ALTER TEXT EDITOR USER'S GUIDE

Order Number: 121956-001

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

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

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update nor to keep current the information contained in this document.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

The following are trademarks of Intel Corporation and its affiliates and may be used only to identify Intel products:

BXP	Intel	iSBX	Multichannel
CREDIT	Int_el	Library Manager	Multimodule
i	Intelevision	MCS	Plug-A-Bubble
ICE	Intellec	Megachassis	PROMPT
iCS	iOSP	Micromainframe	RMX/80
i_m	iRMX	Micromap	System 2000
iMMX	iSBC	Multibus	UPI
Insite			

REV.	REVISION HISTORY	DATE
-001	Original issue. —Series III "RUN" utility V1.3	4/82

PREFACE

This manual provides straightforward instructions on the interactive text editor for use with the Intellec Series III Microcomputer Development System. Since ALTER runs under the Series III operating system, knowledge of ISIS is assumed.

This manual is for new users as well as those who are already familiar with microcomputers and text editors.

This manual contains four chapters and four appendixes.

Chapter 1, Introduction and Tutorial, defines hardware and software necessary to run ALTER, describes invoking and leaving the editor, and provides a brief tutorial session.

Chapter 2, The Editor Basics, describes concepts essential to using ALTER and provides an in-depth description (with photos) of the ALTER display, keyboard, etc.

Chapter 3, Editing Commands, describes editing commands that act directly on text.

Chapter 4, Macro Commands, describes commands that allow you to create macros.

Appendix A, ALTER Command Summary, lists ALTER commands and command format, and provides a brief explanation of each command.

Appendix B, ALTER Error Messages, lists ALTER error messages with an explanation of the probable cause and the state in which the system is left.

Appendix C, ASCII Codes, lists ASCII codes with their hexadecimal values.

Appendix D, Configuring ALTER for non-Intel Terminals, describes how to configure ALTER for non-Intel terminals and provides tested configurations for specific terminals.

Related Publications

For more information on the Series III Microcomputer Development System, see the following manuals:

- *Intellec Series III Microcomputer Development System Product Overview*, 121575
- *Intellec Series III Microcomputer Development System Console Operating Instructions*, 121609
- *Intellec Series III Microcomputer Development System Programmer's Reference Manual*, 121618
- *Intellec Series III Microcomputer Development System Installation and Checkout Manual*, 121612

Notational Conventions

UPPERCASE	Characters shown in uppercase must be entered in the order shown. You may enter the characters in uppercase or lowercase.
italic	Italic indicates a meta symbol which may be replaced with an item that fulfills the rules for that symbol.
[ ]	Brackets indicate optional arguments or parameters.
{ }	One and only one of the enclosed entries must be selected unless the field is also surrounded by brackets, in which case it is optional.
|	The vertical bar separates options within brackets [ ] or braces { }.
input lines	Reverse video (white on black type) differentiates input lines and user responses from system output.
<cr>	Indicates a carriage return.

CONTENTS

CONTENTS (Cont'd.)

TABLES

ILLUSTRATIONS

CHAPTER 1
INTRODUCTION AND TUTORIAL

Introduction

ALTER is an interactive, screen-oriented editor with menu style command prompts that runs under the ISIS operating system on the Intellec Series III Microcomputer Development System. ALTER requires a minimum of 50K RAM.

ALTER takes advantage of CRT capabilities to allow you to:

- Display and scroll text on the screen
- Move to any position in the text file or to any point on the screen instantly
- Correct typing mistakes as you type
- Rewrite text by typing new letters over old ones
- Make insertions and deletions easily

To simplify everyday text editing, ALTER also provides features that allow you to:

- Find any string of characters and substitute another string
- Move or copy sections of text within a file or to another file
- Create macros to execute several commands at once, thereby simplifying repetitive editing tasks
- Scan listing files while editing your primary file
- Indent text automatically
- View lines over 80 characters long

An ALTER Tutorial

This session is a short tutorial that illustrates the most basic ALTER commands. The functions covered are:

- Activating the editor
- Entering text
- Changing text
- Deleting text
- Copying text
- Exiting the editor

The purpose of this session is to get you started, not to teach you the details of the editor. Only a few of the most basic ALTER commands are presented; therefore, it is essential that you read the remaining chapters in this manual to gain a complete understanding and appreciation of the advantages of using ALTER. Chapter 2 describes the editor basics essential to using ALTER. Chapter 3 describes each editing command in detail. Chapter 4 describes Macro commands.

