

Tektronix[®]
COMMITTED TO EXCELLENCE

8500
MODULAR MDL SERIES
Advanced CRT-Oriented
Editor
USERS MANUAL
Version 1.X



This manual supports the following TEKTRONIX products:

8550
Options Products
1S 8300C01

This software module is compatible with:
DOS/50 Version 1.X

**PLEASE CHECK FOR CHANGE INFORMATION
AT THE REAR OF THIS MANUAL.**

8500
MODULAR MDL SERIES
**Advanced CRT-Oriented
Editor**
USERS MANUAL
Version 1.X

Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97077

Serial Number _____

LIMITED RIGHTS LEGEND

Software License No. _____

Contractor: Tektronix, Inc.
Explanation of Limited Rights Data Identification Method
Used: Entire document subject to limited rights.


Those portions of this technical data indicated as limited rights data shall not, without the written permission of the above Tektronix, be either (a) used, released or disclosed in whole or in part outside the Customer, (b) used in whole or in part by the Customer for manufacture or, in the case of computer software documentation, for preparing the same or similar computer software, or (c) used by a party other than the Customer, except for: (i) emergency repair or overhaul work only, by or for the Customer, where the item or process concerned is not otherwise reasonably available to enable timely performance of the work, provided that the release or disclosure hereof outside the Customer shall be made subject to a prohibition against further use, release or disclosure; or (ii) release to a foreign government, as the interest of the United States may require, only for information or evaluation within such government or for emergency repair or overhaul work by or for such government under the conditions of (i) above. This legend, together with the indications of the portions of this data which are subject to such limitations shall be included on any reproduction hereof which includes any part of the portions subject to such limitations.

RESTRICTED RIGHTS IN SOFTWARE

The software described in this document is licensed software and subject to **restricted rights**. The software may be used with the computer for which or with which it was acquired. The software may be used with a backup computer if the computer for which or with which it was acquired is inoperative. The software may be copied for archive or backup purposes. The software may be modified or combined with other software, subject to the provision that those portions of the derivative software incorporating restricted rights software are subject to the same restricted rights.

Copyright © 1981 Tektronix, Inc. All rights reserved. Contents of this publication may not be reproduced in any form without the written permission of Tektronix, Inc.

Products of Tektronix, Inc. and its subsidiaries are covered by U.S. and foreign patents and/or pending patents.

TEKTRONIX, TEK, SCOPE-MOBILE, and  are registered trademarks of Tektronix, Inc. TELEQUIPMENT is a registered trademark of Tektronix U.K. Limited.

Printed in U.S.A. Specification and price change privileges are reserved.

TABLE OF CONTENTS

Section 1 Learning Guide	Page
Introduction	1-1
Editor Overview	1-1
Installing the Editor and Configurator	1-8
Demonstration Run	1-10
For Continued Learning	1-25
 Section 2 Procedures	
Introduction	2-1
The Essentials.....	2-2
Character Manipulation	2-13
Line Manipulation	2-19
Block Manipulation	2-27
Command Shortcuts	2-47
 Section 3 Command Dictionary	
Overview	3-1
Dictionary Page Format	3-1
Terminology.....	3-3
Command Entry	3-4
The Screen	3-4
Special Keys	3-6
Commands.....	3-9
 Section 4 Tables	
Configurable ACE Commands	4-1
Non-Configurable ACE Commands	4-2
ACE Configurable Command Syntax	4-3
ACE Non-Configurable Command Syntax	4-4
How Non-Displayable Characters are Represented by ACE	4-5
ASCII Codes (Hexadecimal)	4-6
CT8500 Keyboard Layout.....	4-7
CT8500 Configurable Key Layout	4-7

Section 5 Technical Notes	Page
Note 1: The Configurator	5-1
Introduction	5-1
Configuration Overview	5-2
Preliminary Considerations	5-4
Using the Configurator	5-14
Error Messages	5-23

Section 6 Error Messages	
Error Messages	6-1
General Messages	6-3

Section 7 Glossary

Section 8 Index

Section 1 LEARNING GUIDE

	Page
Introduction	1-1
Editor Overview	1-1
Uses of the Editor	1-1
General Features	1-2
The Screen	1-3
Editor Commands.....	1-6
Notation Conventions	1-8
Installing the Editor and Configurator	1-8
Equipment Needed	1-8
Install the Software.....	1-9
Demonstration Run	1-10
Introduction	1-10
Start Up and Log On	1-11
How to Correct Input Mistakes.....	1-11
Create a Text File.....	1-11
Modify a Disc File	1-14
Summary	1-24
For Continued Learning	1-25

ILLUSTRATIONS

Fig. No.		Page
1-1	The role of the editor in programming	1-2
1-2	Screen representation	1-3
1-3	Scrolling	1-5
1-4	Paging	1-6
1-5	TEKTRONIX CT8500 terminal keyboard	1-7
1-6	Configurable command locations on the CT8500 keyboard.....	1-7

Section 1

LEARNING GUIDE

INTRODUCTION

This Learning Guide provides a general overview of the 8500 MDL Series Advanced CRT-Oriented Editor (ACE), shows you how to install the editor and configurator, and offers you a simple demonstration for hands-on experience. When you are finished with this Learning Guide, you should be able to do simple editing and be ready to learn more about ACE by reading other sections of this manual.

NOTE

Throughout this manual, "ACE" refers to the 8500 MDL Series Advanced CRT-Oriented Editor.

This Learning Guide is divided into the following topics:

- **Editor Overview.** Gives an overview of the editor and its general features.
- **Installing the Editor and Configurator.** Provides step-by-step instructions for installing ACE and the configurator for use with your system.
- **Demonstration Run.** Shows you how to invoke ACE, enter text, save the text in a file, and reedit an old file.
- **For Continued Learning.** Helps you decide where to go next in this manual to accomplish your own tasks.

EDITOR OVERVIEW

Uses of the Editor

The text editor is an important programming tool. It is useful whenever text must be created and modified. You can use the editor to enter source programs, to prepare command files and data files, and to create documents. Figure 1-1 is a diagram of typical work flow, showing tasks that are suitable for the editor.

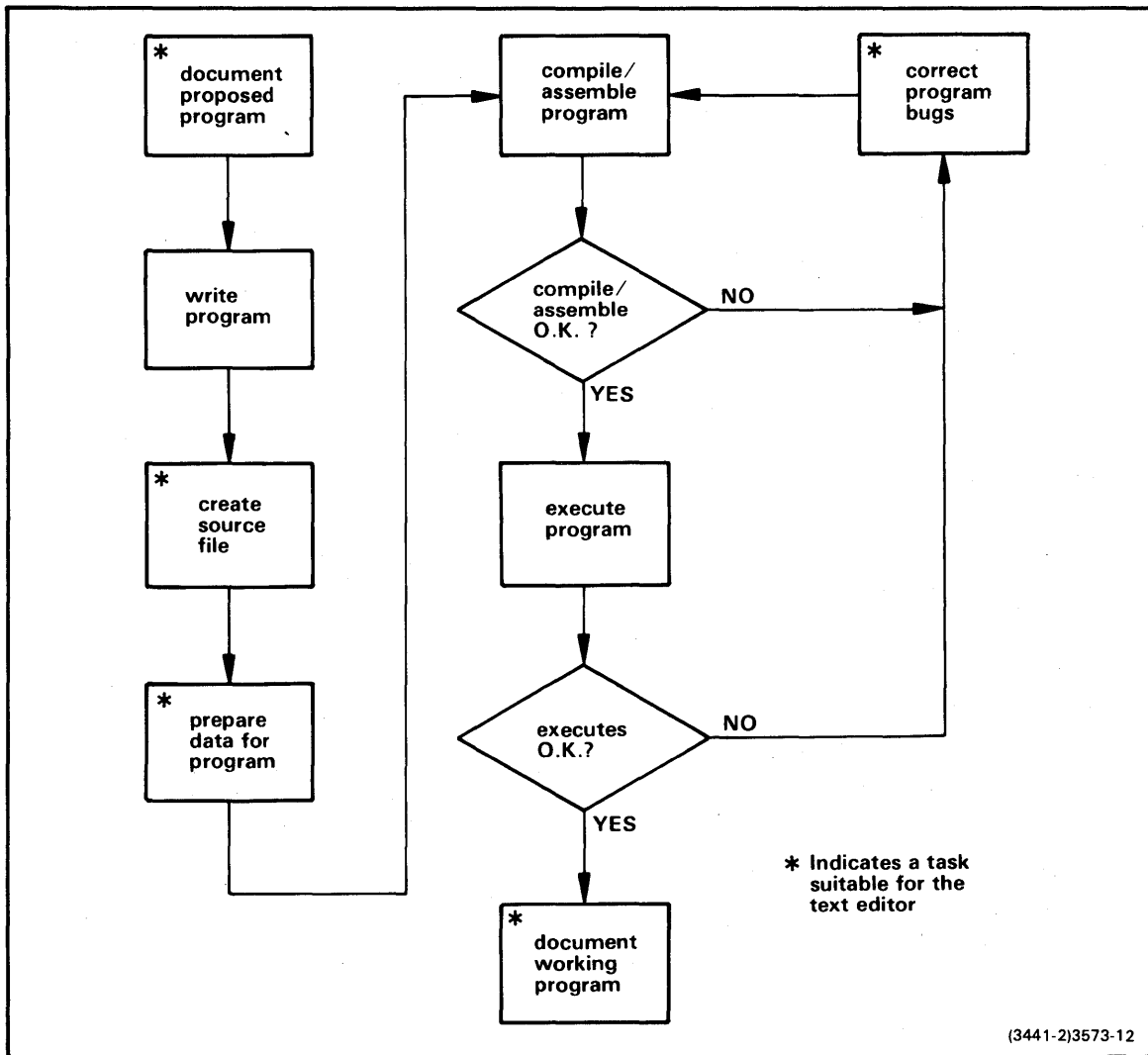


Fig. 1-1. The role of the editor in programming.

The editor is used to create and modify source programs, data files, and documentation. The editor may also be used to create system command files and editor command files.

General Features

ACE is a screen-oriented text editor. It allows you to create, view, and change text using a CRT terminal attached to an 8500-Series system. Using ACE, you can edit your files quite easily.

Here are some features of ACE:

- The screen is divided into two areas: the window and the monitor area.
- The cursor can be moved anywhere within the window.
- Text can be moved (scrolled or paged) through the window forward, backward, left or right.

- The original input file will not be overwritten. A backup copy of the original input file is made implicitly by ACE, or explicitly by the user.
- Text can be read from or written to secondary files.
- Command names can be entered in either upper or lower case.
- A visible end-of-line character is displayed at the end of each line.
- Non-displayable characters may be viewed and manipulated.
- Deleted text is placed into a buffer, and can be restored prior to another deletion.
- The length of a line can be up to 999 characters.
- Lines of text can be split or concatenated.
- Wildcard characters can be defined and used.
- Macros can be defined and executed.
- Command files can be created and executed.
- Several ACE features can be turned on or off to suit the needs of the current editing session.
- ACE can be configured to work with different CRT terminals.

The Screen

ACE is screen-oriented. The screen of your terminal acts as a window into the file that you are editing. Figure 1-2 represents a CRT screen and shows the significant areas of the screen.

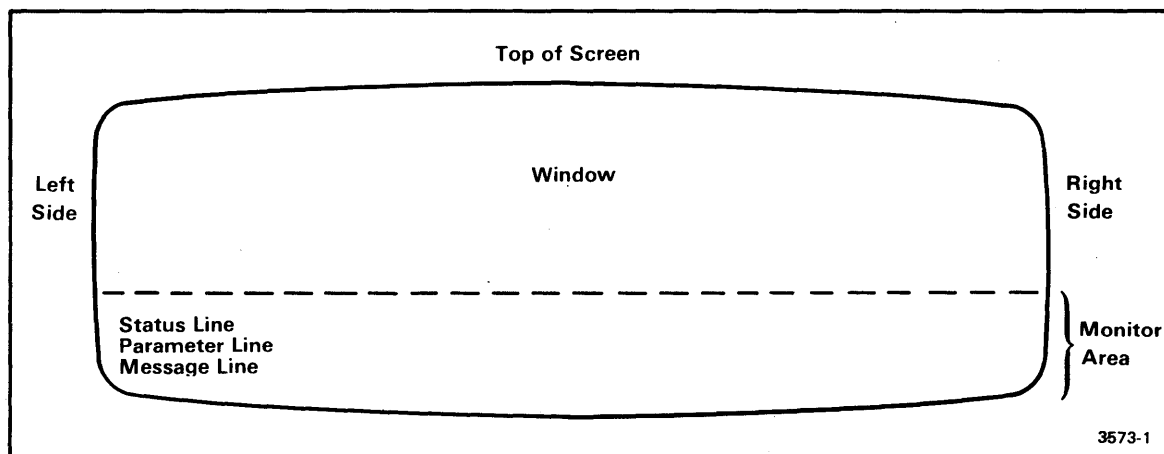


Fig. 1-2. Screen representation.

This illustration represents how ACE uses a CRT screen. The significant areas of the screen are labeled here. The dashed line separates the window from the monitor area.

The **status line** contains the fields that monitor the various states of the editor.

The **parameter line** contains the various parameters that are required by commands.

The **message line** is used to display error or warning messages, to query for acknowledgment of actions, and to display other general messages.

Refer to the Command Dictionary of this manual for more information on each line in the monitor area.

The Cursor

The cursor in the window represents the location of the pointer into the file. It can be moved anywhere within the window to effectively point at a place where editing is to occur.

Different terminals may have different cursor notations. In this manual, the cursor is represented by an underscore ().

Scrolling

Scrolling commands move the text in the window up, down, left, or right. During scrolling, the cursor remains in the same relative position in the window. Figure 1-3 illustrates the scrolling function.

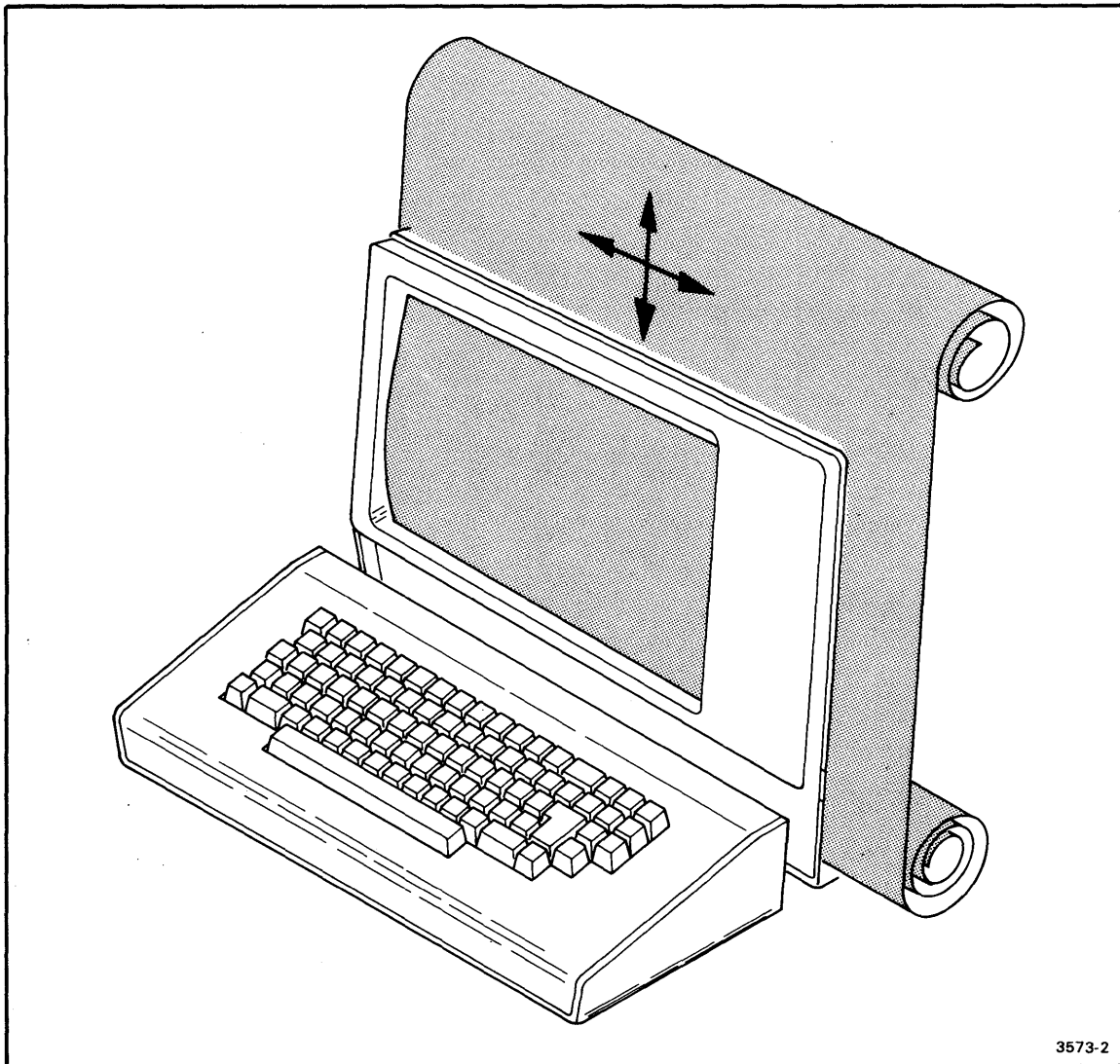


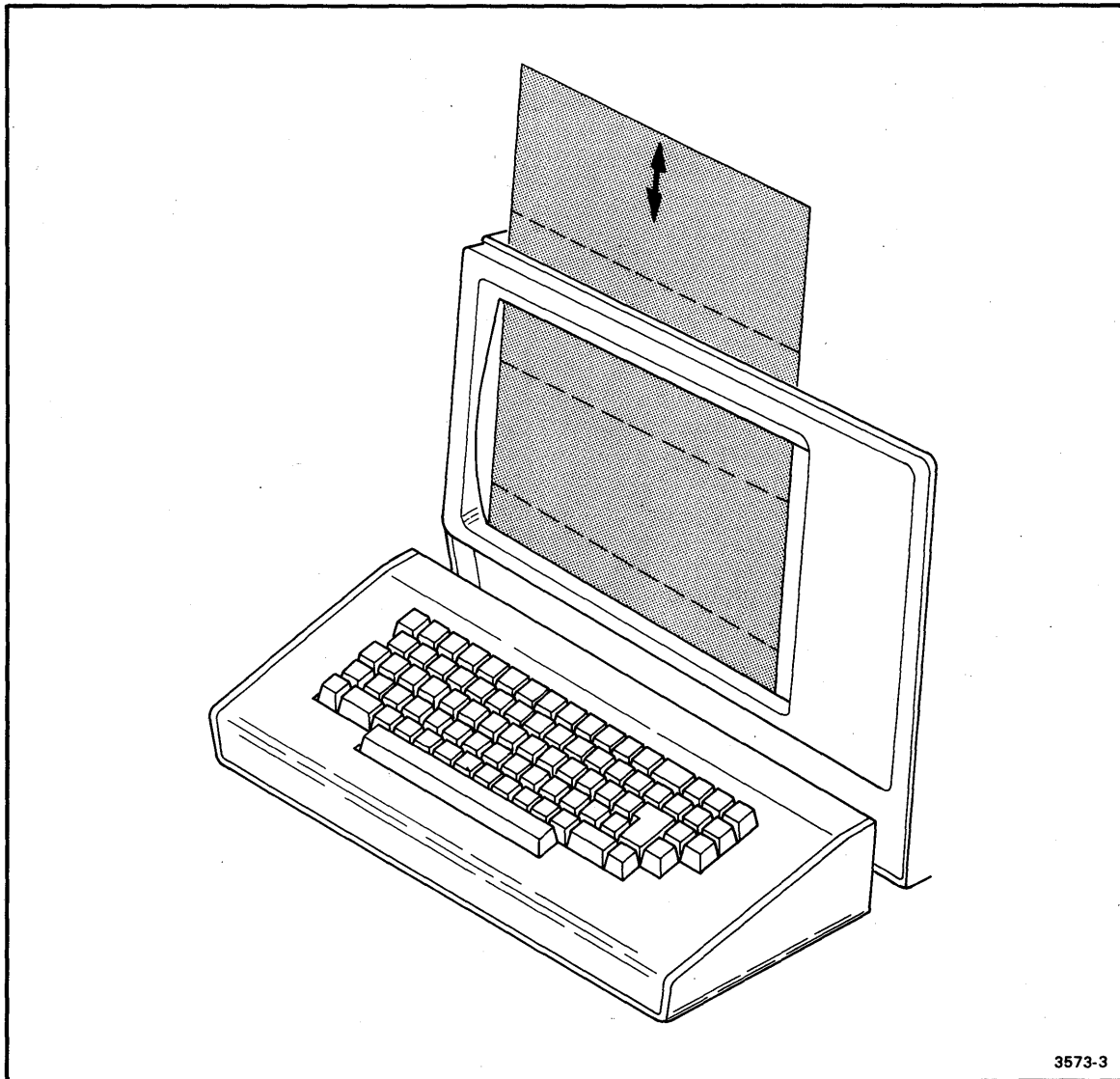
Fig. 1-3. Scrolling.

Paging

Paging commands move the text in the window up or down a page at a time. The size of a page is the same as the size of the window. The cursor remains in the same relative position in the window. Figure 1-4 illustrates the paging function.

NOTE

Make sure that you don't confuse the paging function and the window-sized "page" with actual pages of printed text in a document.



3573-3

Fig. 1-4. Paging.

Editor Commands

There are two types of ACE commands: **configurable** commands and **non-configurable** commands.

Configurable commands are assigned to special keys during the configuration process. Typically, function keys or control key combinations are used. Configurable commands are used for moving the cursor in the window, for scrolling and paging text through the window, for inserting and deleting characters and lines, for entering and exiting revise mode, and for

marking the cursor's position. The keyboard location for each configurable command depends on how your terminal is configured. Technical Note 1, in Section 5 of this manual, describes the configuration process in detail.

In this manual, the configurable commands are represented by **lowercase** letters enclosed **within** parentheses. Refer to Table 4-1, in the Tables section of this manual, for the representation of each configurable command.

Figure 1-5 shows the TEKTRONIX CT8500 terminal keyboard. Figure 1-6 shows where the configurable commands are located on the CT8500 keyboard. If your terminal is **not** a TEKTRONIX CT8500, your keyboard layout may be different.

Non-configurable commands are entered by pressing a specific key or keys on the keyboard. Command names can be entered in either upper or lower case.

In this manual, the non-configurable commands are represented by **uppercase** letters **without** enclosing parentheses.

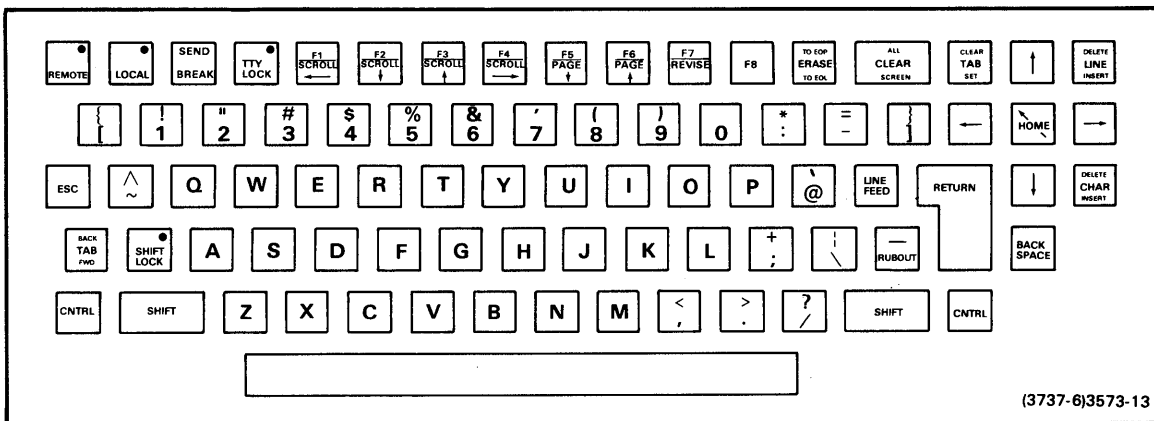


Fig. 1-5. TEKTRONIX CT8500 terminal keyboard.

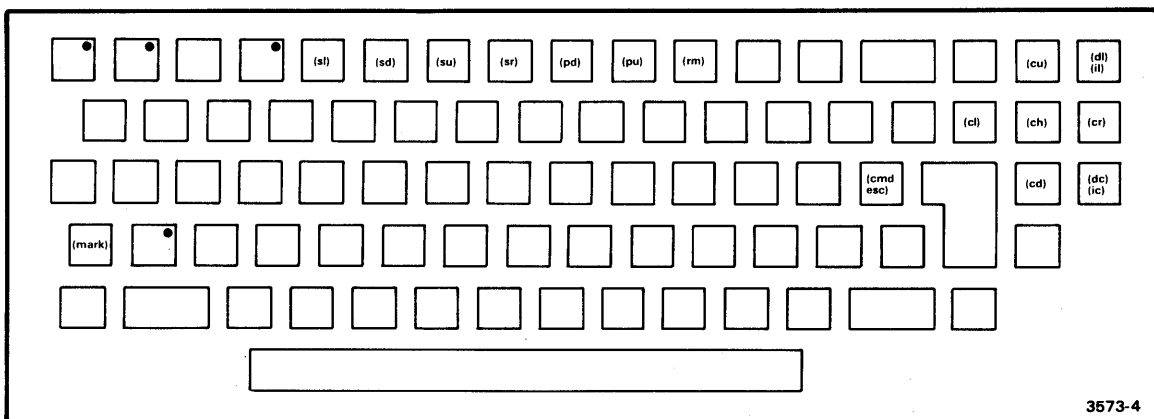


Fig. 1-6. Configurable command locations on the CT8500 keyboard.

Notation Conventions

This manual uses the following conventions in presenting information that is entered or displayed on the system terminal:

- **Underlined text** indicates what you enter from the terminal.
- **Text not underlined** represents the system response displayed on the screen.
- **Lowercase letters enclosed within parentheses** denote the configurable commands. For example, the symbol (cr) indicates the cursor right → command.
- **Uppercase letters enclosed within parentheses** denote the special keys you press. For example, the symbol (CR) indicates the carriage return.
- **Uppercase letters without enclosing parentheses** denote the non-configurable commands. For example, EX is the exit editor command.

INSTALLING THE EDITOR AND CONFIGURATOR

Since ACE can be configured to work with different CRT terminals, you must somehow tell the editor the various codes generated by your terminal's keyboard, the codes required to perform various screen functions, and the physical characteristics of the screen. This process is called **configuration**. The ACE configurator is a program used to create a file that contains the configuration information for a particular CRT terminal.

The ACE editor program and the ACE configurator program are both provided on the ACE installation disc. To use ACE and the configurator, you must transfer both programs to the DOS/50 system disc. This subsection shows you how to install ACE and the configurator for use with an 8550 Microcomputer Development Lab. If you are using any other TEKTRONIX 8500-Series MDL, refer to the Technical Notes section of this manual for a similar procedure.

Equipment Needed

To complete this installation procedure, you will need the following items:

- an 8550 Microcomputer Development Lab with 64K bytes of memory;
- a DOS/50 system disc with a write-enable tab over the write-protect slot; and
- an ACE installation disc.

You will need about ten minutes to complete this installation procedure.

Start up and Set the Date

Turn on your 8550 system, place your system disc in drive 0, and shut the drive 0 door. When you see the > prompt on your system terminal, place your installation disc in drive 1 and shut the drive 1 door.

Use the DATE command to set the date and time. For example, if it is 11:05 am on October 30, 1981, type:

```
> DATE 30-OCT-81/11:05(CR)
```

The system uses this information when it sets the CREATION TIME attribute of each file copied to your system disc.

Do You Have Room on Your System Disc?

This procedure adds 5 files, occupying about 200 blocks, to your system disc. Enter the LDIR command to check the number of free files and free blocks on your system disc.

```
> LDIR(CR)
```

Filename

list of files in your system volume directory

```
Files used   aa
Free files   bbb ← must be at least 5
Free blocks  ccc ← must be at least 200
Bad blocks   0
```

If you have fewer than 5 free files or 200 free blocks, you must delete some files from your system disc or get a new system disc and start the installation procedure over again.

Install the Software

The command file INSTALL, which is used to install the software, resides on the ACE installation disc. To execute this command file, type its filespec:

```
> /VOL/ACE/INSTALL(CR)
```

DOS/50 responds with the following message:

```
* During this installation procedure, one or more of the
* following messages may appear.  IGNORE THESE MESSAGES:
*
*      Error 6E - Directory alteration invalid
*      Error 7E - Error in command execution
*      Error 1D - File not found
*
* If any OTHER error message appears, see your
* Users Manual for further instructions.
*
* If no other error message appears, you'll receive a
* message when the installation procedure is complete.
*
TYPE,OFF
```

The installation process generates a number of error messages that do not affect the success of the installation. However, if any message **other** than 6E, 7E, or 1D appears, take the following steps:

1. Make sure you are using the right discs.
2. Make sure your system disc has a write-enable tab.
3. Make sure there are at least 5 free files and 200 free blocks on your system disc.
4. Begin the installation procedure again.

If the installation procedure fails again, copy down the error message and contact your Tektronix service representative.

"TYPE,OFF", the first command in the INSTALL command file, is displayed before it is executed. This command suppresses subsequent output to your system terminal (except error messages) until INSTALL finishes executing.

Within about five minutes, INSTALL will finish and your system terminal will display the following message:

```
* ACE and the ACE configurator are now installed, please reboot !!!!!
```

Your software is installed, and you can:

- remove your discs and turn off your 8550 system, or
- reboot the system and install more software onto your system disc, or
- reboot the system and continue with the ACE Demonstration Run that follows in this section.

ACE DEMONSTRATION RUN

Introduction

This demonstration presents some basic features of ACE for use with an 8550 Microcomputer Development Lab. If you are using any other TEKTRONIX 8500-Series MDL, refer to the Technical Notes section of this manual for a demonstration run that illustrates how ACE works with your system.

In this demonstration, you use ACE to create and modify a DOS/50 command file named COPYDISC. Refer to the 8550 System Users Manual for information on DOS/50 commands and command files.

In order to perform this demonstration, you need:

- An 8550 Microcomputer Development Lab with 64K bytes of memory.
- A DOS/50 system disc containing ACE and the configurator, with a write-enable tab over the write-protect slot.

This demonstration takes about 30 minutes.

Start Up and Log On

NOTE

If your 8550 is already on, toggle the 8301 RESTART switch upward to establish the initial conditions necessary for this demonstration. Wait for the > prompt before proceeding.

Turn on your 8550 system. Place your system disc in drive 0 and shut the drive 0 door. When your system terminal displays the > prompt, enter the following command line to establish ME as the owner of the files you will create in this demonstration:

```
> USER, ,ME(CR)
```

CAUTION

Be sure you have write permission in the current directory. Otherwise, you cannot save the files you create or the modifications you make.

Now use the DATE command to set today's date and time. For example, if it is 11:05 am on October 31, 1981, enter the following command line:

```
> DATE 31-Oct-81/11:05(CR)
```

How to Correct Input Mistakes

Before you begin to input text, you need to know how to correct your mistakes on the input line.

When you are entering text, you may use the BACKSPACE or RUBOUT key to delete characters one-by-one. Either key will backspace the cursor and erase the deleted character. When you are entering an ACE command or a command parameter, you may also use the CNTRL-x (hold down the CNTRL key and press x) to cancel the current command or parameter being entered.

Create a Text File

Now you know how to correct input mistakes. Let's invoke ACE and create a text file.

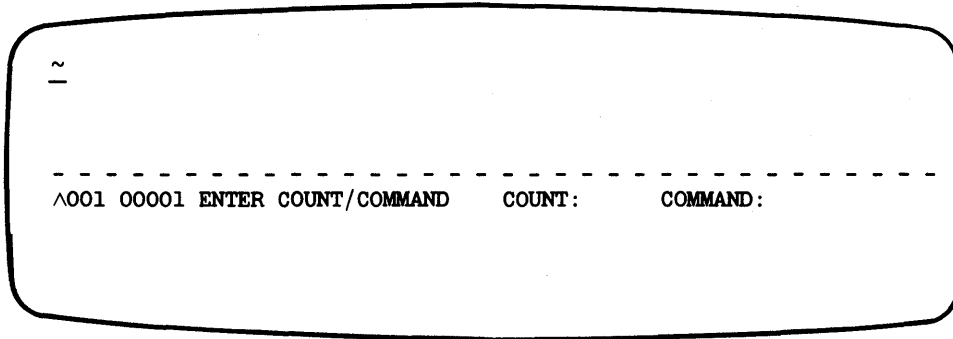
NOTE

If you are using any terminal other than a TEKTRONIX CT8500, you must first make sure that the terminal has been configured for use with ACE. Refer to Technical Note 1 in Section 5 of this manual for information on how to configure your terminal for ACE.

To invoke ACE and create a new file in the current directory, enter the following command:

```
> ACE COPYDISC(CR)  
Version 1.x
```


DOS/50 invokes ACE, creates a new file named COPYDISC on the system disc, clears the window, shows the status line, and moves the cursor to the top left corner of the window (home position). The end-of-line character (~) is supplied automatically by ACE when you create a new file.



If the window is not empty, then a file named COPYDISC already exists in the current directory. Use the S command to exit ACE without harming the file. Then reinvoke ACE, using a new file name.

"ENTER COUNT/COMMAND" in the status line indicates that ACE is ready to accept an editor command.

NOTE

The type-ahead buffer has a capacity of 128 characters. If you type faster than the editor can execute, the buffer will overflow. DOS/50 will sound the bell and ignore any excess characters.

Enter Text

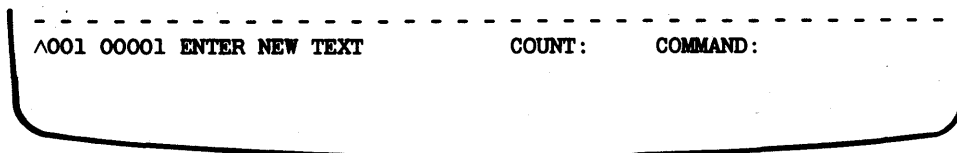
To insert text, press the following key:

(ic)



When you are inserting text or entering a command parameter, do not press any configurable keys. On certain CRT terminals, the key codes of some configurable commands contain an ESC character, which terminates the insert command and parameter entries.

The monitor area now looks like this:



"ENTER NEW TEXT" in the status line indicates that you are now ready to enter your text.

Enter the following text:

```

*** DUPLICATE A SYSTEM DISC *** (CR)
*(TAB) This command file duplicates a system disc. (CR)
*(CR)
*(TAB) Place the system disc in drive 0 and the disc to be (CR)
*(TAB) formatted in drive 1. Type CONT * to continue. (CR)
SUSPEND * (CR)
FORMAT /DEV/FLX1 (CR)
VERIFY /DEV/FLX1 (CR)
COPY:L:B /DEV/FLX0 /DEV/FLX1 (ESC)

```

↑
Press the (ESC) key to
terminate the (ic) command.

When you press the (ESC) key, the editor terminates the (ic) command. Note that when you enter a return (CR), a tilde ~ is displayed and the cursor is moved to the start of the next line. If you do not want the end-of-line character (~) to be visible, you can configure it to a non-displayable character. Refer to Technical Note 1 in Section 5 of this manual for information on the configuration process.

The (TAB) key moves the cursor to the next tab setting. ACE has default tab settings at every 8th column, beginning with column 9.

