444444444
 00003 333333333345555555555
 00004 444444444456666666666

 Limit          4 matches  CF  RE
   44444444456666666666
/
```

Figure 4-50. X (Exchange) Command Examples

Y (Exchange and Search)

Uses

- Use to exchange a pattern with a new pattern on the current line, and then find the next occurrence of the pattern. Use instead of the other exchange commands (G, U, X) when there is a need to check the match before executing the exchange.

- Use with RE (Regular Expressions) set to ON to use the metacharacters for pattern matching. (Refer to Chapter 5 for more information on Regular Expressions.)

Command Default

. Y /last pattern specified/last substitute specified/

Command Syntax

[linespec 1] Y/pattern/substitute/ [N] [Q] [R] [S]

linespec 1 Optional. A beginning line may be specified for the start of the search. May be either absolute numbers or line specification characters. If omitted, the search begins at the current pending line. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.)

/ Delimiter for pattern and substitute. Use any matching punctuation marks with the exception of commas and spaces. All three delimiters must be used when any option or OK? prompt suppression is used. If no options are used, the last delimiter may be omitted.

The default pattern and substitute for exchange commands may be used by omitting the pattern, delimiters, and substitute. To view the current pattern and substitute defaults, use the SH command for any of the exchange commands (G, U, X, and Y). To default the pattern and substitute but specify an option, enter *Y,,<option>*.

pattern Pattern to be located. (See section on pattern specifications in Chapter 3.) If Regular Expressions is set to ON, the metacharacters can be used for unique pattern matches (refer to Chapter 5 for more on Regular Expressions). If a null pattern is entered, the Find default pattern is used.

substitute Pattern to replace the search pattern specified first in the command string. Substitute pattern can be longer or shorter than the search pattern. EDIT inserts or deletes characters as necessary to fill the line.

N Optional no window flag. Enter the N option to disregard the horizontal window columns setting and allow a match anywhere on a line. (To see setting for window columns, use SH WC command.)

Q Optional quiet flag. Enter the Q option to suppress listing of the matched lines.

R Optional remove flag. Use the R option to remove a null (blank) line after the exchange.

S Optional single exchange flag. The S option allows, at most, one exchange per line.

Remarks

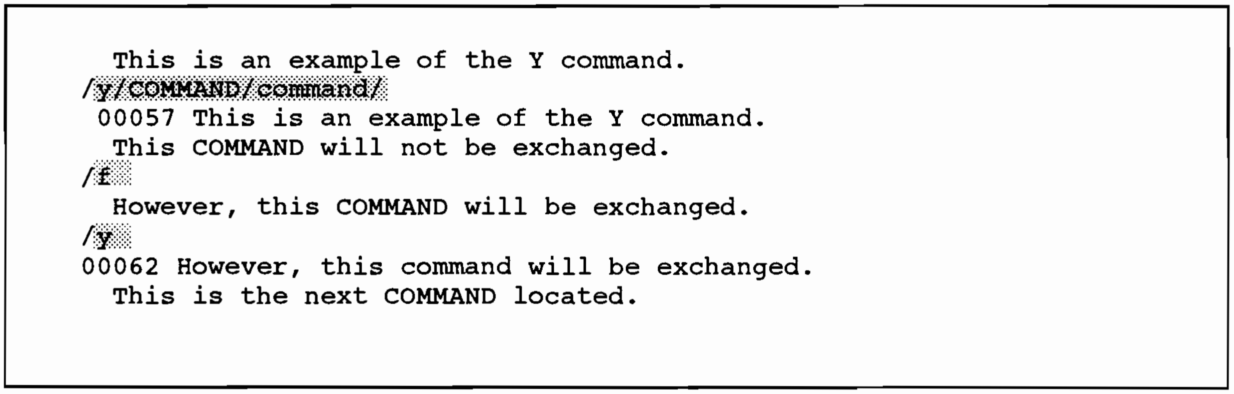
The Y command functions as an exchange (X) command on the pending line and then as a Find (F) command.

The specified search pattern is copied to the default Find buffer. This allows the following find command to use the specified pattern without re-entry. In this way, you can walk through your file, executing exchanges by entering a *Y* or going on to the next match by entering an *F.*

The Undo (UN) command reverses the Y command.

Examples

Figure 4-51 illustrates the use of the Y command to exchange the string *COMMAND* to *command.*



```
  This is an example of the Y command.
/y/COMMAND/command/
 00057 This is an example of the Y command.
  This COMMAND will not be exchanged.
/f
  However, this COMMAND will be exchanged.
/y
00062 However, this command will be exchanged.
  This is the next COMMAND located.
```

Figure 4-51. Y (Exchange and Search) Command Example

EDIT/1000 Non-Alphanumeric Commands

EDIT/1000 provides several non-alphanumeric commands that are used in the editing process. They include

Used to place sequence numbers at the end of all lines in the file.

? Used to access online quick reference, or Help. (Same as the H command mnemonic.)

?? Used to display the version of the EDIT program being run, the source file being edited, and the action that an ER command would perform on the current file.

/ Used to list a review of the last 20 commands used. Referred to as the *Command Stack.*

<space> Used to append (add) a line of text after the pending line, making the appended line the new pending line.

_ Repeat command, used to repeat a command string or a line of text.

* Used to enter a comment line in a command file. All characters on the line following this command are ignored by EDIT.

A description of each of the above commands follows.

(Sequence Numbers)

Uses

Used to place sequence numbers at the end of all lines in the file. Places numbers in columns 76 through 80.

Command Syntax

#xxx[n1 [,n2]]

xxx Three-character identifier field. Can be any three printing characters. Follows the # command immediately, without any delimiting characters.

n1 Optional starting sequence number to be entered. Defaults to 00000.

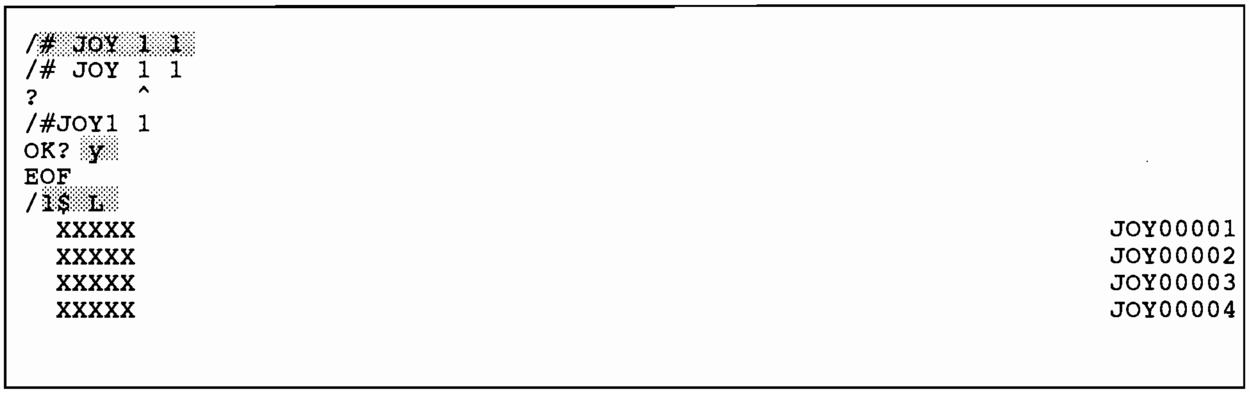
n2 Optional incremental value. Defaults to 10.

Remarks

This command is useful in adding sequence numbers to FORTRAN programs. It can also be used in combination with a Regular Expressions exchange command to be unique numbers in lines.

Examples

Figure 4-52 illustrates the use of the # command. In the first attempt, there is an illegal space between the # command and the 3-character identifier. The second attempt is successful. The L (List) command is used to display the text after execution of the # command.



```
/# JOY 1 1
/# JOY 1 1
?      ^
/#JOY1 1
OK? y
EOF
/1$ L
  XXXXX                                                             JOY00001
  XXXXX                                                             JOY00002
  XXXXX                                                             JOY00003
  XXXXX                                                             JOY00004
```

Figure 4-52. # (Sequence Numbers) Command Example

? (Online Quick Reference)

Uses

Used to access the online quick reference, or Help, for any EDIT/1000 command or option.

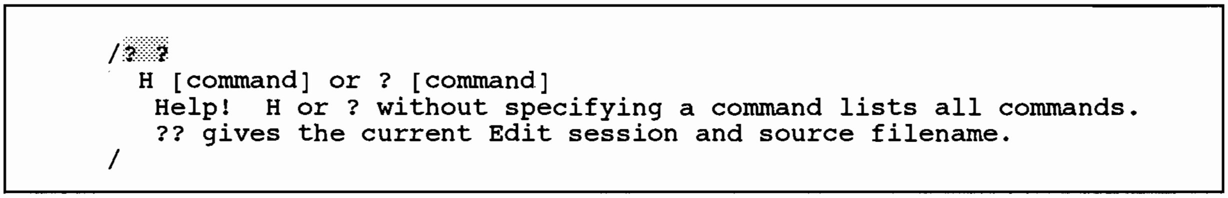
Command Syntax

? [command or option]

Remarks

This command is identical to the H (Help) command. For more information, refer to the detailed description under mnemonic command H.

Examples



```
/? ?
  H [command] or ? [command]
   Help!  H or ? without specifying a command lists all commands.
   ?? gives the current Edit session and source filename.
/
```

Figure 4-53. ? (Online Quick Reference) Command Example

?? (Display Source File Status)

Uses

Used to display the name (version) of the EDIT program, source file being edited, and the action an ER command performs.

Command Syntax

??

Remarks

If the source file is closed (FCS) or will be created, this state is also reported.

Examples

```
/??
  EDIT on NEWCHAPTER::HERMAN:4:960:44
/
/??
  EDIT on XYZ::HERMAN:4:960:44, to be created

/??
  EDIT on WORT::xx, closed
```

Figure 4-54. ?? (Display Source File Status) Command Examples

/ (Command Stack)

Uses

Used to display last 20 commands entered during EDIT session. Listed commands can then be edited and re-executed by placing the cursor on the desired command and pressing the carriage return.

Command Syntax

/ [n]

or

//[/...]

n Optional number of commands to display. Default is 20 commands.

Remarks

Commands are listed in sequence with the cursor positioned at the last blank line. The cursor can be moved to any command in the stack with the cursor arrow keys. The command can then be re-executed with a carriage return, or edited and then executed. Enter carriage return immediately after display of the stack to terminate the command and return to the EDIT prompt.

Only the first 79 characters of a command line are placed in the command stack. The use of command separators does not influence what is saved in the command stack. The last 20 command lines entered, including illegal commands, are saved.

The control character commands entered in screen mode are not recorded in the command stack.

If a command matches one already in the stack, the old command is removed from the stack.

The command stack only works with the terminal and serial driver combination that support screen mode. If the terminal does not support screen mode, use of the "/" command merely advances the pending line.

For additional information of command stack features, see the *RTE-A User's Manual* or the *RTE-6/VM CI User's Manual.*

Examples

```
//

---Commands---
se df of
1 $ f/.sect
100 $ f/.sect
1 $ f/.sect F
1 $ f/# (Sequence N
421
429
:A:B co
```

Figure 4-55. / (Command Stack) Command Example

<space> (Append a Line)

Uses

Used in line mode to add a line of text following the pending line, making the appended line the new pending line.

Command Syntax

<space>text

Remarks

A <space> as the first character in the command line is the command to append a line of text to the work file. The space is removed, tabs are expanded and escape processing is done. The resulting line is put into the work file and becomes the new pending line. If this command is entered without any text, a zero length line is inserted into the text.

Examples

Figure 4-56 illustrates the use of the space command to add a new line of text. The L (List) command is used to display the text before and after the execution of the space command.

```
/L3
  Line one
  Line two
  Line three
/ New line added
/-3L 4
  Line one
  Line two
  Line three
  New line added
/
```

Figure 4-56. <space> (Append a Line) Command Example

_ (Repeat)

Uses

Used to repeat the execution of a command line a specific number of times. Must be at the end of the command line, separated from the other commands on the line by a command separator.

Command Default

|1 $+1_<number entered>

Command Syntax

|[linespec 1] [linespec 2] _[n] [Q]

| Command separator. Must precede the repeat command, separating it from the rest of the command or text.

linespec 1
linespec 2 Line range for execution. If the pending line is outside of this line range, the command line is not repeated. Note that the pending line can never be outside of the default line range. (Refer to the section on line specification in Chapter 3 for a complete list of possible line specifications.)

n Optional number of times the command line is to be repeated.

Q Optional. Enter the Quiet (Q) option to suppress listing of commands repeated.

Remarks

EDIT counts backward to zero, therefore the number of repetitions is one greater than the number specified.

Unless the Q option is used, EDIT displays the command string or text after each repetition, along with the number of repetitions remaining.

The command line is repeated only if the pending line is between the line spec 1 and line spec 2, inclusive.

The number of times the command line is repeated is specified by the numeric argument, n. If the numeric argument is omitted, the command line repeats until the pending line is outside the line range specified. If the numeric argument is omitted, a line range must be specified to prevent a continuous loop.

An operator break terminates the repeat.

Any error terminates the repeat.

EDIT does not allow both a TR (Transfer) and _ (Repeat) command on the same command line.

Examples

Figure 4-57 illustrates the use of the repeat command to make 11 copies of a line containing a vertical bar (for use in a table). *SE CS !* is entered to set a new command separator, then the <space> command is entered to append a line of text that contains five vertical bars with the repeat command.

```
/se cs !
     Command Separator..... CS =!
    /
    / |          |           |          |          |!_10
    /_10
    / |          |           |          |          |
    /_00009
    / |          |           |          |          |
    /_00008
    / |          |           |          |          |
    /_00007
```

Figure 4-57. _ (Repeat) Command Example

Figure 4-58 illustrates a command that finds all BEGIN-END blocks and appends them to the list file (which must be opened before this command line is entered). For this to function properly, the work file cannot have nested BEGIN-END blocks that start in column 1.

The SE RT OFF command causes the Find command to remain at the EOF when there are no more lines that start with BEGIN. This example starts by positioning the file to the first line that starts with BEGIN. The repeated command lines mark the pending line with an “a”, finds the next line that starts with END, and marks it with a “b”. It then appends the blocks marked with “a” and “b” to the list file. The work file is then positioned to the next line that starts with BEGIN and the process is repeated until the pending line is after the last line in the file.

```
/se rt off
/b/^begin/
/ka|f/^end/|kb/|:a:b l,,+|f/^begin/|1 $_
```

Figure 4-58. _ (Repeat) Command Example

* (Comment)

Uses

Used to enter a comment line in an EDIT command file. All characters after the * up to the end of the command line are ignored by EDIT. This includes ignoring a command separator character (|) if it appears after the comment character.

Command Syntax

*<text>

Remarks

EDIT ignores the remaining text on the command line once it encounters the comment command. No further changes are made to the work file.

The * command provides a method of commenting an EDIT transfer file.

Examples

Following is an example of a portion of an EDIT transfer file. The first line is a comment by itself, the second line contains both a command and a comment.