To use this section, you should be at a Series III system. See the *Intellec Series III Console Operating Instructions* for additional operating system information.

Activating the Editor

Activate the editor by typing:

```
RUN ALTER <cr>
```

The editor displays the following menu prompt at the bottom of the screen.

```
---- SERIES-III ALTER V1.0
Again   Block    Delete   Execute   Find    -find    Get   --more--
```

The word "more" indicates that pressing the TAB key will display the next line of prompts.

The vertical bar ' | ' (EOF marker) marks the end of text in the file. Since the new file holds no text yet, this symbol appears at the beginning of the file. As you type text into the file, this symbol moves and continues to mark the end of the file.

The cursor (a blinking underline) covers the EOF marker. The cursor moves in the direction indicated by the arrows on the cursor control keys (the cursor does not move past the EOF marker, however). The HOME key, in conjunction with the cursor control keys, allows for fast cursor movement, depending on the prior cursor movement command.

When first invoked, ALTER is at main command level and is waiting for your input. The menu prompt line offers a selection of main commands or modes (XCHANGE and INSERT are considered modes). When invoked, several main commands offer subcommands. You must be at main command level to execute commands, except for the cursor movement commands and the delete commands, which work as at main command level while in INSERT and XCHANGE mode. If ALTER does not return automatically to main command level upon execution of a command, press the ESC key. (Throughout the manual, main commands and modes are set in uppercase to distinguish them from subcommands, which are set in upper/lowercase; e.g., QUIT Exit.)

To specify a menu selection (i.e., command or mode), press the initial letter of that selection.

NOTE

If you accidentally type CNTL D, control is passed to DEBUG-86. To return to your ALTER session, type G<cr>.

Entering, Changing, and Deleting Text

Before you type text into the file, you must first enter INSERT mode by pressing I.

The word "[insert]" is displayed at the bottom of the screen, indicating that you are now in INSERT mode.

Type in a word but misspell it. To correct the error, press the RUBOUT key. Each time you press RUBOUT, the cursor backs up one column and erases the character. When the erroneous character is erased, simply type the correct characters.

Delete the line you just typed by holding down the CNTL key and typing X. Control X is the DELETE LEFT command; it deletes all text on the line to the left of the cursor. The EOF marker is now back at the beginning of the file.

Now type in the following sentence. Several of the words are deliberately misspelled. Enter text exactly as shown.

```
High-levell languages (Pacal in particular) more<cr>
closely modal the human thought process than low-level<cr>
languages such as assembly language.<cr>
```

The first word in the sentence is misspelled "levell." To correct this error, position the cursor, using the cursor control keys, over the erroneous "l," hold down the CNTL key, and type F. Control F is the CHAR DELETE command. This command deletes the character under the cursor.

The "s" in Pascal has been omitted. To correct this error, position the cursor on the "c" in Pascal and type "s." Text is automatically moved to the right as the "s" is inserted. Hit the ESC key to leave INSERT mode and return to main command level.

The word model is misspelled "modal." To correct the error, enter XCHANGE mode by typing X. Position the cursor over the "a" and type "e." Hit the ESC key to leave XCHANGE mode and return to main command level.

You have learned how to insert text, exchange text, and delete individual characters. Now type the following sentence exactly as shown. Remember first to enter INSERT mode by typing I. Position the cursor below the sentence you just typed, at the beginning of the line.

```
Thus, high-level languages are easier and faster to<cr>
write than low-level languages, since one less less<cr>
translation step is required from concept to code.<cr>
```

Hit the ESC key to leave INSERT mode and return to command level.