Now the screen looks like this:

```

*** DUPLICATE A SYSTEM DISC ***~
*   This command file duplicates a system disc.~
*~
*   Place the system disc in drive 0 and the disc to be~
*   formatted in drive 1. Type CONT * to continue.~
SUSPEND *~
FORMAT /DEV/FLX1~
VERIFY /DEV/FLX1~
COPY:L:B /DEV/FLX0 /DEV/FLX1~

```

```

-----
^001 00001 ENTER COUNT/COMMAND   COUNT:   COMMAND:

```

"ENTER COUNT/COMMAND" appears again in the status line, indicating that you may enter the next ACE command.

Save the Text in a File

Suppose you want to store this text into the file COPYDISC. Enter:

EX

```

-----
^O01 00001 ENTER                COUNT:  COMMAND:EX
FILE ACCESS IN PROGRESS

```

The EX command saves the contents of the workspace into the file and exits from the editor. The screen is cleared and control is returned to the operating system. The > prompt tells you that DOS/50 is ready for the next DOS/50 command.

List Directory

Enter the following DOS/50 command to list the files in the current directory:

> LDIR(CR)

```

Filename
:
COPYDISC ← the new file you created with ACE
:

```

```

Files used      85
Free files      171
Free blocks    1000
Bad blocks      0

```

— refers to your system disc

Modify a Disc File

Let's alter the text that you just created. (You could have changed the text before you saved it, but we'll begin a new editing session to illustrate certain features of the ACE editor.)

NOTE

ACE creates some temporary files during the editing session. These files are deleted when you exit the editor. However, if the system crashes while you are editing in ACE, these temporary files remain on the disc. The temporary file names all begin with ###.ace.temp. Use the DOS/50 command DELETE:N ###.ace.temp to delete these files.*

To invoke ACE, enter the following command:

```
> ACE COPYDISC(CR)
Version 1.x
```

```
*** DUPLICATE A SYSTEM DISC ***~
*   This command file duplicates a system disc.~
*~
*   Place the system disc in drive 0 and the disc to be~
*   formatted in drive 1. Type CONT * to continue.~
SUSPEND *~
FORMAT /DEV/FLX1~
VERIFY /DEV/FLX1~
COPY:L:B /DEV/FLX0 /DEV/FLX1~
```

```
-----
^001 00001 ENTER COUNT/COMMAND   COUNT:   COMMAND:
```

Notice that the window is **not** empty: the file named COPYDISC already exists in the current directory. The window contains the first page of COPYDISC. The cursor is at the home position.

Move the Cursor

There are five cursor motion commands: (cd), (cu), (cr), (cl), and (ch). These commands move the cursor down ↓, up ↑, right →, left ←, and home (to the upper left position of the screen). Experiment with these commands and watch the cursor move.

Now press the following key sequence to move the cursor to the second column of the second line.

```
(ch)
(cd)
(cr)
```

Move the Cursor in the Tabbed Area

Remember that the blank areas after the asterisks (*) in lines 2, 4, and 5 are caused by tab expansion. Press the following key and watch the cursor move right:

(cr)

```

*** DUPLICATE A SYSTEM DISC ***~
*   This command file duplicates a system disc.~
*~
*   Place the system disc in drive 0 and the disc to be~
*   formatted in drive 1. Type CONT * to continue.~
SUSPEND *~
FORMAT /DEV/FLX1~
VERIFY /DEV/FLX1~
COPY:L:B /DEV/FLX0 /DEV/FLX1~

-----
^001 00001 ENTER COUNT/COMMAND   COUNT:   COMMAND:
    
```

Notice that even though you pressed (cr) only once, the cursor moved more than one column. Since the entire blank area is represented by a single tab character, only one (cr) is required to move the cursor past the blank area.

Now press the following key to move the cursor left one character:

(cl)

The cursor is now returned to the second column of the second line.

Delete a Tabbed Area

The (dc) command deletes the current character. Since the entire tabbed area is represented by a single tab character, only one (dc) is required to remove the tab character, thereby removing the tabbed area as well.

To delete the current tabbed area, enter:

(dc)

The tabbed area is removed. The remaining text of the line is shifted left, as shown in the following display:

```

*** DUPLICATE A SYSTEM DISC ***~
*This command file duplicates a system disc.~
*~
*      Place the system disc in drive 0 and the disc to be~
*      formatted in drive 1. Type CONT * to continue.~
SUSPEND *~
FORMAT /DEV/FLX1~
VERIFY /DEV/FLX1~
COPY:L:B /DEV/FLX0 /DEV/FLX1~

```

```

-----
^001 0001 ENTER COUNT/COMMAND   COUNT:   COMMAND:

```

Execute a Command Repetitively

Most ACE commands can be executed repetitively by preceding the command with a repetition count. For example, to move the cursor right 18 characters, enter the following command:

18(cr)

```

*** DUPLICATE A SYSTEM DISC ***~
*This command file duplicates a system disc.~
*~
*      Place the system disc in drive 0 and the disc to be~
*      formatted in drive 1. Type CONT * to continue.~
SUSPEND *~
FORMAT /DEV/FLX1~
VERIFY /DEV/FLX1~
COPY:L:B /DEV/FLX0 /DEV/FLX1~

```

```

-----
^001 0001 ENTER COUNT/COMMAND   COUNT:18  COMMAND:

```

The cursor is moved 18 characters right, to the beginning of the word "duplicates". The number "18" in the status line indicates that a non-default repetition count was entered.

Revise (Overwrite) Text

The cursor is at the beginning of the word "duplicates". Suppose you want to change the remainder of the line to be more descriptive. You can use revise mode.

When the editor is in revise mode, most characters that are entered from the keyboard will overwrite characters already in the window. The (rm) command is used to enter and exit the revise mode. Enter:

(rm)

```
-----
^001 00001 ENTER COUNT/COMMAND      COUNT:      COMMAND:      REVISE
```

The word "REVISE" in the status line shows that the editor is in revise mode. "REVISE" will remain in the status line until you press the (rm) key again, to exit the revise mode.

Watch how ACE overwrites the current line as you enter the following text:

In revise mode, (CR) overwrites the
current character, which is blank,
and splits the line at this point.

```
prepares a blank disc, then(CR)
*copies the contents of the system disc in drive 0 to(CR)
*the blank disc in drive 1.(rm)
```

Press the (rm) key
to exit revise mode.

Now the screen looks like this:

```
*** DUPLICATE A SYSTEM DISC ***~
*This command file prepares a blank disc, then~
*copies the contents of the system disc in drive 0 to~
*the blank disc in drive 1.~
*~
*      Place the system disc in drive 0 and the disc to be~
*      formatted in drive 1. Type CONT * to continue.~
SUSPEND *~
FORMAT /DEV/FLX1~
VERIFY /DEV/FLX1~
COPY:L:B /DEV/FLX0 /DEV/FLX1~

-----
^001 00001 ENTER COUNT/COMMAND      COUNT:      COMMAND:
```

Notice that "REVISE" is no longer present in the status line.

Find a String

Suppose you want to include the :N parameter in the FORMAT command line, so that DOS/50 will format the disc without querying you. The F command searches for a specified string from the cursor towards the end-of-file. To locate the string "FORMAT", first enter:

F

ACE then prompts you to enter the search string. You enter:

```
-----
^001 00001 ENTER SEARCH STRING      COUNT:      COMMAND:F
FORMAT(ESC)
```

The cursor moves to the right of the string "FORMAT", as shown in the following display:

```
*** DUPLICATE A SYSTEM DISC ***~
*This command file prepares a blank disc, then~
*copies the contents of the system disc in drive 0 to~
*the blank disc in drive 1.~
*~
*      Place the system disc in drive 0 and the disc to be~
*      formatted in drive 1. Type CONT * to continue.~
SUSPEND *~
FORMAT_/DEV/FLX1~
VERIFY /DEV/FLX1~
COPY:L:B /DEV/FLX0 /DEV/FLX1~

-----
^001 00001 ENTER COUNT/COMMAND      COUNT:      COMMAND:F
FORMAT
```

Notice that the search only looked for uppercase "FORMAT". Uppercase and lowercase letters are **not** interchangeable in parameter entries.

Insert Characters

Now, you can insert the :N parameter in the FORMAT command line. Enter:

(ic)
:N(ESC)

Here's how the screen looks now:

```

*** DUPLICATE A SYSTEM DISC ***~
*This command file prepares a blank disc, then~
*copies the contents of the system disc in drive 0 to~
*the blank disc in drive 1.~
*~
*      Place the system disc in drive 0 and the disc to be~
*      formatted in drive 1. Type CONT * to continue.~
SUSPEND *~
FORMAT:N_/DEV/FLX1~
VERIFY /DEV/FLX1~
COPY:L:B /DEV/FLX0 /DEV/FLX1~

-----
^001 00001 ENTER COUNT/COMMAND      COUNT:      COMMAND:
    
```

Insert a Line

Now suppose you want to suppress display of the DOS/50 commands during the command file execution. To do this, you need to insert the TYPE OFF command line before the SUSPEND command.

The cursor is at the line containing the FORMAT command. Enter the following command to move the cursor up two lines:

2(cu)

The (il) command inserts a new line below the cursor. Enter:

(il)

```

-----
^001 00001 ENTER NEW TEXT      COUNT:      COMMAND:
    
```

The (il) command has created a blank line with a end-of-line character (~). Now, you can insert text. Enter:

TYPE OFF(ESC)

The (ESC) key terminates the (il) command. The screen now looks like this:

```

*** DUPLICATE A SYSTEM DISC ***~
*This command file prepares a blank disc, then~
*copies the contents of the system disc in drive 0 to~
*the blank disc in drive 1.~
*~
*           Place the system disc in drive 0 and the disc to be~
*           formatted in drive 1. Type CONT * to continue.~
TYPE OFF~
SUSPEND *~
FORMAT:N /DEV/FLX1~
VERIFY /DEV/FLX1~
COPY:L:B /DEV/FLX0 /DEV/FLX1~

-----
^O01 00001 ENTER COUNT/COMMAND   COUNT:   COMMAND:

```

Move a Block of Text

The TYPE OFF command suppresses display of the command file as it is executed. Suppose you want to suppress the following text during command file execution also:

```

*This command file prepares a blank disc, then
*copies the contents of the system disc in drive 0 to
*the blank disc in drive 1.
*

```

You can accomplish this by moving the block of text to follow the TYPE OFF command.

To move a block of text, you first delete the block of text, then move the cursor to the desired position and restore (undelete) the deleted text.

First, move the cursor up 6 lines, so that the cursor is at the second line. Enter:

6(cu)

To delete the current line and the next 3 lines, enter:

4(dl)

Now move the cursor to the position where you want to insert the deleted block of text, as shown in the following display:

```

*** DUPLICATE A SYSTEM DISC ***~
*      Place the system disc in drive 0 and the disc to be~
*      formatted in drive 1. Type CONT * to continue.~
TYPE OFF~
SUSPEND *~
FORMAT:N /DEV/FLX1~
VERIFY /DEV/FLX1~
COPY:L:B /DEV/FLX0 /DEV/FLX1~

-----
^O01 0001 ENTER COUNT/COMMAND      COUNT:      COMMAND:
    
```

Text will be restored (undeleted) before the cursor. To insert the deleted text before the new cursor position, enter:

U

Now the screen looks like this:

```

*** DUPLICATE A SYSTEM DISC ***~
*      Place the system disc in drive 0 and the disc to be~
*      formatted in drive 1. Type CONT * to continue.~
TYPE OFF~
*This command file prepares a blank disc, then~
*copies the contents of the system disc in drive 0 to~
*the blank disc in drive 1.~
*~
SUSPEND *~
FORMAT:N /DEV/FLX1~
VERIFY /DEV/FLX1~
COPY:L:B /DEV/FLX0 /DEV/FLX1~

-----
^O01 0001 ENTER COUNT/COMMAND      COUNT:      COMMAND:U
    
```

Replace a String

Now, suppose you want to replace :L in the COPY command with :N. The :N parameter tells DOS/50 to copy the disc without querying you; DOS/50 will not log each copy operation on the terminal. You enter:

R

Notice that on the status line, the editor prompts for SEARCH STRING. You enter:

```
-----
^O01 00001 ENTER SEARCH STRING      COUNT:      COMMAND:R
:L(ESC)
```

Then the editor prompts for REPLACEMENT STRING, and you enter:

```
-----
^O01 00001 ENTER REPLACEMENT STRING COUNT:      COMMAND:R
:L :N(ESC)
```

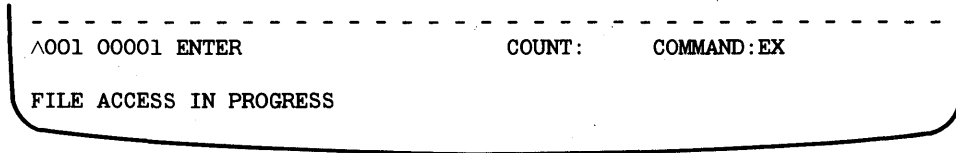
The string :L is replaced by :N, and the screen looks like this:

```
*** DUPLICATE A SYSTEM DISC ***~
*      Place the system disc in drive 0 and the disc to be~
*      formatted in drive 1. Type CONT * to continue.~
TYPE OFF~
*This command file prepares a blank disc, then~
*copies the contents of the system disc in drive 0 to~
*the blank disc in drive 1.~
*~
SUSPEND *~
FORMAT:N /DEV/FLX1~
VERIFY /DEV/FLX1~
COPY:N;B /DEV/FLX0 /DEV/FLX1~
-----
^O01 00001 ENTER COUNT/COMMAND      COUNT:      COMMAND:R
:L :N
```

Exit the Editor

Now use the EX command to save the changed text in the file and exit the editor. The changed text will replace the old text in the file.

EX



The original version of COPYDISC is saved automatically under the backup name #COPYDISC. Enter the following DOS/50 command to list the files in the current directory:

> LDIR(CR)

```

Filename
:
#COPYDISC ← the backup version of COPYDISC
COPYDISC
:

Files used      86
Free files     170
Free blocks    1000
Bad blocks      0
    
```

Summary

You have now finished the Demonstration Run. It showed you how to:

- invoke ACE, using the ACE command;
- exit ACE, using the EX command;
- insert text, using the (ic) and (il) commands;
- position the cursor within the window, using the cursor motion commands;
- delete a tabbed area, using the (dc) command;
- execute a command repetitively by preceding the command with a number;
- modify text in revise mode;
- find a string, using the F command;
- move a block of text, using the delete and undelete commands;
- replace a string, using the R command.

If you prefer, you can leave the two files you have created for future reference. However, if you want to delete these files, you can do so with the following command:

> DELETE:N COPYDISC #COPYDISC(CR)

FOR CONTINUED LEARNING

This Learning Guide explained the elementary ACE concepts. For a more detailed explanation, refer to the following sections of this manual:

- **Section 2, Procedures.** Describes a series of tasks, and lists the commands needed to perform these tasks. Each task includes one or more examples.
- **Section 3, Command Dictionary.** Provides a formal description of each ACE command, its operation, and its syntax. Most command descriptions have illustrative examples. The Command Dictionary is arranged alphabetically by command name.
- **Section 4, Tables.** Summarizes reference information in tables and charts.
- **Section 5, Technical Notes.** Explains the configuration process by which ACE learns the various codes generated by a terminal's keyboard, the codes required to affect the screen, and the physical size of the screen. Contains an installation procedure and demonstration run for other 8500-Series MDLs.
- **Section 6, Error Messages.** Lists the error messages and general messages for ACE. Each message is accompanied by a description of the problem or display.
- **Section 7, Glossary.** Defines special terms used in this manual.
- **Section 8, Index.** Gives you a place to start when you don't know where else to look.

Section 2 PROCEDURES

	Page		Page
Introduction	2-1	Block Manipulation	2-27
The Essentials	2-2	Deleting a Block of Text	2-27
Invoking the Editor	2-2	Moving a Block of Text	2-28
Exiting the Editor	2-2	Saving a Block of Text in a File	2-30
Creating a File	2-3	Duplicating a Block of Text	2-32
Modifying an Existing File	2-5	Printing a Block of Text on the Line Printer (8550 Only).....	2-35
Getting Help	2-7	Splitting a File	2-36
Moving the Cursor to the Beginning of the Line ..	2-7	Concatenating Files	2-38
Moving the Cursor to the End of the Line	2-8	Case Manipulation	2-39
Moving the Cursor to the Beginning of the File ..	2-9	Converting All Uppercase Letters to Lowercase ..	2-39
Moving the Cursor to the End of the File	2-10	Converting All Occurrences of a String to Uppercase	2-41
Resetting the Tab Stops	2-11	Capitalizing All Words Following a Period and Two Spaces	2-43
Entering Commands in Revise Mode	2-12	Capitalizing All Words Following a Period— End-of-Line Sequence	2-45
Character Manipulation	2-13	Command Shortcuts	2-47
Finding a String	2-13	Creating and Using Command Macros	2-47
Counting the Number of Occurrences of a String	2-14	Creating and Using Command Files	2-51
Replacing a String	2-16		
Making Global Replacements	2-17		
Line Manipulation	2-19		
Viewing Blanks in Lines	2-19		
Manipulating Long Lines	2-21		
Moving a Line	2-21		
Splitting a Line	2-22		
Concatenating Lines	2-23		
Changing Text from Cursor to the Beginning of Line	2-24		
Changing Text from Cursor to the End of Line ..	2-25		

Section 2

PROCEDURES

Section 1, the Learning Guide, gave you a general overview of ACE and presented a simple demonstration run. This section presents some common procedures for using ACE with your 8500-Series MDL.

Each procedure in this section is presented in the following format:

Description: A summary of the action(s) performed by the procedure.

Procedure: The information entered or displayed at the system terminal. The following conventions are used in the procedure description:

Underlined: A character sequence you enter.

No Underline: A character sequence displayed by ACE or by your system.

UPPERCASE: An exact character sequence; if these characters are underlined, enter them exactly as shown.

lowercase: A parameter you supply when you perform the procedure.

Braces { }: A required parameter you enter.

Parentheses (): A configurable or special key you press. Lowercase letters enclosed within parentheses represent configurable keys. Uppercase letters enclosed within parentheses represent special keys. Table 4-1, in the Tables section of this manual, lists the configurable commands.

Parameters: A description of the values you supply.

Comments: The operating limits and options for the procedure.

Examples: One or more demonstrations of correct entry format. For ease of illustration, the display windows in the examples are configured to contain only five lines. Otherwise, all configuration parameters are the same as the default (CT8500) configuration file. (See Technical Note 1 in Section 5 of this manual for a discussion of configuration files.)

See also: Cross-references to related procedures.

THE ESSENTIALS

Invoking the Editor

Description: This procedure invokes the editor and specifies the file to be created or edited.

Procedure: > ACE filespec(CR)

Parameters: **filespec**—The disc file to be created or edited.

Comments: The full form of the ACE command is described in the Command Dictionary section of this manual. For routine applications, the short form is sufficient:

ACE filespec

If the specified file does not exist, it is created. If the file already exists, it is modified, and the old version is saved under a backup name.

Examples: > ACE MYFILE(CR)

The editor clears the screen, and positions the cursor in the top left corner of the window (home position).

If MYFILE already exists, it will be opened and made available for editing. The window contains the first page of text. All changes made during the edit session are written into MYFILE. If you exit the editor with the EX command, the editor automatically makes a backup copy (#MYFILE) of the original MYFILE.

If MYFILE did not previously exist, it will be created and the window will contain only the end-of-line character (~) in the home position. All text entered will be written into MYFILE.

See also:

- Creating a File
- Modifying an Existing File
- Exiting the Editor

Exiting the Editor

Description: There are two ways to exit ACE and return control to your system:

- The EX command copies the contents of workspace and the remainder of the unedited input file to the output file, saves the output file, and terminates the editing session.
- The S command terminates the editing session without saving the changes and additions.

Procedure: EX

or

S

Comments: When you exit the editor with the S command, no backup of the original file is made. Any commands following the EX or S command in a command file are ignored.

Examples: To exit the editor and update your file, enter:

EX

```

-----
^001 00001 ENTER                      COUNT:  COMMAND:EX
FILE ACCESS IN PROGRESS
  
```

To exit the editor **without** updating your file, enter:

S

```

-----
^001 00001 ENTER                      COUNT:  COMMAND:S
STOP REQUESTED, ENTER "Y" TO ACKNOWLEDGE:
  
```

Before exiting the editor without file update, ACE asks you whether you really want to exit the editor. You type "Y" or "y" for yes.

See also: ● Invoking the Editor

Creating a File

Description: This procedure creates a new file from text entered from the system terminal.

Procedure: > ACE newfile(CR)
 Version 1.x
 (ic)
 :
 Enter your text here.
 :
 (ESC)
EX

Parameters: **newfile**—The name of the file to be created.

Examples: > ACE MYNEWFILE(CR)
Version 1.x

```

~
-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:

```

MYNEWFILE is created and the window contains only the end-of-line character (~) in the home position. "ENTER COUNT/COMMAND" in the status line indicates ACE is ready for a command.

To enter new text, press:

(ic)

"ENTER NEW TEXT" appears in the status line. You may now enter your text. Use the (BACKSPACE) or (RUBOUT) key to correct input mistakes.

```

I HAVE JUST CREATED MY NEW FILE.(CR)
~
-----
^001 00001 ENTER NEW TEXT           COUNT:  COMMAND:

```

To terminate the insert command, press:

(ESC)

The status line again displays "ENTER COUNT/COMMAND". To exit the editor and save the text in MYNEWFILE, enter:

EX

The screen is cleared and your system prompt is displayed.

- See also:
- Invoking the Editor
 - Modifying an Existing File
 - Exiting the Editor

Modifying an Existing File

Description: This procedure replaces a previously created file (by ACE or otherwise) with an edited version.

Procedure:

```
> ACE oldfile(CR)
Version 1.x
:
Make the necessary changes in the file.
:
EX
```

Parameters: **oldfile**—The name of the file to be modified.

Comments: The file is modified according to the editor commands you enter. The old version is given a backup filename.

Examples:

```
> ACE MYOLDFILE(CR)
Version 1.x
```

```

A is for Advanced.~
C is for CRT.~
E is for Editor.~

-----
^001 0001 ENTER COUNT/COMMAND      COUNT:  COMMAND:
```

MYOLDFILE is open and available for editing. The window contains the first page of MYOLDFILE. Enter the following command sequence to replace the string CRT with CRT-Oriented:

```
R  
CRT(ESC)  
CRT-Oriented(ESC)
```

```
A is for Advanced.~  
C is for CRT-Oriented.~  
E is for Editor.~
```

```
-----  
^001 0001 ENTER COUNT/COMMAND      COUNT:  COMMAND:R  
CRT CRT-Oriented
```

You have made the desired changes in MYOLDFILE. To exit the editor and save the modified version in MYOLDFILE, enter:

```
EX
```

The screen is cleared and your system prompt is displayed. The original version of MYOLDFILE is saved automatically under the backup name #MYOLDFILE.

See also:

- Invoking the Editor
- Creating a File
- Exiting the Editor
- Replacing a String

Getting Help

Description: This procedure displays information about the ACE commands.

Procedure: H

The editor clears the current window and displays the command syntax.

The monitor area looks like this:

```

^001 0001 ENTER                      COUNT:  COMMAND:H
DEPRESS CARRIAGE RETURN TO CONTINUE: (CR)
    
```

Comments: The command syntax is displayed according to the format shown in Tables 4-3 and 4-4, in the Tables section of this manual. If more syntax information exists than can be displayed in one window, the next page of information will be shown after you press the return. After the last page of command syntax has been displayed, the window will return to the display existing at the time you invoked the H command.

Moving the Cursor to the Beginning of the Line

Description: This procedure moves the cursor to the first character of the current line.

Procedure: OA

Comments: If the beginning of the line is outside the window, the text will be scrolled right to show the beginning of the line.

Examples: Assume that the cursor is not at the beginning of the line, as shown in the following display:

```

his is the only line in the file.~
-----
^002 0001 ENTER COUNT/COMMAND      COUNT:  COMMAND:
    
```

"002" in the status line indicates that the left margin of the window is the second column of the line. To move the cursor to the beginning of the line, enter:

OA

Now the screen looks like this:

```
This is the only line in the file.~  
-----  
^001 00001 ENTER COUNT/COMMAND      COUNT:0  COMMAND:A
```

The text is scrolled right to show the beginning of the line.

- See also:
- Moving the Cursor to the End of the Line
 - Moving the Cursor to the Beginning of the File

Moving the Cursor to the End of the Line

Description: This procedure moves the cursor to the last character (end-of-line character) of the current line.

Procedure: /J

Comments: If the end-of-line character (~) is outside the window, the text will be scrolled left to show the end of the line.

Examples: Assume that the cursor is not at the end of line, as shown in the following display:

```
This is the only line in the file.~  
-----  
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:
```

To move the cursor to the end of the line, enter:

/J

Now the screen looks like this:

```

This is the only line in the file.~
-----
^001 00001 ENTER COUNT/COMMAND      COUNT:/  COMMAND:J

```

- See also:
- Moving the Cursor to the Beginning of the Line
 - Moving the Cursor to the End of the File

Moving the Cursor to the Beginning of the File

Description: This procedure moves the cursor to the first character in the file.

Procedure: -/A

Comments: If the beginning of the file is outside the window, the window will be redisplayed to show the beginning of the file.

Examples: Assume that the current screen does not contain the beginning of the file, as shown in the following display:

```

APRIL~
MAY~
JUNE~
JULY~
AUGUST~
-----
^001 00004 ENTER COUNT/COMMAND      COUNT:  COMMAND:

```


"00004" in the status line indicates that the top line in the window is the fourth line in the file. To move the cursor to the beginning of the file, enter:

-/A

Now the screen looks like this:

```

JANUARY~
FEBRUARY~
MARCH~
APRIL~
MAY~
-----
^001 00001 ENTER COUNT/COMMAND      COUNT: -/  COMMAND:A

BEGINNING OF FILE ENCOUNTERED

```

The text is scrolled down to show the beginning of the file.

- See also:
- Moving the Cursor to the Beginning of the Line
 - Moving the Cursor to the End of the File

Moving the Cursor to the End of the File

Description: This procedure moves the cursor to the last character in the file.

Procedure: /A
/J

Comments: The /A command advances the cursor to the beginning of the last line in the file. The /J command moves the cursor to the end of the line.

If the last character in the file is outside the window, the window will be redisplayed to show the end of file.

Examples: Assume that the current screen looks like this:

```

JANUARY~
FEBRUARY~
MARCH~
APRIL~
MAY~
-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:

```

To move the cursor to the beginning of the last line in the file, enter:

/A

```
-----
^O01 0001 ENTER COUNT/COMMAND      COUNT:/  COMMAND:A
END OF FILE ENCOUNTERED
```

Now enter the following command to move the cursor to the end of the line:

/J

```
AUGUST~
SEPTEMBER~
OCTOBER~
NOVEMBER~
DECEMBER~
-----
^O01 0008 ENTER COUNT/COMMAND      COUNT:/  COMMAND:J
```

- See also:
- Moving the Cursor to the End of the Line
 - Moving the Cursor to the Beginning of the File

Resetting the Tab Stops

Description: This procedure sets the tab stops to the columns you specify.

Procedure: T{column}(ESC)

Parameters: **column**—A series of decimal column numbers separated by commas.

Comments: When you invoke ACE, the default tab stops are set at every 8th column, beginning with column 9. You may specify up to 24 new tab stops. Each time you set new tab stops, all existing tab stops are lost. The column numbers must be in ascending order and in the range 2 to 998. Any text that you display is aligned to the most recently specified set of tab stops. After you exit the editor, all tab stops are set back to the default values.

Examples: T

```

-----
^001 00001 ENTER TAB STOPS          COUNT:  COMMAND:T
 10,20,30(ESC)
 3 TAB STOPS DEFINED
    
```

This command deletes the existing tab stops and sets new tab stops at columns 10, 20, and 30.

See also: ● Viewing Blanks in Lines

Entering Commands in Revise Mode

Description: When the editor is in revise mode, most characters that you enter will overwrite characters already in the window. Non-configurable commands are not recognized as such by the editor when you are in revise mode. This procedure shows you how to enter these commands during revise mode so that they do not overwrite characters in the window.

Procedure: (cmd-esc)
command

Parameters: **command**—A valid ACE command.

Comments: Configurable commands, used for moving the cursor, scrolling, paging, inserting, deleting, and exiting from revise mode, do not require the (cmd-esc) if they are configured to single keys that begin with a control-code sequence. Refer to Technical Note 1 in Section 5 of this manual for a discussion of the configuration process.

Examples: Assume that the current screen looks like this:

```

ONE~
TWO~
THREE~

-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:  REVISE
    
```

The word "REVISE" in the status line indicates that the editor is in revise mode. To enter the `-/A` command without exiting revise mode, enter:

(cmd-esc)

-/A

Now the screen looks like this:

```

ONE~
TWO~
THREE~

-----
^001 00001 ENTER COUNT/COMMAND      COUNT:-/ COMMAND:A REVISE

BEGINNING OF FILE ENCOUNTERED

```

See also:

- Moving the Cursor to the Beginning of the File

CHARACTER MANIPULATION

Finding a String

Description: This procedure finds the first occurrence of a specified string of characters starting at the cursor and continuing towards the end-of-file.

Procedure: F{string}(ESC)

Parameters: **string**—The string of characters to be found.

Comments: If a match is found, the cursor will be moved to the right of the found string. If the **string** is not found, the window will be unaffected, and the following message will be displayed:

```
NO OCCURRENCES OF string FOUND.
```

Examples: Assume that the current file contains the following text:

```

ALPHA          BETA          GAMMA
DELTA          EPSILON        ZETA
THETA          LAMBDA         SIGMA
PHI            PSI            OMEGA

```

The cursor is at the beginning of the file. To find the first occurrence of the string "PHI", enter:

F

```

-----
^001 00001 ENTER SEARCH STRING      COUNT:  COMMAND:F
PHI(ESC)
    
```

Now the screen looks like this:

```

ALPHA      BETA      GAMMA~
DELTA      EPSILON   ZETA~
THETA      LAMBDA    SIGMA~
PHI_       PSI       OMEGA~
-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:F
PHI
    
```

The cursor is to the right of the string "PHI".

See also: ● Counting the Number of Occurrences of a String

Counting the Number of Occurrences of a String

Description: This procedure counts the number of occurrences of a specified string of characters in the file.

Procedure: -/A
/F{string}(ESC)

Parameters: **string**—The string of characters to be counted.

Comments: The **-/A** command moves the cursor to the beginning of the file. The **/F** command counts the number of occurrences of the specified string. The position of the cursor and the window are not changed. A message showing the number of matches is displayed in the message line.

Examples: Assume that the current file contains the following text:

```
SUNDAY
MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY
```

To count the number of occurrences of the string "DAY" in the file, enter:

```
-/A
/F
```

```
-----
^O01 00001 ENTER SEARCH STRING      COUNT:/  COMMAND:F
DAY(ESC)
```

Now the screen looks like this:

```
SUNDAY~
MONDAY~
TUESDAY~
WEDNESDAY~
THURSDAY~
-----
^O01 00001 ENTER COUNT/COMMAND      COUNT:/  COMMAND:F

7 OCCURRENCES OF DAY FOUND.
```

The message line indicates that there are 7 occurrences of the string "DAY" in the file.

- See also:
- Moving the Cursor to the Beginning of the File
 - Finding a String

Replacing a String

Description: This procedure replaces the first occurrence of a specified string with a new string. The search begins at the cursor and continues towards the end-of-file.

Procedure: R{oldstring}{ESC}{newstring}{ESC}

Parameters: **oldstring**—The string to be replaced.

newstring—The string that replaces **oldstring**.

Comments: If a match is found, **oldstring** will be replaced by **newstring** and the cursor will be moved to the right of **newstring**. If **oldstring** is not found in the file, the window will be unaffected, and the following message will be displayed:

NO REPLACEMENTS OF string DONE.

Examples: Assume that the current screen looks like this:

```

Happy Birthday, Fred!~
-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:
    
```

To replace the string "Fred" with the string "George", enter:

R

The editor prompts for SEARCH STRING. You enter:

```

-----
^001 00001 ENTER SEARCH STRING      COUNT:  COMMAND:R
Fred(ESC)
    
```

Then the editor prompts for REPLACEMENT STRING. You enter:

```

-----
^001 00001 ENTER REPLACEMENT STRING COUNT:  COMMAND:R
Fred George(ESC)
    
```

Now the screen looks like this:

```

Happy Birthday, George!~

-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:R
Fred George

```

See also: ● Making Global Replacements

Making Global Replacements

Description: This procedure replaces all occurrences of a specified string in the file with a new string. The cursor will be positioned to the right of the last replacement.

Procedure: -/A
/R{oldstring}(ESC){newstring}(ESC)

Parameters: **oldstring**—The string to be replaced.

newstring—The string that replaces **oldstring**.

Comments: If query mode is one, ACE will prompt you to confirm each replacement. (See the Q command in the Command Dictionary section of this manual).

CAUTION

Making global replacements can be very destructive if you choose your search string carelessly. A search string that is too general may lead to unexpected replacements.

Examples: Assume that the current file contains the following text:

```

I LIKE GREEN SALAD AND THOUSAND ISLAND DRESSING.
HE LIKES A HAM SANDWICH AND MILK.

```


To replace all occurrences of "AND" with "WITH", enter:

-/A
/R

```
-----
^001 00001 ENTER SEARCH STRING      COUNT:/  COMMAND:R
AND(ESC)
```

```
-----
^001 00001 ENTER REPLACEMENT STRING COUNT:/  COMMAND:R
AND WITH(ESC)
```

Now the window looks like this:

```
I LIKE GREEN SALAD WITH THOUSWITH ISLWITH DRESSING.~
HE LIKES A HAM SWITHWICH WITH_MILK.~

-----
^001 00001 ENTER COUNT/COMMAND      COUNT:/  COMMAND:R
AND WITH
5 REPLACEMENTS OF AND DONE.
```

Every occurrence of the string "AND" in the file is replaced by the string "WITH".

See also:

- Moving the Cursor to the Beginning of the File
- Replacing a String

LINE MANIPULATION

Viewing Blanks in Lines

Description: Blanks in the window exist for one of the following reasons:

- actual spaces in the text
- tab characters in the file expanded to an equivalent number of spaces
- blank area to the right of the end-of-line
- non-displayable characters in the text

This procedure determines which types of blanks are present in the current line.

Procedure: V

The editor clears the window and displays the current line. Any non-displayable characters are converted into a two-character representation (a circumflex ^ followed by a character). Table 4-5 (in the Tables section of this manual) lists the two-character representation of each non-displayable character.

```

-----
^001 0001 ENTER                COUNT:  COMMAND:V
DEPRESS CARRIAGE RETURN TO CONTINUE: (CR)

```

The editor redisplay the window before the V command was invoked.

Comments: The following rules apply to changes made to blank areas of the window. "Changes" include deleting and replacing characters.

- When the blank area consists of **actual spaces**, any changes do not affect the unchanged blank area. Inserting text in front of spaces preserves the number of spaces.
- When the blank area is due to **tab expansion**, any changes delete the entire area. Since only one tab character represents the blank area, removing the tab character thereby removes the tabbed area as well. Inserting text in front of the tab character alters the number of blanks in the tabbed area.
- In the blank area **to the right of the end-of-line character**, any changes can only be made by deleting the end-of-line or by inserting in front of the end-of-line. Overwriting the end-of-line in revise mode extends the line (effectively inserting).
- When the blank area is due to **non-displayable characters**, any changes will overwrite the non-displayable characters.