```
  *This is a comment.
/f/xyzzy/       |*This is also a comment.
```

Figure 4-59. * (Comment) Command Example

5

EDIT/1000 Regular Expressions

Introduction

The information in this chapter will enable you to

- Define the terms: Regular Expressions, Metacharacters, Literal Matching, Anchoring, Character Class, Negated Character Class, Alphanumeric Transition (or Word Delimiting), Switch or Exchange, and Tag Field.
- Distinguish between Regular Expressions ON mode and Regular Expressions OFF mode, including the metacharacters valid for each mode, and longest- versus shortest-possible matching.
- Successfully use Regular Expressions to exchange columns of data, to delete and add text to a group of similar lines in a file, to align text, to delete all blank lines in a file, to capitalize only the first characters of each word in a file, to insert a character in a specific column, to change uppercase to lowercase, to add text to the end of variable length lines, to make all lines a minimum length and to find similar numbers.
- Use Regular Expressions in conjunction with the RTE CI user interface to build file manipulation commands and to gather data from several different files.

Using Regular Expressions

Introduction to Regular Expressions

Regular Expressions is a pattern specification method for search and exchange commands that uses regular character strings and metacharacters to define the patterns to be matched. Metacharacters are a set of non-alphanumeric characters used to develop generalized patterns that match a wider variety of items than is possible with mere literal pattern matching.

There are two types of pattern-matching available: Regular Expressions OFF matching and Regular Expressions ON matching. Both types are available for use with the EDIT/1000 commands B and F (Find), G and X (Exchange), and Y (Exchange and Search). Regular Expressions OFF matching satisfies the need for a fast, simple system of pattern matching. Regular Expressions ON matching provides a complicated but more powerful method of pattern matching.

When the Regular Expressions option is set to OFF, EDIT/1000 allows the use of literal pattern strings (described in more detail following), and patterns that incorporate the caret (^) and wildcard (@) metacharacters. When Regular Expressions are enabled (the option is set to ON using the Set command), EDIT matches literal pattern strings, as well as pattern strings consisting of one or more of the 15 metacharacters listed in Table 5-1 and Table 5-2.

Table 5-1. Metacharacters for Standard Pattern Matching

Metacharacter	Description
^	The anchor character. Use at the beginning of a pattern string to match only those lines where the pattern occurs at the beginning of the line.
@	The indefinite, or *wildcard* character. Use to match any pattern string, making the shortest match possible in the line.

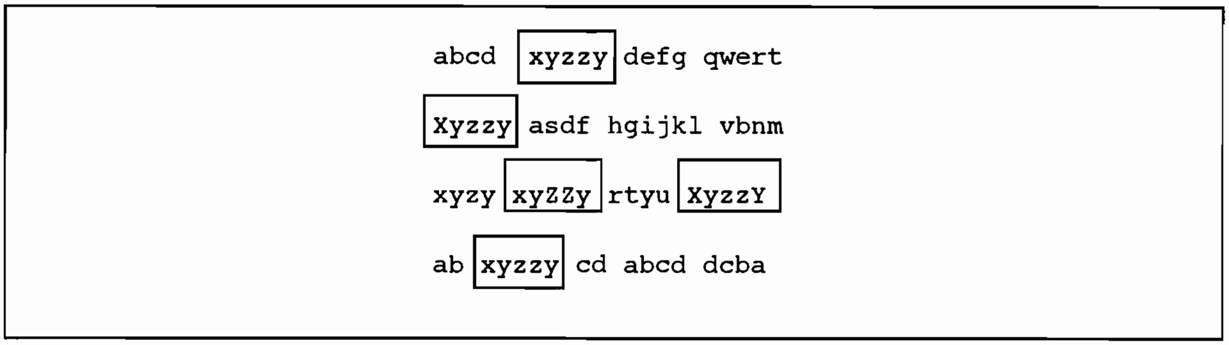
Table 5-2. Metacharacters for Regular Expressions ON Mode

Metacharacter	Description
.	The single-character wildcard. Use to match any character except the end of a line.
^	The beginning anchor. Use to anchor search to the beginning of the line or search window.
$	The end anchor. Use to anchor search to the end of the line or window, whichever is shorter.
[xyz]	Square brackets. Defines a class of characters. Use to find any of the enclosed characters.
[^xyz]	Square brackets and caret. Negates a class of characters. Use to find all but the enclosed characters.
*	Use to match zero or more occurrences of preceding pattern.
+	Use to match one or more occurrences of the preceding pattern.
<n>	Angle brackets. Use to match a specific number (n) of occurrences of the preceding pattern.
@	The indefinite or wildcard character. Use to match zero or more occurrences of any character (short for .*).
:	The alphanumeric transition character. Use to delimit a pattern string so that it matches only that string in the file (as offset by spaces or punctuation) and not the string within other alphanumeric strings. (Example: use :and: to find the word *and* but not *band*.)
{xyz}	Tagged string braces. Use to indicate a string to be recalled in a different location on the line. New location is indicated by &n in the substitute string of the exchange command.
&n	Tagged string recall. Use on the substitute side of an exchange command to recall a tagged string (see above) in a new position (the n'th position) of the line. 1 <= n <= 9. '&' by itself recalls the entire original string.
>n	Tagged string recall in uppercase. Same as &n but shifts recalled string to uppercase.
<n	Tagged string recall in lowercase. Same as &n but shifts recalled string to lowercase.
<$>	Line break character. Use to break a line into two lines at the position of the metacharacter.

Literal Matching

When EDIT/1000 is using literal matching, any literal character in a pattern matches that same character in the text being scanned. A sequence of literal characters, such as *XYZZY*, matches text in a scanned line if the line contains the same sequence.

For example, the command *F/xyzzy/* would match the characters surrounded by boxes in the following text:



```
   abcd  xyzzy defg qwert
Xyzzy asdf hgijkl vbnm
   xyzy xyZZy rtyu XyzzY
   ab xyzzy cd abcd dcba
```

Figure 5-1. Example of Literal Matching

The first line illustrates that the literal characters are matched at any position on the line. The second line shows that both upper- and lowercase versions of alphabetic characters match the pattern. (EDIT provides a Case Folding option that can be set so that EDIT searches are case sensitive using the *SE CF OFF* command.)

The starting characters on the third line do not match the pattern string because the fourth characters of the pattern and text are different. The highlighted matches on the third line show that pattern matching is case insensitive at all positions, not just the first letter of the pattern or text.

The fourth line illustrates that matching occurs at any character boundary. The matched characters do not need to be surrounded by spaces or other punctuation. However, no pattern can match text that extends across a line boundary.

Escape Character

EDIT/1000 also provides a provision for the literal matching of a character that also happens to be a metacharacter. The backslash (\) is the *escape* character that is entered before a metacharacter to indicate to EDIT that the character is to be literally matched. The escape character is an option that can be changed using the SE (Set) command. For example, to specify a literal @ sign, enter \@. For a literal backslash, use \\.

Shortest- and Longest- Possible Matching

The choice of whether to use search and exchange commands with Regular Expressions ON or OFF is determined by your needs. If you have a simple pattern to match or exchange and prefer to do it quickly, Regular Expressions OFF is definitely the best choice. However, if you require more complex matching or a complicated exchange, you will benefit from the flexibility and power of Regular Expressions ON matching.

One major difference between Regular Expressions OFF and ON is the speed with which EDIT scans text lines for matches to the pattern. When Regular Expressions are OFF, EDIT matches the shortest possible sequence of the pattern and is therefore much faster. When Regular Expressions are ON, EDIT conducts repeated scans of the text line to find the longest possible sequence of the pattern. This becomes most critical when the @ wildcard character is used to match more than one text character. For example, consider the text line:

```
Yes, we have no bananas.
```

The command *F/a@a/* with Regular Expressions OFF (shortest possible match) would match at the following sequences:

```
Yes, we have no bananas
```

The same command with Regular Expressions ON (longest possible match) would match this sequence:

```
Yes, we have no bananas
```

Exchange Command Characters

Several of the metacharacters are used for patterns specified in exchange commands. Within the exchange commands (G, X, and Y) you specify both the pattern to be matched and the character string to be substituted. The metacharacters ., ^, $, [], [^], *, +, <n>, @, :, and {} are used on the pattern match side of an exchange command. The characters &n, >n, and <n are used on the substitute side of exchange commands.

Regular Expression Length

Regular Expression operators are limited in length to 256 characters. If the -L:n run string option is used to increase EDIT's line length beyond the default 256 characters, there are some limitations on the use of Regular Expressions. Regular Expressions are limited to 256 characters, both when specifying the Regular Expression and when scanning or substituting text. During specification, if the Regular Expression is longer than 256 characters, or if its interal form exceeds the internal buffer size, EDIT will report an error with its normal "?^" prompt. The up arrow will point at the end of the Regular Expression. During substitution, if the text being manipulated exceeds 256 characters, EDIT will truncate the text without warning.

It is possible for Regular Expressions to operate on a subset of long lines, without truncation, by using the set window columns (SE WC) command. The limitation is that the width of the window column must be less than 256 characters. The window column can start at any position on the line; characters before or after the window column are not scanned and remain unchanged during Regular Expression substitution. For example, to remove leading digits on long lines, provided there are no more than 256 leading digits, use:

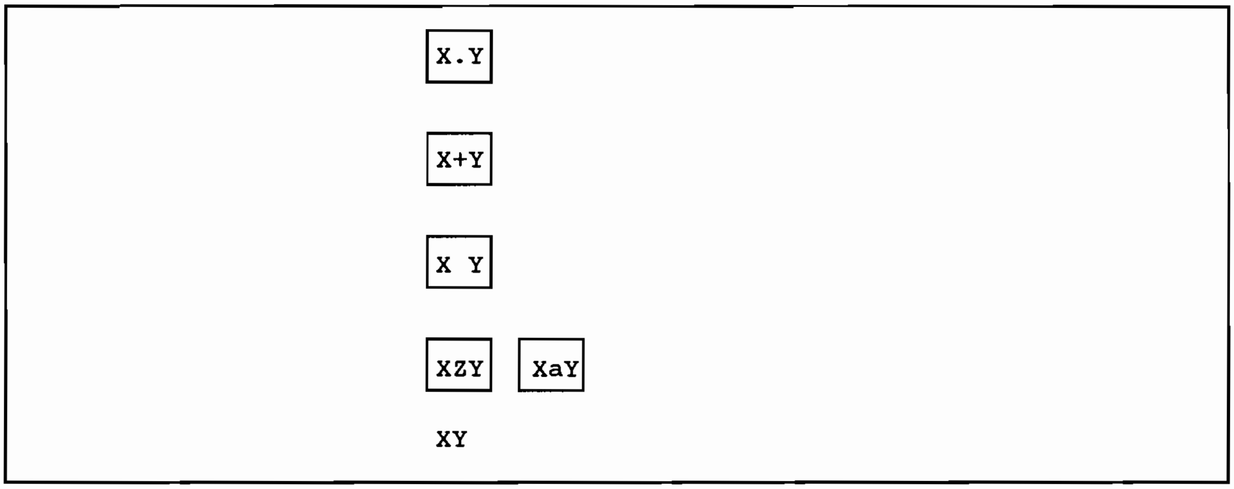
```
/SE RE ON
/SE WC 1 256
/1,$X/^[0-9]+//
```

Metacharacter Descriptions

A description and examples of each of the metacharacters follows.

The Dot (.) Character

The dot character is used with Regular Expressions enabled (ON) as a single-character wild card. When this character is included in a pattern, any single text character located in the dot's position is matched. For example, the command *F/x.y/* matches the boxed items in the following text:



```
X.Y

X+Y

X Y

XZY  XaY

XY
```

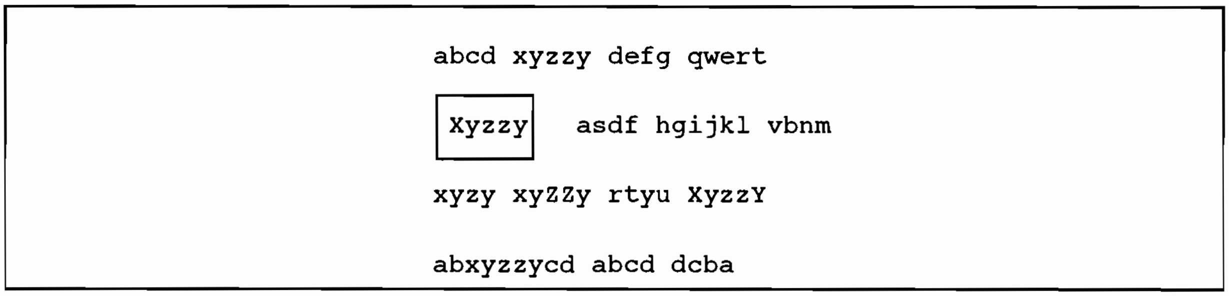
Figure 5-2. Matches to the Regular Expressions Command F/x.y/

Before this pattern matches text, there must be a group of three characters on a line. The first character in the group must be an X, and the third must be a Y. However, the center character can be any character. The preceding example shows that the dot metacharacter matches the character dot (.), the character plus (+), the character space ( ), and the letters Z and a. The last line does not match because there is no character between the X and Y.

The Beginning Anchor Character (^)

The caret (^) is used to anchor a search to the beginning of a line or search window. To anchor a pattern, the caret must be the *first* character in the pattern. If the caret occurs in some other position of the pattern, it becomes a literal caret, or a character class negation (see section following).

The simplest use of the beginning anchor character is to start a pattern search at the beginning of each line. For example, the command *F/^xyzzy/* matches the boxed items in the following text:



```
abcd xyzzy defg qwert

Xyzzy   asdf hgijkl vbnm

xyzy xyZZy rtyu XyzzY

abxyzzycd abcd dcba
```

Figure 5-3. Matches to the Regular Expressions Command F/^xyzzy/

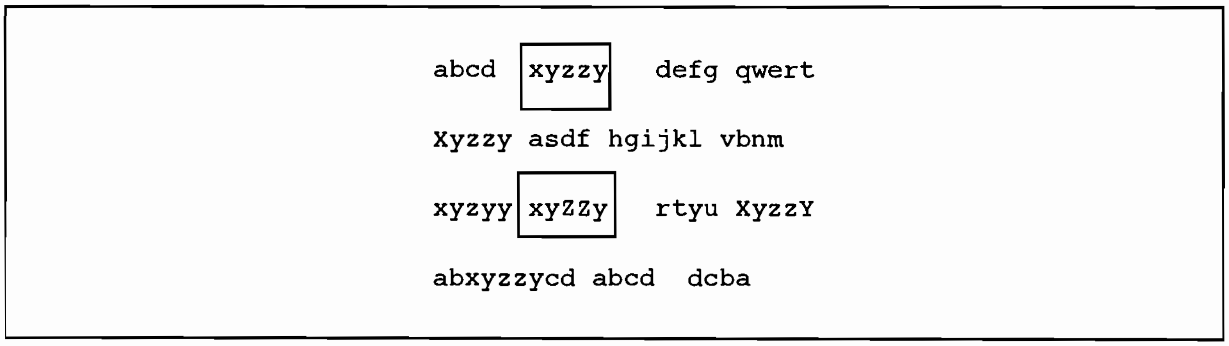
The only string that matches the pattern is the one that occurs at the beginning of the line.

The anchor metacharacter does not anchor the pattern search to the start of a line, but rather to the start of the search window. The search window is an EDIT option that enables you to define a horizontal window for pattern searches. This option allows you to match patterns starting at any specific column, not just column one. To set a new search window (the default is column 1 through column 256), use the *SE WC n n* command (Set Window Column to column range).

To match the string *XYZZY* starting in column six, enter (note EDIT's system response on line two, confirming the command):

```
/ se wc 6
  Search window columns...WC =  6  256
/ f/^xyzzy/
```

This find command matches the boxed items in the following text:



```
abcd  xyzzy   defg qwert

Xyzzy asdf hgijkl vbnm

xyzyy xyZZy   rtyu XyzzY

abxyzzycd abcd  dcba
```

Figure 5-4. Matches to the Regular Expressions Command F/^xyzzy/ with WC=6

For the first line that contains a match, there is a space in column 5 and the text starts in column 6. The second match is a portion of a string, but only the portion beginning in column 6 is matched.

Once a search window is set, it remains in effect for all searches and exchanges until it is reset or the EDIT session is terminated. If the search window is not reset to the default and the search is unsuccessful, EDIT displays the search window setting after all subsequent searches/exchanges that do not have any matches. This serves as a reminder in case you have forgotten to reset to the default. If it is not reset, the search is successful, and it used the All (A) option to locate all occurrences, the search window setting is displayed. If the search is successful but the A option is not used, the window setting is not shown.

The caret metacharacter has the same meaning whether the Regular Expressions option is set to ON or OFF. The caret (^) and the indefinite character @ are the only metacharacters that can be used for both Regular Expressions ON and OFF pattern definition.

The End Anchor Character ($)

The dollar sign metacharacter ($) is used (with the Regular Expressions option set to ON) at the end of a pattern to anchor the match to the end of the line. A dollar sign in any position other than at the end of the pattern is a literal dollar sign and matches a dollar sign in the text of the file. For example, the command *F/xyzzy$/* would match the boxed item in the following text:

```
abcd xyzzy defg qwert

Xyzzy asdf hgijkl vbnm

xyzy xyZZy rtyu [XyzzY]

abxyzzycd abcd dcba
```

Figure 5-5. Matches to the Regular Expressions Command F/xyzzy$/

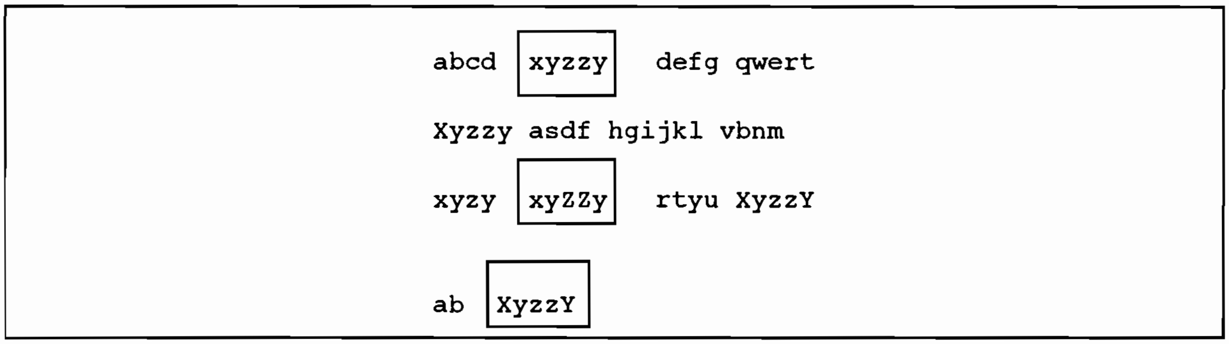
There is only one match in the text sample because there is is only one line that contains a match to the pattern that occurs at the end of the line.

Similar to the caret (^) character, the dollar sign does not match the end of the line but rather the end of the search window. However, the situation is somewhat more complicated for matches anchored to the end of the search window. The end of the line may be to the left or the right of the end of the search window. For such situations, the dollar sign anchors the match to the smaller of the two. If the text line ends at column 50 and the right limit of the search window is at column 150, the dollar sign anchors the match to the end of the line at column 50.

For example, the following command:

```
/ se wc 1 10
  Search window column...WC =  1  10
/ f/xyzzy$/
```

matches the boxed items in the following sample text (note that the fourth line of the sample text has been modified for this example):



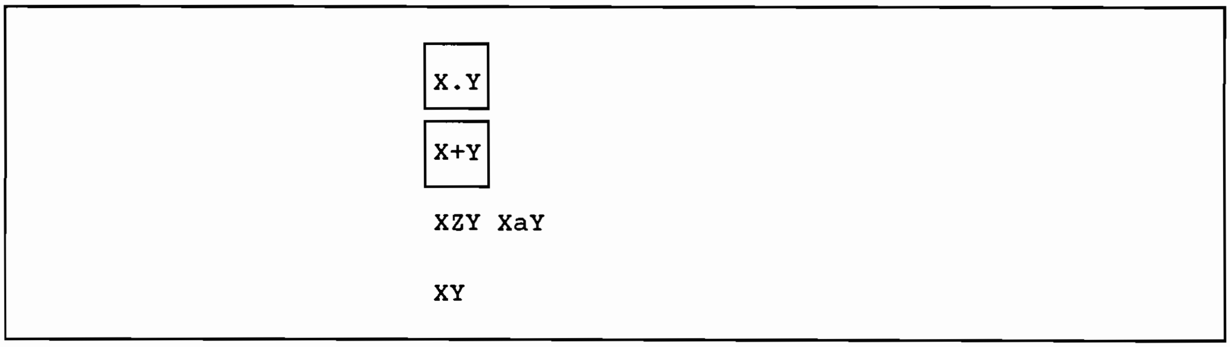
```
abcd  xyzzy  defg qwert

Xyzzy asdf hgijkl vbnm

xyzy  xyZZy  rtyu XyzzY

ab  XyzzY
```

Figure 5-6. Matches to the Regular Expressions Command F/xyzzy$/ with WC=1,10

The first and third lines have strings that match the pattern and end in the same column as the end of the search window. The string on the fourth line is a match to the pattern because it occurs at the end of the line.

Character Class Brackets [xyz]

When the Regular Expressions option is set to ON, a search or exchange pattern set can be used to match any single text character from that set. These pattern sets are referred to as Character Classes. Alphanumeric and special characters to be included in a character class are enclosed in left and right square brackets. This indicates that EDIT is to match any of the characters included between the brackets. When a character class is included in a pattern, a text character occurring in the corresponding position matches any one of the characters from the bracketed class. For example, the command *F/x[.+–]y/* matches the boxed items in the following sample text:



```
X.Y
X+Y
XZY XaY

XY
```

Figure 5-7. Matches to the Regular Expressions Command F/x[.+–]y/

The command directed EDIT to match the letter X, followed by either a dot, plus, or minus character, and then the letter Y. A match is located on the first and second line, however, the third line does not match because the *Z* and *a* do not match any of the characters in the . – + character class. The last line does not match because there is no character between the X and Y.

Within the brackets of a character class, there are only three characters that have special meaning. They are the caret (^), the minus sign (–), and the closing square bracket (]). The caret has special meaning only if it is the first character after the left bracket. This meaning is described in the following section on Negated Character Class.

The minus sign specifies a range of characters if it is between digits or the same case letters (if CF=off). For example, 0-9 refers to the range of zero through nine, and matches any single number within that range. In the previous example, the minus sign does not have a special meaning because there is a plus sign before it.

Left square brackets ([) within a character class are literal left square brackets. A right square bracket (]) outside of the character class is a literal right square bracket. If there is not a closing right square bracket, the character class is terminated by the end delimiter of the pattern. For example, the command

```
/1$ X/^[CD[*/C/
```

performs exchanges on all lines that start with any of the four characters C, D, [, or *, and replaces the pattern with a C.

To expand on the explanation of character class ranges, note that the pattern

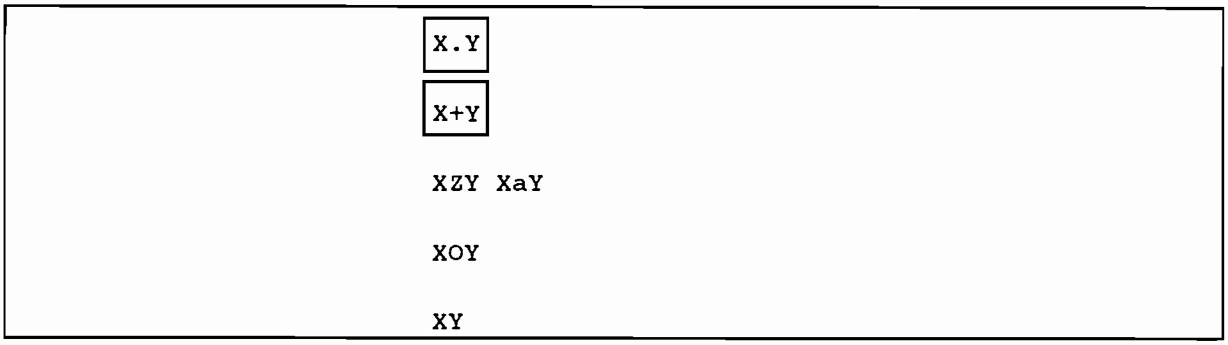
```
[a-z0-9]
```

matches any character in the range a through z or in the range 0 through 9. Character classes are an OR function; if any *one* of the members of the character class set matches the corresponding character in the line of text, then the character class has been matched.

Negated Character Class Brackets [^xyz]

Character class negation allows you to specify the characters that you do *not* want to be matched. A character class is negated by including an caret (^) as the first character after the left square bracket. EDIT then matches any line except those that contain the characters in the negated character class.

For example, the command *F/x[^ A-Z0-9]y/* matches the boxed items in the following sample text:



```
X.Y

X+Y

XZY XaY

XOY

XY
```

Figure 5-8. Matches to the Regular Expressions Command F/x[^ A-Z0-9]y/

The character class specifies characters that are non-alphabetic and non-numeric. The first two lines match because the dot and plus sign between the X and Y fit the requirement. The third and fourth lines do not match because there are alphabetic and numeric characters between the X and Y. The fifth line does not match because there is no character between the X and Y.

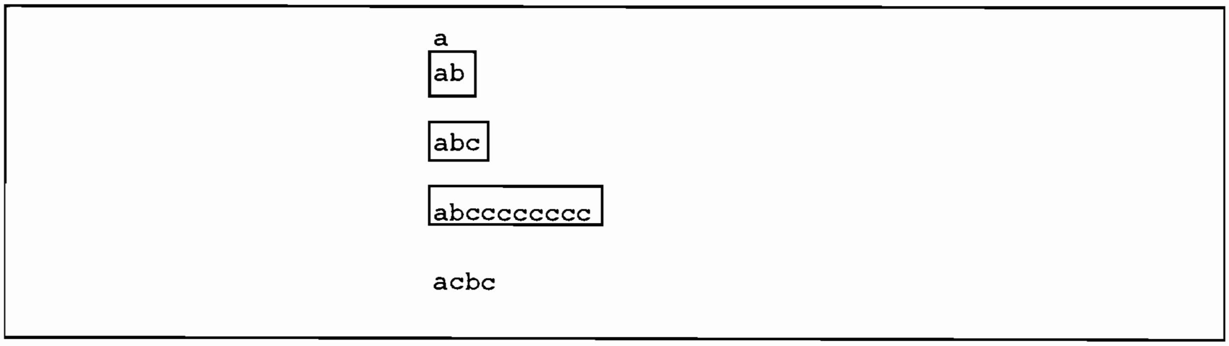
A caret (^) in a character class that is not immediately after the left square bracket is the caret itself, and not a negation of the character class. To specify a character class which consists of only a caret, you can use the pattern

```
[\^]
```

However, this is overkill, because a character class that contains only one character is equivalent to entering the character itself without the brackets. Simple is better, and EDIT is faster at matching a literal character than matching a character class of one.

Zero or More Match Character *

The asterisk (*) is used (with Regular Expressions ON) to match zero or more occurrences of the preceding character or character class. For example, the command *F/ABC*/* matches the boxed items in the following sample text:

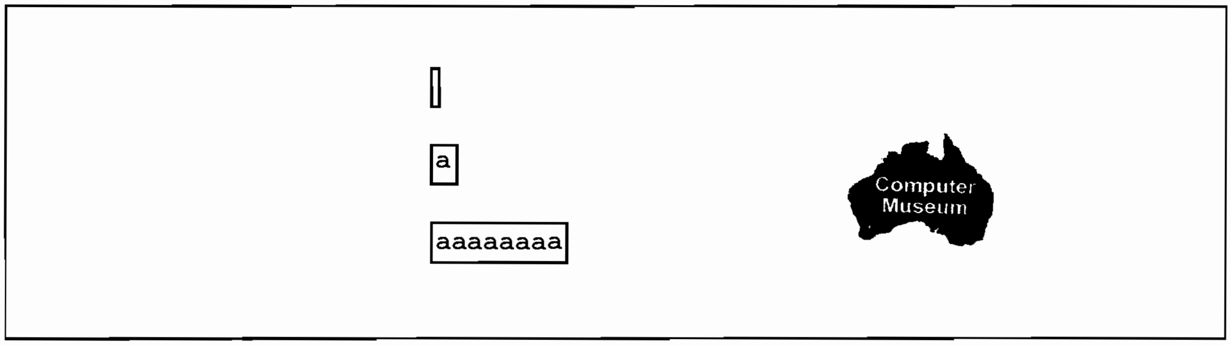


```
a
ab

abc

abccccccccc

acbc
```

Figure 5-9. Matches to the Regular Expressions Command F/ABC*/

The asterisk affects only the preceding character or character class; in this case the C. The pattern specifies a match to lines containing an A, followed by a B, followed by zero or more occurrences of the letter C. The first line does not match because it does not contain the literal character B. The second, third, and fourth line match. The fifth line does not match because the A and B are not next to each other.

The command *F/A*/* matches the boxed items in the following sample text:



```

a
aaaaaaaa
```

Figure 5-10. Matches to the Regular Expressions Command F/A*/

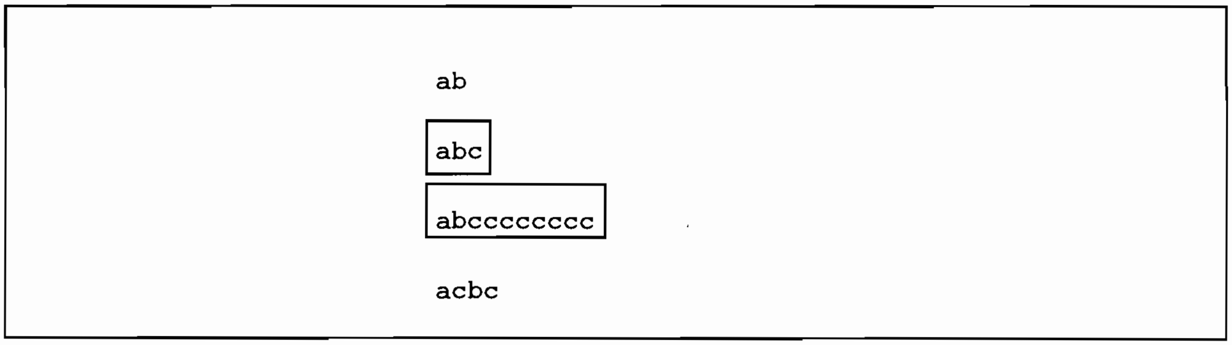
The pattern requires a match to zero or more occurrences of the letter A. The first line, a null (blank) line, matches because it contains zero occurrences of A. The second and third lines match because they contain one or more A's.

One or More Match Character +

The zero or more occurrence (*) metacharacter often matches more than is expected. The plus sign (+) metacharacter can be used to match one or more occurrences of a specified character or character class. For example, the command

```
/f/abc+/
```

matches the boxed items in the following text:



```
ab
abc
abcccccccc
acbc
```

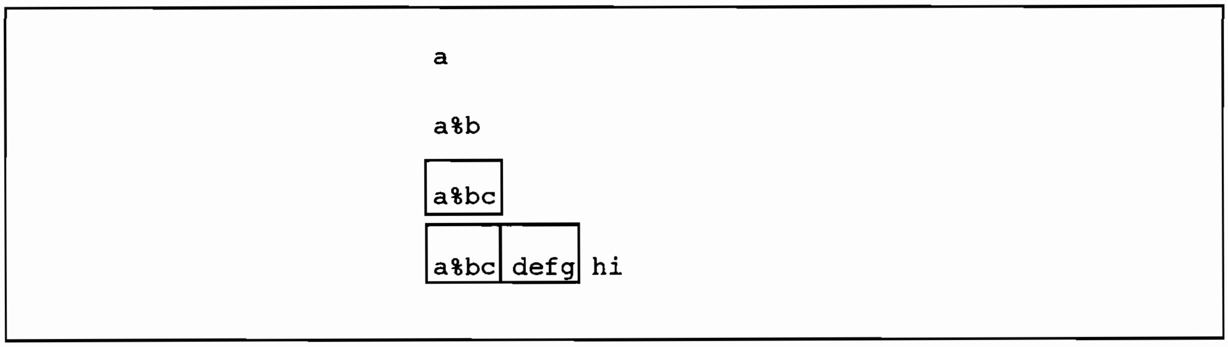
Figure 5-11. Matches to the Regular Expressions Command F/abc+/

In the preceding example, the line with just the AB does not match because the pattern requires there to be at least one C. The match can extend to many C's, as shown by the fourth line.

Note that use of the plus metacharacter has the same effect as entering the character or character class twice, followed by a asterisk. However, the plus is easier to type.

Pattern Repeat Angle Brackets <n>

The angle brackets (<n>) are used to match a specific number of occurrences of a preceding character or character class. For example, the command *1 $ F/.<4>/* would match the boxed items in the following text:

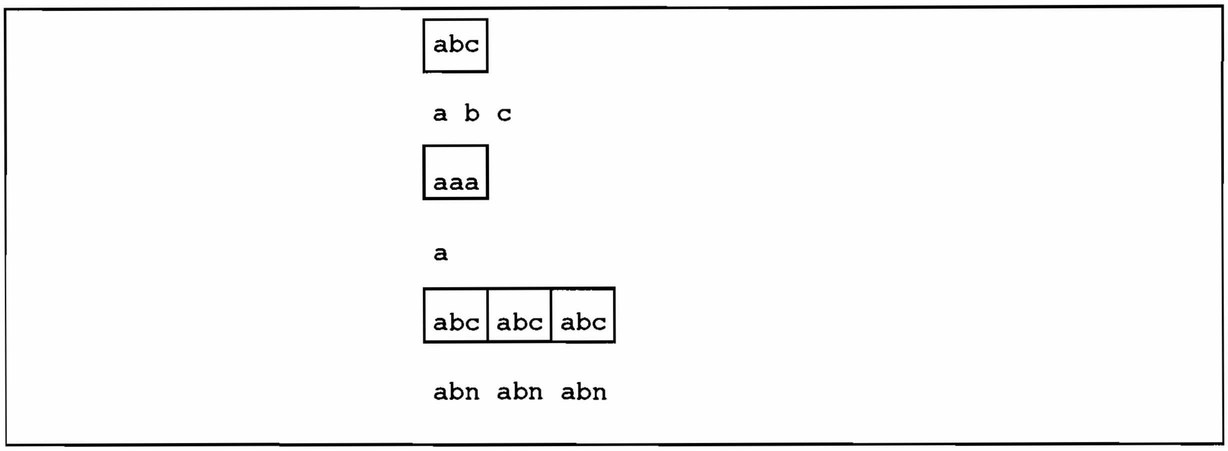