Examples: Assume that the current screen looks like this:

```

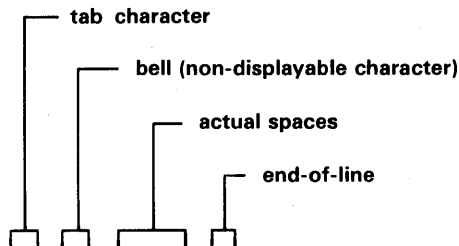
AB   CD EF   GH~
1234567890123456789012345~

-----
^O01 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:

```

To determine the types of blank areas in the current line, enter:

V



```

AB^ICD^GEF   GH^M

-----
^O01 00001 ENTER      COUNT:  COMMAND:V

DEPRESS CARRIAGE RETURN TO CONTINUE: (CR)

```

After you press RETURN, the editor redisplay the window as it was before the V command was invoked.

Manipulating Long Lines

ACE allows you to edit lines containing up to 999 characters. Since most CRT terminals have lines less than this size, data that does not fit into a single window line is truncated at both the left and the right margins.

- If the status line contains a number other than "001" in its first field, lines in the window will be truncated on the left.
- Since the end-of-line character is displayable, it is easily seen if the complete line is displayed. If a tilde is visible at the end of a line, the line is completely displayed. If a tilde is not visible, the line shown is truncated on the right.

Commands (for example, scrolling commands) allow you to move the text in the window left and right so that long lines may be seen in the window.

Moving a Line

Description: This procedure deletes a line and inserts it elsewhere in the file being edited.

Procedure: Move the cursor to the line you want to move, and press:
(dl)

Then move the cursor to the location where the line is to be moved, and enter:
U

Comments: The (dl) command deletes the line. The U command restores the deleted text.

Examples: Assume that the current file contains the following text:

```
Pancake cooking instructions:
Turn on the stove.
Mix all ingredients.
Pour the mix into a pan and cook until done.
```

You want to move the second line to appear after the third line. You move the cursor to the second line, and enter:

(dl)

Then move the cursor to the location where you want to insert the line, as shown in the following:

```
Pancake cooking instructions:~
Mix all ingredients.~
Pour the mix into a pan and cook until done.~
```

```
-----
^001 0001 ENTER COUNT/COMMAND      COUNT:  COMMAND:
```

Enter the following command:

U

Now the screen looks like this:

```

Pancake cooking instructions:~
Mix all ingredients.~
Turn on the stove.~
Pour the mix into a pan and cook until done.~

-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:U

```

See also: ● Moving a Block of Text

Splitting a Line

Description: This procedure splits a line into two lines.

Procedure: Move the cursor to the character that you want to be the first character in the new line and enter the following key sequence:

(ic)(CR)(ESC)

Comments: When you insert a carriage return in a line, you split that line into two lines.

Examples: Move the cursor to the character where you want to start the new line.

```

IT TAKES TIME, EFFORT, AND A GOOD DEAL OF MONEY.~

-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:

```

Then enter:

(ic)(CR)(ESC)

Now the screen looks like this:

```

IT TAKES TIME, EFFORT, ~
AND A GOOD DEAL OF MONEY.~

-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:

```

See also: ● Concatenating Lines

Concatenating Lines

Description: This procedure concatenates the current line and the next line into one line.

Procedure: Move the cursor to the end-of-line character of the first line of the two that you want to concatenate. Then enter:

(dc)

Comments: When the cursor is at the end-of-line character, the (dc) key deletes the end-of-line character, thus concatenating the two lines.

Examples: Move the cursor to the end-of-line character (~) of the first line of the two that you want to concatenate.

```

IT TAKES TIME, EFFORT, ~
AND A GOOD DEAL OF MONEY.~

-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:

```

Then enter:

(dc)

Now the screen looks like this:

```

IT TAKES TIME, EFFORT, AND A GOOD DEAL OF MONEY.~

-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:

```

- See also:
- Moving the Cursor to the End of the Line
 - Splitting Lines

Changing Text from Cursor to the Beginning of Line

Description: This procedure replaces the text preceding the cursor in the current line with new text.

Procedure: -/C{newtext}(ESC)

Parameters: **newtext**—The string that replaces the text preceding the cursor in the current line.

Examples: Assume that the current screen looks like this:

```

MOV      A,B      ; Store the answer~

-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:

```

Suppose you want to change the assembler instruction in the current line to the following instruction:

```
STO    ANS,B
```

You enter:

```
-/C
```

```
-----
^001 00001 ENTER REPLACEMENT STRING COUNT: -/ COMMAND: C
(TAB)STO(TAB)ANS,B(TAB)(ESC)
```

Now the screen looks like this:

```
STO    ANS,B    ; Store the answer~
```

```
-----
^001 00001 ENTER COUNT/COMMAND    COUNT: -/ COMMAND: C
STO ANS,B
```

See Also: • Changing Text from Cursor to the End of Line

Changing Text from Cursor to the End of Line

Description: This procedure replaces the text between the cursor and the end-of-line with new text.

Procedure: /C{newtext}(ESC)

Parameters: **newtext**—The string that replaces the text between the cursor and the end-of-line.

Examples: Assume that the current screen looks like this:

```
Once upon a time, a little girl lived in a big house~
in the big woods.~
```

```
-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:
```

Suppose you want to change the rest of the line to:
there was a little house

You enter:

/c

```
-----
^001 00001 ENTER REPLACEMENT STRING COUNT:/  COMMAND:C
there was a little house(ESC)
```

Now the screen looks like this:

```
Once upon a time, there was a little house~
in the big woods.~
```

```
-----
^001 00001 ENTER COUNT/COMMAND      COUNT:/  COMMAND:C
a little girl lived in a big house
```

See Also: ● Changing Text from Cursor to the Beginning of Line

BLOCK MANIPULATION

Deleting a Block of Text

Description: This procedure deletes a block of text.

Procedure: Move the cursor to the first character of the block that you want to delete. Then press the (mark) key to mark the beginning of the block.

(mark)

Now move the cursor to the last character of the block that you want to delete, and press:

(dc)

Examples: Assume that the current file contains the following text:

```
do re mi fa sol la ti
do mi mi
mi sol sol
re fa fa
la ti ti
```

You want to delete the block of text as indicated above. Move the cursor to the beginning of the block, and press:

(mark)

An at-sign (@) is displayed at the cursor position and the monitor area looks like this:

```
-----
^001 0001 ENTER MOTION          COUNT:  COMMAND:
```

You move the cursor to the end of the block to be deleted, as shown in the following display:

```
do re mi fa sol la ti~
do @i mi~
mi sol sol~
re fa fa~
la_ti ti~
-----
^001 0001 ENTER MOTION          COUNT:  COMMAND:
```

Then press the following key and the block of text is deleted.

(dc)

Now the screen looks like this:

```
do re mi fa sol la ti~
do ti ti~

-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:
```

See Also: ● Moving a Block of Text

Moving a Block of Text

Description: This procedure deletes a block of text and inserts it elsewhere in the file being edited.

Procedure: Move the cursor to the first character of the block that you want to move, and press:

(mark)

Then move the cursor to the last character of the block that you want to move, and press:

(dc)

Now move the cursor to the location where you want to insert the block of text, and enter:

U

Comments: The (dc) command deletes the text. The U command restores the deleted text.

Examples: Assume that the screen looks like this:

```

XXXXXXXX~
XXX~
X~
XXX~
~
-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:

```

You want to move the first two lines to appear after the last line. The cursor is at the beginning of the first line. Press:

(mark)

An at-sign (@) is displayed at the cursor position and the monitor area looks like this:

```

-----
^001 00001 ENTER MOTION              COUNT:  COMMAND:

```

You move the cursor to the end of the second line, as shown in the following display:

```

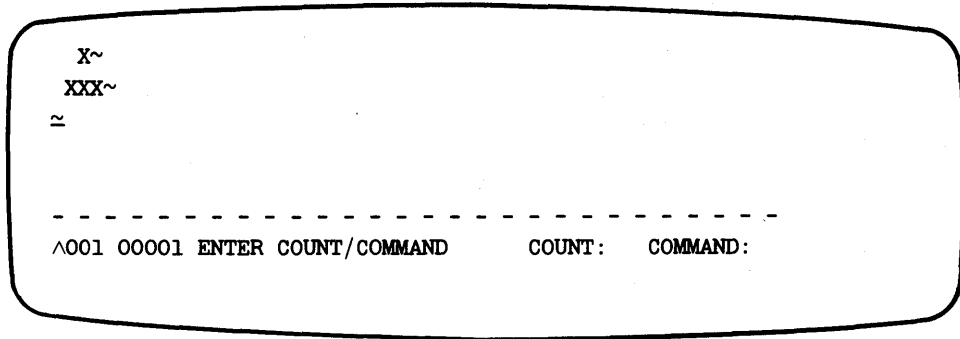
@XXXXX~
XXX~
X~
XXX~
~
-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:

```

Press the following key to delete the block of text:

(dc)

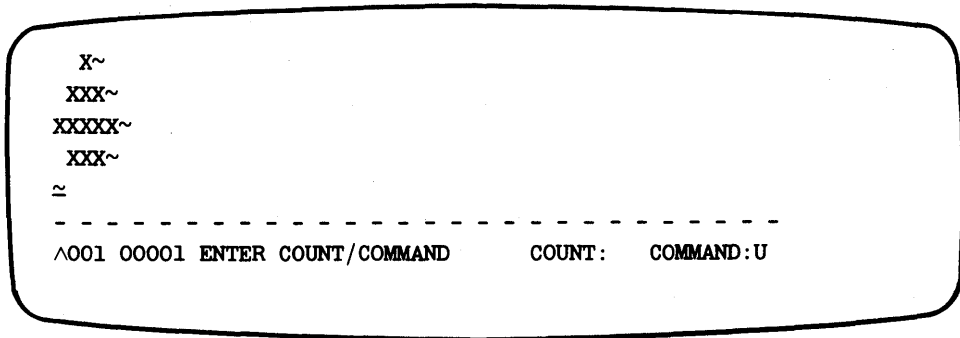
Then move the cursor to the location where you want to insert the block:



Enter the following command to restore the deleted text:

U

Now the screen looks like this:



- See also:
- Moving a Line
 - Deleting a Block of Text

Saving a Block of Text in a File

Description: This procedure copies a block of text to a disc file.

Procedure: O{filespec}(ESC)

Move the cursor to the first character of the block that you want to save, and press:

(mark)

Then move the cursor to the last character of the block that you want to save, and enter:

EW

Parameters: **filespec**—The name of the file to which the block of text is copied.

Examples: Assume that the current file contains the following text and the cursor is at the beginning of the file.

PLEASE SAVE THE FOLLOWING 2 LINES.
BEGINNING OF BLOCK.
END OF BLOCK.
THANK YOU.

Enter the O command followed by the filename, as shown:

O

```

-----
^O01 00001 ENTER FILE NAME          COUNT:  COMMAND:O
  THEFILE(ESC)
FILE THEFILE CREATED.

```

Move the cursor to the beginning of the block you want to save, and press:

(mark)

Then move the cursor to the end of the block, and the screen looks like this:

```

PLEASE SAVE THE FOLLOWING 2 LINES.~
@BEGINNING OF BLOCK.~
END OF BLOCK.~
THANK YOU.~
-----
^O01 00001 ENTER MOTION          COUNT:  COMMAND:

```

Enter the following command to save the block of text between the @ sign and the cursor:

EW

Now the screen looks like this:

```

PLEASE SAVE THE FOLLOWING 2 LINES.~
BEGINNING OF BLOCK.~
END OF BLOCK.~
THANK YOU.~

-----
^O01 0001 ENTER COUNT/COMMAND      COUNT:  COMMAND:EW

```

The window is unchanged. The specified block is saved in the file named THEFILE.

- See also:
- Duplicating a Block of Text
 - Splitting a File

Duplicating a Block of Text

Description: This procedure duplicates a block of text in the file.

Procedure: O{tempfile}(ESC)

Move the cursor to the first character of the block that you want to duplicate, and press:

(mark)

Then move the cursor to the end of the block that you want to duplicate, and enter:

EW

Now move the cursor to the location where you want to insert the duplicated text, and enter:

ER

Parameters: **tempfile**—The name of the temporary file that is used to store the block of duplicated text.

Examples: Assume that the current screen looks like this:

```

+-----+-----+-----+~
|         |         |         |~
~
-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:

```

You want to duplicate these two lines. Enter the following command to open the file TEMP for use with the EW and ER commands.

Q

```

-----
^001 00001 ENTER FILE NAME          COUNT:  COMMAND:0
TEMP(ESC)
FILE TEMP CREATED.

```

Then move the cursor to the beginning of the first line, and press:

(mark)

Now move the cursor to the end of the second line. The screen looks like this:

```

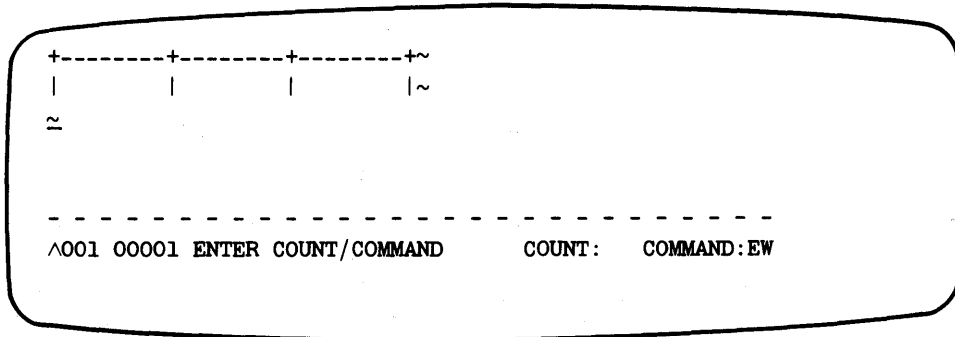
@-----+-----+-----+~
|         |         |         |~
~
-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:

```


Now enter the following command:

EW

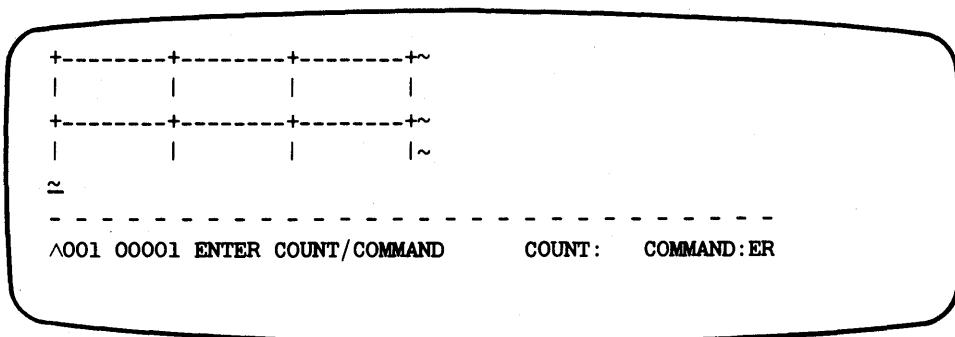
Move the cursor to the position where you want to insert the duplicated text:



Enter the following command:

ER

Now the screen looks like this:



See also: ● Saving a Block of Text in a File

Printing a Block of Text on the Line Printer (8550 Only)

Description: This procedure prints a block of text by copying the block to the spooling directory. For information on setting up the spooling directory, refer to the 8550 System Users Manual.

Procedure:

O{/SPOOL1/filename}(ESC)

Move the cursor to the first character of the block that you want to print, and enter:

(mark)

Then move the cursor to the last character of the block that you want to print, and enter:

EW

Parameters: **filename**—A name to distinguish the file from other files in the spooling directory.

Comments: This procedure shows how to print a block of text. To print the entire file, copy the entire file to the spooling directory.

Examples: Assume that you have already set up a DOS/50 spooling directory before invoking ACE.

Enter the following command to open the file ACEFILE in the spooling directory:

O

```

-----
^001 0001 ENTER FILE NAME          COUNT:  COMMAND:0
/SPOOL1/ACEFILE(ESC)
FILE /SPOOL1/ACEFILE CREATED.

```

Move the cursor to the beginning of the block you want to print, and press:

(mark)

An at-sign (@) is displayed at the cursor position. Move the cursor to the end of the block. Then enter the following command to copy the block of text between the at-sign (@) and the cursor:

EW

The block of text is placed in file ACEFILE in the spooling directory. It will then be sent to the line printer.

See Also:

- Saving a Block of Text in a File

Splitting a File

Description: This procedure splits a file into two files. The first part is saved in the file opened with the O command. The second part remains in the current file.

Procedure: O{filespec}(ESC)

Move the cursor to the beginning of the file, and press:

(mark)

Then move the cursor to the last character of the first part of the file, and enter:

EW

-(dl)

(dl)

Parameters: **filespec**—The name of the file to which the block of text is copied.

Comments: The -(dl) command deletes all preceding lines. The (dl) command deletes the current line.

Examples: Assume that the current file contains the following text:

```
IF ANYTHING CAN POSSIBLY GO WRONG,
IT WILL.
IF A SENTENCE CAN POSSIBLY BE MISUNDERSTOOD,
IT WILL BE.
```

You want to split the file into two files so that the file named MURPHY contains the first two lines and the current file contains the last two lines. Open the file MURPHY with the following command:

O

```
-----
^001 0001 ENTER FILE NAME          COUNT:  COMMAND:O
MURPHY(ESC)
FILE MURPHY CREATED.
```

Move the cursor to the beginning of the file and press:

(mark)

Then move the cursor to the end of the second line. The screen looks like this:

```

@F ANYTHING CAN POSSIBLY GO WRONG,~
IT WILL.~
IF A SENTENCE CAN POSSIBLY BE MISUNDERSTOOD,~
IT WILL BE.~

-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:

```

Enter the following command sequence:

```

EW
-/(d1)
(d1)

```

Now the screen looks like this:

```

IF A SENTENCE CAN POSSIBLY BE MISUNDERSTOOD,~
IT WILL BE.~

-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:

```

The first two lines are saved in the file MURPHY. The last two lines remain in the current file.

See also:

- Moving the Cursor to the Beginning of the File
- Saving a Block of Text in a File
- Concatenating Files

Concatenating Files

Description: This procedure concatenates the current file and another file into one file.

Procedure: /A
 (i1)(ESC)
 O{filespec}(ESC)
 ER

Parameters: **filespec**—The name of the file to be concatenated.

Comments: The current file will be the first part of the concatenated file. The file opened with the O command will be the second part of the concatenated file.

Examples: Assume that the current file contains the following text:

```
This is my file.
It is the first file.
```

Assume also that the file YOURFILE in the current directory contains the following text:

```
This is your file.
It is the second file.
```

Suppose you want to concatenate the current file and the file YOURFILE into one file. First, move the cursor to the end of the current file:

/A

Then insert a blank line:

(i1)(ESC)

Now open the file YOURFILE with the O command:

O

```
-----
^001 0001 ENTER FILE NAME          COUNT:  COMMAND:O
  YOURFILE(ESC)
  FILE YOURFILE OPENED.
```

Enter the following command to insert the contents of the file YOURFILE into the current file:

ER

Now the screen looks like this:

```

This is my file.~
It is the first file.~
This is your file.~
It is the second file.~
~
-----
^O01 0001 ENTER COUNT/COMMAND      COUNT:  COMMAND:ER

```

See Also: • Splitting a File

CASE MANIPULATION

Converting All Uppercase Letters to Lowercase

Description: This procedure converts all uppercase letters in the file to lowercase letters.

Procedure: -/A
 MD{n}(ESC)/XLA(ESC)
 /MX{n}(ESC)

Parameters: n—The name of the macro: any integer in the range 0 to 9.

Examples: Assume that the current file contains the following text:
 I studied Computer Programming at McDonald's College
 in the Winter of 1975.

To move the cursor to the beginning of the file, enter:

-/A

Now define macro number 1. Macro 1 converts all uppercase letters to lowercase, proceeding from the cursor towards the end-of-line, and then advances one line. Enter:

MD

```
-----
^001 00001 ENTER MACRO NAME          COUNT:  COMMAND:MD
1(ESC)
```

```
-----
^001 00001 ENTER MACRO DEFINITION    COUNT:  COMMAND:MD
1 /XLA(ESC)
MACRO 1 DEFINED.
```

Then execute macro 1 until the end-of-file is encountered:

/MX

```
-----
^001 00001 ENTER MACRO NAME          COUNT:/  COMMAND:MX
1(ESC)
```

Now the screen looks like this:

```
i studied computer programming at mcdonald's college~
in the winter of 1975.~
```

```
-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:A
1 MACRO EXECUTIONS PERFORMED.
MACRO TERMINATED AT EOF.
```

All uppercase letters in the file are converted to lowercase. Lowercase letters, digits, and special characters are not changed.

See Also:

- Moving the Cursor to the Beginning of the File
- Creating and Using Command Macros

Converting All Occurrences of a String to Uppercase

Description: This procedure finds a specified string and converts all lowercase letters in the string to uppercase. This procedure repeats until the end-of-file is encountered.

Procedure:

```
-/A
WI{iwild}(ESC)
WE{ewild}(ESC)
MD{n}(ESC)F{iwild}{string}{iwild}{ewild}<XU(ESC)
/MX{n}(ESC)
```

Parameters:

- iwild**—A character that is defined to be the case ignore delimiter.
- ewild**—A character that is defined to be the escape wildcard character.
- n**—The name of the macro: any integer in the range 0 to 9.
- string**—A string of characters that is to be converted to uppercase.

Examples: Assume that the current file contains the following text:
 The Tektronix CT8500 terminal is particularly useful
 when used with tektronix screen-oriented editors,
 such as ACE.

To move the cursor to the beginning of the file, enter:

```
-/A
```

To define the double quote (") as the case ignore delimiter, enter:

```
WI
```

```
-----
^O01 0001 ENTER WILDCARD CHARACTER COUNT:  COMMAND:WI
"(ESC)
CASE IGNORE WILDCARD IS <">
```

To define the dollar sign (\$) as the escape wildcard character, enter:

```
WE
```

```
-----
^O01 0001 ENTER WILDCARD CHARACTER COUNT:  COMMAND:WE
$(ESC)
ESCAPE WILDCARD IS <$>
```


Now define macro number 2, which finds the string "TEKTRONIX", ignoring the case (uppercase or lowercase), and then converts the string to uppercase. Enter:

MD

```
-----
^001 00001 ENTER MACRO NAME          COUNT:  COMMAND:MD
2(ESC)
```

```
-----
^001 00001 ENTER MACRO DEFINITION  COUNT:  COMMAND:MD
2 F"TEKTRONIX"$<XU(ESC)
MACRO 2 DEFINED.
```

Then execute macro 2 until the search fails:

/MX

```
-----
^001 00001 ENTER MACRO NAME          COUNT:/  COMMAND:MX
2(ESC)
```

Now the screen looks like this:

```
-----
The TEKTRONIX CT8500 terminal is particularly useful~
when used with TEKTRONIX_screen-oriented editors,~
such as ACE.~
```

```
-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:F
2 MACRO EXECUTIONS PERFORMED.
NO OCCURRENCES OF "TEKTRONIX" FOUND.
```

All occurrences of the string "Tektronix" in the file are converted to "TEKTRONIX". The message line indicates what condition caused the macro execution to terminate.

See Also:

- Moving the Cursor to the Beginning of the File
- Creating and Using Command Macros

Capitalizing All Words Following a Period and Two Spaces

Description: This procedure capitalizes all words that follow a period and two spaces in the file.

Procedure: -/A
WG{gwild}(ESC)
WE{ewild}(ESC)
MD{n}(ESC)F.(SP)(SP){gwild}{ewild}-XU(ESC)
/MX{n}(ESC)

Parameters: **gwild**—A character that is defined to be the general wildcard character.

ewild—A character that is defined to be the escape wildcard character.

n—The name of the macro: any integer in the range 0 to 9.

Examples: Assume that the current file contains the following text:
 this is the first sentence. this is the second
 sentence. this is the third sentence.
 this is the fourth sentence.

To move the cursor to the beginning of the file, enter:

-/A

To define the asterisk (*) to be the general wildcard character, enter:

WG

```
-----
^001 0001 ENTER WILDCARD CHARACTER COUNT:  COMMAND:WG
*(ESC)
GENERAL WILDCARD IS <*>
```

To define the dollar sign (\$) to be the escape wildcard character, enter:

WE

```
-----
^001 0001 ENTER WILDCARD CHARACTER COUNT:  COMMAND:WE
$(ESC)
ESCAPE WILDCARD IS <${>
```


All words that follow a period and two spaces in the file are capitalized.

- See Also:
- Moving the Cursor to the Beginning of the File
 - Capitalizing All Words Following a Period—End-of-Line Sequence
 - Creating and Using Command Macros

Capitalizing All Words Following a Period—End-of-Line Sequence

Description: This procedure capitalizes all words that follow a period and end-of-line (~) sequence in the file.

Procedure:

```
-/A
WG{gwild}(ESC)
WE{ewild}(ESC)
MD{n}(ESC)F.(CR){gwild}{ewild}-XU(ESC)
/MX{n}(ESC)
```

Parameters: **gwild**—A character that is defined to be the general wildcard character.

ewild—A character that is defined to be the escape wildcard character.

n—The name of the macro: any integer in the range 0 to 9.

Examples: Assume that the current file contains the following text:

```
I don't know when I'll be done.
programming's really such fun.
but when I do quit,
it won't hurt a bit.
because it's left me so thoroughly numb.
```

To move the cursor to the beginning of the file, enter:

```
-/A
```

To define the asterisk (*) as the general wildcard character, enter:

```
WG
```

```
-----
^001 0001 ENTER WILDCARD CHARACTER COUNT:  COMMAND: WG
*(ESC)
GENERAL WILDCARD IS <*>
```

To define the dollar sign (\$) as the escape wildcard character, enter:

WE

```
-----
^001 00001 ENTER WILDCARD CHARACTER COUNT:  COMMAND:WE
$(ESC)
ESCAPE WILDCARD IS <$>
```

Now define macro number 4. Macro 4 finds a string that contains a period followed by an end-of-line character (~) and a character, then converts that character to uppercase (if it is not already uppercase). Enter:

MD

```
-----
^001 00001 ENTER MACRO NAME          COUNT:  COMMAND:MD
4(ESC)
```

The editor prompts for macro definition. You enter:

F.(CR)*\$-XU(ESC)

When you press the return key (CR), a tilde ~ is displayed. The monitor area looks like this:

```
-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:MD
4 F.~*$-XU
MACRO 4 DEFINED.
```

Then enter the following command to execute macro 4 until the search fails:

/MX

```
-----
^001 00001 ENTER MACRO NAME          COUNT:/  COMMAND:MX
4(ESC)
```

Now the screen looks like this:

```

I don't know when I'll be done.~
Programming's really such fun.~
But when I do quit,~
it won't hurt a bit.~
Because it's left me so thoroughly numb.~
-----
^001 0001 ENTER COUNT/COMMAND      COUNT:  COMMAND:F
3 MACRO EXECUTIONS PERFORMED.
NO OCCURRENCES OF .* FOUND.

```

All words that follow a period and an end-of-line character are capitalized.

- See Also:
- Moving the Cursor to the Beginning of the File
 - Finding a String
 - Capitalizing All Words Following a Period and Two Spaces
 - Creating and Using Command Macros

COMMAND SHORTCUTS

Creating and Using Command Macros

Description: This procedure stores a frequently-used ACE command sequence (called a macro) and assigns it a name. The macro is executed whenever its name is specified in a MX command.

Procedure: To create a macro:
MD{n}(ESC){commands}(ESC)

To execute that macro:
MX{n}(ESC)

To list all currently defined macros:
ML

Parameters: n—The name of the macro: any integer in the range 0 to 9.

commands—A sequence of editor commands to be associated with macro name n. To include a configurable command, you must enter the command's key code. If the key code or the command contains an (ESC) character, you must use the escape wildcard character. (The WE command is used to define an escape wildcard character.)

Comments: There is no command separator character. Multiple commands on a line are processed from left to right.

Examples: Suppose you want to create macro number 1, which inserts the following two lines:

```
+-----+
|         |
```

To do this, you'll use the MD command to define macro 1 to contain the following (ic) command line:

```
(ic)+-----+(CR) | (CR)(ESC)
```

Whenever macro 1 is invoked, two lines will be inserted. Notice that this (ic) command line contains an (ESC) character. Since (ESC) is used to terminate the MD command, you must define the escape wildcard character with the WE command. The escape wildcard character will be converted to 01BH (ESC) in memory but will not terminate the MD command.

WE

```
-----
^001 0001 ENTER WILDCARD CHARACTER COUNT:  COMMAND:WE
$(ESC)
ESCAPE WILDCARD IS <$>
```

\$ is defined as the escape wildcard character. You replace any (ESC) character in your command sequence with \$.

(ic) is a configurable command. You must enter the key code of (ic) in defining the macro. Let's assume that your terminal is a TEKTRONIX CT8500. The CT8500 key code for (ic) is 1B,1C which is equivalent to the (ESC) character followed by (CNTRL- \backslash). See Technical Note 1 in Section 5 of this manual.

Now, let's define macro 1. Enter:

MD

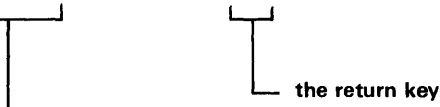
```
-----
^001 0001 ENTER MACRO NAME          COUNT:  COMMAND:MD
1(ESC)
```

Then the editor prompts for macro definition:

```
-----
^001 0001 ENTER MACRO DEFINITION  COUNT:  COMMAND:MD
1
```

You enter:

(CNTRL-\)+-----+(CR) | (CR)



\$ is the escape wildcard character.
 (ESC) followed by (CNTRL-\) is the (ic) key code.
 To enter (CNTRL-\), hold down the control key
 and press \.

(CNTRL-\) is a non-displayable character and (CR) is displayed as a tilde ~.
 The monitor area looks like this:

```
-----
^001 00001 ENTER MACRO DEFINITION   COUNT:  COMMAND:MD
1 $ +-----+~|                    |~$
```

Now press the (ESC) key to terminate the MD command:

(ESC)

```
-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:MD
1 $ +-----+~|                    |~$
MACRO 1 DEFINED.
```

Assume that the file is empty. You invoke macro 1 with the MX command.

MX

```
-----
^001 00001 ENTER MACRO NAME          COUNT:  COMMAND:MX
1(ESC)
```


Macro 1 is executed and the screen looks like this:

```
+-----+~  
|           |~  
R  
  
-----  
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:
```

To invoke macro 1 three more times, you enter:

3MX

```
-----  
^001 00001 ENTER MACRO NAME          COUNT:3  COMMAND:MX  
1(ESC)
```

Now the screen looks like this:

```
+-----+~  
|           |~  
+-----+~  
|           |~  
+-----+~  
|           |~  
+-----+~  
|           |~  
R  
  
-----  
^001 00001 ENTER COUNT/COMMAND      COUNT:3  COMMAND:MX
```

You can invoke the macro as many times as you want and build a handy table quite easily.

See also: ● Creating a File

Creating and Using Command Files

Description: This procedure causes the editor to initially execute commands from a file instead of from the system terminal.

Procedure: > ACE infile outfile comfile(CR)

Parameters: **infile**—The name of the input file.

outfile—The name of the output file. This parameter is optional. (To omit an optional parameter, enter two commas in its place.)

comfile—The name of the editor command file.

Comments: The specified editor command file is executed at the beginning of the editing session.

If the echo flag is on, ACE will show the results of each command executed. (See the EC command in the Command Dictionary section of this manual.)

You may use the editor to create a command file, just as you would create any other text file.

Examples: Suppose you want to create an editor command file, MYCOMMAND, to define three wildcard characters. The command file MYCOMMAND will contain the following editor commands:

```
WE$(ESC)
WG%(ESC)
WI\ (ESC)
```

First, create the command file:

```
> ACE MYCOMMAND(CR)
```

To enter the three editor commands into the file, you'll need to use the (ic) command. However, each of these three commands contains the (ESC) character, which terminates the insert command. In order to enter (ESC) without terminating the insert command, you must first use the WE command to define the escape wildcard character:

WE

```
-----
^O01 0001 ENTER WILDCARD CHARACTER COUNT:  COMMAND:WE
*(ESC)
ESCAPE WILDCARD IS <*>
```

* is defined as the escape wildcard character. You'll replace each (ESC) character in the command lines with *. The * character will be converted to 01BH (ESC) in memory, but will not terminate the insert command.

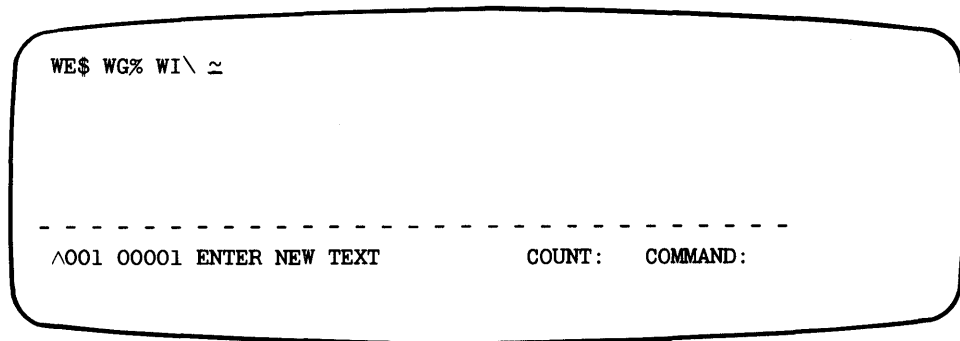
Now enter the insert character command:

(ic)

The editor prompts for new text. Enter:

WE\$*WG%*WI*

The escape wildcard character * is **not** displayed in the window. The screen looks like this:



To terminate the (ic) command, press:

(ESC)

Enter the EX command to save the text in disc file and exit the editor:

EX

You have just created an editor command file, MYCOMMAND, which defines three wildcard characters.

The following command begins an editing session that creates or modifies a file called MYFILE:

> ACE MYFILE, ,MYCOMMAND(CR)

ACE executes the commands in MYCOMMAND, and then prompts for an editor command. At that point, three wildcard characters have been defined.

See Also:

- Invoking the Editor
- Creating a File
- Exiting the Editor

Section 3 COMMAND DICTIONARY

Overview	Page
Section Overview	3-1
Dictionary Page Format	3-1
Terminology	3-3
Command Entry	3-4
The Screen	3-4
Special Keys	3-6

Command Index

ACE—Invoke the editor	3-9
A—Advance lines	3-13
C—Change characters	3-15
(cd)—Move cursor down	3-18
(ch)—Move cursor home	3-19
(cl)—Move cursor left	3-20
(cmd-esc)—Command-escape	3-21
(cr)—Move cursor right	3-22
(cu)—Move cursor up	3-23
(dc)—Delete characters	3-24
(dl)—Delete lines	3-27
EC—Echo commands	3-29
EP—Display editor profile	3-30
ER—Read secondary file	3-31
EW—Write to secondary file	3-33
EX—Exit editor and update file	3-35
F—Find characters	3-36
H—Help (display command syntax)	3-39
(ic)—Insert characters	3-41
(il)—Insert new line	3-43
J—Jump characters	3-44
(mark)—Mark command	3-46
MD—Define/delete macro	3-48
ML—List macros	3-49
MX—Execute macro	3-50
O—Open edit read/write file	3-52
(pd)—Page down	3-54
(pu)—Page up	3-55
Q—Query (conditional command execution)	3-56
R—Replace characters	3-57
(rm)—Enter/exit revise mode	3-60
S—Stop—Exit editor without file update	3-61
(sd)—Scroll down	3-62

Command Index (cont.)	Page
(sl)—Scroll left	3-63
(sr)—Scroll right	3-64
(su)—Scroll up	3-65
T—Define tab stops	3-66
U—Undelete characters	3-67
V—View non-printing characters	3-69
WE—Define escape wildcard character	3-71
WG—Define general wildcard character	3-72
WI—Define case-ignore wildcard character	3-73
XL—Exchange uppercase characters with lowercase	3-74
XU—Exchange lowercase characters with uppercase	3-76

Movement Commands

A—Advance lines	3-13
(cd)—Move cursor down	3-18
(ch)—Move cursor home	3-19
(cl)—Move cursor left	3-20
(cr)—Move cursor right	3-22
(cu)—Move cursor up	3-23
J—Jump characters	3-44
(pd)—Page down	3-54
(pu)—Page up	3-55
(sd)—Scroll down	3-62
(sl)—Scroll left	3-63
(sr)—Scroll right	3-64
(su)—Scroll up	3-65

Character string editing commands

C—Change characters	3-15
(dc)—Delete characters	3-24
F—Find characters	3-36
(ic)—Insert characters	3-41
R—Replace characters	3-57
(rm)—Enter/exit revise mode	3-60
U—Undelete characters	3-67
XL—Exchange uppercase characters with lowercase	3-74
XU—Exchange lowercase characters with uppercase	3-76

Line Editing Commands	Page
A—Advance lines	3-13
(dl)—Delete lines	3-27
(il)—Insert new line	3-43
(pd)—Page down	3-54
(pu)—Page up	3-55
(sd)—Scroll down	3-62
(sl)—Scroll left	3-63
(sr)—Scroll right	3-64
(su)—Scroll up	3-65

Display Commands

EC—Echo commands	3-29
EP—Display editor profile	3-30
H—Help (display command syntax)	3-39
V—View non-printing characters	3-69

File Manipulation Commands

ER—Read secondary file	3-31
EW—Write to secondary file	3-33
O—Open edit read/write file	3-52

Utility Commands

(cmd-esc)—Command-escape	3-21
EX—Exit editor and update file	3-35
(mark)—Mark command	3-46
MD—Define/delete macro	3-48
ML—List macros	3-49

Utility Commands (cont.)	Page
MX—Execute macro	3-50
Q—Query (conditional command execution)	3-56
S—Stop—Exit editor without file update	3-61
T—Define tab stops	3-66
WE—Define (ESC) wildcard character	3-71
WG—Define general wildcard character	3-72
WI—Define case-ignore wildcard character	3-73

TABLES

Table No.		Page
3-1	The Editor Commands	3-8
3-2	How Non-Displayable Characters Are Represented by ACE	3-70

ILLUSTRATIONS

Fig. No.		Page
3-1	Screen Representation	3-4
3-2	Status Line Format	3-5
3-3	Editor Profile Display Format	3-30
3-4	Configurable Editor Commands as Displayed by the H Command	3-39
3-5	Non-Configurable Editor Commands as Displayed by the H Command	3-40

Section 3

COMMAND DICTIONARY

SECTION OVERVIEW

This Command Dictionary alphabetically lists and describes in detail the commands for the Advanced CRT-Oriented Editor. **ACE**, the operating system command that invokes the editor, is also described in this dictionary. The editor can be configured to execute using different terminals. Refer to the Technical Notes section of this manual for information regarding the configurator.

Refer to the Learning Guide Section of this manual for a detailed overview of this editor.

DICTIONARY PAGE FORMAT

Each command entry contains a syntax block, a general explanation, and one or more examples.

Syntax Block

The syntax block contains the valid command entry formats. These formats consist of the command, required parameters, the repetition count, and mark-motion commands.

Configurable and Non-Configurable Commands

Configurable commands are represented in this manual by lowercase letters enclosed within parentheses. These keys are configured to your terminal keyboard when you enter the **ACE** command.

For information regarding the function of individual keys on the CT8500 keyboard, refer to the Tables section of this manual. For information regarding the use of other terminals with this editor, refer to the Technical Notes section of this manual.

Repetition Count

With most commands, a repetition count may be entered to execute a command more than once. For example, entering a **3A** advances the cursor 3 lines.

Here are some facts concerning the use of the repetition count:

- The count has a possible range of the integers -9999 through +9999.
- If no count parameter is entered, the count defaults to +1.
- If a minus sign (-) is entered, the count defaults to -1.
- Positive count values move toward the end of the file.
- Negative count values move toward the beginning of the file.

- In addition to positive and negative count values, you may specify non-numeric count values: / (performs the command to the end of the current line), -/ (performs the command to the beginning of the current line), < (performs the command to the beginning of the line using the length of search string from the previous **F** or **R** command as the count), and > (performs the command to the end of the line using the length of search string from the previous **F** or **R** command as the count),

Mark-Motion Commands

Mark-motion commands let you specify a block of text.

You may use the (dc) and EW commands to delete or write a block of text, respectively. The block is specified by using the following procedure: first, position the cursor at the beginning of the block of text, and enter the (mark) command. Use (**motion**) commands (cursor movement, scrolling, paging) to move the cursor to the end of the text block. Then perform the desired (**dc**) or **EW** command. The command is performed on the entire block of text.

Explanation

A detailed explanation of the command and its functions.

Example

Some common examples of how the command is used. Generally the example will contain:

1. a sample window containing the text before the command is executed;
2. an explanation of what function the command is to perform;
3. a command line showing the required keyboard input;
4. another sample window showing the edited text and any messages or prompts displayed by the editor.

Notation Conventions

The following conventions are used in describing command entry.

Underlined: A character sequence you enter.

No Underline: A character sequence displayed by ACE or by your system.

UPPERCASE: An exact character sequence; if these characters are underlined, enter them exactly as shown.

lowercase: A parameter you supply when you enter the command.

Braces { }: A required parameter you enter.

Brackets []: An optional parameter you may supply when you enter the command.

Parentheses (): A configurable or special key you press. Lowercase letters enclosed within parentheses represent configurable keys. Uppercase letters enclosed within parentheses represent special keys. Refer to the Tables section of this manual for a list of the configurable commands.

The following paragraphs define some of the terms used in this Command Dictionary. Refer to the Glossary section in this manual for a more complete list of terms.

TERMINOLOGY

You use the the editor to modify an **input file**, or to create a new file. The updated version of the file is written to the **output file**, which may or may not have the same name as that of the input file. You can specify an alternate output file. You can read text from or write text to a **secondary file**. If no alternate output file is specified, the editor creates a backup file, containing the unmodified input file, renamed to "**#inputfile**".

The CRT screen is is divided into two portions: the **window**, where editing is done, and the **monitor area**, where information is displayed by the editor and you enter **commands** and **parameters**. The term **string** refers to a character string requested by the editor. This string must not contain any configurable commands and must be terminated by an (ESC).

The **cursor** can be moved to any position in the window, pointing to the place where editing is to occur. The text can be **scrolled** to the left or right, or toward the beginning or end of the file. You can scroll one or more lines or columns at a time. In this manual, the cursor is represented by an underscore "**_**". The term **current character** refers to the character pointed to by the cursor. The term **current line** refers to the line that contains the cursor.

The term **page** refers to a full window of text, and does not correspond to your printed page of text. Text can be moved through the window one page at a time, either forward or backward. This is known as **paging**.

The monitor area contains a **status line** which provides information about the window's position in the file. In the status line the **top line number** is the number of the top line of the window, with respect to the beginning of the file. The **column number** refers to the leftmost visible character of the current window. When the column number is not equal to 001, the leftmost portion of the lines in the window are not shown. You can use the **(sr)** command to view the beginning of the line.

COMMAND ENTRY

When you enter an editor command, your entries must be in the following order: **[count]command[parameter]**. Commands are preceded by an optional count and followed by optional parameters.

THE SCREEN

ACE is screen-oriented. The screen of your terminal acts as a window into the file that you are editing. Figure 3-1 represents a CRT screen and shows the significant areas of the screen.

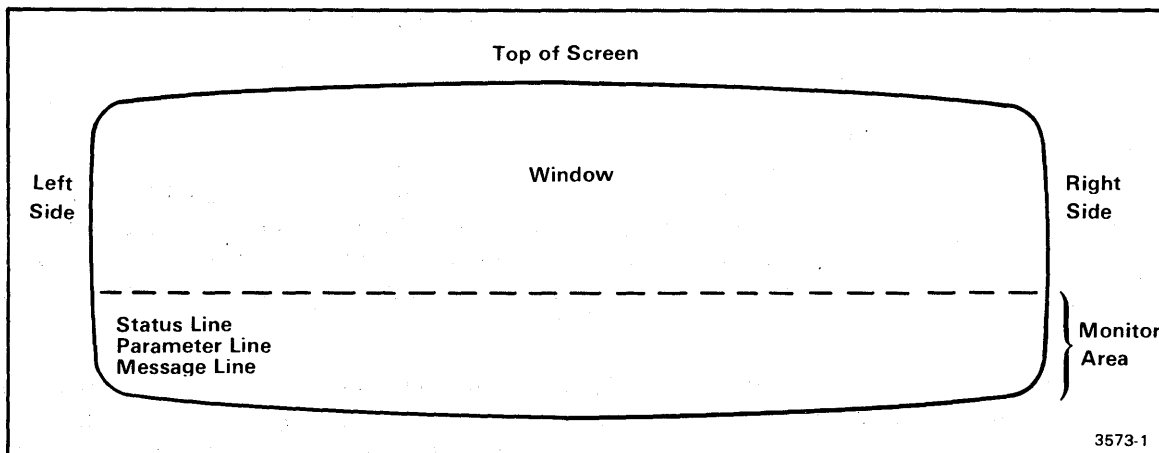


Fig. 3-1. Screen representation.

This illustration represents a CRT screen, as displayed by ACE. The significant areas of the screen are labeled here. The dashed line separates the window from the monitor area.

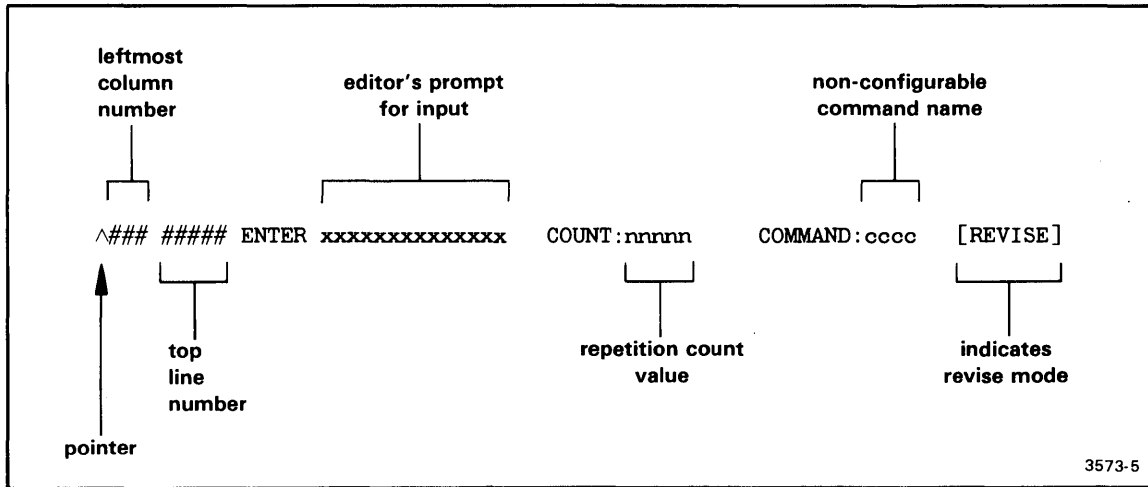


Fig. 3-2. Status Line Format.

The **status line** contains the following information: column number, top line number, editor input prompt, repetition count, non-configurable command name, and revise mode indicator. Figure 3-2 shows the status line format.

In Figure 3-2, the first three digits of the status line indicate the physical column number of the column shown in the leftmost window position. Any number other than 001 indicates that text extends to the left of the window.

The next five digits indicate the line number of the top line in the window with respect to the beginning of the file.

Next, the status line displays the word "ENTER" followed by a prompt string requesting input. Examples of prompt strings include: "COUNT/COMMAND", "SEARCH STRING", "REPLACEMENT STRING", "FILE NAME", "NEW TEXT", "MACRO NAME", "MACRO DEFINITION", "WILDCARD CHARACTER", "TAB STOPS", AND "MOTION".

The "COUNT" field heading is always displayed. If a non-default repetition count is entered, it is displayed. Otherwise, this field is left blank.

The "COMMAND" field heading is always displayed, followed by the name of the non-configurable command that is being executed.

The "REVISE" message is displayed only if the editor is in revise mode.

After you enter the command name, the editor uses the **parameter entry line** to show the various parameters as you enter them.

The **message line** is used to display error or warning messages, to query for confirmation of potentially destructive actions, and to display monitor messages.

When you access text that is outside of the workspace, the following message is displayed on the message line:

```
FILE ACCESS IN PROGRESS
```

When the command that is executing is completed, the message is erased.

SPECIAL KEYS

Carriage return

A visible end-of-line character is displayed at the end of each line. With the TEKTRONIX CT8500 terminal a tilde "~" is used to represent the end-of-line or return character. The end-of-line character is configurable to other characters. Refer to the Technical Notes section of this manual for information about the configurator.

In revise mode, entering a return overwrites the current character and splits the line at that point. While inserting characters, entering a return creates a blank line.

In this manual, the return key is represented by "(CR)".

Space

While inserting or revising text, or entering parameters, pressing the space bar erases the character at the cursor before moving the cursor right.

Backspace/Rubout

While inserting text or entering parameters, the backspace or rubout key erases the character at the cursor before moving the cursor left. In revise mode, the backspace or rubout key moves the cursor left but does not erase any characters.

Escape (ESC)

The (ESC) key signifies the end of parameter entry. Be sure that you press the (ESC) key to terminate any command that requires it.

TAB

During insert and revise mode, the tab character is expanded to the number of spaces indicated by the current tab stop definitions. At all other times, the tab character is not displayed. The tab character is treated as a single character by the editor.

NOTE

Tab stops are reset to the default values when you invoke the editor. User-defined tab stops are not remembered by the editor between edit sessions.

CNTRL-X

The CNTRL-x key cancels the current command or parameter being entered.

Displayable and Non-displayable characters

While in revise mode or insert mode, or when entering parameters, non-displayable characters may be included as valid characters. Otherwise, each non-displayable character occupies one blank position in the window.

NOTE

The characters CNTRL-S and CNTRL-Q are detected by the operating system. CNTRL-S halts the display. After entering CNTRL-S, you cannot change the screen until you enter CNTRL-Q to resume.

While in revise mode, or insert mode, or when entering parameters, displayable characters are treated literally. Otherwise they are treated as editor command mnemonics, in which case only valid mnemonics are accepted. The following rules apply to changes made to blank areas of the window. "Changes" include deleting and replacing characters.

- When the blank area consists of **actual spaces**, any changes do not affect the unchanged blank area. Inserting text in front of spaces preserves the number of spaces.
- When the blank area is due to **tab expansion**, any changes delete the entire area. Since only one tab character represents the blank area, removing the tab character thereby removes the tabbed area as well. Inserting text in front of the tab character alters the number of blanks in the tabbed area.
- In the blank area **to the right of the end-of-line character**, any changes can only be made by deleting the end-of-line or by inserting in front of the end-of-line. Overwriting the end-of-line in revise mode extends the line (effectively inserting).
- When the blank area is due to **non-displayable characters**, any changes will overwrite the non-displayable characters.

LIST OF EDITOR COMMANDS

Table 3-1 shows a list of all the ACE commands.

Table 3-1
ACE Editor Commands

Command	Description
A	Advance lines
C	Change characters
(cd)	Move cursor down
(ch)	Move cursor home
(cl)	Move cursor left
(cmd-esc)	Command-Escape
(cr)	Move cursor right
(cu)	Move cursor up
(dc)	Delete characters
(dl)	Delete lines
EC	Echo commands
EP	Display editor profile
ER	Read secondary file
EW	Write to secondary file
EX	Exit editor and update file
F	Find characters
H	Help (display command syntax)
(ic)	Insert characters
(il)	Insert new line
J	Jump characters
(mark)	Mark cursor position
MD	Define/delete macro
ML	List macros
MX	Execute macro
O	Open edit read/write file
(pd)	Page down
(pu)	Page up
Q	Query (conditional command execution)
R	Replace characters
(rm)	Enter/exit revise mode
S	Stop: exit editor without file update
(sd)	Scroll down
(sl)	Scroll left
(sr)	Scroll right
(su)	Scroll up
T	Define tab stops
U	Undelete characters
V	View non-displayable characters
WE	Define (ESC) wildcard character
WG	Define general wildcard character
WI	Define case-ignore wildcard character
XL	Exchange uppercase characters with lowercase
XU	Exchange lowercase characters with uppercase

SYNTAX

ACE Interactive form of editor invocation.

ACE {input file} [, [output file], [command file], [configuration file]]
Command line form of editor invocation.

EXPLANATION

There are two forms of the **ACE** command line: interactive and command line. In **interactive** invocation, you directly invoke the editor without having to enter a complex command line. In **command line** invocation, you invoke the editor from within a command file.

INTERACTIVE FORM

In interactive invocation, you simply enter **ACE**. The editor then displays the following message:

```
Version 1.X  
ENTER NAME OF FILE TO BE EDITED:
```

You respond by entering the name of the disc file that you want to edit.



The name of your file must be a valid disc file name. Names of system devices such as "CONO", "LPT", "REMO", etc. are not valid file names for use with the editor. Also file names that begin with "<" or ">" are not valid.

If the specified file does not already exist, the operating system creates a new file with the specified name. If the file that you specified already exists, the editor displays the following message:

```
ENTER NAME OF FILE TO RECEIVE RESULTS OF EDIT SESSION:
```

If you want the results of your editing session to be placed in any file other than the input file that you specified, enter that name here. Otherwise, enter a return; **ACE** will place all editing changes into the specified input file, and will retain a backup version of the input file renamed to "**#inputfile**".

If you specify a different output file and that file already exists, **ACE** displays the following message:

```
OUTPUT FILE ALREADY EXISTS.  
DELETE FILE?
```

If you respond with a "Y" or "y", the existing file is deleted and a new one is created. If you respond with any other character, ACE displays the following message:

ENTER NAME OF FILE TO RECEIVE RESULTS OF EDIT SESSION:

The editor then asks for a command file and a configuration file by displaying the following messages:

ENTER NAME OF COMMAND FILE:

and

ENTER NAME OF CONFIGURATION FILE:

Both the command file and the configuration file are optional. To not specify either file, answer the prompt by entering a return. However, if a file is specified for either response, it must be the name of an existing file. Otherwise, ACE displays the appropriate messages:

COMMAND FILE {name} DOES NOT EXIST.

ENTER NAME OF COMMAND FILE:

or

CONFIGURATION FILE {name} DOES NOT EXIST.

ENTER NAME OF CONFIGURATION FILE:

COMMAND LINE INVOCATION

In command line invocation, you must enter all of the required and optional parameters on the one ACE command line. This form of invocation is typically used to invoke the editor from a command file. The typical ACE invocation is as follows:

ACE filename

The first parameter, input file, is the name of the disc file that you want to edit. This is the only required parameter. If this is the only parameter that you enter, all edit changes are written back to the input file after the backup file is created. If the input file does not already exist, it is created. If the first parameter is omitted, ACE will display an error message.

The second parameter, output file, may be specified only if an existing input file is being edited. If you specify an output file when a new file is being created with the editor, ACE displays the following message and then returns control to the system:

EDITING NEW FILE, OUTPUT FILE SHOULD NOT BE SPECIFIED.

The output file parameter is used to specify the name of the file that is to receive the results of the edit session. In other words, all changes made to input file are written to output file and input file remains unchanged.

If you do not specify an output file, but want to specify a command file or configuration file, enter two commas in place of the output file to indicate a null entry. When you do not specify an output

file, all changes made during the edit session are written to your input file. However, a backup copy of the original input file is made first.

If you specify a command file, it must contain a series of editor commands that are to be executed before any manual input from the keyboard is obtained. If the command file contains the editor termination command, no user input is ever requested. However, if the command file does not contain the editor termination command, when the last editor command is processed, the editor will display the contents of the window reflecting the current position in the file. You must then continue or terminate the edit session by entering editor commands.

While the command file is executing, the state of the echo option determines whether the editor displays its progress. The echo option is initially set to off, and no output is shown during command file processing. If the echo option is set to on, the editor will show the result of each command as it is processed. The status line, parameter line, and message line will also be updated for each command.

If you specify a command file that does not already exist, the editor will display an error message. If you do not specify a command file, but want to specify a configuration file, you must enter two commas in place of the command file to indicate a null entry.

If specified, the configuration file contains configuration parameters for the editor. Configuration file must be an existing file created by the configurator (see the Technical Notes section of this manual for further information). If the {configuration file} parameter is omitted, the editor uses the default filespec /EOS/NO.EMULATOR/ace.cfg.

If any of the filespecs entered is longer than 64 characters, the following message is displayed:

```
FILE NAME {filespec} IS TOO LONG.
```

Any error found in the ACE command line causes an error message to be displayed, and control is returned to the operating system.

EXAMPLE

Suppose you want to invoke the editor interactively to edit the file PROG1, and you want to place the resulting edited file in the file PROG1.1. No command file is to be used and the default configuration file is to be used. The following example shows you how this is done. All underlined text is entered exactly as shown. (The term "(CR)" in command entry refers to the return key.)

```
ACE(CR)
```

```
Version 1.X
```

```
ENTER NAME OF FILE TO BE EDITED:
```

```
PROG1(CR)
```

```
ENTER NAME OF FILE TO RECEIVE RESULTS OF EDIT SESSION:
```

```
PROG1.1(CR)
```


ENTER NAME OF COMMAND FILE:

(CR)

ENTER NAME OF CONFIGURATION FILE:

(CR)

The terminal screen now displays the editor status lines and the first part of your file, as shown in the following display. You are now ready to begin editing.

```
*This is a sample window display of the ACE editor.~
* SAMPLE PROGRAM~
if (tabindx > Tab.set) {~
    tabindx = -1;~
~
-----
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:
```

The next example shows how you would invoke the editor to edit the file AB1.S, and to place the resulting edited file in AC1.S1. All editing is done by the command file EDIT.COMM1, and you want to use the alternate configuration file ADM-3A.CFG.

ACE AB1.S AC1.S1 EDIT.COMM1 ADM-3A.CFG(CR)

If the echo option is set off and the command file contains an editor termination command, the editing occurs without changing the screen. However, if the echo option is on, all changes made to the file are shown on the screen. If the command file does not contain an editor termination command, you can continue editing at the place in the file where the command file finished.

SYNTAX

A	Advances the cursor to the beginning of the next line.
OA	Advances the cursor to the beginning of the current line.
-A	Advances the cursor to the beginning of the previous line.
[count]A	Advances the cursor to the beginning of the count -th next line.
-[count]A	Advances the cursor to the beginning of the count -th preceding line.
/A	Advances the cursor to the beginning of the last line of the file.
-/A	Advances the cursor to the beginning of the file.

EXPLANATION

The **A** command moves the cursor to the beginning of the current line and then advances the cursor forward or backward the specified number of lines. A positive count moves the cursor toward the end of the file. A negative count moves the cursor toward the beginning of the file. Zero count moves the cursor to the beginning of the current line.

For a negative count, if **count** exceeds the number of lines between the cursor and the beginning of the file, the cursor is moved to the first line of the file.

For a positive count, if **count** exceeds the number of lines between the cursor and the end of the file, the cursor is moved to the last line of the file.

If the **A** command moves the cursor beyond the window, the window is redisplayed. Otherwise, the window is unchanged. If the beginning or end of the file is encountered while the **A** command is in progress, the appropriate message is displayed on the message line:

BEGINNING OF FILE ENCOUNTERED.

or

END OF FILE ENCOUNTERED.

The window will contain the page of text at the beginning or end of the file.

The **A** command causes the leftmost column of text to be displayed in column one of the window.

EXAMPLE

```

while (cur-col >= Tab~
  tab-ndx++;~
  if (tab-ndx > Tab.set) {~
    tab-ndx = -1;~
  while (cur-char >= TAB~
  -----
^001 00017 ENTER COUNT/COMMAND      COUNT:  COMMAND:

```

To move the cursor down 2 lines in the file, enter:

2A

The new cursor position is shown in the following window display:

```

while (cur-col >= Tab~
  tab-ndx++;~
_ if (tab-ndx > Tab.set) {~
  tab-ndx = -1;~
while (cur-char >= TAB~
  -----
^001 00017 ENTER COUNT/COMMAND      COUNT:2  COMMAND:A

```

To move the cursor to beginning of the last line of the file, enter:

/A

The following window is now displayed:

```

  tab-ndx++;~
  if (tab-ndx > Tab.set) {~
    tab-ndx = -1;~
while (cur-char >= TAB~
_ this is the end of the program
  -----
^001 00018 ENTER COUNT/COMMAND      COUNT:/  COMMAND:A

END OF FILE ENCOUNTERED.

```

SYNTAX

C[{string}](ESC)	Replaces the character at the cursor with the specified string.
OC[{string}](ESC)	Replaces zero characters with the specified string. In effect, inserts the specified string before the cursor.
-C[{string}](ESC)	Replaces the character preceding the cursor with the specified string.
[count]C[{string}](ESC)	Replaces the specified number of characters including and following the cursor with the specified string.
-[count]C[{string}](ESC)	Replaces the specified number of characters preceding the cursor with the specified string.
>C[{string}](ESC)	Replaces characters including and following the cursor with the specified string. The number of characters is determined by the length of the last string found with the F or R command. If no prior F or R command has been used, the value of ">" is zero.
<C[{string}](ESC)	Replaces characters preceding the cursor with the specified string. The number of characters is determined by the length of the last string found with the F or R command. If no prior F or R command has been used, the value of "<" is zero.
/C[{string}](ESC)	Replaces the characters from the cursor to the end of the current line with the specified string. Does not change the end-of-line character.
-/C[{string}](ESC)	Replaces the characters between the cursor and the beginning of the current line with the specified string. Does not change the current character.

EXPLANATION

The **C** command replaces the specified number of characters before or after the cursor with the specified string. A negative count changes characters preceding the cursor. A positive count changes characters starting with the cursor toward the end of the file. The string may contain the escape wildcard character, which is converted into the (ESC) character. The general and case ignore wildcard characters, however, are treated literally. For information regarding the general, case-ignore, and escape wildcards, refer to the **F**, **R**, **WE**, **WG**, or **WI** commands in this Command Dictionary.

EXAMPLE

This argument holds for any value of K up to and~
including x58. Consider the next cases K=x15 through~
K=x45.~

~

~

^001 00021 ENTER COUNT/COMMAND COUNT: COMMAND:

To change the 58 at the cursor in the example window to a 278, enter:

C27(ESC)

The 58 is changed to a 278 and the following window is displayed:

This argument holds for any value of K up to and~
including x278. Consider the next cases K=x15 through~
K=x45.~

~

~

^001 00021 ENTER COUNT/COMMAND COUNT: COMMAND:C

7

not completed and the contents of P upon reset may not~
necessarily be the address of the instruction that~
would have been executed next. It may, for instance,~
point to an immediate operand if the reset occurred~
during the second cycle of a~

^001 00013 ENTER COUNT/COMMAND COUNT: COMMAND:

If you want to change the two words "for instance," to read "for example," enter the following command line:

/Cfor example,(ESC)

The following window is displayed:

not completed and the contents of P upon reset may not~
necessarily be the address of the instruction that~
would have been executed next. It may, for example,~
point to an immediate operand if the reset occurred~
during the second cycle of a~

^001 00013 ENTER COUNT/COMMAND COUNT:/ COMMAND:C
for example,

(cd)

Move cursor down

SYNTAX

- (cd) Moves the cursor down one line.
- [count](cd) Moves the cursor down the specified number of lines.

EXPLANATION

The **(cd)** command moves the cursor toward the bottom of the screen. If the cursor is already at the bottom of the window or on the last line of a partially filled window, the cursor will roll over to the top of the window, remaining in the same column position. The window remains unchanged.

SYNTAX

(ch) Moves the cursor to the home position.

EXPLANATION

The **(ch)** command moves the cursor to the home position of the window. If the cursor is already in the home position, no action occurs.

EXAMPLE

```

      ORG      100H~
PLAN LXI    H,500H~
      MVI     B,5~
      XRA    A~
LOOP ADD    M~
-----
^001 00001 ENTER COUNT/COMMAND   COUNT:  COMMAND:

```

To move the cursor to the upper left corner of the window, enter the **(ch)** command.

```

-   ORG      100H~
PLAN LXI    H,500H~
      MVI     B,5~
      XRA    A~
LOOP ADD    M~
-----
^001 00001 ENTER COUNT/COMMAND   COUNT:  COMMAND:

```


SYNTAX

- (cl) Moves the cursor left one character.
- [count](cl) Moves the cursor left the specified number of characters.

EXPLANATION

The **(cl)** command moves the cursor toward the left side of the screen. If the cursor is already in the leftmost column of the screen, the cursor is moved up to the rightmost position in the preceding line. If the cursor is in the home position, the **(cl)** command moves the cursor to the end-of-line character of the last line of the window.

SYNTAX

(cmd-esc)[count](command)[parameters]

Allows entry of a command while in revise mode.

EXPLANATION

The command-escape (**cmd-esc**) command is used to differentiate between command entry and text entry while the editor is in revise mode. The status line indicates whether or not the editor is in revise mode. When the editor is in revise mode, characters that are entered from the keyboard overwrite characters that are already in the window. As each character is entered, the cursor is moved right one position (as in the cursor right command). This allows modifications to text to be made and seen at the same time.

Other editor commands are valid during revise mode. However, to enter a non-configurable command or a repetition count, you must first enter the (**cmd-esc**) command. Otherwise, the command mnemonic and any other parameters to the command will overwrite characters on the screen. Thus you can effectively revise, insert, and delete characters at the same time without having to switch between modes.

Configurable commands (moving the cursor, scrolling, paging) do not require the (**cmd-esc**) if they are configured to single keys that begin with the control character sequence. The key that enters the (**cmd-esc**) command is defined during the configuration process. Refer to the Technical Notes section of this manual for information regarding the Configurator.

If (**cmd-esc**) is entered twice in succession while in revise mode, or if (**cmd-esc**) is entered while not in revise mode, the following message is displayed on the message line:

```
INVALID USE OF (CMD-ESC) COMMAND.
```

EXAMPLE

To advance the cursor 3 lines while in revise mode, enter:

(cmd-esc)3A

(cr)

Move cursor right

SYNTAX

- (cr) Moves the cursor right one character.
- [count](cr) Moves the cursor right the specified number of characters.

EXPLANATION

The **(cr)** command moves the cursor toward the right side of the screen. If the cursor is already in the rightmost column of the screen or at the end-of-line character, the cursor is moved to the leftmost character in the following line. If the cursor is on the end-of-line character of the last line of the window, the **(cr)** command moves the cursor to the home position.

SYNTAX

- (cu) Moves the cursor up one line.
- [count](cu) Moves the cursor up the specified number of lines.

EXPLANATION

The **(cu)** command moves the cursor toward the top of the screen. If the cursor is already at the top of the screen, the cursor will wrap around to the bottom of the window or to the last line of a partially filled window, remaining in the same column position. The window remains unchanged.

SYNTAX

- (mark)(motion)(dc) Deletes the characters between the mark and the cursor.
- (dc) Deletes the character at the cursor.
- (dc) Deletes the character preceding the cursor.
- [count](dc) Deletes the specified number of characters including and following the cursor.
- [count](dc) Deletes the specified number of characters preceding the cursor.
- >(dc) Deletes characters including and following the cursor. The number of characters is determined by the length of the last string found with the **F** or **R** command. If no prior **F** or **R** command has been used, the **>(dc)** command has no effect.
- <(dc) Deletes characters preceding the cursor. The number of characters is determined by the length of the last string found with the **F** or **R** command. If no prior **F** or **R** command has been used, the **<(dc)** command has no effect.
- /(dc) Deletes the characters from the cursor to the end of the current line. The end-of-line character is not deleted.
- (dc) Deletes the characters between the cursor and the beginning of the current line.

EXPLANATION

The **(dc)** command has two forms: block deletion and simple deletion.

Block Deletion. When you enter (mark)(motion) as a parameter, all characters between and including the mark and the cursor are deleted when you enter the **(dc)** command. The cursor's position in the file is remembered by the editor and an at-sign (@) is displayed at the cursor position. (The at-sign (@) is configurable and can be changed using the configurator. Refer to the Technical Notes section of this manual for information regarding the configurator.) Then when you enter the **(dc)** command, all text between and including the cursor position and the @ (at-sign) is deleted from the window. If the two cursor positions are not contained in the window, and if the query option is on, the editor asks you to confirm the deletion by displaying the following message:

DELETE REQUESTED. ENTER "Y" TO ACKNOWLEDGE:

Any response other than "Y" or "y" bypasses the deletion. If you enter a "Y" or "y", the text is deleted from the window.

Simple Deletion. The second form of the **(dc)** command deletes the specified number of characters before or after the cursor. A positive count deletes the specified number of characters including and following the cursor. A negative count deletes the specified number of characters preceding the cursor.

Two adjacent lines can be concatenated by deleting the end-of-line character of the first line.

Any characters that are deleted by a single **(dc)** command are saved in a buffer. The most recently deleted text can be restored with the **U** (undelete) command.

You can combine the **F** command with the **(dc)** command to find and delete text:

F{string}(ESC) <(dc) Find and delete text.

EXAMPLE

```

0000000ADEE62023ACDB83499D9FDDACBCDFEDCA~
111131422748773094477A6636CAABBE87C5DDA~
DC8335198F8E77ABCE5DFA7EFA6285D74C65AA84~
17DEC8294117AC7EFEAFECBDA63802F8E7AEE83B~
7372958790A87698F987E556EFFF76897CE98798~
-----
^001 00011 ENTER COUNT/COMMAND    COUNT:    COMMAND:
    
```

To delete 25 characters from the cursor toward the end of the file, enter:

25(dc)

The characters are deleted and the following window is displayed:

```

0000000ADEE62023ACDB83499D9FDDACBCDFEDCA~
111131422748773094477A6636CAABBE87C5DDA~
A6285D74C65AA84~
17DEC8294117AC7EFEAFECBDA63802F8E7AEE83B~
7372958790A87698F987E556EFFF76897CE98798~
-----
^001 00011 ENTER COUNT/COMMAND    COUNT:25 COMMAND:
    
```

(dc)

Delete characters

```

This paragraph_of text is the one~
that I want to delete.~
This paragraph is not going to be deleted. It~
will remain in the file.~
~

```

```
-----
^001 00015 ENTER COUNT/COMMAND      COUNT:  COMMAND:
```

If you want to delete the paragraph or block of text that starts with the cursor, first enter the **(mark)** command. The editor then displays an at-sign "@" at the cursor position, and displays a prompt message on the status line:

```

This paragraph@of text is the one~
that I want to delete.~
This paragraph_is not going to be deleted. It~
will remain in the file.~
~

```

```
-----
^001 00015 ENTER MOTION              COUNT:  COMMAND:
```

Move the cursor to the end of the text that you want to delete, and enter the **(dc)** command. The following window is displayed:

```

This paragraph~
is not going to be deleted. It~
will remain in the file.~
~
~
~

```

```
-----
^001 00015 ENTER COUNT/COMMAND      COUNT:  COMMAND:
```

SYNTAX

(dl)	Deletes the current line.
-(dl)	Deletes the preceding line.
[count](dl)	Deletes the current line and the following count-1 lines.
-[count](dl)	Deletes the specified number of lines preceding the current line.
/(dl)	Deletes the current line and all the following lines to the end of the file.
-(/dl)	Deletes all the preceding lines.