```
a

a%b

a%bc

a%bc defg hi
```

Figure 5-12. Matches to the Regular Expressions Command 1 $ F/.<4>/

The pattern specifies to match four occurrences of any character. The first and second lines do not match because there are not enough characters. The third and fourth lines match because there are at least four characters to match on each line.

The command *1 $ F/[abc]<3>/a/* matches the boxed items in the following text:



```
abc

a b c

aaa

a

abc abc abc

abn abn abn
```

Figure 5-13. Matches to the Regular Expressions Command 1 $ F/[abc]<3>/a/

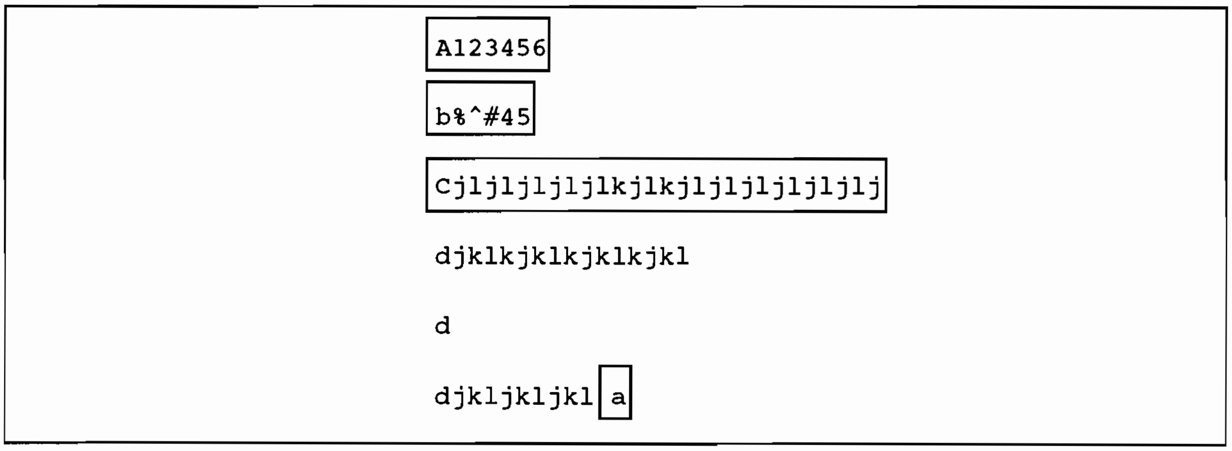
In the preceding example, the first and third lines match because they contain three occurrences of the characters specified in the character class. The second line does not match because there are blanks between the characters specified in the character class. No blanks are specified in the class. The fourth line does not match because it contains only one character, and the pattern specified three occurrences. The fifth line matches the pattern since it contains three occurrences of the characters specified, one right after the other. The sixth line does not match because there are n's and spaces between three possible occurrences of a pattern match. If the sixth line had read abb abn abn, it would have matched.

Indefinite Character @

The metacharacter @ is the indefinite, or wildcard character used in Regular Expression patterns. It is used to match zero or more occurrences of any character.

When an @ is first encountered in a pattern, EDIT matches everything up to the end of the line. Then the rest of the pattern is compared to the remainder of the line to ensure that the line contains a complete match to the pattern. Thus, the rest of the pattern requires EDIT to shorten the text matched by the @ metacharacter. This elimination of characters matched by the @ is repeated until the pattern matches or until there are no characters matched by the @. EDIT backtracks over the entire line for each portion of the pattern following the @. This is referred to as *longest possible matching*, and it is described in more detail at the beginning of the Regular Expressions section. If more than one @ is used in a pattern, the matching process takes longer to complete, since a full set of backtracking is done for each successive @ every time a single character backtracking is done for the first @.

For example, the command *1 $ F/[abc]@/a/* matches the boxed items in the following text:



```
A123456

b%^#45

Cjljljljljlkjlkjljljljljlj

djklkjklkjklkjkl

d

djkljkljkl a
```

Figure 5-14. Matches to the Regular Expressions Command 1 $ F/[abc]@/a/

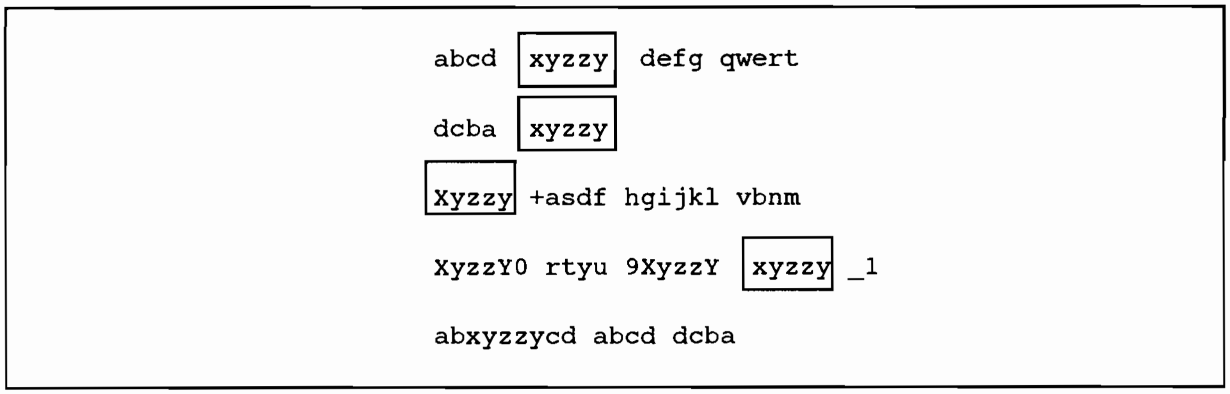
The pattern requires that the lines matched should consist of a, b, or c, followed by zero or more occurrences of any character. The first, second, and third lines match since they all consist of a, b, or c followed by zero or more other characters. The fourth and fifth lines do not match because they do not contain one of the letters specified in the pattern. The sixth line matches because it does contain one of the letters specified in the character class, followed by zero occurrences of other characters.

Alphanumeric Transition Character :

It is useful to have the ability to match a *word* of text. A word here is defined as a string of characters that is delimited by punctuation characters, such as spaces, commas or parentheses. To illustrate, suppose you want to find all occurrences of the word *and*. Using the command *F/and/* would find the word *and*, but would also find the words *hand*, *command*, and *grand*.

To avoid finding words that contain the word *and*, EDIT provides the Alphanumeric Transition metacharacter (:) for use as a delimiter for the desired string. The alphanumeric transition character specifies that no match is to be made when alphanumerics (A through Z and digits 0 through 9) occur immediately before or after the desired string. Everything else (punctuation, blanks, etc.) is matched. The alphanumeric transition metacharacter matches the zero length string between alphanumeric to punctuation characters. It also matches the zero length string between the start of each line and the first character, as well as the one at the end of each line.

For example, the command *F/:xyzzy:/* matches the boxed items in the following text:



```
abcd  xyzzy  defg qwert

dcba  xyzzy

Xyzzy +asdf hgijkl vbnm

XyzzY0 rtyu 9XyzzY  xyzzy _1

abxyzzycd abcd dcba
```

Figure 5-15. Matches to the Regular Expressions Command F/:xyzzy:/

The first line in the preceding example illustrates that a match can occur in the center of a line. The second shows that it can occur at the end of a line. The third line illustrates that a match can occur at the start of a line, and that a plus sign (+) is considered a non-alphanumeric.

In the fourth line, the first occurrence of *xyzzy* does not match because it is followed by the digit zero, and the second occurrence does not match because it is preceded by the digit nine. The third occurrence on the fourth line is a match since it is followed by an underscore (_).

The fifth line illustrates that the alphanumeric transition metacharacter is useful for building patterns that match complete words, and do not match sub-strings within words.

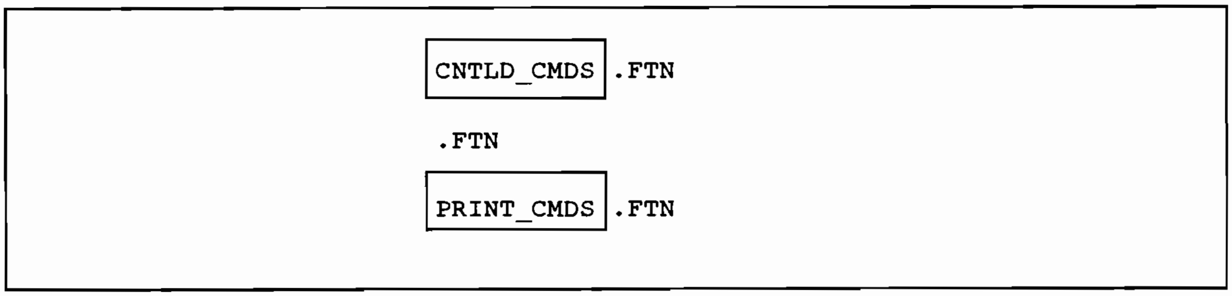
The alphanumeric transition metacharacter can be used to find occurrences of variable names in source code, unless the underscore is used as part of a variable name.

Tagged String Braces {xyz}

EDIT Regular Expressions provides a notation for exchange commands that enables you to identify a string or field that is to be switched or moved on the line. The string to be moved is referred to as a *tag field* or *tag string*, and right and left curly braces are used to enclose it in the pattern. For every left curly brace, there must be a matching right curly brace.

Within the exchange commands (G, X, and Y) you specify both the pattern to be matched and the character string to be substituted. The tagged string curly braces are used on the pattern match side of an exchange command. The characters *&n*, *>n*, and *<n* are used on the substitute side of the exchange command to indicate where and how the tagged string is to be recalled on the newly edited line. (See the Tagged String Recall metacharacters, following.)

For example, the command *1 $ F/^{[^.]+}/* matches the boxed items in the following text:



```
CNTLD_CMDS .FTN

.FTN

PRINT_CMDS .FTN
```

Figure 5-16. Matches to the Regular Expressions Command 1 $ F/^{[^.]+/

The pattern indicates that EDIT is to find the longest possible string that begins in column one of the line and does not contain a dot. It is to find at least one and as many as possible, non-dot characters.

The first and third lines match the pattern because they consist of one or more non-dot characters that begin at column one of the line. The second line does not match, since the line begins with a dot.

Tagged String Recall &n

The &n notation, where n is a position number, is used on the substitute side of an exchange command to retrieve a tagged string enclosed in a set of curly braces. The position number used corresponds to the order of tagged string sets on the pattern match side of the exchange command. For example, use the command *1 $ X/^{....}@/&1.LST/* to change the following file names from the way they appear in the first paragraph of Figure 5-17 to the way they appear in the second paragraph.

```
DRAW.FTN
TIME.MAC
CTRL.ADA
COPY.FTN


DRAW.LST
TIME.LST
CTRL.LST
COPY.LST
```

Figure 5-17. Matches to the Regular Expressions Command 1 $ X/^{....}@/&1.LST/

On the substitute side of the command, the &1 recalls the tagged field that contains the first four characters of each line. This field is followed by the string *.LST.* It is important to note that the same tagged string can be used more than once on the substitute side of the exchange command.

Upper- and Lower-Case String Recall >n <n

If desired, a tagged string can be retrieved in upper- or lowercase. This is accomplished using the notation >n or <n, where n is the position of the tagged string on the pattern match side of the command.

For example, the command *1 $ X/^{[^.]+}\.{.<5>}/>2.&1/* changes the text in the first paragraph below to the order in the second paragraph below:

```
ab222.radm8
mr24354.radm7
gh3459809809098.radm3
dm3.radm2


RADM8.ab222
RADM7.mr24354
RADM3.gh3459809809098
RADM2.dm3
```

Figure 5-18. Matches to the Regular Expressions Command 1 $ X/^{[^.]+}\.{.<5>}/>2.&1/

Break Line Character <$>

The break line character <$> can be used on the substitute side of an exchange command to insert a new line at the point it occurs. There may be more than one <$> in the substitute pattern and the tagged string recall can be used on either side of it. If there is a search window in effect when the exchange is executed, text outside the window is copied onto each created line.

For example, *1$X/^{.<20>[^]*}{@}/&1<$>&2* breaks lines at the first space after column 20 in the following text:

```
This is a line with more than 20 characters

This is a line with
 more than 20 characters.
```

Figure 5-19. Matches to the Regular Expressions Command 1$X/^{.<20>[^]*}{@}/&1<$>&2

Creating Patterns

This section describes how EDIT reads (parses) a command and the effect this has on the results of the search/exchange. It also supplies examples of patterns created to accomplish specific tasks.

Pattern Parsing

EDIT reads patterns in the same way that it reads commands: from left to right. This process is referred to as *parsing,* and it is necessary to understand this process before effective patterns can be created. The order used by EDIT to translate search and exchange commands follows:

1. EDIT notes the line range, if specified, then recognizes that it is a command that contains a pattern (commands B, F, ‘ ‘,’ ’, G, U, X, and Y).

2. EDIT identifies the pattern by noting the first non-blank, non-alphanumeric character after the command mnemonic, and uses that character as the pattern delimiter.

3. EDIT reads the pattern from left to right to the next delimiter (the delimiter must match the first delimiter used and must not be preceded by the escape character).

4. EDIT examines the pattern to see if it is legal. If it is not legal, EDIT indicates an error and does not finish parsing the command. To indicate an error, EDIT displays the command, placing an arrow under the end of the illegal pattern (not under the illegal character). For example:

   ```
   /1$ F/{[.<2>]@/
   ?             ^
   ```

   In the preceding example, EDIT indicates the error by placing the arrow under the end of the illegal pattern string to indicate that there is not a closing curly brace.

5. If the pattern is legal, EDIT scans and parses the rest of the command for the substitute string (if it is an exchange command) and options.

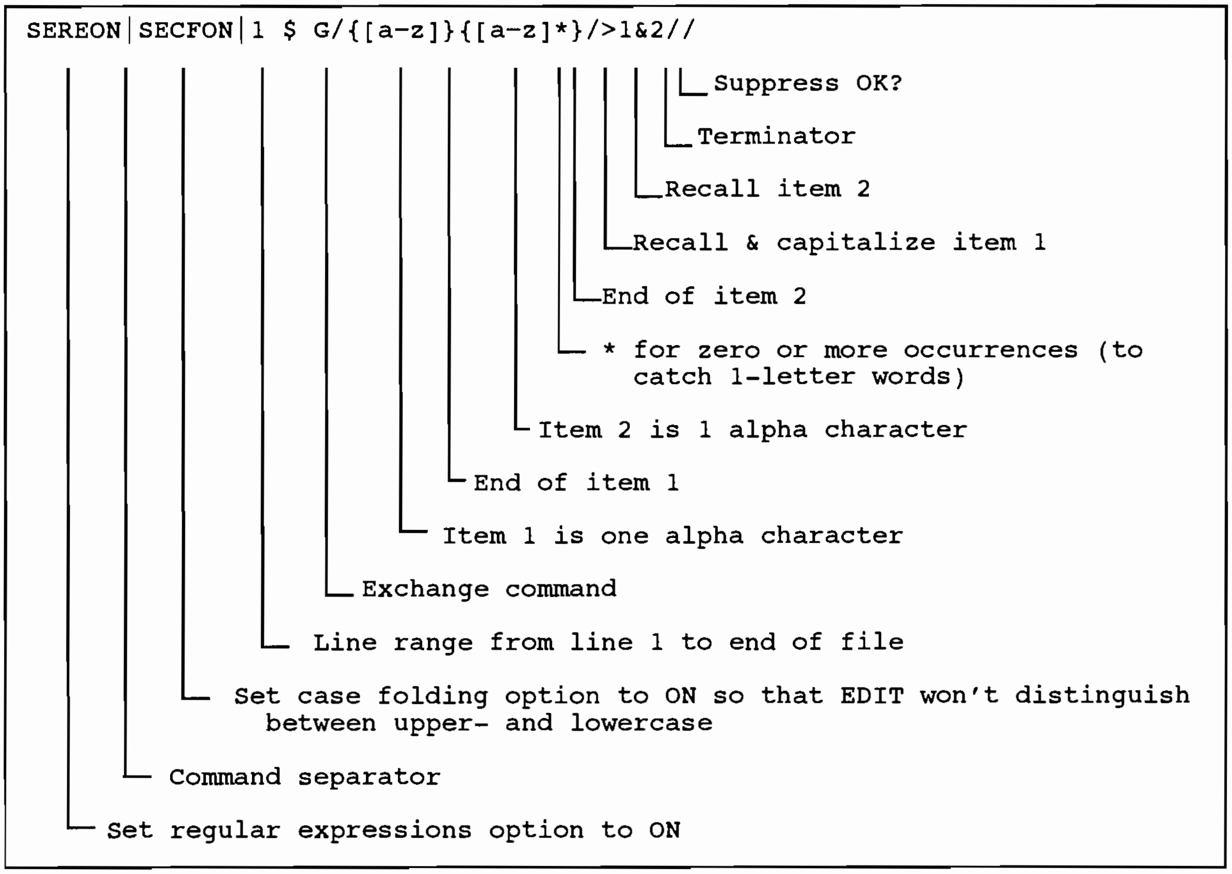
Changing Cases

To change uppercase characters to lowercase, or vice versa, use the command *SE RE on* (Set Regular Expressions ON). Then enter the first command following to change uppercase letters to lowercase, or the second command to change lower to uppercase:

```
/1 $ X/@/<

/1 $ X/@/>
```

Capitalizing First Letters of Words

To capitalize the first letter of all words in a file of all lowercase characters, and suppress asking (suppress the OK? prompt), enter the following command:



```
SEREON|SECFON|1 $ G/{[a-z]}{[a-z]*}/>1&2//
```

Suppress OK?

Terminator

Recall item 2

Recall & capitalize item 1

End of item 2

* for zero or more occurrences (to catch 1-letter words)

Item 2 is 1 alpha character

End of item 1

Item 1 is one alpha character

Exchange command

Line range from line 1 to end of file

Set case folding option to ON so that EDIT won't distinguish between upper- and lowercase

Command separator

Set regular expressions option to ON

Figure 5-20. Capitalizing First Letter of All Words in a File

Add Text to Variable Length Lines

To add text to the end of lines in the file that have varying lengths, (including blank lines) enter the command

```
SEREON|1 $ G/$/text/Q/
    | ||   | | |    |└ Terminator
    | ||   | | |    └ Suppress listing of change
    | ||   | | └ Enter desired text
    | ||   | └ To exchange at the end of the line
    | ||   └ Exchange command
    | |└ Line range from line 1 to end of file
    | └ Command separator
    └ Set regular expressions option to ON
```

Figure 5-21. Adding Text to the End of Lines with Varying Lengths

To add text to the end of all lines *except* the blank lines, enter the command

```
SEREON|1 $ G/.$/&text/Q/
```

Insert at a Specific Column

To insert a character at a specific column (column 40) throughout the file, enter the following command:

```
SEWC40|1 $ U//character/Q/SEWC
   |   |   | |    |     |   └ To return to default window settings
   |   |   | |    |     └ Suppress listing of change
   |   |   | |    └ Character to insert at column 40
   |   |   | └ The // indicates replace with no characters
   |   |   └ Unconditional exchange command
   |   └ Line range from line 1 to end of file
   └ Set window column option to start at column 40
```

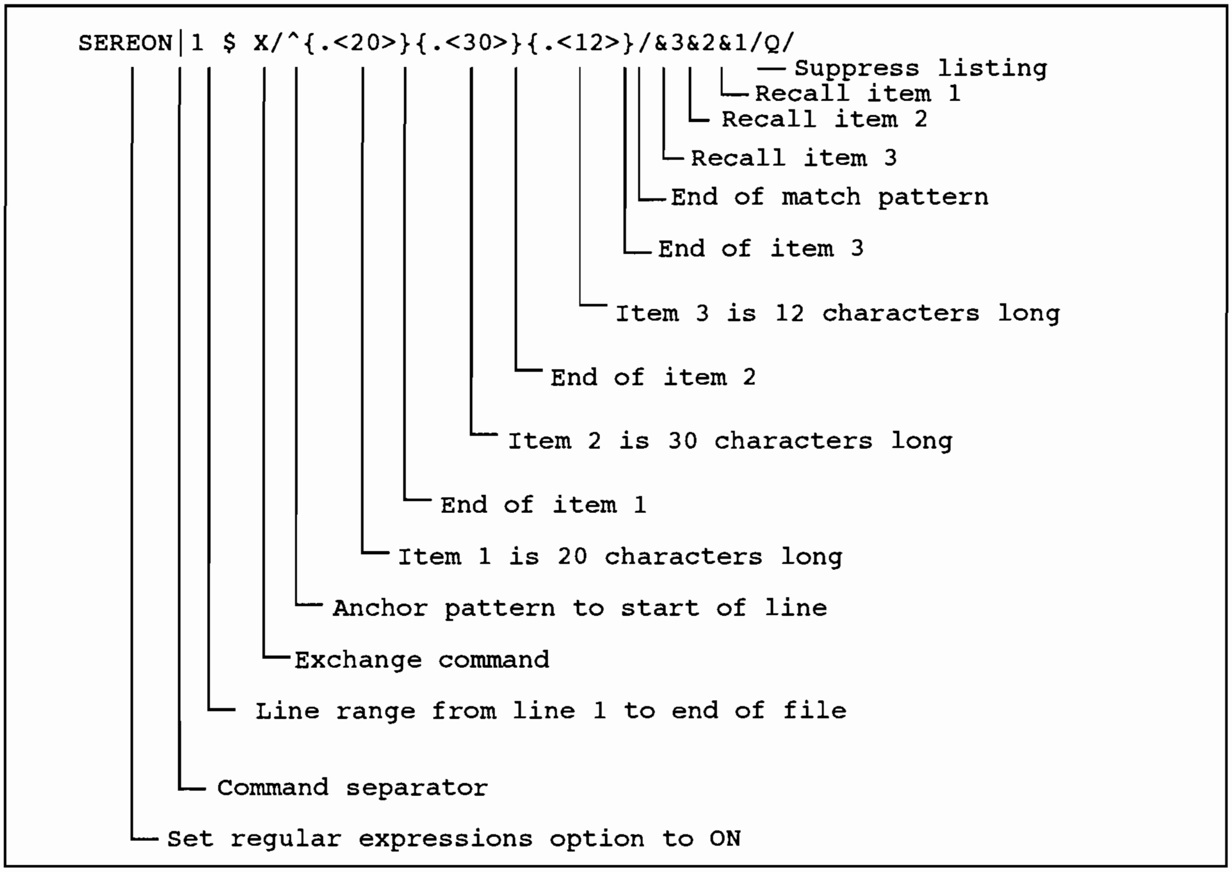
Figure 5-22. Inserting a Character at a Specific Column

To insert a character at column 40 on all lines except those shorter than 40 characters, enter the following command:

```
SEREON|1 $ X/^.<40>/&character/
```

Move Columns of Data

To switch or move columns of data (for example, switching column 1 with column 2, where column 1 is figures and column 2 is descriptions), enter the following command:



```
SEREON|1 $ X/^{.<20>}{.<30>}{.<12>}/&3&2&1/Q/
```

Suppress listing
Recall item 1
Recall item 2
Recall item 3
End of match pattern
End of item 3
Item 3 is 12 characters long
End of item 2
Item 2 is 30 characters long
End of item 1
Item 1 is 20 characters long
Anchor pattern to start of line
Exchange command
Line range from line 1 to end of file
Command separator
Set regular expressions option to ON

Figure 5-23. Moving or Switching Columns of Data

Note that all lines to be moved must be at least as long as the total of all the defined field lengths (62 characters in the preceding example). Characters to the right of the defined fields are left in place while shorter lines are not changed because they do not match the pattern. Text or blanks can be inserted between the columns by inserting the desired text/blanks in the substitute string (between &3, &2, and &1 in the preceding example).

Make All Lines a Minimum Length

To make all lines in a file a minimum length (4 characters in the example following), enter the command

```
SEWC4|1 $ U///Q|SEWC
 |   ||   | | | |
 |   ||   | | | └─Reset window columns to default
 |   ||   | | |
 |   ||   | | └─ Suppress listing of changes
 |   ||   | |
 |   ||   | └─ The /// replaces nothing w/nothing; add blanks if needed
 |   ||   |
 |   ||   └─ Unconditional exchange command
 |   ||
 |   |└─ Line range from line 1 to end of file
 |   |
 |   └─ Command separator
 |
 └─ Set window column option to column 4
```

Figure 5-24. Making All Lines a Minimum Length

After the preceding example, all lines are at least four characters long. Any line shorter than four characters is padded with blanks out to column four.

Learning Exercises

To test your understanding, try developing one or more commands and match pattern to accomplish the following tasks (answers are provided on the following page):

1. Develop a search command that searches for part numbers within a file. A sample of the numbers to be located is listed below. Each part number consists of a number, followed by an alpha character, followed by three numbers, followed by two alpha characters, as follows:

   ```
   1F333AM
   3U698JG
   8N723MB
   5S979ML
   1T777JJ
   2U612JD
   3F987WE
   ```

2. List the commands you would use to add line numbers to a BASIC program.

3. For a file that includes text of telephone numbers and names that start in column 15 and do not extend beyond column 50 (as listed following), develop an exchange command with a pattern that switches the columns to list names in the first column and phone numbers in the second column. The phone numbers should line up after the switch. (Hint: first set your horizontal search window.)

   ```
   (703)525-8911  Gertrude Magillicutty
   (609)892-4356  Cliff Hanson
   (807)567-7333  Amanda Swampbreath
   (605)899-1212  Remark Able
   ```

Answers

1. One version of the command that searches for part numbers similar to those in the example, follows:

```
     1 $ F/[0-9][A-Z][0-9]<3>[A-Z][A-Z]/
```

2. For this exercise, you would first use the Sequence Number (#) command to place numbers in columns 73 through 75 (in the example, they are shifted left to fit within the right margin of the page). Then set Regular Expressions ON (SE RE on) and enter the following command:

```
     1$X/{@} {[^ ]*}/&2 &1//
```

   You would then probably want to use the Kill Blanks (BK) command to delete all trailing blanks.

3. There are many commands that accomplish the task described in exercise 3. Two possible commands are listed below:

```
     SEREON|SEWC 50
     1 $ U///Q/|SEWC
     1 $ X/^{.<14>}{@}/&2&1
```

   In the preceding command, the window column was set to 50 to ensure that all lines are 50 characters long (since the U command pads blanks up to column 50). Then the U command replaces nothing with nothing at column 50. Then reset the margins. The X command matched the 14-character telephone number and switched it with everything else on the line.

```
     SEREON|SEWC 50
     1 $ U///Q/|SEWC
     1 $ X/{[0-9-()]+} *{@}/&2&1
```

   The second version of the command again, sets the window to column 50. However, it checks specifically for telephone number syntax.

EDIT Regular Expressions and the RTE User Interface

The first portion of this chapter provided an introduction to using Regular Expressions to accomplish sophisticated pattern matching while editing. Regular Expressions can also be used in conjunction with the RTE Command Interpreter (CI). This section provides examples of using EDIT's Regular Expressions with CI to build command files and to construct summary files containing data from several different files.

Building File Manipulation Commands

The following example explains how to build a CI command file to perform a specific operation on many files within a CI directory. The operation illustrated involves compiling and printing many of the FORTRAN sources in the directory. A CI command file must be created that runs the FTN7X compiler on each file and then runs Print on each list file.

The first step in building such a command file is to create a file containing the pertinent filenames, one name per line. This is done using the DL (Directory List) command as follows:

```
CI> dl @.ftn s makelst.cmd
```

This command creates the file makelst.cmd, which contains the name of all the files in current directory that have the type extension .FTN. The S (Size) option causes DL to include the file size next to the filename. It is used here because it places one filename per line. Before issuing this command, make sure the list file does not already exist. If it does, the DL command reports a FILE ALREADY EXISTS error.

The DL command also places a few extra lines in the file that should be deleted before beginning operations on the filenames. An example of the output follows:

```
directory /EDIT/E4/
          name        blks

CNTLD_CMDS.FTN         22
CNTLD_COPY_MOVE.FTN    22
CNTLD_SUBS.FTN         77
DO_CNTLD.FTN           34
```

The next step in building the desired command file is to use EDIT to convert the list of filenames into proper CI commands. Start by entering

```
CI> edit makelst.cmd
```

When EDIT starts up, it reports several lines of information, and prompts for the first EDIT command. To delete the extra lines DL inserted at the beginning and end of the listing, and to scan to see if the list includes the files expected, enter the EDIT command

```
/1$s
```

This places EDIT in screen mode and puts the entire file on the screen (well, at least as much as comfortably fits in the terminal's memory). Then use the delete line key to delete the title lines and the blank line at the end of the file. In addition, delete all lines containing names of files are not to be compiled and printed. Use the control and U command to exit the screen mode.

Next, issue the EDIT command to turn Regular Expressions ON. The shortest form of this command is

```
/sere
```

The full SE RE ON command is not necessary because the default setting for the Regular Expressions option is OFF. When the command is issued, EDIT toggles the setting if the ON/OFF parameter is omitted from the command.

The file now contains the desired filenames, one per line. The next step involves issuing the EDIT exchange command to converts the filenames into the required CI commands. The following example illustrates the type of command that is needed:

```
ftn7x CNTLD_CMD - 0,,qc;print CNTLD_CMD.lst
```

At this point, a Regular Expressions command must be issued to match all of the characters from the start of each line up to the first dot. All the characters on the line are then replaced by the desired characters (that is, ftn7x, etc.) and the filename is put in both input filename positions. To do this, enter the command

```
/1 $ X/^{[^.]+}@/ftn7x &1 - 0,,qc;print &1.lst//
```

This command starts off specifying a line range of 1$, which is the first line in the file through the last, inclusive. The match side of this exchange command begins with the anchor character, ^. This tells EDIT to start matching at the start of the line. By anchoring the pattern, you can safeguard against matching lines that you do not want to match.

The next character used in the pattern is the left curly brace {, which specifies the start of a tag field (or an item to be reinserted in a different position on the exchanged line). A tag field consists of a group of characters from the text being searched. These characters can be used in the substitute side of an exchange command. The tagged field used in this example is the filename that is to be copied to the substitute side of the exchange. The end of the tag field is specified by the matching right curly brace }, just after the plus metacharacter.

EDIT requires that curly brace metacharacters match up. That is, for each left curly brace, there must be a matching right curly brace at the same nesting level. If curly braces are not matched, EDIT reports an error, as in the example following:

```
1 $ X/^{[^.]+\.@/ftn7x &1 - 0,,q;print &1//
?              ^
```

The caret, ^, points at the bad pattern and not at the first incorrect character.

Note also that the anchor character (^) in the preceding pattern is before the tag character ({). The caret is an anchor character only if it is the first character in the pattern. If it had been placed on the other side of the curly brace, it would have been interpreted by EDIT as the literal caret character.

Matching of the filename itself is accomplished with the next five characters in the pattern. The [^.]+ instructs EDIT to match the longest possible string not containing a dot. This pattern is a combination of three regular expression features. First, a character class is specified by the use of matching square brackets ([...]). The characters to be matched are specified using the character class negation (^) meaning anything but a dot. Finally, the plus (+) instructs EDIT to match at least one, and as many as possible, non-dot characters.

On the exchange side of the command, the literal characters ftn7x are used, followed by &1. The &1 instructs EDIT to retrieve the tagged text corresponding to the first set of curly braces ({}), and insert that text at the position of the &1 in the substituted text. As illustrated in this example, a tagged text item can be used more than once in the substitute pattern.

Note the two slashes at the end of the pattern, used to suppress the confirmation prompt, OK?.

Issuing the preceding exchange command results in the file listed as follows:

```
ftn7x CNTLD_CMDS - 0,,qc;print CNTLD_CMDS.lst
ftn7x CNTLD_COPY_MOVE - 0,,qc;print CNTLD_COPY_MOVE.lst
ftn7x CNTLD_SUBS - 0,,qc;print CNTLD_SUBS.lst
ftn7x DO_CNTLD - 0,,qc;print DO_CNTLD.lst
```

To exit EDIT, enter an ER (Exit and Replace) command. Then type MAKE.LST at the CI> prompt to obtain a printout of the listing.

Gathering Data from Files

EDIT's pattern matching and exchange capabilities can also be used to gather data from many different files into one summary table. For example, suppose you had written a simulator that produced text output files describing the results of each simulation. Each simulation run uses different input data and creates a different results file. You would like to gather several numbers from each of the results files and build them into one summary table file. You would like to time stamp the table and format the table with headers. The numbers you require are on different lines of the simulator results files.

Figure 5-25 lists the results files produced by different runs of the simulator. The naming convention used to signify a results file is the starting equal sign. The next three letters describe the input data set and the last two letters describe parameters to the simulator.

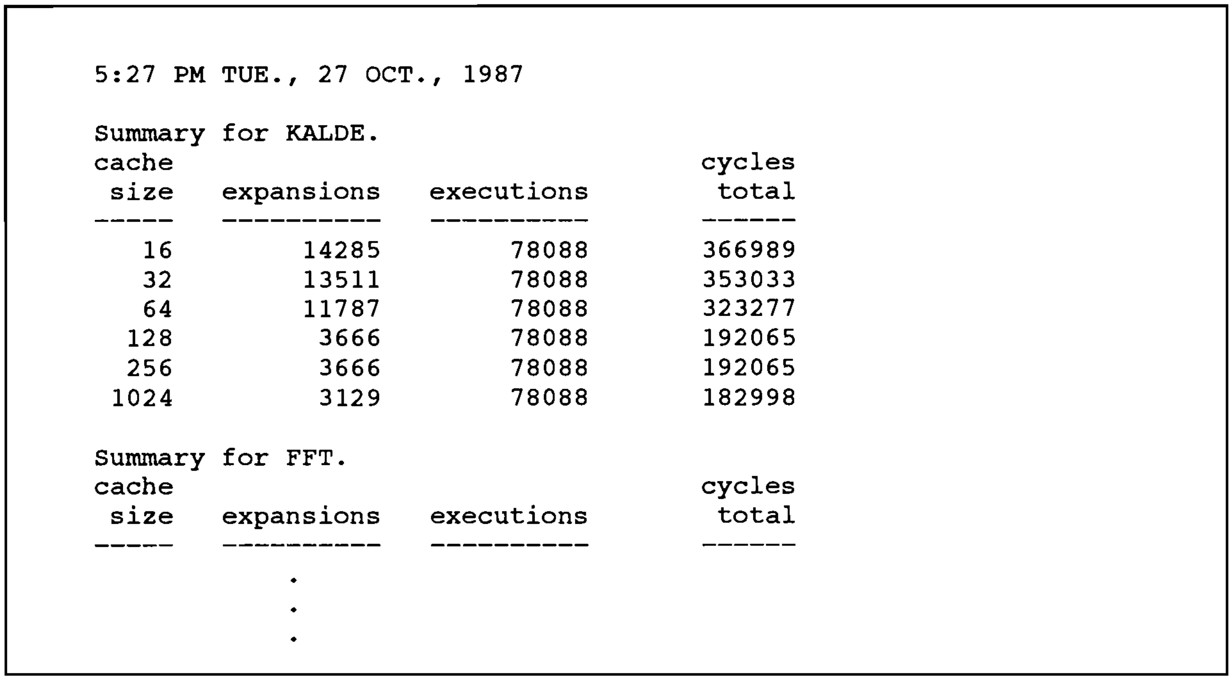
```
directory /JDJ/AM/AMDAT/
=FFT34              =FFT44              =FFT54
=FFT57              =FFT58              =FFT59
=FFT5C              =FFT66              =FFT77
=FFTAA              =FFTBB              =FFTCC
=KAL54              =KAL55              =KAL56
=KAL59              =KAL5A              =KAL5B
=KAL77              =KAL88              =KAL99
=KALCC              =PUZ34              =PUZ44
=PUZ77              =PUZ88              =PUZ99
=PUZCC              =SOR34              =SOR44
=SOR56              =SOR57              =SOR58
=SOR5B              =SOR5C              =SOR66
=SOR99              =SORAA              =SORBB
=WAL44              =WAL54              =WAL55
=WAL58              =WAL59              =WAL5A
=WAL66              =WAL77              =WAL88
=WALBB              =WALCC
```

Figure 5-25. List of Input Files

```
RU,AM,+X:16,"KALDE,>KALDE,=KAL54
Start time:   Mon Oct 26, 1987 11:43 pm
Stop time:    Mon Oct 26, 1987 11:49 pm
  .
  .
  .
RMEM size                                  32
cache size                                 16
  .
  .
  .
Address Lookups:
  Instruction Expansions                14285
  Syllables Extracted                   79060
  .
  .
  .
  address lookups(1 per)                34324
  pipeline breaks(1 per taken jump)      1704
  execution(1 per expanded inst.)       78088
                                       ------
                     Cycles Total      366989
```

Figure 5-26. Sample Data File

Figure 5-26 provides an example of one of the results files. This file starts a line that describes the set of input data used. In this case, +X:16 is a parameter to the simulator and the data set called KALDE. The name of the data set is to be included in the summary table header. (The stack of dots indicates that lines of the file have been deleted to provide a smaller figure for this text.) Several of the lines displayed are retrieved to be included in the summary table. A Regular Expressions pattern is constructed to enable EDIT to match the lines by searching for the text used to label the required numbers.



```
5:27 PM TUE., 27 OCT., 1987

Summary for KALDE.
cache                                 cycles
 size   expansions   executions        total
-----   ----------   ----------       ------
   16        14285        78088       366989
   32        13511        78088       353033
   64        11787        78088       323277
  128         3666        78088       192065
  256         3666        78088       192065
 1024         3129        78088       182998

Summary for FFT.
cache                                 cycles
 size   expansions   executions        total
-----   ----------   ----------       ------
                .
                .
                .
```

Figure 5-27. Desired Output

Figure 5-27 shows the summary table file that is to be constructed. It consists of a time stamp, followed by a label describing the data set (Summary for KALDE), and a table of numbers showing the results of each simulation. Each line in the table comes from one run of the simulator.

The table file is built by running EDIT several times, once for each output file produced by the simulator. An EDIT command file must be written to extract the required lines from each results file and append them to the end of the table file. This multiple scheduling of EDIT is done using the CI command file illustrated in the following figure.

```
pu,"cycs
cr,"cycs:::3:24
edit,"cycs,ti 40|er

edit,=kal54,tr *cycs1/
edit,=kal54,tr *cycs2/
edit,=kal55,tr *cycs2/
edit,=kal58,tr *cycs2/
edit,=kal59,tr *cycs2/
edit,=kal5a,tr *cycs2/

edit,=fft54,tr *cycs1/
edit,=fft54,tr *cycs2/
   .
   .
   .
edit,=wal5b,tr *cycs2/
edit,=wal5c,tr *cycs2/
```

Figure 5-28. CI Command File

The summary table file created is named "CYCS. The first line of the CI command file purges the old version of the table file, and the second line creates a new, empty version of it. The third line runs EDIT, causing it to put a time stamp into the table file. A separate EDIT command file is not needed to perform this, as the required EDIT commands are short enough to include in EDIT's runstring. The remaining CI command file lines run EDIT on each simulator results file. Each run of EDIT reads commands from one of two EDIT command files.

Two versions of the EDIT command file are constructed: *CYCS1 and *CYCS2. The *CYCS1 EDIT command file is used to build table the header and to extract the first line of the summary table. The *CYCS2 command file serves a similar function, except that it merely extracts and formats one line of the table and appends it to the summary table file. *CYCS2 does not build the table headers.

```
sere
b/ru,am/
i |-ka|g/@"{[^,]+}@/Summary for &1./
 cache                               cycles
  size   expansions   executions      total
 -----   ----------   ----------     ------
+d/cache size/q/
g/@{.<8>[0-9]} *$/   &1/
+d/instruction expansions/q/
g/@{.<8>[0-9]} *$/      &1/
-j
+d/expanded instruction executions/q/
g/@{.<8>[0-9]} *$/    &1/
-j
+d/cycles/q/
d/total/q/
g/@.<8>[0-9]} *$/    &1/
-j
:a.1,,+"cycs/
a/
```

Figure 5-29. EDIT Command File *CYCS1

Figure 5-29 illustrates the EDIT command file *CYCS1 in its entirety. The first step of building the required table is adding in the summary name. The line to be extracted from a simulation results file is

```
RU,AM,+X:16,"KALDE,>KALDE,=KAL54
```

The commands in the *CYCS1 EDIT command file that extract the summary name are

```
/sere
/b/ru,am/
/i |-ka|g/@"{[^,]+}@/Summary for &1./
```

The command SERE turns on the Regular Expressions option. The command b/ru,am/, finds the line in the work file that contains the required summary name.

The last command in this group consists of three commands on one line. The i command inserts a blank line before the title line. The next command (−ka), marks the line just inserted with the mark a. The inserted line is marked because it is to be the first line appended to the summary table file. (This command does not change the work file position.) Finally, the G command changes the retrieved line into the required header text. It does this by matching the entire line with the beginning and ending @ signs and tagging the character between the double quote and comma. This exchange produces the line:

```
Summary for KALDE.
```

Next, the fixed text headers are put into the table file. All the text is fixed, so the EDIT space ( ) command can be used to append lines of text. The commands from the *CYCS1 EDIT command file are listed following. (Note the space in front of each line of text in the figure of *CYCS1.)

```
cache   expansions  executions    cycles
size                              total
-----   ----------  ----------    ------
```

Next, the first column of the summary table is constructed. The number for this column is from a labeled line in the results file. Note the following example data file:

```
   .
   .
RMEM size                               32
Cache size                              16
   .
   .
```

and the relevant commands for the command file:

```
+d/cache size/Q/
g/@{.<8>[0-9]} *$/  &1/
```

All lines in the results file that follow the last header line and precede the text line "cache size" are deleted with the D command. The Q (quiet) option used with the D command tells EDIT not to list the lines deleted. The G command finds and tags a nine-character string that ends with the last digit on the line. It then exchanges the string, inserting two blanks before the tagged blanks and number, and deleting all other characters on the line. This sequence of commands has created the first column of the table.

The next column of the summary table is built by deleting text in the results file and joining the number onto the table line. The relevant text in the results file is

```
   .
   .
Address Lookups:
  Instruction Expansions             14285
```

and the instructions from the EDIT command file are

```
+d/instruction expansions/q/
g/@{.<8>[0-9]} *$/    &1/
-j
```

The first two steps are identical to those used for the first column. All lines from the one after the column just built to the one containing the next required number are deleted in the results files. Next, all text on the line is deleted except for a numeric field consisting of nine characters. The -j (Join) command moves the work file position up to the line containing the first column and joins it with the nine-character-wide column just created.

There are two more nine-character columns in each table line. These are built using commands similar to those just presented.

The final commands used to build the table are