EXPLANATION

The **(dl)** command moves the cursor to the beginning of the current line and then deletes the specified number of lines forward or backward from the current cursor position. If you specify a positive count, the lines are deleted toward the end of the file. If you specify a negative count, the lines are deleted toward the beginning of the file.

For a negative count, if **count** exceeds the number of lines between the cursor and the beginning of the file, the cursor is moved to the first line of the file.

For a positive count, if **count** exceeds the number of lines between the cursor and the end of the file, the cursor is moved to the last line of the file. The lines deleted by the **(dl)** command are removed from the window. The deleted lines are saved in a buffer. The most recently lines deleted can be restored with the **U** (undelete) command.

If more than a full page is to be deleted, and if the query option is **ON**, the editor asks you to confirm the deletion, by displaying the following message:

DELETE REQUESTED. ENTER "Y" TO ACKNOWLEDGE:

If you enter a **"Y"** or **"y"**, the deletion is performed. Any other response bypasses the deletion. The **(dl)** command causes physical column one to be displayed in the leftmost column of the window.

EXAMPLE

```
Portland~  
Seattle~  
Vancouver~  
Tacoma~  
Olympia~  
-----  
^001 00015 ENTER COUNT/COMMAND    COUNT:  COMMAND:
```

To delete the 3 lines of text following and including the current line, enter:

3(dl)

The lines are deleted and the following window is displayed.

```
Portland~  
Olympia~  
San Francisco~  
San Diego~  
Los Angeles~  
-----  
^001 00015 ENTER COUNT/COMMAND    COUNT:3  COMMAND:
```

To delete the current line and all of the following lines to the end of the file, enter:

/(dl)

The rest of the file is deleted and the following window is displayed:

```
Anchorage~  
Sitka~  
Fairbanks~  
Victoria~  
Portland~  
-----  
^001 00011 ENTER COUNT/COMMAND    COUNT:/  COMMAND:
```

SYNTAX

EC	Turns the echo option ON.
-EC	Turns the echo option OFF.

EXPLANATION

The **EC** command is used to turn the echo option ON or OFF. When the echo option is OFF, if commands are being processed from a command file, no display is generated by the editor until all commands have been processed or until an error is detected. If the echo option is OFF, however, the effect of each subsequent command is shown in the status line and window as if the command had been entered by the user on the keyboard. After you enter the **EC** or **-EC** command, the respective message is displayed on the message line:

ECHO ON

or

ECHO OFF

The **EC** command affects command echoing only when a command file is being processed.

SYNTAX

EP Displays the editor profile.

EXPLANATION

The **EP** command displays all of the user-definable parameters that have been entered by previously issued commands. These parameters include: file names, search string, search string length, replacement string, tab stops, query and echo options, and wildcard characters. (Macros are displayed with the **ML** command.)

When you enter the **EP** command, the current window is cleared and used to display the editor profile information. The following message is displayed on the message line:

```
DEPRESS CARRIAGE RETURN TO CONTINUE:
```

The profile information remains in the window until you press the return key. At that time, the editor redisplay the contents of the window before **EP** was entered.

Figure 3-3 is a sample editor profile display generated by the **EP** command.

```

-- FILE NAMES --
INPUT      : (input file name)
OUTPUT     : (output file name)
COMMAND    : (command file name)
READ/WRITE : (read/write file name)
CONFIG     : (config file name)

-- TAB STOPS DEFINED --
  9 17 25 33 41 49 57 65 73 81 89 97
105 113 121 129 137 145 153 161 169 177 185 193
-- F AND R SEARCH STRINGS --
LENGTH=[count]
{string}
-- R REPLACEMENT STRING --
{string}
-- OPTIONS --
QUERY=[ON/OFF] ECHO=[ON/OFF]
-- WILDCARD CHARACTERS --
GENERAL <C>  ESCAPE <C>  CASE IGNORE <C>

```

3573-6

Fig. 3-3. Editor Profile Display Format.

SYNTAX

ER Reads all text from the edit read/write file.

[count]ER Reads all text from the edit read/write file, the specified number of times.

EXPLANATION

The **ER** command reads the file that has been opened with the **O** (open) command. The contents of the file are inserted preceding the cursor. The file is rewound by **ER**; thus, successive invocations of **ER** cause multiple copies of the file to be inserted into the window. The **count** parameter specifies the number of times the file is to be inserted. If no file has been opened with the **O** (open) command, the window is not affected and, the following message is displayed on the message line:

NO EDIT READ/WRITE FILE HAS BEEN OPENED.

The **ER** and **EW** commands can be combined to copy portions of text to other parts of the current file. To do this, use the **EW** command to write the block of text to be copied into the edit read/write file. Then, move the cursor to the place in the file where the copy is to be made, and use the **ER** command to read the block back into the file.

EXAMPLE

The file SUB12 contains the following information:

Tokyo
Hong Kong

To open the file SUB12, enter:

OSUB12(ESC)

The file is opened for reading (or writing). The following information is displayed:

```
Peking~
Singapore~
Shanghai~
Sapporo~
Kyoto~
-----
^O01 00109 ENTER COUNT/COMMAND COUNT: COMMAND:0
FILE SUB12 OPENED.
```

Position the cursor at the place in the file that you want to insert the file and enter:

ER

The entire contents of the file SUB12 are inserted into the window preceding the cursor, as shown in the following window display.

```
Peking~  
Singapore~  
Tokyo~  
Hong Kong~  
Shanghai~  
-----  
^001 00109 ENTER COUNT/COMMAND      COUNT:  COMMAND:ER
```

SYNTAX

(mark)(motion)EW	Writes the block of text between and including the position of the cursor when the mark key was pressed and the position of the cursor when the EW command was entered to the edit read/write file.
EW	Write the character at the cursor to the edit read/write file.
-EW	Writes the character preceding the cursor to the edit read/write file.
[count]EW	Writes the specified number of characters including and following the cursor to the edit read/write file.
-[count]EW	Writes the specified number of characters preceding the cursor to the edit read/write file.
/EW	Writes the characters from the cursor to the end of the current line. Does not include the end-of-line character to the edit read/write file.
-/EW	Writes the characters between the cursor and the beginning of the current line to the edit read/write file.

EXPLANATION

The **EW** command has two forms. In the first form, the editor marks the cursor position at the instant you entered the (mark) command; that is, the editor remembers the cursor's position in the file and displays an at-sign (@) at the cursor position. (The at-sign (@) is configurable and can be changed using the configurator. Refer to the Technical Notes section of this manual for information regarding the configuration process.) Then when you enter the **EW** command all text between and including the cursor position and the @ (at-sign) is written to the file opened with the **O** (open) command.

The second form of the **EW** command writes the specified number of characters before or after the cursor. A negative count causes the specified number of characters preceding the cursor to be written to the file. A positive count causes the specified number of characters from the cursor position to be written to the file. The position of the cursor is not changed by the **EW** command.

The contents of the file that you are writing to can be "read" and inserted back into the window at a different place using the **ER** command. The **EW** command does not rewind the file before writing; thus, successive **EW** commands will append text to the file.

If no file has been opened with the **O** (open) command, the following message is displayed on the message line:

NO EDIT READ/WRITE FILE HAS BEEN OPENED

The **ER** and **EW** commands can be combined to copy portions of text to other parts of the current file or even another file. To do this, use the **EW** command to write the block of text to be copied into the edit read/write file. Then move the cursor to the place in the file where the copy is to be made, and enter the **ER** command to read the block back into the file.

EXAMPLE

Before using the **EW** (edit write) command, use the **O** (open) command to create a file named TESTDATA to receive the text. To do this, enter:

OTESTDATA(ESC)

The following monitor area is displayed:

```
-----  
^001 00001 ENTER COUNT/COMMAND    COUNT:  COMMAND:O  
  
FILE TESTDATA CREATED.
```

To write 100 characters starting with the current character, enter:

100EW

The characters are written to the file TESTDATA.

SYNTAX

EX Exits the editor.

EXPLANATION

The **EX** command ends the edit session. The remainder of the unedited input file is copied into the output file. The following message is displayed on the message line:

FILE ACCESS IN PROGRESS.

All files are closed, any temporary files are deleted, the backup copy of input file is renamed to "**#inputfile**" if no alternate output file is specified, and the editor returns control to the operating system. A backup file is not created if the editor is being used to create a new file.

To exit the editor without saving the file being edited, use the **S** (stop) editor command.

SYNTAX

F{string}(ESC)	Finds the next occurrence of the specified string following the cursor.
[count]F(string)(ESC)	Finds the count -th occurrence of the specified string following the cursor.
/F{string}(ESC)	Counts the number of occurrences of the string following the cursor. The cursor is not moved.
F(ESC)	Finds the next occurrence of the string specified with a previous F or R command.
[count]F(ESC)	Finds the count -th occurrence of the string specified with the previous F or R command.
/F(ESC)	Counts the number of occurrences of the string specified with the previous F or R command.

EXPLANATION

The **F** command searches for the specified string, starting after the cursor and continuing to the end of the file. If string is null, the last string found with a previous **F** or **R** command is searched for again. When a null string is specified and no prior **F** or **R** has been issued, the following message is displayed:

NO PREVIOUS "F" OR "R" COMMAND USED. SEARCH STRING IS UNDEFINED.

If a search is performed, and a match is found, the window then shows the lines surrounding the found string. The cursor is located at the character following the found string. If string is not found in the file, the window is unaffected and the following message is displayed on the message line:

NO OCCURRENCES OF {string} FOUND.

If the optional count is entered, **F** searches for the specified number of occurrences of the string before redisplaying the window. The window shows the lines surrounding the **count**-th occurrence of the string. If the specified count exceeds the actual number of occurrences of the string between the cursor and the end of the file, the window shows the lines surrounding the last matched occurrence, and the following message is displayed:

xx OCCURRENCES OF {string} FOUND.

When you count all occurrences of the string, the cursor and window are not changed, but a message showing the total number of matches found is displayed on the message line.

The length of the matched string is retained by the editor for use with the "<" and ">" forms of the repetition count.

The **F** command may cause the window to show the text shifted right or left from its original column position.

Wildcard Characters. The search string may contain the general, escape, or case-ignore wildcard characters. You must enter an even number of case ignore-wildcard characters in the search string: if you do not, the **F** command is terminated and the following message is displayed:

UNBALANCED CASE IGNORE WILDCARD CHARACTER, COMMAND ABORTED.

If two successive case-ignore wildcard characters are included in the search string, the **F** command is terminated, and the following message is displayed:

CASE IGNORE STRING MUST NOT BE NULL.

NOTE

*The use of wildcards in search strings causes the **F** command to run slower. The **F** command is more efficient when longer or more specific search strings are used.*

EXAMPLE

To find the 14th occurrence of the character string "can" in the file, first move the cursor to the beginning of the file:

-/A

Now enter:

14Fcan(ESC)

The following window is displayed. The cursor is located following the 14th occurrence of the string "can".

```
tests of numerical data include various combinations~
of "less than", "greater than", "equal to", and~
"not equal to"; alphabetic strings can also be~
compared, but only for equivalence.~
~
```

```
-----
^001 00131 ENTER COUNT/COMMAND      COUNT:14  COMMAND:F
can
```

F

If you want to count how many times the character string "can" occurs in your file, move the cursor to the beginning of the file by entering:

-/A

Then enter the following command line to count the occurrences:

/Fcan(ESC)

The window displays the beginning of the file; the message line contains the number of occurrences.

```
_Labels within programs are generally local~  
so that the same labels can be repeated in ~  
different programs without interference.~  
Subroutine calls and branches can be made only to~  
a label; there is no absolute addressing~  
-----  
^001 0001 ENTER COUNT/COMMAND      COUNT:/  COMMAND:F  
can  
21 OCCURRENCES OF can FOUND.
```

SYNTAX

H Displays the command syntax.

EXPLANATION

The H command displays the syntax of all the editor commands. The current window is cleared and used to display the command syntax information. The following message is displayed on the message line:

DEPRESS CARRIAGE RETURN TO CONTINUE:

The syntax information remains in the window until you enter a return. At that time, if more syntax information exists than can be displayed in one window, the next page of information is shown. When all of the syntax information has been displayed, the window resumes its contents before H was invoked. The format of the command syntax as shown by the H command is presented in Figures 3-4 and 3-5.

CONFIGURABLE EDITOR COMMANDS			
Function		Optional Count	Cmd
-----		-----	---
Cursor Up		n	(cu)
Cursor Down		n	(cd)
Cursor Left		n	(cl)
Cursor Right		n	(cr)
Cursor Home			(ch)
Scroll Down		n	(sd)
Scroll Up		n	(su)
Scroll Right		n	(sr)
Scroll Left		n	(sl)
Page Down		n	(pd)
Page Up		n	(pu)
Delete Characters	> < / -/ - -n	n	(dc)
	(mark) (motion)	(dc)	
Delete Lines	/ -/ - -n	n	(dl)
Insert Characters			(ic) {string} (ESC)
Insert Line			(il) {string} (ESC)
Revise Mode			(rm) {string} (rm)
			(rm) (cmd-esc) (cmd) (rm)
Command Escape			(cmd-esc)
Mark Cursor			(mark)

3573-7

Fig. 3-4. Configurable editor commands as displayed by the H command.

NON-CONFIGURABLE EDITOR COMMANDS			
Function		Optional Count	Cmd
-----		-----	---
Advance lines	0	/ -/ n -n -	A
Change characters	0 > <	/ -/ n -n -	C {string} (ESC)
ECho on/off		-	EC
Editor Profile			EP
Edit Read		n	ER
Edit Write		/ -/ n -n -	EW
	(mark)	(motion)	EW
EXit editor, write file			EX
Find string		/ n	F {string} (ESC)
Help			H
Jump characters	> <	/ -/ n -n -	J
Macro Define/Delete			MD {name} (ESC) {string} (ESC)
Macro List			ML
Macro eXecute		/ n	MX {name}
Open edit read/write file			O {string} (ESC)
Query on/off		-	Q
Replace characters		/ n	R {string1} (ESC) {string2} (ESC)
Stop editor			S
define Tab stops			T {string} (ESC)
Undelete characters		n	U
View lines		/ -/ n -n -	V
Escape Wild card define/delete			WE {string} (ESC)
General Wild card define/delete			WG {string} (ESC)
case Ignore Wild card define/delete			WI {string} (ESC)
eXchange Lower case	> <	/ -/ n -n -	XL
eXchange Upper case	> <	/ -/ n -n -	XU

3573-8

Fig. 3-5. Non-configurable editor commands as displayed by the H command.

SYNTAX

(ic){string}(ESC) Inserts string before the cursor.

EXPLANATION

The **(ic)** command is used to enter new characters into the window. When **(ic)** is entered, the current character and the text following the cursor is shifted right one column for each subsequent character that is entered. New text is inserted in this manner until (ESC), return, or CNTRL-X is depressed.

If (ESC) or CNTRL-X is entered while inserting, the insert characters command is terminated. If a return is entered, the text is scrolled down one line from the line containing the cursor unless you are at the bottom line of the window, and a line is created with the cursor at the left side. Any characters shifted to the right before return was entered are located to the right of the cursor on the next line.

The (ESC) character can be inserted into the file by entering the escape wildcard character. When the wildcard is entered, an (ESC) is written to the file, but the **(ic)** command is not terminated. The general and case ignore wildcard characters, however, are treated literally.

The **(ic)** command causes the window to show text shifted left or right from its original column position if text is inserted at the end of a long line and a return is entered.

If the cursor reaches the right side of the window while inserting, the lines are scrolled left to accommodate the additional text.

EXAMPLE

```

termcount := 0;~
sum:= 0;~
read();~
REPEAT~
    termcount := termcount + 1;~
-----
^001 00014 ENTER COUNT/COMMAND    COUNT:  COMMAND:

```

To insert the character string "limit" before the cursor, enter:

(ic)limit(ESC)

The string is inserted, and the following window is displayed:

```
termcount := 0;~  
sum:= 0;~  
read(limit);~  
REPEAT~  
    termcount := termcount + 1;~  
-----  
^001 00014 ENTER COUNT/COMMAND    COUNT:  COMMAND:
```

SYNTAX

- (il)[{string}]{ESC}** Inserts a new line into the window following the current line.
- (il)[{string}]{ESC}** Inserts a new line into the window before the current line.

EXPLANATION

The **(il)** command is used to enter a new line into the window following the current line. The **-(il)** command is used to insert a new line into the window before the current line.

When **(il)** is entered, the text is scrolled down one line starting with the line following the current line and a blank line is inserted. When **-(il)** is entered, the text is scrolled down one line starting with the current line and a blank line is inserted. The cursor is positioned in the leftmost column of the new line. The specified string is then inserted on the new line and is terminated by the (ESC).

EXAMPLE

```

JAN 12, 1872~
OCT 15, 1901~
AUG 20, 1957~
FEB 13, 1981~
JUN 21, 1981~
-----
^001 00010 ENTER COUNT/COMMAND      COUNT:  COMMAND:

```

To insert the date "JAN 15, 1920" after "OCT 15, 1901" in your file, position the cursor on the line before the place you want to insert the new line, and enter:

(il)(TAB)JAN 15, 1920(ESC)

The line is inserted in your file and the following window is displayed:

```

JAN 12, 1872~
OCT 15, 1901~
JAN 15, 1920~
AUG 20, 1957~
FEB 13, 1981~
-----
^001 00010 ENTER COUNT/COMMAND      COUNT:  COMMAND:

```


SYNTAX

- J Moves the cursor one character to the right.
- J Moves the cursor one character to the left
- [count]J Moves the cursor the specified number of characters to the right.
- [count]J Moves the cursor the specified number of characters to the left.
- /J Moves the cursor to the end-of-line character.
- /J Moves the cursor to the beginning of the current line.
- <J Moves the cursor left. The number of characters is determined by the length of the last string found with the **F** or **R** command. If no prior **F** or **R** command has been entered, the <**J** command has no effect.
- >J Moves the cursor right. The number of characters is determined by the length of the last string found with the **F** or **R** command. If no prior **F** or **R** command has been entered, the >**J** command has no effect.

EXPLANATION

The **J** command moves the cursor forward or backward the specified number of characters. A positive count moves the cursor toward the end of the file. A negative count moves the cursor toward the beginning of the file. If the count is zero, an error message is displayed.

For a negative count, if **count** exceeds the total number of lines between the current line and the beginning of the file, the cursor is moved to the first line of the file.

For a positive count, if **count** exceeds the total number of lines between the current line and the end of the file, the cursor is moved to the last line of the file. If the cursor is moved outside the window, the window is redisplayed. Otherwise, the window remains unchanged except for the position of the cursor. If the beginning or the end of the file is encountered, the appropriate message is displayed on the message line:

BEGINNING OF FILE ENCOUNTERED

or

END OF FILE ENCOUNTERED

and the window contains the page of text at the end or at the beginning of the file.

EXAMPLE

Listing 1 shows a program for creating such designs.~
 Using the rotation algorithm,~
 curved patterns that appear to be three-dimensional~
 will be produced.~
 The main function_of the program are~

 ^001 00011 ENTER SEARCH STRING COUNT: COMMAND:F
 function

The F command has just been used to find the string "function". The F command positions the cursor at the character following the found string. The editor remembers the length of the search string and this value can be used as the count for the J command. So, if you enter:

<J

the cursor is moved left as many places as there are characters in the search string, which places it at the beginning of the string "function" and the following window is displayed:

Listing 1 shows a program for creating such designs.~
 Using the rotation algorithm,~
 curved patterns that appear to be three-dimensional~
 will be produced.~
 The main function of the program are~

 ^001 00011 ENTER COUNT/COMMAND COUNT:< COMMAND:J

The cursor can be moved to the end of the current line by entering:

/J

The following window is displayed:

Listing 1 shows a program for creating such designs.~
 Using the rotation algorithm,~
 curved patterns that appear to be three-dimensional~
 will be produced.~
 The main function of the program are~

 ^001 00011 ENTER COUNT/COMMAND COUNT:/ COMMAND:J

SYNTAX

(mark)(motion) Marks a block of text.

EXPLANATION

The **(mark)** command is used to mark the beginning or end of a block of text that is to be processed by the **(dc)** or the **EW** command.

When you enter the **(mark)** command, an at-sign (@) is displayed in place of the current character to mark its position and the message "MARK DEFINED" is displayed. (The "@" is a configurable character, and can be changed using the configurator. Refer to the Technical Notes section of this manual for information regarding the configurator.) The character at the mark is still there; it is just masked by the mark character.

After you mark the starting point or ending point of a block of text, move the cursor to the other end of the text and enter the **(dc)** or **EW** command. The command is performed on the entire block of text, including the marked character and the current character.

All commands are valid after you have entered the **(mark)** command. However, if you overtype the marked character in revise mode, the mark is removed, the character is replaced, and the message "MARK REMOVED" is displayed.

If you enter CNTRL-X after entering the **(mark)** command and before entering the **(dc)** or **EW** command, the mark is removed. This will occur even if you are using the CNTRL-X to terminate another command. If this occurs, the message "MARK REMOVED, COMMAND ABORTED" is displayed.

When you enter the **(dc)** or **EW** command, do not enter a repetition count; if you do, the message "COUNT INVALID WHEN MARK DEFINED" is displayed and the **(dc)** or **EW** command is aborted.

EXAMPLE

```
    This paragraph_of text is the one~  
    that I want to delete.~  
    This paragraph is not going to be deleted. It~  
    will remain in the file.~  
    ~  
-----  
^001 00015 ENTER COUNT/COMMAND      COUNT:      COMMAND:
```

To delete the block of text that starts with the cursor, first enter the (mark) command. An at-sign "@" is displayed at the cursor position and a prompt message is displayed on the status line.

```
This paragraph@of text is the one~  
that I want to delete.~  
This paragraph_is not going to be deleted. It~  
will remain in the file.~  
~  
-----  
^001 00015 ENTER MOTION          COUNT:  COMMAND:
```

Move the cursor to the end of the block that you want to delete, and enter the (dc) command. The text is deleted and the following window is displayed:

```
This paragraph~  
is not going to be deleted. It~  
will remain in the file.~  
~  
~  
~  
-----  
^001 00015 ENTER COUNT/COMMAND    COUNT:  COMMAND:
```

SYNTAX

- MD{name}(ESC){string}(ESC) Defines a macro name to be the editor commands contained in the specified string.
- MD{name}(ESC)(ESC) Removes definition of macro name.

EXPLANATION

The **MD** command allows you to define a new macro or to delete an existing macro. A macro is a stored list of editor commands that can be executed later by referring to the name associated with the macro. The macro name is a decimal digit (0-4), and is used to identify any one of the five macro commands that can be defined. If any macro name other than 0-4 is entered, the following message is displayed on the message line:

```
INVALID MACRO NAME, COMMAND ABORTED.
```

The (ESC) character may not be included in the specified string. However, string may contain the escape wildcard character, which is converted into its actual representation. The general and case-ignore wildcard characters, however, are treated literally.

The **MX** command may not be included in a macro definition; executing a macro from within another macro is not allowed. However, this error can only be detected during macro execution.

If the macro name is valid, and macro is not null, the following message is displayed on the message line:

```
MACRO {name} DEFINED.
```

If you specify a null string, the following message is displayed on the message line:

```
MACRO {name} IS NOW UNDEFINED.
```

EXAMPLE

Suppose you want to create a macro named "1" to move the cursor down 5 lines in the window and then execute the V (view) command on the current line and the next 4 lines. To do this, enter:

```
MD1(ESC)5A5V(ESC)
```

To delete or remove the macro definition named 4, enter:

```
MD4(ESC)(ESC)
```

The second (ESC) removes the macro definition. The following message is displayed:

```
-----
^001 00015 ENTER MACRO DEFINITION COUNT:  COMMAND:MD
4(ESC)(ESC)
MACRO 4 IS NOW UNDEFINED.
```

SYNTAX

ML Displays all the user-defined macro commands.

EXPLANATION

The **ML** command displays all of the user-defined macro commands in the following format:

```
-- MACROS --  
0 {string}  
1 {string}  
2 {string}  
3 {string}  
4 {string}
```

The macro definitions remain in the window until you enter a return. At that time, the editor redisplay the contents of the window before the **ML** command.

SYNTAX

<code>MX{name}</code>	Executes the specified macro once.
<code>[count]MX{name}</code>	Executes the specified macro the specified number of times or until one of the termination conditions is met.
<code>/MX{name}</code>	Executes the specified macro name until one of the termination conditions is met.

EXPLANATION

The **MX** command executes the specified macro until one of the following termination conditions is satisfied:

1. The macro is executed [count] times.
2. A search fails.
3. During command execution, the beginning or the end of the file is encountered.
4. An unrecognized command or parameter or an **MX** command is encountered in the macro string.

If the specified macro is not defined, the following message is displayed on the message line:

```
MACRO {name} IS NOT DEFINED.
```

If any macro name other than 0-4 is entered, the following message is displayed on the message line:

```
INVALID MACRO NAME, COMMAND ABORTED.
```

If condition 1 causes the macro execution to be terminated, then any remaining commands in the original command sequence are performed, if executing from a command file.

If condition 2 or 3 causes the macro execution to be terminated, any remaining commands in the macro string are ignored. If executing from a command file, any remaining commands in the original sequence are ignored.

The appropriate messages are displayed on the message line:

```
MACRO TERMINATED AT BOF.
```

or

```
MACRO TERMINATED AT EOF.
```

and/or

```
xx MACRO EXECUTIONS PERFORMED.
```

If an **MX** command is encountered in the macro string, the macro execution is terminated, and the following error message is displayed on the message line:

```
MX COMMAND IS INVALID IN MACRO DEFINITION.
```

Any commands in the macro string prior to the detected error will have been performed. The following message is displayed on the parameter line:

```
xx MACRO EXECUTIONS PERFORMED.
```

If the query option is ON, the editor prompts you to confirm certain commands. If you bypass a command via the query option, the macro termination condition may not necessarily be met yet.

EXAMPLE

Macro "1" is defined as **5A5V**. To execute this macro enter:

```
MX1
```

The following window is displayed:

```
m^LBar^G7#^UU*i<{6&^Ytpr^V+\KbK9^?f^L~RD^KO^D^C^M
CdW^P^C^x^HK(O]^PFLU,75L:^Gt-j&^^^TrM4/G<Oy^KLk^M
+J)^T.U'^Yc^M
HDN^?^]^F^`<4^N^O^A^L^f^(.^O^i.^!^?^$^N^A^G^O)n!N^K^M
=WM-#ER ,%EUpj^?^!Nu(hI~'uG^N^O^ms/^JJ3J(t9^M
-----
^O01 0001 ENTER COUNT/COMMAND      COUNT:5  COMMAND:V
DEPRESS CARRIAGE RETURN TO CONTINUE. _
```


SYNTAX

- O{filespec}(ESC) Opens the specified file for use by the **ER** and **EW** commands.
- O(ESC) Closes the file opened with the **O** command.

EXPLANATION

The **O** command opens the specified file for use with the **ER** and **EW** commands. If you try to open a second file, the first file is automatically closed before the second file is opened.

CAUTION

The name of your file must be a valid filespec. Names of system devices such as "CONO", "LPT", "REMO", etc. are not valid filespecs for use with the editor. Also, file names that begin with "<" or ">" are not valid.

The specified filespec must be a valid disc file name and must differ from the names of the files specified in the ACE invocation. Otherwise, the file is not opened, and the following message is displayed on the message line:

FILE {string} IS IN USE. INVALID FOR EDIT READ/WRITE FILE.

If you specify a null filespec, the currently open file is closed and the following message is displayed on the message line:

EDIT READ/WRITE FILE IS NOW UNDEFINED.

If the specified filespec does not exist, the following message is displayed on the message line:

FILE {string} CREATED.

Otherwise, the following message is displayed on the message line:

FILE {string} OPENED.

EXAMPLE

To open the existing file SUB12, enter:

OSUB12(ESC)

The file is opened for reading (or writing) and the following window and message line are displayed:

```
Peking~  
Singapore~  
Shanghai~  
Sapporo~  
Kyoto~  
-----  
001 00109 ENTER COUNT/COMMAND COUNT: COMMAND:0  
  
FILE SUB12 OPENED.
```

To use the **EW** (edit write) command, first use the **O** (open) command to create a file named **TESTDATA** to receive the text. To do this, enter:

OTESTDATA(ESC)

The following monitor area is displayed:

```
-----  
^001 00001 ENTER COUNT/COMMAND COUNT: COMMAND:  
  
FILE TESTDATA CREATED.
```

SYNTAX

- (pd) Moves the text down one page.
- [count](pd) Moves the text down the specified number of pages.

EXPLANATION

The **(pd)** command causes the text in the window to be moved down, bringing into the window text nearer the beginning of the file. If no count is entered, the text is moved down one page. Otherwise, the text is moved down the specified number of pages. The cursor remains in the same relative position in the window. If the window is already showing the beginning of the file or if the beginning of the file is encountered in the new page, the following message is displayed on the message line:

BEGINNING OF FILE ENCOUNTERED.

SYNTAX

- (pu) Moves the text up one page.
- [count](pu) Moves the text up the specified number of pages.

EXPLANATION

The **(pu)** command causes the text in the window to be moved up, bringing into the window text nearer the end of the file. If no count is entered, the text is moved up one page. Otherwise, the text is moved up the specified number of pages. The cursor remains in the same relative position in the window. If the window is already showing the end of the file or if the end of the file is encountered in the new page, the following message is displayed on the message line:

END OF FILE ENCOUNTERED.

Q

SYNTAX

- Q Turns the query option ON.
- Q Turns the query option OFF.

EXPLANATION

The **Q** command is used to turn the query option on or off. The query option affects the **(dc)**, **(dl)**, and **R** commands. The query option prompts you to confirm an action that is potentially destructive to the file. The appropriate message is displayed on the message line:

QUERY ON.

or

QUERY OFF.

EXAMPLE

Enter the following command:

Q

The query option is turned on and the following display is generated:

```
-----  
^001 00001 ENTER COUNT/COMMAND    COUNT:  COMMAND:Q  
  
QUERY ON
```

Then if you try to delete 35 lines of text, the editor will prompt you to confirm the deletion:

```
-----  
^001 00001 ENTER COUNT/COMMAND    COUNT:35 COMMAND:  
  
DELETE REQUESTED. ENTER Y TO ACKNOWLEDGE: _
```

If you enter a "Y" or "y", the lines are deleted. Any other response aborts the **(dl)** command.

SYNTAX

- R{string1}{ESC}{string2}{ESC}**
Replaces the next occurrence of string1, starting with the cursor, with string2.
- [count]R{string1}{ESC}{string2}{ESC}**
Replaces the specified number of occurrences of string1, starting with the cursor, with string2.
- /R{string1}{ESC}{string2}{ESC}**
Replaces all occurrences of string1, starting with the cursor, with string2.
- R(ESC){string2}{ESC}**
Replaces the next occurrence of string1 (found with the last **F** or **R** command) with string2.
- R(ESC){string2}(ESC)**
Replaces the next occurrence of string1 (found with the last **F** or **R** command) with string2.
- R{string1}{ESC}(ESC)**
Replaces the next occurrence of string1 with the last **R** command replacement string.
- R(ESC)(ESC)**
Replaces the next occurrence of the last **F** or **R** search string the last **R** command replacement string.
- [count]R(ESC){string2}{ESC}**
Replaces the specified number of occurrences of string1 (from the last **F** or **R** command) with string2, starting from the cursor position.
- /R(ESC){string2}{ESC}**
Replaces all occurrences of string1 (from the last **F** or **R** command) with string2.
- [count]R{string1}{ESC}(ESC)**
Replaces the specified number of occurrences of string1 with the last **R** command replacement string, starting from the cursor position.
- /R{string1}{ESC}(ESC)**
Replaces all occurrences of string1 with the replacement string from the last **R** command.
- [count]R(ESC)(ESC)**
Replaces the specified number of occurrences of the search string from the last **F** or **R** command with the replacement string from the last **R** command, starting from the cursor position.
- /R(ESC)(ESC)**
Replaces all occurrences of the search string from the last **F** or **R** command with the replacement string from the last **R** command.

EXPLANATION

The **R** command searches for string1 and replaces it with string2. If string1 is a null string, the last string1 specified with a previous **F** or **R** command is used again. If no previous **F** or **R** command has been used, the command has no effect and the following message is displayed on the message line:

NO PREVIOUS "F" OR "R" COMMAND USED. SEARCH STRING IS UNDEFINED.

If string2 is a null string, the replacement string from the previous **R** command is used again. If no previous **R** command has been used, the command has no effect and the following message is displayed on the message line:

NO PREVIOUS "R" COMMAND USED. REPLACEMENT STRING IS UNDEFINED.

NOTE

*You cannot use the **R** command to replace a string with a null string. Use **F{string}<(dc)** instead.*

The search begins at the cursor and continues toward the end of the file. If a match is found, the window is redisplayed. The window then shows the lines surrounding the replaced string. The cursor is located to the right of the replaced string. If string1 is not found in the file, the window is unaffected and the following message is displayed on the message line:

NO REPLACEMENTS OF {string} DONE.

If the optional count parameter is entered, the **R** command searches for the specified number of occurrences of string1 and replaces them with string2 before redisplaying the window. The window shows the lines surrounding the count-th replacement of string1. If the specified count exceeds the actual number of occurrences of string1 between the cursor and the end of the file, the cursor is positioned to the right of the last replacement and the following message is displayed on the message line:

xx REPLACEMENTS OF {string1} DONE.

If the query option is on, and if multiple replacements of string1 have been specified, the window is redisplayed for each occurrence of string1 and the following message is displayed on the message line:

REPLACEMENT REQUESTED. ENTER "Y" TO ACKNOWLEDGE.

If you enter "Y" or "y", the replacement is performed. Any other response causes the particular replacement to be bypassed.

As each window is displayed, the cursor is positioned after string1. The **R** command may cause the text to be moved right or left in the window from its originally displayed position.

Both string1 and string2 may contain wildcard characters. If more general wildcard characters appear in string2 than in string1, the excess wildcard characters are treated literally. For example, "*" is the general wildcard character, the command "Rabc*(ESC)x**(ESC)" causes the string "abcd" to be replaced by the string "xd**".

EXAMPLE

Listing 1 shows a program for creating such designs.~
 Using the rotation algorithm,~
 curved patterns that appear to be three-dimensional~
 will be prduced.~

The main function of the program are~

 ^001 00011 ENTER COUNT/COMMAND COUNT: COMMAND:

If you want to replace the "prduced" with the word "produced", enter the following command sequence:

Rprduced(ESC)produced(ESC)

The word is replaced and the following window is displayed:

Listing 1 shows a program for creating such designs.~
 Using the rotation algorithm,~
 curved patterns that appear to be three-dimensional~
 will be produced.~

The main function of the program are~

 ^001 00011 ENTER COUNT/COMMAND COUNT: COMMAND:R
 prduced(ESC)produced(ESC)

SYNTAX

(rm) Enters or exits revise mode.

EXPLANATION

The status line indicates whether or not the editor is in revise mode. The **(rm)** command is used to enter and exit revise mode. When the editor is in revise mode, characters that are entered from the keyboard overwrite characters that are already in the window. As each character is entered, the cursor is moved right one position (as in the cursor right command). This allows modifications to text to be made and seen at the same time.

Other editor commands are valid during revise mode. However, to enter a non-configurable command or a repetition count, you must first enter the (cmd-esc) command. Otherwise, the command mnemonic and any other parameters to the command will overwrite characters on the screen. Thus you can effectively revise, insert, and delete characters at the same time without having to switch between modes.

Configurable commands (moving the cursor, scrolling, paging) do not require the (cmd-esc) if they are configured to single keys that begin with the control character sequence. The key that enters the (cmd-esc) command is defined during the configuration process. Refer to the Technical Notes section of this manual for information regarding the Configurator.

If (cmd-esc) is entered twice in succession while in revise mode, or if (cmd-esc) is entered while not in revise mode, the following message is displayed on the message line:

INVALID USE OF (CMD-ESC) COMMAND.

If a configurable command begins with a character that has special meaning to the editor (backspace/rubout, tab, CNTRL-X, return), the special meaning takes precedence over the configurable command. For example, if the TAB key is configured to be the (mark) command (as for the CT8500), then pressing the TAB key while in revise mode will cause the current character to be replaced with a TAB character, rather than causing the (mark) command to be executed. To execute the **(mark)** command, you would enter **(cmd-esc) (mark)**. Thus, the **(cmd-esc)** command should be configured to a key that has no special meaning.

Wildcard characters are treated literally in revise mode.

SYNTAX

S Exit the editor without updating files.

EXPLANATION

The **S** command terminates editor execution without saving changes to the file being edited. A backup copy of the original input file is not created and the output file (if specified) is deleted. However, any files that were written to with the **EW** command remain written.

When the **S** command is entered, the editor displays the following message on the message line:

STOP REQUESTED , ENTER "Y" TO ACKNOWLEDGE:

Any response other than "Y" or "y" aborts the **S** command, and causes the following message to be displayed on the message line:

COMMAND ABORTED.

Any commands following a **S** command in a command file are ignored.

SYNTAX

- (sd) Scrolls text down one line.
- [count](sd) Scrolls text down the specified number of lines.

EXPLANATION

The **(sd)** command causes the text in the window to be moved down, bringing into the window text nearer the beginning of the file. If no count is entered, the text is moved down one line. Otherwise, the text is moved down the specified number of lines. The cursor remains in the same relative position in the window. If the window is already showing the first line of the file, the window remains unchanged and the following message is displayed on the message line:

 BEGINNING OF FILE ENCOUNTERED.

SYNTAX

- (sl) Scrolls text left one column.
- [count](sl) Scrolls text left the specified number of columns.

EXPLANATION

The **(sl)** command causes the text in the window to be moved left, bringing into the window text nearer to column 999. If no count is entered, the text is moved left one column. Otherwise, the text is moved left the specified number of columns. The cursor remains in the same relative position in the window. If the window is already showing column 999, the window remains unchanged and the following message is displayed on the message line:

MAXIMUM COLUMN ENCOUNTERED.

SYNTAX

- (sr) Scrolls text right one column.
- [count](sr) Scrolls text right the specified number of columns.

EXPLANATION

The **(sr)** command causes the text in the window to be moved right, bringing into the window text nearer to column 1. If no count is entered, the text is moved right one column. Otherwise, the text is moved right the specified number of columns. The cursor remains in the same relative position in the window. If the window is already showing column 1, the window remains unchanged and the following message is displayed on the message line.

BEGINNING OF LINE ENCOUNTERED.

SYNTAX

- (su) Scrolls text up one line.
- [count](su) Scrolls text up the specified number of lines.

EXPLANATION

The **(su)** command causes the text in the window to be moved up, bringing into the window text nearer the end of the file. If no count is entered, the text is moved up one line. Otherwise, the text is moved up the specified number of lines. The cursor remains in the same relative position in the window. If the window is already showing the last line of the file, the window remains unchanged and the following message is displayed on the message line:

END OF FILE ENCOUNTERED

SYNTAX

T{string}(ESC)	Defines tab stops.
T(ESC)	Clears all tab stops.

EXPLANATION

The **T** command is used to define the positions of tab stops during an edit session. Initially, tab stops are set to the default values 9, 17, 25, . . . 177, 185, 193. The locations of the tab stops are defined by the specified string, which must be a series of decimal column numbers separated by commas. If you specify a null string, all tab stops are cleared. A maximum of 24 tab stops can be defined. If more than 24 tab stops are defined, the following message is displayed on the message line:

TOO MANY TAB STOPS, COMMAND IGNORED.

The sequence of tab stops must be defined in ascending column numbers in the range of 2–998. If the tab stops are defined out of order or if an invalid tab stop is defined, the following message is displayed on the message line:

INVALID TAB STOP, COMMAND IGNORED.

If valid tab stops are entered, the following message is displayed on the message line:

xx TAB STOPS DEFINED.

When a tab character is entered during the **(ic)** command or in revise mode, a single tab character (CNTRL-I) is written to the file. However, the number of spaces specified by the current tab stop definition are inserted into the window. If the tab is not defined, a single space is inserted in the window.

SYNTAX

- U** Inserts the text from the last deletion into the window, preceding the cursor.
- [count]U** Inserts the text from the last deletion into the window the specified number of times.

EXPLANATION

The **U** command causes the text from the most recent (**dc**) or (**dl**) command to be inserted immediately to the left of the cursor. Thus, if you accidentally delete text, you can use a single **U** command to restore the deleted text without having to reposition the cursor. To undelete the text more than once you can specify a count.

The **U** command works in the following way:

1. Deleted text is stored in the undelete buffer.
2. The **U** command reads the contents of this buffer into the file at preceding the current character.
3. The undelete buffer is not affected until another delete occurs.

The **U** command can be used with the (**dc**) and (**dl**) commands to move text from one location in a file to another. To do this, delete the text that you want to move, move the cursor to the destination location, and enter the **U** command. The deleted text is inserted before the current character. This text can be undeleted as many times as you want until another delete command is entered.

EXAMPLE

```

Don't lose ~
Your head~
To save a minute~
You need your head~
Your brains are in it.~

```

```

-----
^001 00015 ENTER COUNT/COMMAND    COUNT:  COMMAND:

```


U

Undelete characters

Command Dictionary—8500 Series ACE Users

The following commands delete the line, move the cursor to the beginning of the file, and undelete the text at the desired location:

```
/(dl)  
-/A  
U
```

The line is moved to the beginning of the file and the following window is displayed:

```
You need your head~  
This is the beginning of the file *****~  
  This file contains~  
data related to the number~  
of occurrences of the~  
-----  
^001 00001 ENTER COUNT/COMMAND      COUNT:  COMMAND:U
```

SYNTAX

V	View non-displayable characters in the current line.
-V	View non-displayable characters in the preceding line.
[count]V	View non-displayable characters in the current line and the next count-1 lines.
-[count]V	View non-displayable characters in the specified number of lines preceding the current line.
/V	View non-displayable characters in the current line and all of the following lines to the end of the file.
-/V	View non-displayable characters in all of the preceding lines in the file.

EXPLANATION

The **V** command displays the specified number of lines with any non-displayable characters converted into a displayable form. This command does not modify the lines; it simply allows you to see characters that are part of the line, but have no displayable representation. If you enter a positive count, lines are displayed from the current line toward the end of the file. If you enter a negative count, lines are displayed from the preceding line toward the beginning of the file.

Non-displayable characters can be generated from a standard ASCII keyboard by holding down the "CNTRL" key simultaneously with one of the displayable characters. The **V** command makes non-displayable characters visible by displaying them as a two-character sequence. Table 3-2 lists the two-character representation of the non-displayable characters.

When **V** is invoked, the window is cleared, the lines are displayed, and the following message is displayed on the message line:

DEPRESS CARRIAGE RETURN TO CONTINUE.

The lines remain in the window until you press the return key. If more lines are to be displayed than can be shown in the window, pressing the return key causes the next page of lines to be displayed. When all of the lines have been displayed the, the editor redisplay the contents of the window before the **V** command was invoked.

If the display length of the line exceeds the number of columns in the window, any remaining characters are truncated.

Table 3-2
How Non-Displayable Characters Are Represented by ACE

Hex value	Name	Shown as	Hex value	Name	Shown as
00	NUL	^@	10	DLE	^P
01	SOH	^A	11	DC1	^Q
02	STX	^B	12	DC2	^R
03	ETX	^C	13	DC3	^S
04	EOT	^D	14	DC4	^T
05	ENQ	^E	15	NAK	^U
06	ACK	^F	16	SYN	^V
07	BEL	^G	17	ETB	^W
08	BS	^H	18	CAN	^X
09	HT	^I	19	EM	^Y
0A	LF	^J	1A	SUB	^Z
0B	VT	^K	1B	ESC	^[
0C	FF	^L	1C	FS	^\
0D	CR	^M	1C	FS	^]
0E	SO	^N	1E	RS	^^
0F	SI	^O	1F	US	^_
			7F	DEL	^?

EXAMPLE

To view 5 lines of a special file that contains control characters, enter:

5V

```

m^LBar^G7#\^UU*i<{6&^Ytpr^V+\^KbK9^?f^L~RD^KO^D^C^M
CdW^P^C^x^HK(0)^PFLU,75L:^Gt-j&^^^TrM4/G<Oy^KLk^M
+J)^`T.U'^Yc^M
HDN^?^]^F`<4^N^O^A^Lf((^Oi.!^?^$^N^GO)n!NAK ^M
=WM-#ER ,% ^EUpj^?^!Nu(hI~'uG^N^ms/^JJ3J(t9^M
-----
^O01 00014 ENTER                COUNT:5  COMMAND:V
DEPRESS CARRIAGE RETURN TO CONTINUE.
    
```

SYNTAX

- WE{string}(ESC)** Defines the first character of string to be the escape wildcard character.
- WE(ESC)** Causes the escape wildcard character to be undefined.

EXPLANATION

The **WE** command defines the escape wildcard character. The escape wildcard can be used in place of the (ESC) character in search and replacement strings of the **F** and **R** commands, and in strings of the **(il)**, **(ic)**, and **MD** commands.

The escape wildcard character is initially undefined. The **WE** command defines it to be the first character in the specified string. If more than one character is included in the string, only the first character is used. The following message is displayed on the message line:

```
ESCAPE WILDCARD CHARACTER IS <char>.
```

If you want the escape wildcard character to be undefined, enter the **WE** command followed by an (ESC). The following message is then displayed on the message line:

```
ESCAPE WILDCARD IS NOW UNDEFINED.
```

If you choose a wildcard character that is a non-displayable character or has already been defined by the **WI** or **WG** commands, the following message is displayed on the message line:

```
INVALID WILDCARD CHARACTER, IGNORED.
```

SYNTAX

- WG{string}(ESC)** Defines the first character of string to be the general wildcard character.
- WG(ESC)** Causes the general wildcard character to be undefined.

EXPLANATION

The **WG** command defines the general wildcard character. The general wildcard character can be used in search and replacement strings of the **F** and **R** commands to match any character other than the end-of-line character.

The general wildcard character is initially undefined. The **WG** command defines it to be the first character in the specified string. If more than one character is included in the string, only the first character is used. The following message is displayed on the message line:

GENERAL WILDCARD CHARACTER IS <char>.

If you want the general wildcard character to be undefined, enter the **WG** command followed by an (ESC). The following message is then displayed on the message line:

GENERAL WILDCARD CHARACTER IS NOW UNDEFINED.

If you choose a wildcard character that is a non-displayable character or has already been defined by the **WI** or **WE** commands, the following message is displayed on the message line:

INVALID WILDCARD CHARACTER, IGNORED.

EXAMPLE

If you defined the "*" as the general wildcard character, the command **FABC*EFG(ESC)** would find the first occurrence of a string of seven characters, where the first three must be "ABC", the last three must be "EFG", and the middle character may be any character (except end-of-line).

SYNTAX

- WI{string}(ESC)** Defines the case ignore delimiter to be the first character of the specified string.
- WI(ESC)** Causes the case ignore wildcard character to be undefined.

EXPLANATION

The **WI** command defines the case-ignore wildcard character. The case-ignore wildcard character can be used in search strings of the **F** and **R** commands.

The case ignore wildcard character is initially undefined. The **WI** command defines it to be the first character in the specified string. If more than one character is included in the string, only the first character is used. The following message is displayed on the message line:

CASE IGNORE WILDCARD CHARACTER IS <char>

If you want the case-ignore wildcard character to be undefined, enter the **WI** command followed by an (ESC). The following message is then displayed on the message line:

CASE IGNORE WILDCARD CHARACTER IS NOW UNDEFINED.

If you choose a wildcard character that is a non-displayable character or has already been defined by the **WE** or **WG** commands, the following message is displayed on the message line:

INVALID WILDCARD CHARACTER, IGNORED.

EXAMPLE

If you defined the "%" to be the case-ignore wildcard character, the command **F%ABC%(ESC)** would find the first occurrence of any one of the following strings: "abc", "abC", "aBc", "aBC", "Abc", "AbC", "ABc", or ABC.

SYNTAX

XL	Examines the character at the cursor and converts it to lowercase if it is uppercase.
-XL	Examines the character preceding the cursor and converts it to lowercase if it is uppercase.
[count]XL	Examines the specified number of characters starting at the cursor and converts them to lowercase if they are uppercase.
-[count]XL	Examines the specified number of characters preceding the cursor and converts them to lowercase if they are uppercase.
/XL	Examines all characters from the cursor to the end of the line and converts them to lowercase if they are uppercase.
-/XL	Examines all characters between the cursor and the beginning of the line and converts them to lowercase if they are uppercase.
>XL	Examines characters starting at and following the cursor and converts them to lowercase if they are uppercase. The number of characters is determined by the length of the last matched string found with the F or R command.
<XL	Examines characters starting at and preceding the cursor and converts them to lowercase if they are uppercase. The number of characters is determined by the length of the last matched string found with the F or R command.

EXPLANATION

The **XL** command converts uppercase letters with equivalent lowercase letters. Lowercase letters, digits, and special characters are not changed. The cursor is not affected by the **XL** command.

EXAMPLE

```

ABCDEF GHIJKLMNOPQRSTUVWXYZ0123456789!@#%&*( )+ ~
BCDEF GHIJKLMNOPQRSTUVWXYZ0123456789!@#%&*( )+ ~
CDEF GHIJKLMNOPQRSTUVWXYZ0123456789!@#%&*( )+ ~
DEF GHIJKLMNOPQRSTUVWXYZ0123456789!@#%&*( )+ ~
EF GHIJKLMNOPQRSTUVWXYZ0123456789!@#%&*( )+ ~
-----
^O01 00021 ENTER COUNT/COMMAND      COUNT:  COMMAND:

```

To change all uppercase characters on the current line to lowercase, enter:

/XL

The case is changed, and the following window is displayed:

```

ABCDEF GHIJKLMNOPQRSTUVWXYZ0123456789!@#%&*( )+ ~
bcdefghijklmnopqrstuvwxyz0123456789!@#%&*( )+ ~
CDEF GHIJKLMNOPQRSTUVWXYZ0123456789!@#%&*( )+ ~
DEF GHIJKLMNOPQRSTUVWXYZ0123456789!@#%&*( )+ ~
EF GHIJKLMNOPQRSTUVWXYZ0123456789!@#%&*( )+ ~
-----
^O01 00021 ENTER COUNT/COMMAND      COUNT:/  COMMAND:XL

```


SYNTAX

XU	Examines the character at the cursor and converts it to uppercase if it is lowercase.
-XU	Examines the character preceding the cursor and converts it to uppercase if it is lowercase.
[count]XU	Examines the specified number of characters starting at the cursor and converts them to uppercase if they are lowercase.
-[count]XU	Examines the specified number of characters preceding the cursor and converts them to uppercase if they are lowercase.
/XU	Examines all characters from the cursor to the end of the line and converts of them to uppercase if they are lowercase.
-/XU	Examines all characters between the cursor and the beginning of the line and converts them to uppercase if they are lowercase.
>XU	Examines characters starting at and following the cursor and converts them to uppercase if they are lowercase. The number of characters is determined by the length of the last matched string found with the F or R command.
<XU	Examines characters starting at and preceding the cursor and converts them to uppercase if they are lowercase. The number of characters is determined by the length of the last matched string found with the F or R command.

EXPLANATION

The **XU** command exchanges lowercase letters with equivalent uppercase letters. Uppercase letters, digits, and special characters are not changed. The cursor is not affected by the **XU** command.

EXAMPLE

not completed and the contents of P upon reset may not~
necessarily be the address of the instruction that~
would have been executed next. It may, for instance,~
point to an immediate_operand if the reset occurred~
during the second cycle of a~

^001 00025 ENTER COUNT/COMMAND COUNT: COMMAND:F
immediate

The F command has been used to find the string "immediate", which is shown preceding the cursor. To change the string to uppercase letters, enter:

<XU

The following window is displayed:

not completed and the contents of P upon reset may not~
necessarily be the address of the instruction that~
would have been executed next. It may, for instance,~
point to an IMMEDIATE_operand if the reset occurred~
during the second cycle of a~

^001 00025 ENTER COUNT/COMMAND COUNT:< COMMAND:XU

Section 4

TABLES

Table 4-1
Configurable ACE Commands

Command	Explanation
(cd)	Cursor Down
(ch)	Cursor Home
(cl)	Cursor Left
(cmd-esc)	CoMmanD-ESCAPE
(cr)	Cursor Right
(cu)	Cursor Up
(dc)	Deletes Character
(dl)	Delete Line
(ic)	Insert Character
(il)	Insert Line
(mark)	MARK cursor position
(pd)	Page Down
(pu)	Page Up
(rm)	Revise Mode
(sd)	Scroll Down
(sl)	Scroll Left
(sr)	Scroll Right
(su)	Scroll Up

**Table 4-2
Non-configurable ACE Commands**

Command	Explanation
A	Advance lines
C	Change characters
EC	Echo Commands
EP	display Editor Profile
ER	Edit, Read secondary file
EW	Edit, Write secondary file
EX	EXit editor, update file
F	Find characters
H	Help
J	Jump characters
MD	Macro, Define/Delete
ML	Macro, List
MX	Macro, eXecute
O	Open secondary file
Q	Query
R	Replace characters
S	Stop: exit, no file update
T	Tab stop definition
U	Undelete characters
V	View non-displayable characters
WE	Wildcard, Escape definition
WG	Wildcard, General definition
WI	Wildcard, case-Ignore definition
XL	eXchange with Lowercase
XU	eXchange with Uppercase

Table 4-3
ACE Configurable Command Syntax

Optional Count	Command	Parameters
	(cu)	
	(cd)	
	(cl)	
	(cr)	
	(ch)	
	(sd)	
	(su)	
	(sr)	
	(sl)	
	(pd)	
	(pu)	
> < / -/ - -n n	(dc)	
(mark) (motion)	(dc)	
/ -/ - -n n	(dl)	
	(ic)	{string} (ESC)
	(il)	{string} (ESC)
-	(rm)	[{string} ... (rm) (cmd-esc) (cmd) ^a]
	(cmd-esc)	
	(mark)	

^a (cmd) represents a non-configurable command

Table 4-4
ACE Non-Configurable Command Syntax

Optional Count							Command	Parameters	
0			/	-/	n	-n	-	A	
0	>	<	/	-/	n	-n	-	C	{string} (ESC)
								EC	
								EP	
					n			ER	
			/	-/	n	-n	-	EW	
			(mark)		(motion)			EW	
								EX	
			/		n			F	{string} (ESC)
								H	
	>	<	/	-/	n	-n	-	J	
								MD	{name} (ESC) {string} (ESC)
								ML	
					n			MX	{name}
								O	{string} (ESC)
								Q	
			/		n			R	{string} (ESC) {string} (ESC)
								S	
								T	{string} (ESC)
								U	
			/	-/	n	-n	-	V	
								WE	{string} (ESC)
								WG	{string} (ESC)
								WI	{string} (ESC)
	>	<	/	-/	n	-n	-	XL	
	>	<	/	-/	n	-n	-	XU	

Table 4-5
How Non-Displayable Characters Are Represented by ACE

Hex Value	Name	Shown As	Hex Value	Name	Shown As
00	NUL	^@	10	DLE	^P
01	SOH	^A	11	DC1	^Q
02	STX	^B	12	DC2	^R
03	ETX	^C	13	DC3	^S
04	EOT	^D	14	DC4	^T
05	ENQ	^E	15	NAK	^U
06	ACK	^F	16	SYN	^V
07	BEL	^G	17	ETB	^W
08	BS	^H	18	CAN	^X
09	HT	^I	19	EM	^Y
0A	LF	^J	1A	SUB	^Z
0B	VT	^K	1B	ESC	^[
0C	FF	^L	1C	FS	^\
0D	CR	^M	1D	GS	^]
0E	SO	^N	E1	RS	^^
0F	SI	^O	1F	US	^_
			7F	DEL	^?

Table 4-6
ASCII Codes (Hexadecimal)

		HIGH-ORDER BITS							
		0	1	2	3	4	5	6	7
LOW-ORDER BITS		CONTROL		SYMBOLS		UPPERCASE		LOWERCASE	
		0	NUL	DLE	SP	ø	@	P	`
1	SOH	DC1	!	1	A	Q	a	q	
2	STX	DC2	"	2	B	R	b	r	
3	ETX	DC3	#	3	C	S	c	s	
4	EOT	DC4	\$	4	D	T	d	t	
5	ENQ	NAK	%	5	E	U	e	u	
6	ACK	SYN	&	6	F	V	f	v	
7	BEL <small>BELL</small>	ETB	'	7	G	W	g	w	
8	BS <small>BACKSPACE</small>	CAN	(8	H	X	h	x	
9	HT	EM)	9	I	Y	i	y	
A	LF	SUB	*	:	J	Z	j	z	
B	VT	ESC	+	;	K	[k	{	
C	FF	FS	,	<	L	\	l	:	
D	CR <small>RETURN</small>	GS	-	=	M]	m	}	
E	SO	RS	.	>	N	^	n	~	
F	SI	US	/	?	O	_	o	DEL <small>RUBOUT</small>	

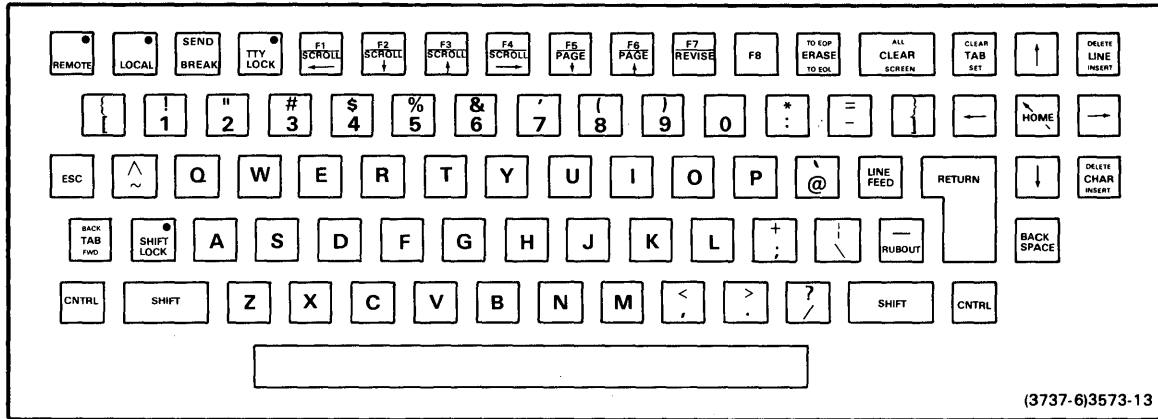


Fig. 4-1. TEKTRONIX CT8500 Terminal Keyboard Layout.

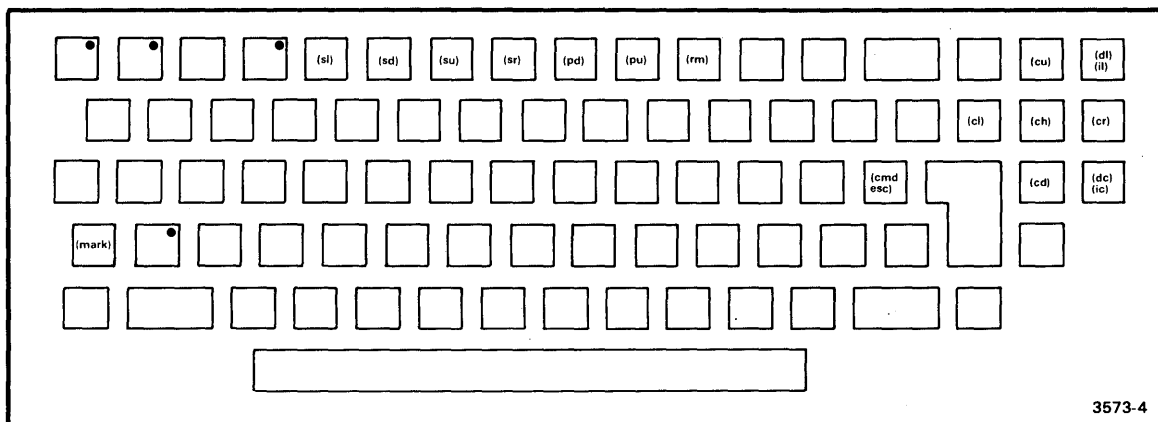


Fig. 4-2. Configurable Command Locations on the CT8500 Keyboard.



Section 5 TECHNICAL NOTES

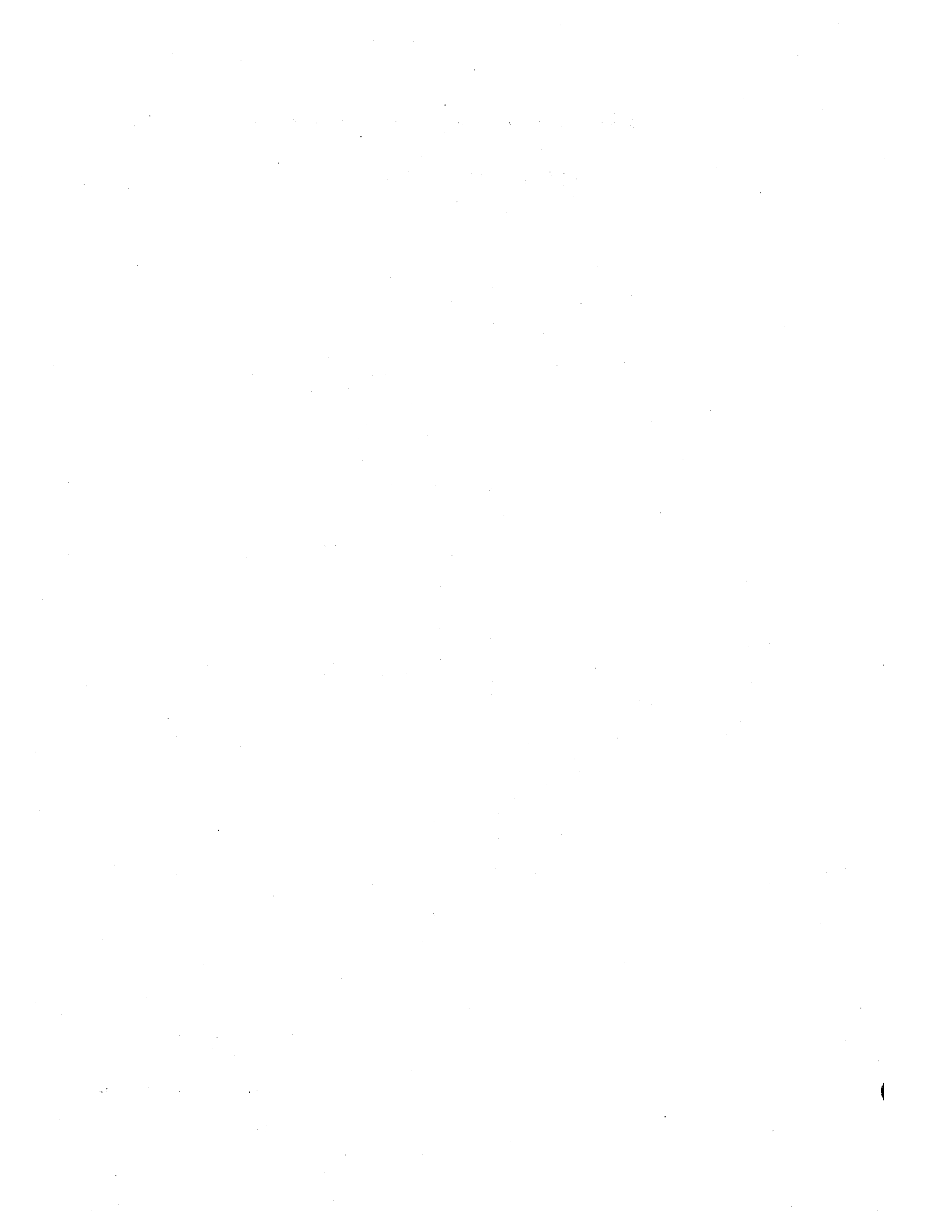
NOTE 1. THE ACE CONFIGURATOR	Page
Introduction	5-1
How This Note Is Organized	5-1
Minimum Requirements	5-1
 Configuration Overview	 5-2
Preliminary Considerations	5-4
Remote and Local Modes	5-4
Terminal Name	5-4
Key Codes for Configurable Commands	5-4
Valid Key Codes	5-5
Revise Mode Considerations	5-7
Terminal Command Sequences	5-7
Available Terminal Commands	5-8
Constructing Terminal Command Sequences	5-8
Command Sequence Prefix and	
Termination Characters	5-9
Terminal Initialization and Termination	5-10
Command Sequence Functions	5-11
Terminal Characteristics	5-13
 Using The Configurator	 5-14
Invoking the Configurator	5-14
Viewing a Configuration File	5-15
Obtaining a Printout of a Configuration File	5-17
Creating a Configuration File	5-17
Configurator Input	5-18
Demonstration of Configurator input	5-20
Modifying a File	5-21
ACE and Configuration Files	5-21
The Default File	5-22
Other Configuration Files	5-22
Changing the Contents of the Default File	5-22
 Error Messages	 5-23

TABLES

Table No.		Page
5-1	Configurable Parameters	5-3
5-2	Configurable Commands	5-5
5-3	Valid Configurable Command Key	
	Code Prefixes	5-6
5-4	Invalid Configurable Command Key	
	Code Prefixes	5-6
5-5	Key Codes for Revise Mode Functions ...	5-7
5-6	CT8500 Terminal Remote Commands	
	(in hexadecimal)	5-8
5-7	CT8500 Configuration File Display	5-16

ILLUSTRATIONS

Fig. No.		
5-1	Configuration Relationships	5-2
5-2	CT8500 Terminal initialization	
	command sequence	5-10
5-3	CT8500 Terminal termination	
	command sequence	5-11



Section 5

TECHNICAL NOTES

NOTE 1. THE ACE CONFIGURATOR.

INTRODUCTION

The ACE Configurator is used to create, modify, and view the interface between ACE and each terminal that you want to use with ACE. Each interface is called a **configuration file**, and contains the values of **configurable parameters** for a single terminal. The ACE Configurator and the configuration file for the TEKTRONIX CT8500 are included on the ACE installation disc. Both are installed along with ACE as part of the normal ACE installation procedure. The Learning Guide section of this manual describes that installation procedure.

How This Note Is Organized

Before you use the Configurator to create a configuration file for a new terminal, you need to understand the configuration process. This technical note describes the actual mechanics of using the Configurator, and then gives you this basic understanding. Consequently, this technical note is organized as follows:

- **Overview.** A discussion giving you a "big picture" of the configuration process.
- **Preliminary Considerations.** How to select the values for configurable parameters.
- **Using the Configurator.** How to invoke the Configurator and how to view, create, and modify configuration files.
- **Configuration Files.** How to invoke ACE with non-default configuration files, and how to change the contents of your default configuration file.
- **Error Messages.** A description of each error message.

Minimum Requirements

If you want to create a configuration file for a terminal other than the TEKTRONIX CT8500, then your terminal must satisfy the following minimum requirements:

1. It must have an ASCII keyboard.
2. It must not transmit byte values that use the high-order bit to encode information. ACE assumes 7-bit ASCII key code values.
3. It must be able to erase the terminal screen. *with a code*
4. It must be able to destructively write on the terminal screen.
5. It must be able to position the cursor using X-Y coordinates.
6. It must have a screen display at least 80 columns wide.

CONFIGURATION OVERVIEW

You use the Configurator to create a unique interface between ACE and your terminal. This interface is a **configuration file**. When ACE is invoked, a configuration file is read into **configuration tables** contained within ACE. These tables remain resident in memory throughout your editing session, and are accessed by ACE when interpreting key codes from the keyboard, and when returning subsequent commands to your terminal. Figure 5-1 shows the relationships between the Configurator, ACE, and your terminal.

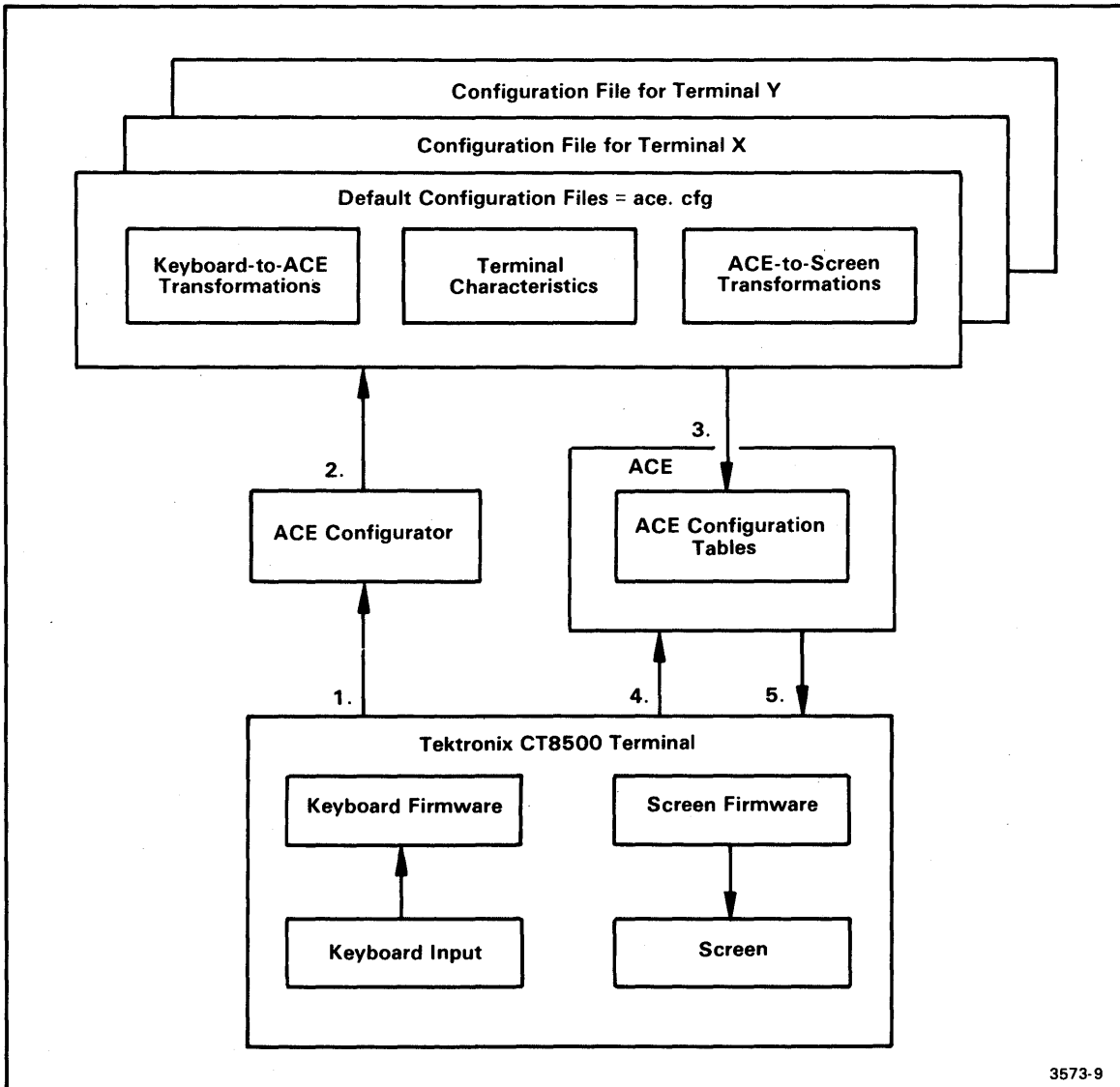


Fig. 5-1. Configuration relationships.