```
/:a.1,,+"cycs/
/a/
```

The :a.l,,+"cycs/ command opens the summary table file and appends all the formatted lines that were just built into the data file. Using the :a mark permits changes to the number of lines appended without requiring modification to this command. The a/ command in the *CYCS1 file aborts EDIT, leaving the results files unchanged.

A

Error and Information Messages

This portion of the appendix provides a listing of the EDIT/1000 error messages and those system messages that most often occur while using EDIT. The errors are listed alphabetically in chart form. Those errors beginning with a number or variable string are listed first. Each error is listed as it is displayed, and an explanation of the error and suggestions for remedial action are provided.

EDIT/1000 Program Messages

Message **n lines read.**

Cause Informational message displayed after a file is read in to be edited or in response to a M (Merge) command. Indicates the number of lines copied from the file. Message indicates number of lines at the position of the n.

Action None required.

Message **n lines recovered.**

Cause Informational message displayed when EDIT is running a recovery operation, indicating the number of lines recovered in the file at the position of the n.

Action None required.

Message **Appending to file.**

Cause Informational message displayed when the append option (+) is used with the L (List) or K (Kill) command.

Action None required.

Message **Auto LF must be off.**

Cause Error message indicating that the automatic line feed is enabled on your terminal. The automatic line feed must be turned off before EDIT will function correctly.

Action Turn off automatic line feed or see System Manager.

Message **Block mode must be off.**

Cause Error message indicating that the block mode is enabled on your terminal. The block mode must be turned off before EDIT will function correctly.

Action Turn off block mode or see System Manager.

Message **Break – Workspace unchanged.**

Cause Informational message. Displayed in response to a break command used during a MO (Move) or CO (Copy) command, if the break command is detected prior to file modification.

Action None required.

Message **Can not recover because of incorrect data in the work file.**
To avoid re-entering recover mode, purge <filename>
Should this file be purged now?_

Cause Informational message displayed during a recovery operation indicating that the file was so corrupt that EDIT could not recover any data. This message indicates that there probably isn't any data in the file at all. Message instructs user to purge the file to avoid re-entering recovery operation. The name of the file to be purged is displayed at the position of the <filename>. Requires user to enter a *yes* or *y* to purge the file, or a *no* or *n* to postpone a decision.

Action Enter an affirmative response to purge the work file, or enter a *no* or *n* to postpone the decision and see your System Manager.

Message **Closed file <filename>.**

Cause Informational message indicating that a file has been closed as a result of a command entered by the user. The filename is displayed at the position of <filename>.

Action None required.

Message **Command not executed.**

Cause Informational message displayed when a non-affirmative response is entered in response to the OK? prompt described above.

Action None required.

Message **– – –Commands– – –**

Cause Header for the list of commands displayed in response to the command stack review (/ command).

Action To re-execute a command listed in the command stack, position the cursor on the desired command and press the carriage return. User may also edit any command in the stack and execute in the same way.

Message **Created file <filename>.**

Cause Informational message displayed after user enters an FI (File Input), L (List), EC (Exit and Create), WC (Write and Create), M (Merge), or TR (Transfer) command to create a file. Filename is listed at the position of <filename>.

Action None required.

Message **Disk error.**

Cause Error message displayed in response to a disk (hardware) failure.

Action Attempt an EC (Exit and Create) command to exit EDIT. Ask your System Manager to run a diagnostic test on your hardware to determine the nature of the problem.

Message **EDIT on <filename>.**

Cause Informational message displayed in response to the ?? command. Provides the current user's version of the EDIT program (for example, EDI53) and the file being edited.

Action None required.

Message **EDIT on <filename>, closed.**

Cause Informational message displayed in response to the ?? command (and the SH ER, Show Exit and Replace option command) after an FCS (Close Source File) command has been used to close the source file. At this point, EDIT will not allow the use of an ER (Exit and Replace) or WR (Write and Replace) command until the source file has been re-opened.

Action None required.

Message **EDIT on <filename>, to be created.**

Cause Informational message displayed in response to the ?? command (and the SH ER, Show Exit and Replace option command). Indicates that the current work file does not yet have a source file and that the use of an Exit (ER) command or a Write and Replace (WR) command will create the source file.

Action None required.

Message **EDIT aborted.**

Cause Informational message displayed when EDIT aborts as a result of an internal error. Preceded by a message indicating the reason for the abort.

Action Contact your HP representative.

Message **EDIT aborted by user.**

Cause Informational message displayed when EDIT has been aborted (Abort command) by the user.

Action None required.

Message **EDIT : Use ? for help.**

Cause Informational message displayed when the EDIT program is accessed.

Action None required.

Message **end of edit.**

Cause Informational message displayed in response to an EC (Exit and Create) or ER (Exit and Replace) command.

Action None required.

Message **Entering recover mode.**

Cause Informational message displayed when the −R option (to initiate recovery) is used with the EDIT runstring or when the last EDIT session was abnormally terminated.

Action Determine what should be saved from the recovered work file, then exit EDIT.

Message **EOF.**

Cause Informational message indicating that the current pending line is at the end of the file.

Action None required.

Message **EOF n matches <options>.**

Cause Informational message displayed after execution of a Find (B or F) command or an exchange (G, U, X, Y). Indicates that EDIT reached the end of the file and found the number of matches listed. The number of matches or the word *No* is displayed at the position of the n. For Finds, EDIT reports the number of lines containing one or more matches. For exchanges, EDIT reports the number of exchanges performed. If the CF (Case Folding) or RE (Regular Expressions) options were set to ON during the search, a CF and/or RE is displayed at the position of <option>.

Action None required.

Message **EXEC ERROR. P1 – P6, A, B, RTN.**
ADDRESS DATA
Suspending.

Cause Error messages displayed as a result of an internal error.

Action Contact your System Manager. Have your System Manager "off" EDIT (terminate the session) and then use the recovery mode to recover your work file.

Message **FI, <filename> specifies file to edit.**

Cause Informational message displayed when the EDIT runstring is used without specifying a filename to be edited.

Action Use FI (File Input) command to access an existing file for editing or enter text for a new file that is created by using the EC (Exit and Create) or WC (Write and Create) command after editing is complete.

Message **File is write protected.**

Cause Error message displayed when user attempts to open a write-protected file. EDIT can read the file, but will not allow you to write over the file using an ER(Exit and Replace) or WR (Write and Replace) command.

Action Check typing; specify a different file; or for File Manager files, access by specifying the correct security code for the file.

Message **If your terminal has them, disable straps d, g and enable straps l, T (closed/lower case is disable).**

Cause Informational message indicating that your terminal is not strapped correctly for the EDIT/1000 program and that EDIT was unable to programmatically re-strap the terminal.

Action See the System Manager to have the terminal manually re-strapped.

Message **Illegal file type on file <filename>**
Type <1,2, or 6> files are illegal.

Cause Error message displayed when user attempts to edit a file with a file type of 1, 2, or 6. Only file types 3, 4, 5, 7 and above can be edited using EDIT/1000. The filename entered is displayed in the first line of the message. The file type is displayed in the second line of the message.

Action Check typing.

Message **Limit n matches <option>.**

Cause Informational message displayed after execution of a Find (B or F) or an Exchange (G, U, X, or Y) command. Indicates that EDIT reached the end of the line range specified and found the number of matches listed. The number of matches or the word *No* is printed at the position of the n. For Finds, EDIT reports the number of lines containing one or more matches. For exchanges, EDIT reports the number of exchanges performed. If the CF (Case Folding) or RE (Regular Expressions) options were set to ON during the search, a CF and/or RE is displayed.

Action None required.

Message **Line n patched.**

Cause Informational message displayed during a recovery operation, indicating that a specific line was corrupted and that EDIT had to interpret it during the recovery. The line number of the corrupted line is displayed at the position of the n.

Action Examine the line specified following the recovery operation to ensure that it contains the proper number of characters.

Message **Line limit exceeded.**

Cause Error message displayed when file contains more than 32,500 lines. No additional modifications can be made to the file.

Action Use the ER (Exit and Replace) or A (Abort) command to exit EDIT. For more information, see the sections on File Size Limit in Chapters 1 and 3.

Message **Line order patched at line n.**

Cause Error message displayed after recovery operation, indicating that the recovery file is not consistent and that EDIT started duplicating lines at the line number displayed.

Action Use the A (Abort) command to abort the recovery if the file is expendable; or examine the file for duplicate lines around the line number displayed in the error message. For more information, refer to the Recovering From Errors section in Chapter 3.

Message **No changes allowed during recover mode.**

Cause Error message displayed when user attempts to modify a file when EDIT is running a recovery operation. (The purpose of the recovery is to reconstruct a work file that was being edited when EDIT was abnormally terminated.)

Action Exit EDIT to terminate the recovery operation. To exit EDIT, use an EC (Exit and Create) command to create a copy of the recovered file; or use an A (Abort) command to delete the recovered file; or, if you are unsure whether to save the work file or abort and would like to consult the System Manager, use an AS (Abort and Save Work File) command to postpone the decision. If the AS command is used, the recovery operation is executed again the next time EDIT is run.

Message **No help for <command/option>. Use ? for general help.**

Cause Message displayed when there is no help available for the command or option entered after the H or ? (Help) command.

Action Check typing. Use ? to view the help available. If online help is not available for the command or option entered, consult the Manual.

Message **No marks.**

Cause Informational message displayed in response to the SH MA (Show Marks) command. Indicates that there are no lines in the file that have been marked using the CTRL and K, or Kx (Mark Line) command.

Action None required.

Message **No space in scratch file.**

Cause Error message displayed if user attempts to modify a file after receiving the *Out of disk work space* message.

Action Use an ER (Exit and Replace) or A (Abort) command to exit EDIT, then proceed to obtain additional disk space as suggested under the *Out of disk work space* message, above.

Message **No such file <filename>**
An ER or the first WR will create it.

Cause Message displayed when a non-existing file is specified in the EDIT runstring. The filename specified is listed at the position of <filename>.

Action If file is to be created, proceed to enter text and use the ER (Exit and Replace) command without a filename to terminate the session. Otherwise, check typing of filename entered.

Message **non HP26XX terminal – screen mode unavailable.**

Cause Error/informational message displayed in response to the S (Screen Mode) command. Indicates that your terminal does not support the EDIT/1000 screen mode.

Action Use the EDIT/1000 line mode to edit.

Message **Not found.**

Cause Informational message displayed following execution of a line specification search command. Indicates that the pattern specified in a search or exchange command was not found.

Action None required.

Message **OK?**

Cause Command confirmation prompt displayed by EDIT when a command is used that could potentially delete or significantly alter the file text. Requires a *yes* or *no* response.

Action Type *y* or *yes* to execute the entered command, or *n* or *no* to cancel the command.

Message **Opened file <filename>.**

Cause Informational message displayed after user enters an M (Merge) or TR (Transfer) command. The file opened is listed at the position of <filename>.

Action None required.

Message **Operator break.**

Cause Informational message indicating that user terminated execution of a command before completion.

Action None required.

Message **Out of disk work space.**

Cause Error message displayed during an EDIT command when there is insufficient space available on the disk to complete the command. No additional modifications can be made to the work file.

Action First, attempt an EC (Exit and Create) or ER (Exit and Replace) command; then consult with the System Manager to obtain more disk space. If EC or ER is unsuccessful, use the EC command and specify a directory on an LU (Logical storage Unit of the disk) that has sufficient space. Your System Manager may choose to run EDIT specifying the −S option and a directory on a different LU to place your work file in a directory that has adequate space. For more information on how to recover from the *Out of Disk Work Space* message, refer to the section on Recovering From Errors in Chapter 3.

Message **Overlap.**

Cause Error message indicating that the destination specified in a MO (Move) command lies within the range of lines specified to be moved.

Action Check typing of the MO command line specifications and compare with the pending line number. Specify a different destination or a different range of lines to be moved.

Message **Posted file <filename>.**

Cause Informational message displayed in response to a WR (Write and Replace) command, indicating that the write operation is complete.

Action None required.

Message **Purging file <filename>.**

Cause Informational message displayed when user entered an affirmative response to the message described above: *Can not recover because of incorrect data in the work file. To avoid re-entering recover mode, purge <filename> Should this file be purged now?_.*

Action None required.

Message **Put cursor on first line when edit is complete.**

Cause Instructional message displayed in line mode when more than 78 characters are entered on a line during use of the Q (Single Line Screen Mode) command. Indicates that additional characters entered on the line will be wrapped to the next line on the screen display.

Action After the Q command edit is complete, position the cursor on the first line of the wrapped text before pressing the carriage return. If the cursor is not positioned on the first line of the wrapped text, the lines above the cursor are lost.

Message **Read terminated before end of file found.**

Cause Informational/error message displayed when EDIT did not completely read in the file. Occurs if user enters the BR (Break) command, if a corrupted file is accessed, or if EDIT runs out of work space when reading in the file. Message is preceded by a message indicating the reason for termination of the read. This message is displayed if the file exceeds the file size limit of 32,500 lines.

Action Correct the error which resulted in termination of the read. If message was received in response to an FI (File Input) command, use of the ER (Exit and Replace) command *truncates* any data that was not read in prior to the read termination. Use the A (Abort) command to avoid truncating data. To save the data, use the Merge (M) command to create several smaller files that can be edited and then re-merged using the CI (Command Interpreter) merge program. Refer also to the FMP (File Management Package) error message, *RAN OUT OF DISK SPACE.*

Message **RESTORE SEGMENT AND DO A “GO”.**

Cause Instructional message displayed as a result of an internal error in the EDIT/1000 program.

Action Consult with System Manager.

Message **Resume EDIT on <filename>.**

Cause Informational message displayed when a run operation initiated by the RU (Run) command is complete. The source file name is displayed at the position of <filename>.

Action None required.

Message **Segment load error for *EDITx*.**

Cause Error message displayed as a result of an internal error in the EDIT/1000 program. The segment number is provided in the position of *EDITx.*

Action Consult with System Manager.

Message **Setting terminal straps to d g I T.**

Cause Informational message indicating that your terminal is not strapped correctly for the EDIT/1000 program and that EDIT is attempting to programmatically re-strap the terminal.

Action None required.

Message **Size Edit up.**

Cause Error message indicating that the EDIT/1000 program has not been loaded properly.

Action See the System Manager.

Message **Sorry, no help. File EDIT.HLP not found.**

Cause Error message displayed when EDIT cannot locate the help file. Indicates that the EDIT program has not been installed properly.

Action See the System Manager.

Message **Start and/or stop line not found**
O saves original text written to screen
S saves text just read from screen
B saves both (inserts screen text before original text)
What should be saved ?

Cause Error message displayed in response to a control key combination (^F, ^P, ^S, ^X) used from the screen mode, indicating that EDIT was unable to locate either the beginning or ending screen brackets, or both brackets. Requires user to indicate information to save by entering an O, S, or B.

Action Enter an O to save the screen text as originally written to the screen for this screen mode session, without the most recent edits; enter an S to save the text read from the screen; or enter a B to save both the original screen text and the text just read from the screen. If unsure as to the status of the text, enter a B and then delete any duplicated lines.

Message **Start > EOF.**

Cause Error message referring to the line range specification entered for a command. Indicates that the starting line number entered is beyond the last line of the file.

Action Check typing and/or verify the line numbers of the lines to be moved/copied.

Message **Start > stop.**

Cause Error message referring to the line range specification entered for a command. Indicates that the starting line number entered is larger than the ending line number for the range.

Action Check typing.

Message **System type of your terminal? _**

Cause Prompt message displayed when EDIT cannot determine the system type during an I/O mapping operation. Indicates that the distributed system to which the terminal is mapped does not have the current DS software installed.

Action Enter an A if terminal is connected to an A Series system, or enter an E if terminal is connected to an E or F Series system. Install current software.

Message **Time stamp found.**

Cause Informational message displayed during the read operation, indicating that a time stamp was located in the file.

Action None required.

Message **Time stamping output file.**

Cause Informational message displayed in response to a WR (Write and Replace), ER (Exit and Replace), WC (Write and Create), or EC (Exit and Create) command issued for a file that has been time stamped by EDIT.

Action None required.

Message **To put work file on cartridge "xx" use "RU, EDIT, –S::xx, file".**

Cause Informational message displayed when EDIT run command cannot be completed due to insufficient disk space.

Action Consult with System Manager to obtain additional disk space, or run EDIT using the –S option, specifying a directory (indicated by the "xx" in the message) on an LU (Logical disk storage Unit) that contains sufficient space. For more information, refer to the section on Recovering From Errors in Chapter 1.

Message **Window n, n.**

Cause Informational message displayed after execution of an F (Find) or X (Exchange) command. Indicates the search window size that was used for the search. Displayed only if no matches to the pattern were located and the window size used for the search was different from the default window size. Serves to remind user to change window size back to default before continuing the editing session.

Action Attempt Find or Exchange command with a different window size to locate desired pattern. After Find or Exchange operation is complete, change window size back to default before continuing the editing session.

Message **Work file <filename>.**

Cause Informational message displayed when the −S option (create work file in a different directory) is used in the EDIT runstring. Indicates the work file to be created at the position of <filename>.

Action None required.

Message **Work file error.**

Cause Error message displayed during EDIT initialization. Indicates that EDIT was unable to create the work file. Followed by a message indicating why the work file could not be created.

Action Correct the error.

Message **You are using I/O mapping with a pre-A.85 version of IOMAP. Please enter "E" if your terminal is on a M, E or F−Series machine or "A" if it is on an L or A−Series machine.**

Cause Informational/error message displayed when user fails to enter an A or E in response to the message *System type of your terminal?* _, described above.

Action Enter an A if terminal is connected to an A Series system, or enter an E if terminal is connected to an E or F Series system. Install current software.

RTE System and FMP Messages

There are several FMP (File Management Package) messages issued by the RTE user interface that may be encountered when accessing EDIT. The most common FMP messages are listed and described below. For a more complete list, refer to the *RTE User's Manual*.

Message **BAD RECORD LENGTH**

Cause Error message displayed when EDIT is reading in a file and detects that the record structure of the file is corrupted. This is usually the result of an abnormal program termination or a system error.

Action Examine file to determine what information has been lost. An ER (Exit and Replace) command updates the file to have consistent record structure, saving only the uncorrupted data currently in EDIT's work file.

Message **DIRECTORY IS FULL**

Cause Error message indicating that there is insufficient space in the current directory to add another file.

Action Delete unnecessary files, use another directory, or see your System Manager.

Message **DIRECTORY READ PROTECTED**

Cause Error message displayed when user attempts to use an FI (File Input), EDIT, TR (Transfer), or M (Merge) command to access a directory that is read protected.

Action Check typing to ensure that the correct directory was specified; log on as the owner of the file; or see the System Manager to investigate possible access to the file.

Message **DIRECTORY WRITE PROTECTED**

Cause Error message displayed when an input command (FI, EDIT, TR, or M) or a write command (ER or WR) is used on a directory that is write protected.

Action Check typing to ensure that the correct directory was specified; log on as the owner of the directory; or see the System Manager to investigate possible access to the directory.

Message **FILE ALREADY EXISTS**

Cause Displayed when user attempts an EC (Exit and Create) or WC (Write and Create) command on an existing file.

Action Use an ER (Exit and Replace) or WR (Write and Replace) command to overwrite the existing file or select a new filename and use the EC or WC command.

Message **FILE IS ALREADY OPEN**

Cause Information message indicating that another program is using the file specified.

Action Use the Command Interpreter DL (Directory List) command with the O option to find out which program is using the file.

Message **FILE READ PROTECTED**

Cause Error message displayed when user attempts to use an FI (File Input), EDIT, TR (Transfer), or M (Merge) command to access a file that is read protected.

Action Check typing to ensure that the correct file was specified; log on as the owner of the file; or see the System Manager to investigate possible access to the file.

Message **FILE WRITE PROTECTED**

Cause Error message displayed when an input command (FI, EDIT, TR, or M) or a write command (ER or WR) is used on a file that is write protected.

Action Check typing to ensure that the correct file was specified; log on as the owner of the file; or see the System Manager to investigate possible access to the file. An EC (Exit and Create) command can be used to create an unprotected copy of the file.

Message **ILLEGAL FILE POSITION**

Cause Error message displayed when EDIT is reading in a file and detects that the record structure of the file is corrupted. This is usually the result of an abnormal program termination or a system error.

Action Examine file to determine what information has been lost. An ER (Exit and Replace) command updates the file to have consistent record structure, saving only the uncorrupted data currently in EDIT's work file.

Message **ILLEGAL LU**

Cause Error message indicating that device number (for example, tape drive, printer, etc.) specified is not accessible or capable of performing the specified function, or does not exist.

Action Check typing.

Message **ILLEGAL NAME**

Cause Error message indicating that filename specified contains illegal characters (for example, leading digits or non-printing characters such as control characters).

Action Correct filename used.

Message **INCORRECT PASSWORD**

Cause Error message displayed when the password specified for a remote account is incorrect.

Action Check typing.

Message **INCORRECT SECURITY CODE**

Cause A File Manager error message indicating that the security code specified is incorrect.

Action If file is to be updated, use an FI (File Input) command with the correct security code to input the file.

Message **NO SUCH ACCOUNT**

Cause Error message displayed when a non-existent account is specified during log on to a different distributed system.

Action Check typing.

Message **NO SUCH CARTRIDGE**

Cause File Manager error message indicating that the cartridge number specified does not exist.

Action Check typing.

Message **NO SUCH DIRECTORY**

Cause Error message displayed when user specifies a directory that is non-existent.

Action Check typing or use system commands (DL) to find the correct name of the directory.

Message **NO SUCH FILE**

Cause Error message displayed when an FI (File Input), ER (Exit and Replace), M (Merge), or TR (Transfer) command is used to input a file that does not exist in the current working directory or the directory specified in the command.

Action Check typing; ensure that the correct directory was specified.

Message **NO SUCH NODE**

Cause Error message displayed when a non-existent node is specified while attempting to access a file on a different distributed system.

Action Check typing.

Message **RAN OUT OF DISK SPACE**

Cause Error message indicating that there is insufficient disk space to complete the operation, and that the source file has been corrupted at the point where the system ran out of disk space. Note that the work file now contains the ONLY VALID VERSION of your file.

Action DON'T ABORT EDIT!! Either (1) use the RU (Run) command to run CI (Command Interpreter) from EDIT and enter commands to create more space (that is, purge unneeded files or pack the directory); or (2) use the EC (Exit and Create) command specifying a different directory, to create a new file on a directory that has sufficient space.

B

Glossary

Following is a glossary of terms that you will encounter throughout this manual. Many of the terms are further explained in Chapter 1 – EDIT/1000 Basics and Chapter 3 – EDIT/1000 Operations.

anchor

A character used to start a search or a match at the beginning of the search window.

case folding

An option initially turned on to allow recognition of either upper or lower case letters.

character class

A group of characters any one of which can be used as the character to be searched/ matched.

closure

The metacharacter "*" used to make a text pattern which matches zero or more successive occurrences of the single character pattern. For example, x* matches zero or more x's; xx* matches one or more x's; [a–z]* matches any string of zero or more lower case letters. If a number of combinations are present, the longest one is used. Note that for @ in pattern specification, the shortest is used.

command stack

A list of EDIT commands entered during any one session. The list is accessible with the command stack command (/).

command string

The various fields entered in an EDIT command. These are typically the additional EDIT commands, the optional (or required) line specifications, pattern and exchange fields, and command options.

control character

A non-printing character entered by pressing both the CTRL key and the appropriate character. Control characters are used as sub-commands in line and screen mode editing.

current prompt

The EDIT prompt character in effect. Default is a slash (/). It may be changed using the SE PC command described in Chapter 3.

default

The initial or reset condition for an optional parameter in the EDIT command string.

delimiter

A character used to separate command and parameters. Required for some commands and optional for others.

escape character

A character used to revert a character with special meaning to its original meaning.

line specification

Parameters that specify one line or a range of lines. Usually they precede certain commands. Special characters may be used to indicate offsets. These characters may be checked online with *? LS*.

list file

A file created or opened by the L and K commands. Specified number of lines may be written to or appended to the list file.

logical unit number

A number (referred to as LU) used by the operating system to identify an I/O device.

marks

Alphabetic line labels set by the user for future reference to the line. Maintained by EDIT. Marks are valid for each session.

match

A pattern to be located matching either a specified pattern or all except the specified pattern.

metacharacter

A set of special characters with unique meanings to EDIT. These characters are used in regular expression searches and exchanges. These characters can be checked online with *? RE*.

namr

Standard RTE FMGR disk file designation. In EDIT, usually means a filename with optional security code and cartridge reference number separated by ":". File type and size are sometimes used in creating a file from EDIT.

options

Either an optional parameter in a command string or an initial EDIT condition that can be changed with the set option command. Refer to the SE command description for details.

pattern

A character string to be searched/exchanged.

range

A number of lines with a defined beginning and ending line specification. Can be either absolute numbers or line specification characters, with or without offsets.

recovery mode

EDIT enters the recovery mode whenever a scratch file is found to already exist. No changes can be made; screen mode is disabled. All other commands are allowed. Must abort or exit restoring or creating with a filename before running any editing session.

regular expressions

Special patterns using the metacharacters for unique searches and exchanges. Use of regular expressions provides a means of specifying alternates and developing generalized patterns. For example, *X[+−]Y* which matches X+Y or X−Y. To use the regular expressions EDIT session option, use the Set command to turn the option ON.

reverse match

A search in which a match occurs only if the specified pattern is not found within a line.

runstring

Usually a run program command with all the permissible command parameters. For example: *EDIT,file,120|S.*

scratch file

Term often used to refer to the work file, or temporary work area on disk where EDIT operates from during the editing session. This area contains the text read into memory.

screen mode

An editing mode in which a number of lines of text is displayed to allow local editing using the editing keys available on the terminal keyboard.

source file

An ASCII file opened and read by EDIT.

string

A group of ASCII characters.

substitute

A replacement pattern when a match occurs during an exchange command execution.

vertical window

The number of lines above and below the pending line to be displayed with the vertical window (W) command.

window

Usually means the search window which confines the search to within two columns defined by the horizontal window settings. See also "Vertical Window".

Loading EDIT/1000

The EDIT program must be loaded online. EDIT should be loaded by the system manager or a person appointed by the manager. The following components are required for the applicable systems:

Table C-1. EDIT/1000 Components

MODULE NAME	PART NUMBER	DESCRIPTION
%EDIT	92074-12008	RTE-6/VM & -A EDIT program module (combined)
%ED000	92074-16055	RTE-6/VM & -A EDIT message module
#ED1K6	92074-17003	RTE-6/VM load file
#ED1KA	92074-17005	RTE-A load file
$ED1K6	92074-12005	RTE-6/VM library
$ED1KA	92074-12011	RTE-A library
"EDIT	92074-17004	EDIT help file

Memory Requirements

EDIT uses all memory available in its partition, up to 32 pages. The best performance is achieved if EDIT is sized at 32 pages. If EDIT is loaded into a partition of insufficient size (approximately 21 pages), EDIT issues a message that requests a larger partition and terminates.

Help File

The file "EDIT contains all of the text for the EDIT help commands. This file must be accessible to EDIT, or the help command does not work.

On RTE-A and RTE-6/VM, the user can move the file "EDIT to the global SYSTEM directory under the following name: EDIT.HLP::SYSTEM. When a help command is executed, EDIT searches the SYSTEM directory for EDIT.HLP. If EDIT does not find it, it searches the FMGR cartridges for the file "EDIT and uses that file instead.

Work File

EDIT creates a type 1 file called the work file (sometimes called the scratch file) when you run it. If a source file is specified, it is copied into this file. It is the work file, not the source file, that you modify during an EDIT session. The type 1 work file is reformatted and copied to the source file or another file by the EC, ER, WC, or WR commands. The name of the work file depends on the operating system under which you are running EDIT.

Under any operating system, the scratch file cartridge can be specified in the EDIT runstring by specifying the -S option. An existing work file can be specified for recover with the -R runstring option.

RTE-A Work File Considerations

Ennnx+0000000000.EDIT::SCRATCH:1:size
−OR−
Ennnx+:0:crn:1:size

where:

Ennnx – File name constructed as follows:

The clone name under RTE-A is usually “EDIT”. The session ID number replaces characters 2-4, filling from the right: a single-digit ID number replaces only character 4, a two-digit number replaces characters 3 and 4, and a three-digit ID number replaces characters 2-4. If the clone name contains fewer than five characters (as in “EDIT”), the fifth character in Ennnx is replaced with an asterisk.

For example, EDIT in session 32 creates the following file:

```
ED32*+0000000000.EDIT::SCRATCH:1:512
```

EDIT in session 178 creates:

```
E178*+0000000000.EDIT::SCRATCH:1:512
```

SCRATCH – The /SCRATCH or /SCRATCHEDIT directory.

First EDIT tries to create a file on /SCRATCH. If is it successful, EDIT then checks to see if the /SCRATCH directory is on RAM disk. /SCRATCH must be of adequate size to accommodate any files created. If /SCRATCH is not large enough, EDIT/1000 will not create the file, report an error, and terminate.

If /SCRATCH is on RAM disk, EDIT will purge the file off of /SCRATCH and attempts to create the file on directory /SCRATCHEDIT. If /SCRATCHEDIT does not exist, the file will be put back on the /SCRATCH directory. This action is taken to allow the /SCRATCH directory to be put on the RAM disk while still protecting EDIT’s recover mode. It is assumed that a RAM disk will not survive a reboot. For this reason, if you put /SCRATCH on a RAM disk, it is recommended that you create a /SCRATCHEDIT directory on a real disk.

crn – If /SCRATCH does not exist, EDIT puts the work file on the FMGR scratch cartridge specified in $SCRN. The default value of $SCRN is 0, (default to top cartridge in crn list). $SCRN can be changed by the BOOTEX SC command. Refer to the *RTE-A System Generation and Installation Manual,* part number 92077-90034.

size – Usually 512 blocks, depending on the amount of space available in the /SCRATCH directory or the top FMGR cartridge. The work file is automatically extended if more room is needed.

RTE-6/VM Work File Considerations

Under RTE-6/VM, the work file takes one of the following three forms: the first is the CI file system format, the second is the single-CPU FMGR file system format, and the third is the multiple-CPU FMGR file system format:

```
EDIxx+0000000000.EDIT::SCRATCH:1:size
      -OR-
EDIxx+:0:crn:1:size
      -OR-
EyIxx+:0:crn:1:size
```

where:

EDIxx – The FMGR clone name of the EDIT session, where xx is usually the session LU. If the clone name contains fewer than 5 characters, as in "EDIT", one or more asterisks are added to pad the name. A plus (+) is added as the sixth character of the work file name to identify it as the work file.

EyIxx – Same as EDIxx, except that the y is replaced by the number of the CPU in the multiple-CPU system on which the EDIT session is running.

SCRATCH – The /SCRATCH directory, if it exists; otherwise, crn is used.

crn – If /SCRATCH does not exist, EDIT puts the work file on the FMGR scratch cartridge specified in $SCRN. The default value of $SCRN is 0, (default to top cartridge in crn list). $SCRN can be changed by the FMGR VL command (refer to the *RTE-6/VM Terminal User's Guide*). Do not designate LU 2 or LU 3 to be the scratch cartridge; EDIT does not have write access to them. You cannot create a module to change $SCRN; EDIT looks for it only in the system map.

size – Usually 512 blocks, depending upon the amount of space available in the /SCRATCH directory or the top cartridge. The work file is automatically extended if more room is needed.

RTE-6/VM Loading Considerations

Under RTE-6/VM, EDIT can be loaded by LINK, which merges all of its segments into one type 6 file. This eliminates the need to load the T5IDM utility or to execute the RP command to restore the ID segments. If EDIT is loaded by LINK, though, it cannot be loaded as a permanent program.

Alternatively, EDIT can be loaded as described below:

The system manager can load EDIT as a permanent program for security, convenience, and better performance. The manager can load EDIT as a permanent program by specifying the IH option in the LOADR runstring, and the OP,PE command. EDIT must be loaded online; not at generation time. A LOADR command file (called #ED1K6) and a runstring that uses the command file for loading EDIT as a permanent program are shown below:

```
*LOADR command file #ED1K6
OP,PE
OP,EB
SZ,32
LIB,$ED1K6             (Loading EDIT into RTE-6/VM)
LIB,%ED000
RE,%EDIT

:RU,LOADR:IH,#ED1K6,,,PE (Uses command file #ED1K6)
```

The system manager should load EDIT shortly after booting. Doing so prevents EDIT from being loaded in the middle of the RTE system tracks, possibly minimizing the number of contiguous tracks available.

EDIT has five segments, EDIT0 through EDIT4. If EDIT is NOT loaded as a permanent program, the system manager must also load the RTE utility program T5IDM. T5IDM manages ID segments; refer to the *RTE-6/VM System Manager's Reference Manual,* part number 92084-90009, if you need to know more about T5IDM.

If you want to load EDIT as a temporary program, you must execute the FMGR command SP for all five segments as well as the main. If T5IDM is not loaded, you must RP all of the EDIT segments before running EDIT.

D

EDIT/1000 in a Multipoint Environment

In a multipoint environment, several considerations apply to EDIT operations. A terminal is defined to be in a multipoint environment if its I/O is being processed through driver DVR07. To determine if a terminal is operating under multipoint, observe the transmit break light on the terminal. If it is blinking intermittently, the terminal is probably a multipoint terminal.

When EDIT is invoked from a multipoint terminal, several actions are performed for the user. The intrinsic tab function of the terminal is enabled (the editor's tab character (CTRL-I) remains on). The INSERT CHAR, DELETE CHAR, and CLEAR DISPLAY keys on the terminal are enabled for use as editing functions. (Note that the EDIT commands to insert and delete lines should be used, not the INSERT LINE and DELETE LINE keys on the terminal.) Finally, since the CONTROL and ESCAPE keys do not work in a multipoint environment, the Q and O commands rather than the edit commands explained in the preceding sections of this manual should be used to perform character edits.

It should be remembered that in a multipoint environment the ENTER key, and not the carriage return, is the key used to transmit text to the multipoint controller. Furthermore, the characters that EDIT will process are usually delimited by the left margin on the left and the current cursor position on the right. Paying attention to these observations will help greatly.

In the multipoint environment, screen mode and the command stack features are not available. The slash (/) moves EDIT to the next line. All of the other EDIT commands may be used in a multipoint environment.

Q and O Commands

The Q and O commands are used for character edits in a multipoint environment. Both commands are used to edit the pending line. The only difference between them is that the O command immediately sends a copy of the pending line to the destination work area while the Q command does not. Although only the Q command is discussed throughout the rest of this section, the discussion applies to the O command as well.

When the Q command is entered, the pending line is displayed along with the delimiter (GS) to the left of the line. The delimiter is not part of the text string and must be preserved to assure proper operation. EDIT will position the cursor underneath the first character of the line. You may now edit the line using any of the following procedures.

To retain the pending line as it is, immediately hit the ENTER key. For example

```
/Q
  ABCDEFGHIJKL    (Cursor displayed under first
  -                  character in line.
                   Press the ENTER key and line is
                    retained as is.)

/P
  ABCDEFGHIJKL    (Line displayed remains the same)
/
```

To truncate characters from the end of a line, position the cursor immediately after the last character to be retained. Strike the ENTER key to enter all the characters between the left margin and the current cursor position. The intrinsic terminal key CLEAR DISPLAY can also be used to delete characters at the end of a line. After using the CLEAR DISPLAY key, the ENTER key is used to enter the edited line. For example

```
/Q
  ABCDEFGHIJKL    (Position cursor under I,
          -        press ENTER key.)

  ABCDEFGH        (Edited line is displayed.)

/Q
  ABCDEFGH        (Position cursor under F, press
       -           CLEAR DISPLAY, then ENTER.)

  ABCDE           (Edited line is displayed.)
/
```

To add characters in the middle of a line, the INSERT CHAR key may be used. Press the INSERT CHAR key. The red light above the key should come on. Move the cursor to the position where the characters are to be added, and type in the new characters. Finally, position the cursor at the end of the line and hit the ENTER key. The insert light will go off and the edited line will remain as the pending line. For example

```
/Q
  ABCDEFGHIJKL    (Position cursor underneath F,
       -           depress the INSERT CHAR key, and
                   type in the new characters
                   "123".  Position cursor at end
                   of line and press ENTER.)


    ABCDEF123GHIJKL (Edited line is displayed.)
  /
```

To delete one or more characters, position the cursor under each character to be deleted and press the DELETE CHAR key. The character will be deleted from the display and the rest of the line will be shifted left to fill in the gap. After all of the desired deletions have been made, move the cursor to the end of the line and press the ENTER key. Do not delete the delimiter at the beginning of the line. For example

```
/Q
  ABCDEFGHIJKL     (Position cursor under the G,
        -           press the DELETE CHAR key three
                    times, move the cursor to the
                    end of the line and press ENTER)

  ABCDEFJKL        (Edited line is displayed.)
/
```

Tab Control in a Multipoint Environment

When EDIT is invoked in a multipoint environment, the TAB key on the terminal is enabled (the EDIT tab character remains on). The tab stops are initially set to columns 7 and 21. These may be changed using the SET TAB and CLEAR TAB keys on the terminal. The TAB key may be used to position the cursor at any time. It is equivalent to moving the cursor using the terminal keys with arrows on them.

Index