(1) Your input causes the Configurator (2) to create or modify a configuration file. When ACE is invoked, (3) a configuration file is read into ACE's configuration tables. The default file "ace.cfg" is the configuration file for the TEKTRONIX CT8500 terminal in the case above. ACE configuration tables are accessed whenever (4) ACE interprets input from the keyboard and (5) whenever ACE sends sequences of terminal commands to the terminal.

Once you create a configuration file, that file should completely specify the configurable interface to a terminal.

So what does a configuration file contain? Essentially, it contains the values for the configurable parameters listed in Table 5-1. This technical note examines these parameters in the order in which you must input them to the Configurator. Parameters are grouped into four classes:

1. Terminal name
2. Key codes for configurable commands
3. Terminal command sequences
4. Terminal characteristics

Note that many parameters are optional. Each optional parameter is preceded by an asterisk in Table 5-1.

**Table 5-1
Configurable Parameters**

Terminal name	*Insert character command sequence
Cursor up function key code	*Insert character mode termination sequence
Cursor down function key code	*Delete character command sequence
Cursor right function key code	*Insert line command sequence
Cursor left function key code	*Delete line command sequence
Scroll up function key code	*Turn cursor on command sequence
Scroll down function key code	*Turn cursor off command sequence
Scroll right function key code	*Turn blink on command sequence
Scroll left function key code	*Turn blink off command sequence
Page up function key code	*Turn inverse video on command sequence
Page down function key code	*Turn inverse video off command sequence
Revise mode function key code	Cursor positioning command sequence
Insert character function key code	Zero x-coordinate value
Insert line function key code	Zero y-coordinate value
Delete character function key code	At-sign (@) block delimiter replacement
Delete line function key code	End of line replacement character
Command escape function key code	Number of usable lines on screen
Mark cursor motion function key code	Number of physical columns on line
*Command sequence prefix character	*Terminal initialization command sequence
*Command sequence terminator character	*Terminal termination command sequence
Erase screen command sequence	Line insertion at cursor
*Erase from cursor to end of line command sequence	Y coordinate before X coordinate
*Erase from cursor to end of screen command sequence	Cursor coordinate in displayable decimal
	Cursor coordinate increment positive

PRELIMINARY CONSIDERATIONS

Before using the Configurator, you need to select values for the configurable parameters that you intend to input. Remember that the configuration file for the CT8500 is already available, and that you do not need to create a configuration file for it. There are four steps in the parameter selection process:

1. identifying your terminal,
2. determining legal key codes for configurable ACE commands,
3. obtaining a specification of terminal commands, and
4. gathering information about a terminal's characteristics.

Before taking these steps, you should understand the difference between a terminal's local and remote modes. The following paragraphs discuss these two modes.

Remote and Local Modes

When a terminal operates in **local** mode, the keys that are pressed affect the terminal screen directly: the logic of the terminal dictates changes to the terminal screen. For example, pressing the **Q** key causes the letter **Q** to be displayed on the terminal screen: nothing is transmitted to a connected computer.

When a terminal operates in **remote** mode, it is controlled by a program such as ACE running on a host computer. In this case, values are sent to the host computer, where the program then determines which codes are sent to the terminal. Thus, the logic of the controlling program dictates any changes to the terminal screen. For example, assume that you press the key configured to tell ACE to home the cursor. ACE interprets this key code and sends an appropriate command to the terminal to home the cursor. The key you press only indirectly affects your terminal screen.

ACE operates in remote mode, so keep the distinction between local and remote modes in mind as you select parameter values.

Terminal Name

The terminal name parameter is used only to identify the terminal and can be named anything that you like. The terminal name parameter is **not** the same as the name of the configuration file.

Key codes for configurable commands

Each key on the keyboard has a value or values that are transmitted when that key is typed. Most keys also have values for the shifted and control interpretations of each key. One or more of these transmitted ASCII values is called a key code.

One part of the configuration process is to select key codes for ACE commands. ACE accepts two types of commands: configurable and non-configurable. Marked keys will usually exist on a keyboard for configurable commands. ACE allows you to configure these keys to the appropriate ACE commands. When such a key is typed, it tells the editor to execute the command for which the key is configured.

For instance, if you have a HOME key on your terminal, you can use it to execute the ACE command to home the cursor. Remember, though, the HOME key is disabled and cannot be used for any function **unless you configure it**. Table 5-2 lists the configurable ACE commands. You should select key codes for these commands that make using ACE as easy as possible.

Table 5-2
Configurable Commands

Key Code Parameter	ACE Command
Cursor up	(cu)
Cursor down	(cd)
Cursor left	(cl)
Cursor right	(cr)
Cursor home	(ch)
Scroll down	(sd)
Scroll up	(su)
Scroll right	(sr)
Scroll left	(sl)
Page down	(pd)
Page up	(pu)
Delete character	(dc)
Delete line	(dl) "d"
Insert character	(ic) "i"
Insert line	(il)
Revise mode	(rm)
Command-escape ^a	(cmd-esc)
Mark cursor ^a	(mark)

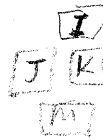
^a These key code parameters must be exactly one byte long.

Valid Key Codes

Several types of keys are recommended as ones you should configure to the ACE configurable commands:

1. Programmable function keys
2. Pre-labeled keys corresponding to terminal functions
3. Fixed-code function keys

If your terminal has no programmable function keys, the "i" and "d" keys are suggested for the insert character (ic) and delete character (dc) commands.



Likewise, if your terminal has no cursor keypad, you might consider four adjacent keys on your keyboard for the cursor commands: for example, (ESC)I, (ESC)J, (ESC)K, and (ESC)M. These key codes are for escaped characters so that they do not conflict with other ACE commands.

Note that there are restrictions on the values you can choose for key code parameters: only the ASCII values for keys in Table 5-3 can be used to **begin** the key codes of configurable key code parameters.

Table 5-3
Valid Configurable Command Key Code Prefixes

Category	ASCII Prefixes
Uppercase	B D G I K L N P Y Z
Lowercase	b d g i k l n p y z
Punctuation	(space) ! " # \$ % & ' () * + , . : ; = ? @ [\] ^ _ ` { } ~
Control Characters	All CNTRL-characters except: CNTRL-H, CNTRL-S, CNTRL-Q, and CNTRL-X

Conversely, the values for the keys in Table 5-4 **cannot** begin a key code parameter. Essentially, this means that you cannot use the key-codes that begin **non-configurable** commands, that are used for command editing such as backspace and CNTRL-X, or that are used in repetition counts (0-9).

Table 5-4
Invalid Configurable Command Key Code Prefixes

Category	ASCII Prefixes
Uppercase	A C E F H J M O Q R S T U V W X
Lowercase	a c e f h j m o q r s t u v w x
Control	CNTRL-H, CNTRL-Q, CNTRL-S, CNTRL-X
Digits	0 1 2 3 4 5 6 7 8 9
Other	/ < > - (RUBOUT)

Remember that conflicts between key code parameters are not allowed: a key code cannot be identical to, or prefix the key code of, another command. This means that although **B** and **BB** are both valid key codes for configurable commands, you can use only **one** in any given configuration file. Note that **BA** and **BB** can both exist as key codes in the same configuration file, since neither key code is the prefix of the other.

Similarly, the key codes (ESC)A and (ESC)B can both exist in the same configuration file, so long as escape (ESC) **alone** is not configured in that file. Note that the escape key (ESC) is often used

to prefix the key codes of configurable commands. So, if you use escape to prefix multiple key codes, be sure **not** to configure it to a configurable command by itself.

Revise Mode Considerations

When you use ACE in revise mode, the key codes in Table 5-5 have functions that override any configured meaning that you may give them.

**Table 5-5
Key Codes for Revise Mode Functions**

Key Code	Revise Mode Function	Usable Prefix
CNTRL-H	(backspace)	no
CNTRL-I	(tab)	yes
CNTRL-M	(return)	yes
CNTRL-Q	(resume output)	no
CNTRL-S	(stop output)	no
CNTRL-X	(cancel)	no
RUBOUT	(backspace)	no

Of the keys listed in Table 5-5, only CNTRL-M (return) and CNTRL-I (tab) are "usable" prefixes for key codes of configurable commands. A configurable command prefixed by either of these key codes can be executed directly when you are **not** in revise mode. However, to execute such a command **within** revise mode, you must precede it by typing a command-escape (cmd-esc). This implies that the key code for command-escape (cmd-esc) **cannot** have CNTRL-M or CNTRL-I as the first character in its key code.

There is another factor to consider in revise mode: namely, a configurable command can be executed in revise mode without a preceding command-escape if it has a **non-displayable** ASCII character as the first character of its key code. Note that the escape key (ESC) is identical to CNTRL-[, and is therefore a non-displayable ASCII character: it is a particularly good prefix for key codes that you want to execute **directly** in revise mode.

In the preceding discussion, remember that a key you use to **prefix** a key code can also be the **only** key contained in that key code.

Terminal Command Sequences

Terminal command sequences are sent by ACE to effect changes in your terminal display. A terminal command sequence consists of one or more terminal commands designed to carry out an ACE editing function.

To correctly specify terminal command sequences, you need to know the following:

1. The commands available for a terminal.
2. How ACE constructs commands before sending them to the terminal.
3. The ACE functions that terminal command sequences are intended to perform.

Available Terminal Commands

Before constructing terminal command sequences, you must first have a list of the available terminal commands. As an example, Table 5-6 lists the terminal commands available for the TEKTRONIX CT8500. Note that the CT8500 has a variety of terminal commands that are suited to ACE's editing capabilities.

Table 5-6
CT8500 Terminal Remote Commands (in hexadecimal)

Function	Code Sequence	Function	Code Sequence
Transmit Line	1B,01	Stop Attribute	1B,20
Transmit Page	1B,02	Enter Monitor Mode	1B,3B
Clear Screen	1B,04	Exit Monitor Mode	1F
Clear All	1B,05	Enter Host Control Mode	1B,3D
Tab Forward	09	Enter Power-Up Mode	1B,5D
Tab Back	1B,09	Enter Local Mode	1B,5F
Set Column Tab	1B,61	Learn Mode Status	1B,38
Clear Column Tab	1B,62	Enter Learn Mode	1B,39
Insert Line	1B,0C	Exit Learn Mode	1F
Delete Line	1B,0D	Split Screen	1B,3E
Erase to End of Line	1B,14	Screen Alignment Test	1B,3F
Erase to End of Page	1B,15	Load Cursor Address	1B,7C
Insert Character	1B,1C	Read Cursor Address	1B,7D
Delete Character	1B,1D	Page Down	1B,44
Begin Underline Attribute	1B,21	Page Up	1B,45
Begin Blink Attribute	1B,22	Page One	1B,59
Begin Blink and Underline	1B,23	Page Two	1B,79
Begin Inverse Attribute	1B,24	Cursor Down	1B,52
Begin Reduce Attribute	1B,28	Cursor Right	1B,53
Begin Reduce and Underline	1B,29	Cursor Home	1B,54
Begin Blink and Reduce	1B,2A	Cursor On	1B,5A
Begin Blink, Reduce, and Underline Attribute	1B,2B	Cursor Off	1B,5B
Begin Inverse and Reduce	1B,2C	Scroll Up	1B,42
Begin Blink, Reduce, and Inverse Attribute	1B,2E	Scroll Down	1B,41
		Keyboard Unlock	1B,6B
		Keyboard Lock	1B,6A

Constructing Terminal Command Sequences

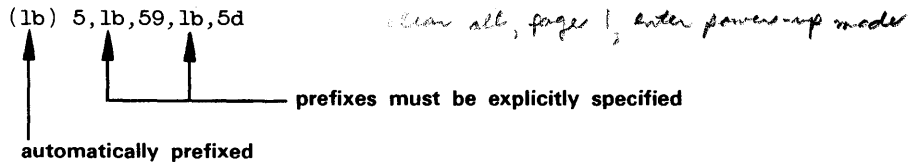
A terminal command sequence consists of one or more terminal commands. You construct each sequence to perform the function described by the appropriate terminal command sequence parameter. In many cases, a single terminal command can perform the function you desire. In other cases, however, a **sequence** of terminal commands must be used.

In this discussion, we will look first at terminal command prefix and termination characters. Then, we will examine the construction of the terminal initialization and terminal termination command sequences, using the CT8500 terminal as an example.

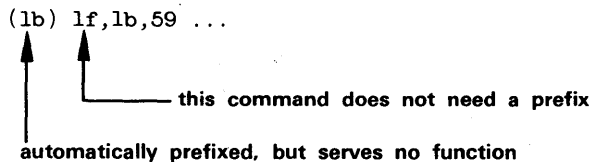
Command Sequence Prefix and Termination Characters. The command prefix and termination characters are used by a terminal to differentiate between ordinary characters and terminal commands. When a string is enclosed within command prefix and termination characters, the terminal is able to recognize the string as a terminal command. In some cases these characters are programmable, and in others they are hard-wired. For the TEKTRONIX CT8500, the prefix character is a non-programmable escape (ESC) character; the termination character is not needed and is omitted.

ACE automatically prefixes terminal command sequences with the prefix character, and automatically terminates terminal command sequences with the termination character. Since both these characters are optional, ACE uses each one only if you **do not** omit it during the configuration process. If more than one terminal command is contained within a terminal command sequence, you must explicitly specify prefix and termination characters **between** terminal commands.

For example, in the following terminal termination command sequence, the command prefix character (shown in parentheses) is implicitly prefixed to the first command of the sequence, but prefix characters **inside** the string are specified explicitly.



In some cases, a terminal command exists that need not be prefixed by the command prefix character. For example, the CT8500 command to exit monitor mode is 1F and does not require a preceding prefix character. Still, the command sequence character is automatically prefixed by ACE, although it serves no function. Some of the values actually sent by ACE to initialize the CT8500 follow:



The command prefix character may vary from command to command. If so, you should omit the command prefix character parameter, and enter it explicitly before each command in a command sequence.

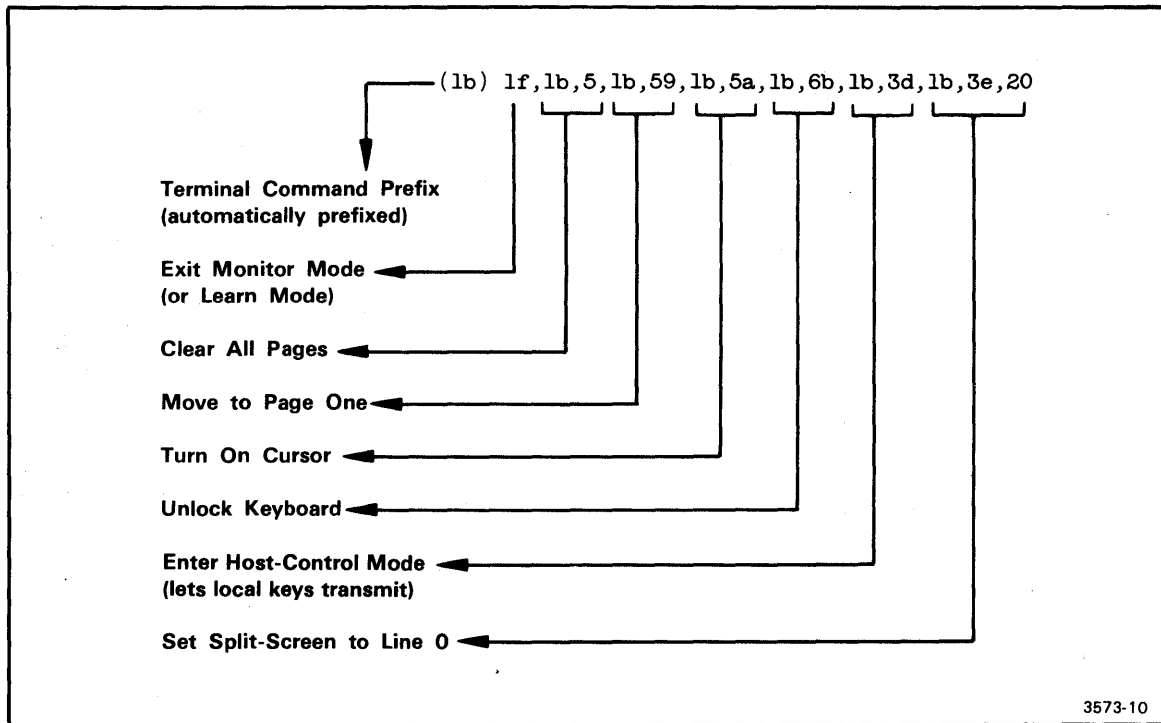


Fig. 5-2. CT8500 Terminal initialization command sequence.

Terminal Initialization and Termination. At the start of your editing session, ACE executes a terminal initialization command, and at the end of your editing session ACE executes a terminal termination command. These commands are executed by sending terminal commands to your terminal to prepare your terminal for entry into ACE and for exit to the system. Figures 5-2 and 5-3 show the kinds of considerations that should be made for the terminal initialization and termination command sequences, using the TEKTRONIX CT8500 as an example.

The terminal initialization command sequence in Fig. 5-2 performs the following functions:

1. Exits the terminal from monitor mode or from learn mode. Also disables the display of control-characters.
2. Clears all screen buffers.
3. Ensures that the correct text buffer (Page One) is being referenced.
4. Turns on the display of the cursor.
5. Unlocks the keyboard so that keyboard input is recognized.
6. Places the terminal in host-control mode so that the terminal can communicate with ACE.
7. Disables the CT8500 split screen option.

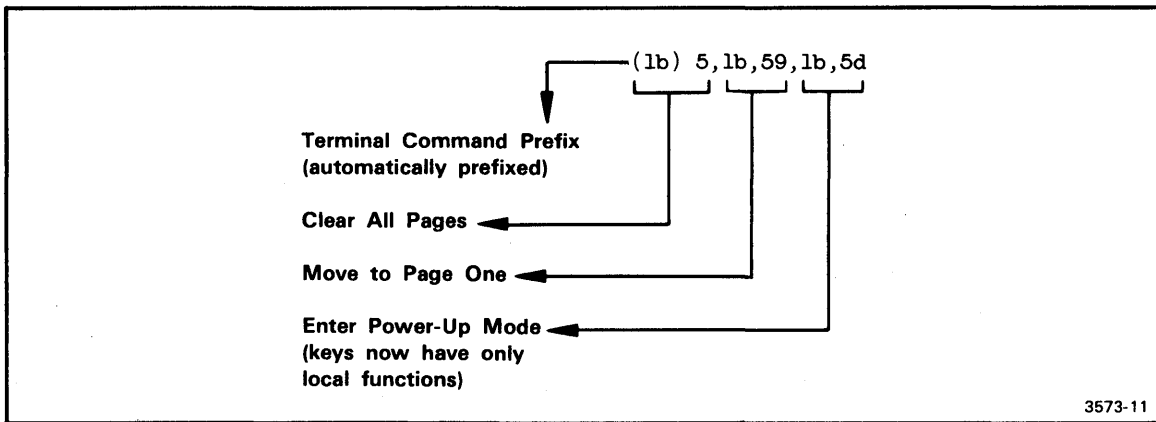


Fig. 5-3. CT8500 Terminal termination command sequence.

The command sequence in Fig. 5-3 performs the following functions:

1. Clears all screen buffers.
2. Moves you to the first screen buffer (Page One).
3. Puts the terminal in Power-Up Mode, thus re-enabling local key functions.

Note that the command sequences shown in Figs. 5-2 and 5-3 take care of a number of exceptional conditions that may be encountered when invoking ACE. When you create command sequences for another terminal, you will want to take care of similar conditions.

Command Sequence Functions

The preceding discussion described **how** to construct a terminal command sequence. The following list tells **what function** each command sequence is supposed to perform. The first two parameters listed, specify the command sequence prefix character and the command sequence termination character. Both of these parameters are described in more detail earlier in this technical note.

Unless otherwise noted, all of the following command sequences are optional parameters.

Command Sequence Prefix Character. If specified, this character is implicitly prefixed to the start of every terminal command sequence in your configuration file.

Command Sequence Terminator Character. If specified, this character is implicitly appended to the end of every terminal command sequence in your configuration file.

Erase Screen Command Sequence. (Required Parameter). Erases the screen.

Erase from Cursor to End of Line Command Sequence. Erases text from the cursor to the end of the line.

Erase from Cursor to End of Screen Command Sequence. Erases text from the cursor to the end of the screen.

Insert Character Command Sequence. Causes entry into terminal insert mode. Insertion occurs at the cursor. Character insertion may cause the number of characters in a line to exceed the physical length of the line on your screen. If it does, ACE assumes that your terminal will **not** wrap the line around to the next line. If your terminal **does** wrap around long lines, ACE will still work properly if you **omit** this parameter.

Insert Character Mode Termination Sequence. Causes exit from terminal insert mode.

Delete Character Command Sequence. Deletes the character at the cursor.

Insert Line Command Sequence. Creates a blank line either at or below the line on which your cursor resides. Where the line is inserted depends on your terminal, and is specified by the Boolean parameter "line insertion at cursor".

Delete Line Command Sequence. Deletes the line containing the cursor from your displayed text.

Turn Cursor On Command Sequence. Turns on display of the cursor.

Turn Cursor Off Command Sequence. Turns off display of the cursor.

Turn Blink On Command Sequence. Causes characters to the right of the cursor to blink.

Turn Blink Off Command Sequence. Turns Off the blinking of characters to the right of the cursor.

Turn Inverse Video On Command Sequence. Causes characters to the right of the cursor to be displayed in inverse video.

Turn Inverse Video Off Command Sequence. Turns Off inverse video display of characters to the right of the cursor.

Cursor Positioning Command Sequence. (Required Parameter). Causes absolute positioning of the cursor. This command sequence is used with X and Y cursor coordinates to specify the column and row of the cursor on the screen. For some terminals, the Y coordinate may be specified before the X coordinate. You use the Boolean "Y Coordinate Before X Coordinate" parameter to select which coordinate ordering is true for your terminal.

Terminal Initialization Command Sequence. Initializes the terminal for use by ACE. See the preceding discussion of Terminal Initialization and Termination Command Sequences for more details.

Terminal Termination Command Sequence. Deconfigures the terminal and returns it to a known state. See the preceding discussion of Terminal Initialization and Termination Command Sequences for more details.

Terminal Characteristics

After you've selected key code and terminal command sequence parameters, the final step in selecting parameter values is to obtain information about a terminal's characteristics. Information can usually be found in the documentation that comes with your terminal. If these characteristics are not documented, you may be able to determine characteristics by experimenting with your terminal. The characteristics that you need to know pertain mainly to your terminal screen's implementation as an X-Y coordinate grid.

The following configuration parameters allow ACE to account for these characteristics. Most of these parameters are Boolean or numeric values. Only the terminal initialization and termination command sequences are commands to the terminal. Each of these two commands is implicitly executed only once per editing session, and is described more fully in the preceding discussion of Terminal Command Sequences.

Zero X-Coordinate Value. The binary value of the zero X-coordinate. Column numbers are presumed to be either monotonically increasing or decreasing from this value.

Zero Y-Coordinate Value. The binary value of the zero Y-coordinate. Row numbers are presumed to be either monotonically increasing or decreasing from this value.

At-Sign (@) Block Delimiter Replacement. The ASCII character assigned to replace the at-sign (@) as a block delimiter for the (dc) and EW commands.

End of Line Replacement Character. The displayable ASCII character used to signify the end of each line displayed on your screen. This character is the tilde (~) in the configuration file for the TEKTRONIX CT8500 terminal.

Number of Usable Lines on Screen. The number of lines on your screen. The minimum number is 5: this accounts for the four monitor lines at the bottom of the screen, and a minimum window of one line. ACE will not work if this value is less than 5.

Number of Physical Columns on Line. The number of physical columns in a line: for most terminals, 80 columns. Note that the maximum number allowed is 127. Terminals capable of supporting longer line lengths can be used, but only the first 127 columns can be used by ACE. ACE does not work correctly if the value you select is less than 80.

Terminal Initialization Command Sequence. See the preceding discussion of Terminal Command Sequences.

Terminal Termination Command Sequence. See the preceding discussion of Terminal Command Sequences.

Line Insertion at Cursor. A Boolean value referring to the "insert line" terminal command sequence. This value is true if a blank line is inserted **at** the line on which the cursor resides, and text scrolled down beginning with that line. This value is false if a blank line is inserted **below** the line on which the cursor resides, and the cursor is moved to that new line. A **Y** or **y** should be input when entering a true value for this parameter; an **N** or **n** should be input when entering a false value.

Y Coordinate Before X Coordinate. A Boolean value that is set to true if the Y coordinate is sent first when sending X and Y coordinates for cursor positioning. Enter a **Y** or **y** to indicate a true value for this parameter; enter an **N** or **n** to indicate a false value.

Cursor Coordinate in Displayable Decimal. A Boolean value that is set to true if cursor coordinates must be converted to displayable decimal form before being sent to the terminal. If this value is false, single byte values are sent instead. Enter a **Y** or **y** to indicate a true value for this parameter; enter an **N** or **n** to indicate a false value.

Cursor Coordinate Increment Positive. A Boolean value that is set to true if cursor coordinates are monotonically incremented by positive values and to false if cursor coordinates are monotonically incremented by negative values. A **Y** or **y** should be input when entering a true value for this parameter; an **N** or **n** should be input when entering a false value.

USING THE CONFIGURATOR

Once you have determined the values that you want for configurable parameters, you have completed the more difficult of the tasks associated with configuration for a terminal. Now, you are ready to invoke the Configurator to perform the following actions:

1. View a configuration file
2. Obtain a printout of a configuration file
3. Create a configuration file
4. Modify a configuration file

Invoking the Configurator

Before invoking the Configurator, it is important to note the following:

- The only acceptable user inputs are those that ACE prompts for. This applies to any prompt string you see when using the Configurator.
- Legal responses can be entered in either upper or lower case.

- All user responses must be terminated by entering a return.
- All user responses are underlined in the examples included here.
- Examples in this technical note are for the 8550. Differences may exist for other 8500-Series systems.

To invoke the Configurator, type:

```
> ACECONFIG
```

The Configurator prompts you with the following message:

```
SELECT FILE NAME-- Quit, Default, Ask (Q, D, A): D
```

If you type **Q**, you will exit the Configurator; if you type **A**, you will be prompted for the name of a configuration file. This file must be a file that you wish to create, or an already existing file that you wish to view or modify.

Viewing a Configuration File

We continue the configuration session started above, and show you how to view a configuration file. If this is your first time using the Configurator, the default file for the TEKTRONIX CT8500 is the only configuration file available for viewing. You select the file `/EOS/NO.EMULATOR/ace.cfg` by typing **D** as shown above. Your response produces a new prompt:

```
SELECT ACTION--  
Quit, Create file, View file, Modify file (Q, C, V, M): V
```

The response **V** allows you to view the configuration file.

A display for the default file is shown in Table 5-7. The display on your terminal screen will be double-spaced.

Table 5-7
CT8500 Configuration File Display

Terminal name IN Ascii IS CT8500
 Cursor up function key code IN Hex IS 1b,52
 Cursor down function key code IN Hex IS 1b,51
 Cursor right function key code IN Hex IS 1b,53
 Cursor left function key code IN Hex IS 1b,50
 Cursor home function key code IN Hex IS 1b,54
 Scroll up function key code IN Hex IS 1b,42
 Scroll down function key code IN Hex IS 1b,41
 Scroll right function key code IN Hex IS 1b,43
 Scroll left function key code IN Hex IS 1b,40
 Page up function key code IN Hex IS 1b,45
 Page down function key code IN Hex IS 1b,44
 Revise mode function key code IN Hex IS 1b,46
 Insert character function key code IN Hex IS 1b,1c
 Insert line function key code IN Hex IS 1b,c
 Delete character function key code IN Hex IS 1b,1d
 Delete line function key code IN Hex IS 1b,d
 Command escape function key code IN Hex IS a
 Mark cursor motion function key code IN Hex IS 9
 Command sequence prefix character IN Hex IS 1b
 Command sequence terminator character IS OMITTED
 Erase screen command sequence IN Hex IS 4
 Erase from cursor to end of line command sequence IN Hex IS 14
 Erase from cursor to end of screen command sequence IN Hex IS 15
 Insert character command sequence IN Hex IS 1c
 Insert character mode termination sequence IS OMITTED
 Delete character command sequence IN Hex IS 1d
 Insert line command sequence IN Hex IS c
 Delete line command sequence IN Hex IS d
 Turn cursor on command sequence IN Hex IS 5a
 Turn cursor off command sequence IN Hex IS 5b
 Turn blink on command sequence IN Hex IS 22
 Turn blink off command sequence IN Hex IS 20
 Turn inverse video on command sequence IN Hex IS 24
 Turn inverse video off command sequence IN Hex IS 20
 Cursor positioning command sequence IN Hex IS 7c
 Zero x-coordinate value IN Hex IS 20
 Zero y-coordinate value IN Hex IS 20
 At-sign (@) block delimiter replacement IN Ascii IS @
 End of line replacement character IN Ascii IS
 Number of usable lines on screen IN Decimal IS 25
 Number of physical columns on line IN Decimal IS 80
 Terminal initialization command sequence IN Hex IS

**Table 5-7 (cont.)
CT8500 Configuration File Display**

```

1f,1b,5,1b,59,1b,5a,1b,6b,1b,3d,1b,3e,20
Terminal termination command sequence IN Hex IS
5,1b,59,1b,5d
Line insertion at cursor IN boolean IS YES
Y coordinate before X coordinate IN boolean IS YES
Cursor coordinate in displayable decimal IN boolean IS NO
Cursor coordinate increment positive IN boolean IS YES
    
```

After the file is displayed on the screen, you are prompted to select an action. To exit the Configurator, you type **Q** to exit the select action process, followed by another **Q** to leave the Configurator. This process is shown below:

```

SELECT ACTION--
  Quit, Create file, View file, Modify file (Q, C, V, M): Q

SELECT FILE NAME-- Quit, Default, Ask (Q, D, A): Q

>
    
```

Obtaining a Printout of a Configuration File

The only way to obtain a human-readable printout of a configuration file is to capture the output of the view process. The following sequence of commands copies the output of the view process to the line printer and to the terminal:

```

> LOG LPT

> ACECONFIG

SELECT FILE NAME-- Quit, Default, Ask (Q, D, A): A
ENTER NAME OF CONFIGURATION FILE: new.cfg

SELECT ACTION--
  Quit, Create file, View file, Modify file (Q, C, V, M): V

SELECT ACTION--
  Quit, Create file, View file, Modify file (Q, C, V, M): Q

SELECT FILE NAME-- Quit, Default, Ask (Q, D, A): Q

> LOG CONO
    
```

Creating A Configuration File

Thus far, you've learned how to select configurable parameter values, how to invoke the Configurator, and how to view a configuration file. Now you are ready to create a new configuration file. You do this by first invoking the Configurator:

```

> ACECONFIG
    
```

Next, respond to the file name prompt with the name of a new configuration file:

```
SELECT FILE NAME-- Quit, Default, Ask (Q, D, A): A
ENTER NAME OF CONFIGURATION FILE: new.cfg
```

The file you have named "new.cfg" will **not** be written out until you have input responses for **all** 48 parameter prompts. If you exit the Configurator before responding to the prompts for all parameters, file creation is aborted and all of your input lost. So remember, you cannot view or write a partially created file: files are written only after all parameters have been specified. (Many parameters are optional, but you must explicitly omit them when prompted: you cannot ignore them altogether.) To begin creation of "new.cfg", enter **C** as your response to the following prompt:

```
SELECT ACTION--
Quit, Create file, View file, Modify file (Q, C, V, M): C
```

At this point, you begin to input configurable parameters. You must respond to the prompt for **every** parameter. Configurator input is described in the following paragraphs.

Configurator Input

The Configurator accepts only certain inputs for each parameter. Where the input mode for a parameter is not fixed, you must select it. Input options are specified on the prompt line. Legal input modes are given below:

ASCII	Any displayable ASCII character. Control characters must be input in decimal or hexadecimal format. Multiple characters are concatenated.
Decimal	Any decimal number from 0 to 127. Multiple decimal numbers must be separated by commas.
Hexadecimal	Any hexadecimal number from 0 to 7F. Alphabetic hexadecimal digits may be input in either upper or lower case. Multiple hexadecimal numbers must be separated by commas.
Boolean	A Y or y for true values; an N or n for false values.
Multi-Line	The optional terminal initialization and termination command sequences may require more than one line of input. After the input of each line, a prompt asks you if you are done (D) or if your wish to add more (M) lines to the parameter. Multi-line parameters are input in either ASCII, hexadecimal, or decimal format.
Optional	Optional parameters are those you may omit. You may omit a parameter only if the "Omitted" response is listed on the prompt line. If you type O the parameter is omitted.

After each parameter is input, you are prompted to type either **O**, **Q**, or **M**.

1. If you type **O** or **o**, the parameter you have input is "OK" and you are prompted for the input of the next parameter.

2. If you type **Q** or **q**, file creation is aborted.
3. If you type **M** or **m**, you are prompted to re-enter and modify the current parameter.

The Configurator **cannot** guarantee that a given configuration file will work for a given terminal. It is up to you to correctly identify the function key codes, terminal command sequences, and terminal characteristics that are appropriate for a particular terminal.

When the Configurator detects an input error, it prints an error message and then reprompts you for correct input. Error messages are described at the end of this technical note. After you have specified all parameters, the new configuration file is written out and the following message appears on the screen:

```
The configuration file has been written.
```

If you decide to quit the Configurator by typing **Q** during the file creation process, the Configurator displays the following message:

```
TERMINATING file creation
```

If this message appears, your configuration file is **not** written out.

Demonstration of Configurator Input

The preceding paragraphs discussed the types of Configurator input. The following demonstration shows how you actually input parameters.

Terminal name FORMAT is Ascii
ENTER Terminal name: terminalX

Terminal name IN Ascii IS terminalX
CONFIRM-- Quit, Ok, Modify (Q, O, M): M

Terminal name FORMAT is Ascii
ENTER Terminal name: terminalY

Terminal name IN Ascii IS terminalY
CONFIRM-- Quit, Ok, Modify (Q, O, M): Q

CHOOSE Page up function key code FORMAT--
Ascii, Decimal, Hex (A, D, H): H
ENTER Page up function key code: 1b,45

Page up function key code IN Hex IS 1b,45
CONFIRM-- Quit, Ok, Modify (Q, O, M): Q

CHOOSE Turn inverse video on command sequence FORMAT--
Ascii, Decimal, Hex, Omitted (A, D, H, O): Q

Turn inverse video on command sequence IN Hex IS OMITTED
CONFIRM-- Quit, Ok, Modify (Q, O, M): Q

CHOOSE Number of usable lines on screen FORMAT--
Ascii, Decimal, Hex (A, D, H): D
ENTER Number of usable lines on screen: 33

Number of usable lines on screen IN Decimal IS 33
CONFIRM-- Quit, Ok, Modify (Q, O, M): Q

CHOOSE Terminal initialization command sequence FORMAT--
Ascii, Decimal, Hex, Omitted (A, D, H, O): H
ENTER Terminal initialization command sequence: 1f,1b,5,1b,59,1b,6b
Done, More (D, M): M
ENTER Terminal initialization command sequence: 1b,3d,1b,3e,20
Done, More (D, M): D

Terminal initialization command sequence IN Hex IS
1f,1f,1b,5,59,1b,6b,1b,3d,1b,3e,20
CONFIRM-- Quit, Ok, Modify (Q, O, M): Q

CHOOSE Line insertion at cursor? (Yes or No) Y
ENTER Line insertion at cursor?: Y

Line insertion at cursor IN boolean IS YES
CONFIRM-- Quit, Ok, Modify (Q, O, M): Q

Modifying a file

Like creating a file, modifying a file requires that **all** parameters be confirmed or modified before the Configurator writes the file out. You cannot simply enter the Configurator, modify a single parameter, and then exit. If you do, the modified file is **not** written out. To begin modifying a file, enter the ACECONFIG command:

```
> ACECONFIG

SELECT FILE NAME-- Quit, Default, Ask (Q, D, A): A
ENTER NAME OF CONFIGURATION FILE: term.cfg

    [this example assumes that term.cfg is an already ]
    [ existing configuration file.                    ]

SELECT ACTION--
    Quit, Create file, View file, Modify file (Q, C, V, M): M
```

Now, in order to modify the file, you must indicate to the Configurator, for each prompt, whether the current parameter is correct or is to be modified. Here is an example of this procedure for the first two parameters in a file:

```
Terminal name IN Ascii IS newterminal
CONFIRM-- Quit, Ok, Modify (Q, O, M): Q

Cursor up function key code IN Ascii IS @^
CONFIRM-- Quit, Ok, Modify (Q, O, M): M

CHOOSE Cursor up function key code FORMAT--
    Ascii, Decimal, Hex (A, D, H): H
ENTER Cursor up function key code: 1b,52

Cursor up function key code IN Hex IS 1b,52
CONFIRM-- Quit, Ok, Modify (Q, O, M): Q
```

A similar dialogue must then occur for each of the remaining parameters. If you type **Q** for quit while modifying a file, file modification is aborted and the following message is displayed on the terminal screen:

```
TERMINATING file modification
```

When you have completed modifying a file, the Configurator writes out the modified file and displays the following message:

```
The configuration file has been written.
```

ACE and Configuration Files

Once you have created or modified a configuration file, you will want to use it with ACE. The following paragraphs discuss the following:

1. The default configuration file.
2. Other configuration files.
3. How to change the contents of the default configuration file.

The Default File

The TEKTRONIX CT8500 terminal is recommended for use with 8500-Series systems. The configuration file for this terminal is contained in the file "/EOS/NO.EMULATOR/ace.cfg." This file is automatically installed when you install ACE. If no configuration file parameter is specified on the ACE invocation line, ACE by default reads /EOS/NO.EMULATOR/ace.cfg into its configuration tables. This is why /EOS/NO.EMULATOR/ace.cfg is also called the default file.

Other Configuration Files

The main purpose of the Configurator is to create new configuration files for terminals **other** than the TEKTRONIX CT8500. Once a new file is created, it can be used by ACE. In the following example, ACE is invoked with a configuration file other than the default file:

```
> ACE file.text,,,new.cfg
```

Here, ACE is configured for the terminal specified by "new.cfg", with "file.text" as the file to be edited. The output file and command file parameters are omitted in this example. If no configuration file parameter is given, then the default configuration file, "/EOS/NO.EMULATOR/ace.cfg", is read in by ACE.

Changing the Contents of the Default File

The following procedure shows you how to copy the contents of the old default file and then read in the contents of a new configuration file. You should only perform this procedure after you have created and debugged a configuration file that you want as the new default. This procedure assumes that you have a directory named "/EOS/config-dir" in which you store configuration files:

```
> USER /EOS,TEKTRONIX
```

```
[This command places you in the EOS directory and allows]
[ you to alter files owned by the user TEKTRONIX.      ]
```

```
> COPY:B NO.EMULATOR/ace.cfg config-dir/CT8500.cfg
```

```
[This command copies the contents of the default file  ]
[ to a file named CT8500.cfg.                          ]
```

```
> COPY:B config-dir/newterm.cfg NO.EMULATOR/ace.cfg
```

```
[This command copies the new configuration file to the ]
[ default file.                                       ]
```

```
> USER , ,NO.NAME
```

```
[Changes the user from TEKTRONIX back to NO.NAME.     ]
```

The preceding procedure changed the contents of the default file **without** writing over the current contents. Remember that you must become the user TEKTRONIX to alter the file, and that you should copy the current contents of the default file to another file before you alter the default file's contents.

ERROR MESSAGES

Error: command table entry conflict. Conflicting parameter strings for function key codes were input. Such conflicts occur when function key code parameters are identical or when one function key code parameter is a prefixing substring of another. For example, you cannot configure the scroll right key code for (ESC) and the scroll left function key code for the sequence (ESC)A.

Error: current file is not a configuration file. An attempt was made to view or modify a file that is not a configuration file.

Error: decimal value greater than 127 encountered. A number greater than 127 was input when entering a parameter in decimal input format.

Error: <filename> already exists. An attempt was made to name a configuration file with the name of a file already existing in the current directory.

Error: <filename> does not exist. An attempt was made to view or modify a file that either does not exist or cannot be opened.

Error: hex value greater than 7f encountered. A number greater than 7F was encountered in a value entered in hexadecimal input format.

Error: illegal Ascii string. A character string input in ASCII input format contained a non-displayable character.

Error: illegal boolean response. A character that is not Y, y, N, or n was entered in Boolean input format.

Error: illegal character found in decimal. A non-decimal digit was encountered in a number entered in decimal input format.

Error: illegal character found in hex string. A non-hexadecimal digit was encountered in a number entered in hexadecimal input format.

Error: no characters input. Only a return was typed, causing a string of zero length to be input as a parameter.

Error: one character response expected. Your response to a selection was not one character long.

Error: this parameter must be exactly one byte long. An attempt was made to enter a multi-byte string for a parameter that must be exactly one byte long.

Error: unrecognized response. The Configurator presented a menu, and your one-letter response was not on the menu.



Function
 cursor up
 cursor down
 cursor right
 cursor left
 cursor home
 scroll up
 scroll down
 scroll right
 scroll left
 page up
 page down
 revise mode
 insert character
 insert line
 delete character
 delete line
 command escape
 mark cursor position

Code
 b
 1b
 c
 1b, 49
 1e
 1, 40, d
 1, 41, d
 1, 42, d
 1, 43, d
 1, 44, d
 1, 45, d
 12
 1b, 51
 1b, 45
 1b, 57
 1b, 52
 a
 9

Key
 up
 down
 right
 back tab
 home
 F1
 F2
 F3
 F4
 F5
 F6
 AR
 ins char
 ins line
 del char
 del line
 ↑
 ↑I

Command Sequence
 prefix character
 terminator character
 erase screen
 erase cursor to end of line
 erase cursor to end of screen
 insert character
 insert character mode terminate
 delete character
 insert line
 delete line
 turn cursor on
 turn cursor off
 turn blink on
 turn blink off
 turn inverse on
 turn inverse off
 cursor positioning

Code
 * 1b
 *
 2a
 * 74
 * 79
 *
 *
 *
 * 45
 * 52
 *
 *
 *
 *
 *
 *
 *
 *
 3d

Characteristic
 Zero x-coordinate
 zero y-coordinate
 @ block delimiter replacement
 End of line replacement character
 number of lines
 number of columns
 initialization command sequence
 termination command sequence
 line insertion at cursor
 y coordinate before x
 cursor coordinate in decimal
 cursor coordinate increment pos

Value
 20
 20
 @
 ~
 24
 80
 * 2a
 * 2a
 yes
 YES
 NO
 YES

* optional sequence

ACE Configuration Worksheet

File: exor

Terminal: exorterm

Function
 cursor up
 cursor down
 cursor right
 cursor left
 cursor home
 scroll up
 scroll down
 scroll right
 scroll left
 page up
 page down
 revise mode
 insert character
 insert line
 delete character
 delete line
 command escape
 mark cursor position

Code
 b
 a
 c
 d
 1b, 40
 1b, 20
 1b, 21
 1b, 22
 1b, 23
 1b, 24
 1b, 25
 12
 1b, 50
 1b, 56
 1b, 51
 1b, 57
 7
 9

Key
 UP ARROW
 DN ARROW
 RI ARROW
 return
 HOME
 F1
 F2
 F3
 F4
 F5
 F6
 ↑R
 INS CHAR
 INS LINE
 DEL CHAR
 DEL LINE
 ↑G
 ↑I

Command Sequence
 prefix character
 terminator character
 erase screen
 erase cursor to end of line
 erase cursor to end of screen
 insert character
 insert character mode terminate
 delete character
 insert line
 delete line
 turn cursor on
 turn cursor off
 turn blink on
 turn blink off
 turn inverse on
 turn inverse off
 cursor positioning

Code
 * 1b
 *
 58
 * 55
 * 54
 *
 *
 * 51
 * 56
 * 57
 *
 *
 * 60
 * 61
 * 62
 * 63
 45

Characteristic
 Zero x-coordinate
 zero y-coordinate
 @ block delimiter replacement
 End of line replacement character
 number of lines
 number of columns
 initialization command sequence
 termination command sequence
 line insertion at cursor
 y coordinate before x
 cursor coordinate in decimal
 cursor coordinate increment pos

Value
 20
 20
 40
 7e
 18
 50
 * 58
 * 58
 Yes
 Yes
 No
 Yes

* optional sequence

ACE Configuration Worksheet

File: avp

Terminal: adds viewpoint

Function
 cursor up
 cursor down
 cursor right
 cursor left
 cursor home
 scroll up
 scroll down
 scroll right
 scroll left
 page up
 page down
 revise mode
 insert character
 insert line
 delete character
 delete line
 command escape
 mark cursor position

Code
2a
Da
6
15
1
d
2,21
2,22
2,23
2,31
2,32
12
69
49
64
44
7
9

Key
up
down
right
left
home
CR
sh F1
sh F2
sh F3
F1
F2
↑R
L
F
d
I
↑G
P

Command Sequence
 prefix character
 terminator character
 erase screen
 erase cursor to end of line
 erase cursor to end of screen
 insert character
 insert character mode terminate
 delete character
 insert line
 delete line
 turn cursor on
 turn cursor off
 turn blink on
 turn blink off
 turn inverse on
 turn inverse off
 cursor positioning

Code
 * _____
 * _____
C
 * 1b,4b
 * 1b,6b
 * _____
 * _____
 * _____
 * _____
 * _____
 * _____
 * _____
 * _____
 * _____
 * _____
1b,59

Characteristic
 Zero x-coordinate
 zero y-coordinate
 @ block delimiter replacement
 End of line replacement character
 number of lines
 number of columns
 initialization command sequence
 termination command sequence
 line insertion at cursor
 y coordinate before x
 cursor coordinate in decimal
 cursor coordinate increment pos

Value
20
20
40
7e
18
50
 * C
 * C
Yes
Yes
No
Yes

* optional sequence

ACE Configuration Worksheet

Function	Code	Key
cursor up		
cursor down		
cursor right		
cursor left		
cursor home		
scroll up		
scroll down		
scroll right		
scroll left		
page up		
page down		
revise mode		
insert character		
insert line		
delete character		
delete line		
command escape		
mark cursor position		

Command Sequence	Code
prefix character	* 1b
terminator character	*
erase screen	1c
erase cursor to end of line	* F
erase cursor to end of screen	* 18
insert character	*
insert character mode terminate	*
delete character	*
insert line	* 1a
delete line	* 13
turn cursor on	*
turn cursor off	*
turn blink on	*
turn blink off	*
turn inverse on	*
turn inverse off	*
cursor positioning	11

Characteristic	Value
Zero x-coordinate	0
zero y-coordinate	0
@ block delimiter replacement	40
End of line replacement character	7e
number of lines	18
number of columns	50
intialization command sequence	* 1c
termination command sequence	* 1c
line insertion at cursor	yes
y coordinate before x	no
cursor coordinate in decimal	no
cursor coordinate increment pos	yes

* optional sequence

ACE Configuration Worksheet

File: wyse

Terminal: WYSE 100

Function	Code	Key
cursor up	_____	↑
cursor down	_____	↓
cursor right	_____	→
cursor left	_____	←
cursor home	_____	HOME
scroll up	1e	_____
scroll down	1b 7b	_____
scroll right	_____	_____
scroll left	_____	_____
page up	_____	_____
page down	_____	_____
revise mode	_____	_____
insert character	_____	_____
insert line	_____	_____
delete character	_____	_____
delete line	_____	_____
command escape	_____	_____
mark cursor position	_____	_____

Command Sequence	Code
prefix character	* _____
terminator character	* _____
erase screen	_____
erase cursor to end of line	* _____
erase cursor to end of screen	* _____
insert character	* _____
insert character mode terminate	* _____
delete character	* _____
insert line	* _____
delete line	* _____
turn cursor on	* _____
turn cursor off	* _____
turn blink on	* _____
turn blink off	* _____
turn inverse on	* _____
turn inverse off	* _____
cursor positioning	_____

Characteristic	Value
Zero x-coordinate	_____
zero y-coordinate	_____
@ block delimiter replacement	_____
End of line replacement character	_____
number of lines	_____
number of columns	_____
intialization command sequence	* _____
termination command sequence	* _____
line insertion at cursor	_____
y coordinate before x	_____
cursor coordinate in decimal	_____
cursor coordinate increment pos	_____

* optional sequence

Function	Code	Key
cursor up	_____	_____
cursor down	_____	_____
cursor right	_____	_____
cursor left	_____	_____
cursor home	_____	_____
scroll up	_____	_____
scroll down	_____	_____
scroll right	_____	_____
scroll left	_____	_____
page up	_____	_____
page down	_____	_____
revise mode	_____	_____
insert character	_____	_____
insert line	_____	_____
delete character	_____	_____
delete line	_____	_____
command escape	_____	_____
mark cursor position	_____	_____

Command Sequence	Code
prefix character	* _____
terminator character	* _____
erase screen	_____
erase cursor to end of line	* _____
erase cursor to end of screen	* _____
insert character	* _____
insert character mode terminate	* _____
delete character	* _____
insert line	* _____
delete line	* _____
turn cursor on	* _____
turn cursor off	* _____
turn blink on	* _____
turn blink off	* _____
turn inverse on	* _____
turn inverse off	* _____
cursor positioning	_____

Characteristic	Value
Zero x-coordinate	_____
zero y-coordinate	_____
@ block delimiter replacement	_____
End of line replacement character	_____
number of lines	_____
number of columns	_____
intialization command sequence	* _____
termination command sequence	* _____
line insertion at cursor	_____
y coordinate before x	_____
cursor coordinate in decimal	_____
cursor coordinate increment pos	_____

* optional sequence

Section 6

ERROR MESSAGES

This section contains general editor messages as well as editor error messages. The general messages follow the error messages. Both lists are in alphabetical order.

EDITOR ERROR MESSAGES

CASE IGNORE STRING MUST NOT BE NULL. The string entered between case ignore delimiters must not be empty.

COMMAND ABORTED. The command was aborted; the reason is provided in an accompanying message except when CTRL-x was entered to terminate command entry.

COMMAND FILE <filespec> DOES NOT EXIST. ENTER NAME OF COMMAND FILE: The filespec entered does not exist. Check spelling.

CONFIGURATION FILE DOES NOT EXIST. ENTER NAME OF CONFIGURATION FILE: The filespec used as the configuration file parameter in the ACE command does not exist. Check spelling.

EDITING NEW FILE, OUTPUT FILE SHOULD NOT BE SPECIFIED. When the input file is a new file, the editor uses the input file as the output file.

FILE NAME TO BE EDITED NOT SPECIFIED. The ACE command was entered with no input file specified.

FILE <filespec> IS IN USE. INVALID FOR EDIT READ/WRITE FILE. The file specified in the O command must differ from the files specified at editor invocation and must be a valid filespec.

FILE NAME <filespec> IS TOO LONG. The maximum length of a filespec is 64 characters.

INVALID COUNT. A repetition count such as //, /-, or more than four digits was entered.

INVALID PARAMETER, RE-ENTER. The repetition count entered is invalid for the given command. (For example: O(dc) or <A.

INVALID TAB STOP, COMMAND IGNORED. Tab stops, defined by the T command, must be an ascending series of decimal-integer column numbers in the range 2-998, separated by commas.

INVALID WILDCARD CHARACTER, IGNORED. The character entered with the WI, WE, or WG command has already been defined as a wildcard character or is a non-displayable character.

MACRO <name> IS NOT DEFINED. The macro name in an MX command has not been previously defined.

MACRO NAME INVALID, COMMAND ABORTED. An invalid number was entered as a macro name either in defining or executing the macro. Only the digits 0 through 9 are valid macro names.

MX COMMAND IS INVALID IN MACRO DEFINITION. Macros may not be nested; the MX command cannot be used within a macro.

NO EDIT READ/WRITE FILE HAS BEEN OPENED. A file must be opened with the O command before an attempt is made to read from or write to the file with the ER or EW command.

NO OCCURRENCES OF <string> FOUND. The search string was not found. The window is unaffected.

NO PREVIOUS "F" OR "R" COMMAND USED. SEARCH STRING IS UNDEFINED. The F or R command was used with a null search string and no prior search string was defined. No search is performed.

NO PREVIOUS "R" COMMAND USED. REPLACEMENT STRING IS UNDEFINED. The R command was used with a null replacement string that and no prior replacement string was defined. No replacement is made.

NO REPLACEMENTS OF <string> DONE. The search string was not found in the file; no replacement is made.

OUTPUT FILE <filespec> ALREADY EXISTS. DELETE FILE? The output file entered already exists. The editor gives you a chance to delete the existing file. If you respond with "Y" or "y", the file is deleted and a new one created. If any other response is given, the editor prompts you for another file name.

SYSTEM FUNCTION ERROR ON <filespec> = <hexadecimal>. This is a operating system related error. Any error of this type will abort the edit session without saving files. Check the write-protect tab on the disc, ATTRIBUTES, or DIRECTORY and filespecs. If the error persists with entry that appears to be valid, contact your Tektronix service representative.

UNBALANCED CASE IGNORE WILDCARD CHARACTER, COMMAND ABORTED. An odd number of case ignore delimiter characters were entered in the string. An even number must be entered: one to begin each portion of the string in which the case is to be ignored, and one to end it.

TOO MANY TAB STOPS, COMMAND IGNORED. A maximum of 24 tab stops may be defined by the T command.

UNRECOGNIZABLE COMMAND, IGNORED. An unrecognizable command mnemonic was entered. Check the correct syntax.

GENERAL MESSAGES

BEGINNING OF FILE ENCOUNTERED. An ACE command has encountered the beginning of the file. The first line of the file is displayed at the top of the window.

BEGINNING OF LINE ENCOUNTERED. The (sr) command has encountered the beginning of the line. The first character in the line is displayed in the first column of the window.

CASE IGNORE WILDCARD CHARACTER IS <char>. The WI command defines the case ignore delimiter. This message is displayed when the character is defined.

CASE IGNORE WILDCARD CHARACTER IS NOW UNDEFINED. The null (empty) string was entered as the wildcard character with the WI command.

DELETE FILE? The file specified for output already exists. If you enter "Y" or "y", the file will be deleted; any other response will leave the file intact.

DELETE REQUESTED ENTER "Y" TO ACKNOWLEDGE. When the query option is on, the editor asks you to confirm a delete request that extends beyond the window size. If you enter "Y" or "y", the text will be deleted; any other response will leave the text intact.

DEPRESS CARRIAGE RETURN TO CONTINUE. The editor prompts you to enter return, to redisplay the window after the EP, H, ML, and V commands.

ECHO ON. The EC command specifies the state of the echo process. "EC" turns the echo on and "-EC" turns the echo off.

ECHO OFF. The EC command specifies the state of the echo process. "EC" turns the echo on and "-EC" turns the echo off.

EDIT READ/WRITE FILE IS NOW UNDEFINED. A null (empty) string was entered as the filespec in an O command.

END OF FILE ENCOUNTERED. An ACE command has encountered the end of the file. If the file contains more lines than the window can display, the last line of the file is displayed at the bottom of the window.

ENTER NAME OF COMMAND FILE: No parameters were entered with the ACE command. Enter a valid command filespec. If a null (empty) string is entered, no command file will be used.

ENTER NAME OF CONFIGURATION FILE: No parameters were entered with the ACE command. Enter a valid configuration filespec. If a null (empty) string is entered, the default configuration file will be used.

ENTER NAME OF FILE TO BE EDITED: No parameters were entered with the ACE command. Enter a valid filespec. If a null (empty) string is entered, ACE will prompt you again.

ENTER NAME OF FILE TO RECEIVE RESULTS OF EDIT SESSION: No parameters were entered with the ACE command. Enter a valid filespec. If a null (empty) string is entered, the output file will be renamed to the input file name after the EX command.

ENTER COUNT/COMMAND. The editor is waiting for you to enter the next editor command.

ENTER SEARCH STRING. The editor is waiting for you to enter the search string.

ENTER REPLACEMENT STRING. The editor is waiting for you to enter the replacement string.

ENTER TAB STOPS. The editor is waiting for you to enter the tab stop string.

ENTER WILDCARD CHARACTER. The editor is waiting for you to enter the wildcard character.

ENTER MOTION. A mark has been entered; the editor is waiting for you to perform cursor motion, scrolling, or paging.

ENTER MACRO NAME. The editor is waiting for you to specify the macro name: an integer from 0 to 9.

ENTER MACRO DEFINITION. The editor is waiting for you to define the new macro.

ENTER NEW TEXT. The editor is waiting for you to enter new text.

ESCAPE WILDCARD CHARACTER IS <char>. The WE command defines the escape wildcard character that is used when you want to enter the (ESC) character into text or search for it. This message is displayed when the character is defined.

ESCAPE WILDCARD CHARACTER IS NOW UNDEFINED. The null string was entered as the escape wildcard character with the WE command.

FILE <filespec> CREATED. A new file was created with the file name specified in the O command.

FILE <filespec> OPENED. The file specified in the O command is now open.

FILE ACCESS IN PROGRESS. Text outside the workspace is being accessed.

GENERAL WILDCARD CHARACTER IS <char>. The WG command defines the general wildcard character that is used in search and replacement strings. This message is displayed when the character is defined.

GENERAL WILDCARD CHARACTER IS NOW UNDEFINED. The null string was entered as the wildcard character with the WG command.

MACRO <name> DEFINED. The editor displays this message when a valid macro name is defined with the MD command.

MACRO <name> IS NOW UNDEFINED. A null (empty) string was entered as the macro definition.

xx MACRO EXECUTIONS PERFORMED. The MX command has terminated before all executions are completed. Either a search has failed, the cursor is at the beginning or the end of the file, or an invalid command was encountered during a macro execution.

MACRO TERMINATED AT BOF. The macro execution terminated because the cursor is at the beginning of the file.

MACRO TERMINATED AT EOF. The macro execution terminated because the cursor is at the end of file.

MAXIMUM COLUMN ENCOUNTERED. The (sl) command has encountered column 999. Column 999 is displayed in the last column of the window.

xx OCCURRENCES OF <string> FOUND. The number of occurrences indicated in the F command exceeds the total number of occurrences of <string> between the cursor and the end of the file.

QUERY [ON/OFF.] The Q command specifies the state of the query option. "Q" turns on the query option, "-Q" turns off the query option.

REPLACE REQUESTED. ENTER "Y" TO ACKNOWLEDGE: When the query option is on, the multiple replacement command causes the window to be displayed for each occurrence of the search string and prompts you to confirm each replacement. A response other than "Y" or "y" will cause the particular replacement to be bypassed.

xx REPLACEMENTS OF <string> DONE. The number of replacements indicated in the R command exceeds the total number of the occurrences of <search> string between the cursor and the end of the file. The cursor is positioned to the right of the last replacement.

STOP REQUESTED, ENTER "Y" TO ACKNOWLEDGE: The S command causes the editor to terminate execution without saving the results of the edit session. Respond with "Y" or "y" to terminate.

nn TAB STOPS DEFINED. The editor displays the number of tab stops defined after the T command.

Section 7

GLOSSARY

ACE. The Advanced CRT-Oriented Editor.

backup file. The file produced by the editor if no output file is specified. (The unmodified input file renamed.)

case ignore wildcard. The character used to delimit search strings in which the case (upper/lower) of letters is to be ignored.

command-escape. The configurable key used to permit execution of non-configurable commands in revise mode.

command file. A file containing ACE commands, executed when the editor is invoked, before any user interaction is allowed.

configurable command. A command whose function is assigned to a special key on a terminal.

configuration file. A file that configures the editor for a specific terminal.

configurator. The program used to create terminal configuration files for ACE.

current character. The character at the cursor.

current line. The line containing the current character

cursor. The visual indicator of your position in a file. Editor commands operate in relation to the position of the cursor.

end-of-line. A configurable character used to indicate the end of a text line. With the TEKTRONIX CT8500 terminal, the tilde (~) is the default character used to visually represent the end-of-line on your screen.

escape. An ASCII character represented by a key on the terminal usually marked ESC. The escape character is used to terminate ACE parameters.

filespec. The "name" of a file. A filespec is either the entire pathname of a file, or a brief name relative to the current directory. See your System Users Manual.

home. The character position at the upper left hand corner of the window.

input file. The file that you want to edit; specified when you invoke the ACE editor. If no output file is specified, the output file is rename to the input file name when the editor is terminated with an EX command.

macro. A specially defined command sequence that is executed by a short command rather than by entering the whole command sequence.

mark. A file position indicator used to delimit blocks of text for later manipulation.

message line. A line reserved in the monitor area for messages to you from ACE.

monitor area. The four lines at the bottom of your screen containing status information about your file and the terminal display.

motion. Any sequence of cursor motion, scrolling, and paging commands that move the cursor from one location in the text to another.

non-configurable command. A command that is entered by a fixed key or keys, regardless of the terminal.

output file. The name of the file to which your editing corrections are written. This parameter is specified when you invoke the ACE editor. If no output file is specified, the output file is renamed to the input file name when the editor is terminated with an EX command.

page. A block of text containing the number of lines that can be displayed in the window. This should not be confused with a physical page of text.

parameter line. A line reserved in the monitor area to display parameters that you enter.

query. An option that requires you to confirm any potentially destructive changes that you may want to perform.

revise mode. A character input mode during which text may be overwritten.

screen. The face of the terminal CRT. The screen is divided into the window and the monitor area.

scrolling. A line-by-line or column-by-column movement of the text in the window, during which the cursor remains stationary.

secondary file. Any file, other than the input file, output file, command file, and configurator file that the editor reads from or writes to. Any file opened with an O command.

status line. A line in the monitor area containing status information.

undelete. The most recent deletion is read into the file ahead of the cursor when the undelete command is entered.

window. That part of the terminal screen that is used to view the contents of a file. The four lines of the monitor area are **not** part of the window.

Section 8

INDEX

A

A command 3-13
 aborting the editor 3-61
 ACE 7-1
 command 3-9
 interactive form 3-9
 configurator relationship 5-2
 advance cursor 3-13
 ASCII codes 4-6

B

backspace 3-6
 backup file 7-1
 blank manipulating 2-19
 block
 deleting 2-27
 duplicating 2-32
 moving 2-28
 saving 2-30

C

C command 3-15
 CT8500 terminal 1-7
 carriage return 3-6
 case
 conversion 2-39, 41, 43, 45
 exchange 3-74, 76
 case-ignore
 delimiter 3-73
 string 3-37
 wildcard 7-1
 (cd) command 3-18
 (ch) command 3-29
 change characters 3-15
 changing
 to line beginning 2-24
 to line end 2-25
 character inserting 1-20, 3-41
 characters, non-displayable 3-69
 (cl) command 3-20
 (cmd-esc) command 3-21
 command
 entry in revise mode 3-21, 60
 list
 configurable 4-1
 non-configurable 4-2
 prefix, configurable 5-6
 sequence functions 5-11
 command escape 7-1
 command files 2-51, 7-1
 command syntax 2-7, 3-39
 command-escape character 3-71

commands
 configurable 1-7
 non-configurable 1-7
 concatenating
 files 2-38
 lines 2-23
 configurable commands 1-6, 7-1
 list 4-1
 parameters 5-3
 prefix 5-6
 syntax 3-39, 4-3
 configurable key layout, CT8500 4-7
 configuration
 file 5-21, 7-1
 creating 5-17
 default 5-22
 modifying 5-21
 configurator 5-1, 7-1
 error messages 5-23
 invocation 5-14
 ace relationship 5-2
 control-x 3-7
 correcting of input errors 1-11
 count 3-2
 (cr) command 3-22
 creating
 files 2-3, 3-52
 macros 2-47
 CT8500 keyboard layout 4-7
 current character 7-1
 current line 7-1
 (cu) command 3-23
 cursor 1-4, 7-1
 cursor movement 1-15, 2-7, 8, 9, 10, 3-18, 19,
 20, 22, 23, 44

D

(dc) command 3-24
 default configuration file 5-22
 defining
 case-ignore delimiter 3-73
 command-escape character 3-71
 general wild-card 3-72
 macros 3-48
 tab stops 3-66
 delete
 characters 3-24
 lines 3-27
 blocks 2-27
 macros 3-48
 delimiter, case-ignore 3-73
 demonstration run 1-10
 demonstration text 1-13
 directory listing 1-14
 (dl) command 3-27
 duplicating blocks 2-32

E

EC command 3-29
 echo commands 3-29
 editor profile 3-30
 end-of-line character 1-12, 7-1
 entering text 1-12
 EP command 3-30
 ER command 3-31
 error messages 6-1
 configurator 5-23
 errors, input, correcting 1-11
 escape 3-6, 7-1
 escape wild-card character 3-71
 EW command 3-33
 EX command 3-35
 exchange with lower case 3-74
 exchange with upper case 3-76
 executing macros 2-47, 3-50
 exiting the editor 1-14, 24, 2-2, 3-35
 without saving 3-61

F

F command 3-36
 file
 concatenating 2-38
 creating 2-3, 3-52
 modifying 2-5
 opening 3-52
 reading 3-31
 splitting 2-36
 writing 3-33
 configuration default 5-22
 files, command 2-51
 filespec 7-1
 finding a string 1-19, 2-13, 3-36
 frequency of occurrence of a string 2-24

G

general messages 6-3
 general wild-card character 3-72
 global replacement 2-17
 glossary 7-1

H

H command 2-7, 3-39
 help 2-7, 3-39
 home position 3-19, 7-1

I

(ic) command 3-41
 (il) command 3-43
 initializing the terminal 5-10
 input errors, correcting 1-11
 input file 7-1

inserting
 characters 1-20, 3-41
 lines 1-20, 3-43
 installation 1-8
 invoking the editor 1-11, 2-2

J

J command 3-44
 jump 3-44

K

keyboard layout, CT8500 4-7

L

lines
 concatenation 2-23
 deleted text restoring 2-31
 deleting 2-30
 inserting 1-20, 3-43
 long 2-21
 moving 2-31
 restoring deleted text 2-31
 splitting 2-22
 listing
 macros 3-49
 the directory 1-14
 local and remote modes 5-4
 long lines 2-21
 lower case exchange 3-74

M

macro 7-2
 definition 2-47, 3-48
 deleting 3-48
 executing 2-47, 3-50
 listing 3-49
 mark 7-2
 (mark) command 3-46
 mark-motion 3-2, 24, 33
 marking a character 3-46
 MD command 3-48
 message line 1-4, 3-5, 7-2
 messages
 error 6-1
 general 6-3
 ML command 3-49
 modes, local and remote 5-4
 modifying a file 2-5
 monitor area 1-4, 3-5, 7-2
 motion 7-2
 moving
 blocks 2-28
 text 1-21, 3-67
 the cursor 1-15, 2-7, 8, 9, 10
 MX command 3-50

N

non-configurable command syntax 3-40, 4-4
 non-configurable commands 1-7, 4-2, 7-2
 non-displaying characters 2-19, 4-5
 viewing 3-69
 notation conventions 1-8
 number of occurrences of a string 2-24

O

O command 3-52
 opening files 3-52
 output file 7-2

P

page 7-2
 paging 1-5, 3-54, 55
 parameter line 7-2
 parameter entry line 1-4, 3-5
 parameters, configurable 5-3
 (pd) command 3-54
 prefix, configurable command 5-6
 printing a block 2-35
 procedure format 2-1
 profile 3-30
 (pu) command 3-55

Q

Q command 3-56
 query 3-56, 7-2

R

R command 3-57
 reading files 3-31
 recovering deleted text 3-67
 remote and local modes 5-4
 repeating blocks 2-32
 repetitive execution 1-17
 repetition count 3-2
 replace string 3-57
 replacing strings 1-23, 2-16
 resetting tabs 2-11
 revise mode 1-18, 2-12, 3-60, 7-2
 command entry 3-21, 60
 (rm) command 3-60
 rubout 3-6

S

S command 3-61
 saving blocks 2-30
 screen 7-2
 screen representation 1-3, 3-4
 scrolling 1-4, 3-62, 63, 64, 65, 7-2
 (sd) command 3-62
 searching 1-19, 2-13, 3-36
 secondary file 7-2

(sl) command 3-63
 space bar 3-6
 splitting
 a file 2-36
 a line 2-22
 (sr) command 3-64
 status line 1-4, 3-5, 7-2
 stop 3-61
 string replacement 1-23, 2-16, 3-57
 global 2-17
 string search 1-19, 2-13, 3-36
 string, number of occurrences 2-24
 (su) command 3-65
 syntax list
 configurable 4-3
 non-configurable 4-4
 syntax of commands 2-7, 3-39

T

T command 3-66
 tab 3-6, 66
 area 1-16
 manipulation 2-19
 resetting 2-11
 temporary files 1-14
 terminal
 commands 5-8
 configuration characteristics 5-13
 CT8500 1-7
 initializing 5-10
 requirements 5-1
 terminating 5-10
 termination character 5-9
 text
 entry 1-12
 moving 1-21

U

U command 3-67
 undelete 3-67, 7-2
 upper case exchange 3-76

V

V command 3-69
 verify command entry 3-56
 viewing non-displaying characters 3-69

W

WE command 3-71
 WG command 3-72
 WI command 3-73
 wild-card
 characters 3-71, 72, 73
 searching 3-37
 window 1-4, 3-4, 7-2
 writing files 3-33

X

XL command 3-74
 XU command 3-76

MANUAL CHANGE INFORMATION

At Tektronix, we continually strive to keep up with latest electronic developments by adding circuit and component improvements to our instruments as soon as they are developed and tested.

Sometimes, due to printing and shipping requirements, we can't get these changes immediately into printed manuals. Hence, your manual may contain new change information on following pages.

A single change may affect several sections. Since the change information sheets are carried in the manual until all changes are permanently entered, some duplication may occur. If no such change pages appear following this page, your manual is correct as printed.

DESCRIPTION

TEXT CORRECTIONS

- Page 6-1 Insert the following error message immediately before EDITING NEW FILE... :
- COUNT INVALID WHEN MARK DEFINED.** A repetition count was entered before a (dc) or EW command after a (mark) was defined.
- Page 6-2 Insert the following error message immediately before MX COMMAND IS INVALID IN MACRO DEFINITION:
- MARK REMOVED, COMMAND ABORTED.** A CNTRL-X was entered after a (mark) and before a (dc) or EW command was entered.
- Page 6-2 The error message **TOO MANY TAB STOPS...** should precede the error message **UNBALANCED CASE IGNORE...**
- Page 6-3 Insert the following error message immediately before the heading GENERAL MESSAGES:
- UNTERMINATED PARAMETER IN MACRO DEFINITION.** The macro being executed contains a parameter not terminated with an (ESC). The commands before the unterminated parameter are executed; commands after the parameter are not terminated.
- Page 6-5 Insert the following error message immediately before MAXIMUM COLUMN ENCOUNTERED:
- MARK REMOVED.** The mark was overwritten in revise mode. The mark is removed and is replaced by the new character.