Note that the word "less" is typed twice. To correct this error (since it appears at the end of the line), position the cursor on the "l" in the second "less," hold down the CNTL key, and type A. Control A is the DELETE RIGHT command, which deletes all text to the right of the cursor.

Suppose you wish to delete the phrase "from concept to code" from the text. To do this, you must "block" (i.e., delimit) this section from the rest of the text using the BLOCK command.

First, position the cursor over the first character of the section (in this case you want the period to close the sentence, so position the cursor on the space before the "f") and press B for BLOCK. The "at" sign (@) covers the space. Then, position the cursor one character past the end of the section you wish to block; i.e., immediately after the "e" in "code."

When you press B for BLOCK, the menu offers several subcommands: Buffer, Delete, Find, -find, Jump, Put. To delete this section, press D. The section is deleted from the text and the space is closed automatically, as shown here:

```
High-level languages (Pascal in particular) more
closely model the human thought process than low-level
languages such as assembly language.
Thus, high-level languages are easier and faster to
write than low-level languages, since one less
translation step is required.
```

Copying Text

Suppose you wanted to copy this text to create a new file (or add to an existing file). Using the BLOCK command, delimit the text by positioning the cursor over the initial letter and pressing B. Now move the cursor one space past the end of the text and press P for Put. The menu will prompt for an output file. Type in the filename and press ESC or RETURN. The text is copied to the specified file.

If you wanted to copy this section to another part of your file, you would delimit the text using the BLOCK command and specify the Buffer subcommand . (A buffer is a temporary holding space for text.) Then, position the cursor where you want the text to appear and press G for the GET command. This command will prompt for an input file. Hitting the RETURN key or the ESC key "gets" the contents of the buffer and places it at the current cursor position.

Exiting the Editor

To exit from the editor, hit Q for QUIT.

The following prompt appears:

```
---- no input file
Abort        Init        Write
```

Since you probably do not want to save the contents of this practice session, press A.

The menu will prompt:

```
all changes lost? (y or [n])
```

Hit y. The editor exits, all input is deleted, and control is returned to the operating system.

To save this file, you would press W. The editor prompts for a filename. After you type in a filename and press RETURN or ESC, the editor writes the named file to the disk.

This tutorial has demonstrated how to insert, exchange, delete, and copy text. For a more detailed explanation of the above commands and other commands available with ALTER, see Chapters 3 and 4.

Starting an Editing Session

You started the tutorial session by typing RUN ALTER <cr>, without specifying a filename. You can specify a filename at the beginning of an editing session, however. The format of the invocation command is:

```
RUN ALTER [input file [TO output file]] [,other input file [TO other output file]]
[{NOMACRO, NOMR} | {MACRO, MR} [(macro file)]]
```

where:

RUN activates the 8086 execution mode.

ALTER is the command name.

input file names the file you wish to edit. If you do not specify a filename, ALTER creates a new file whose name must be supplied with the QUIT command.

output file names a destination file for the edited text. The file is written when you execute either a QUIT Exit or QUIT Update command.

other input file names the file you wish to edit in the secondary buffer.

other output file names the destination file of the edited text in the secondary file.

{NOMACRO, NOMR} stops ALTER from reading any macro files.

{MACRO, MR} changes the default condition in which ALTER searches for a macro file with the name ALTER.MAC. Type MACRO followed by a filename in parentheses to indicate a macro file that should be read instead of ALTER.MAC.

(*macro file*) names the macro file that should be read instead of ALTER.MAC.

The invocation command can be shortened by using a hyphen (-) instead of a comma (,). For example:

ALTER :F1:BLIP.PLM-LST

is the same as:

ALTER :F1:BLIP.PLM, :F1:BLIP.LST.

NOTE

ALTER should not be invoked under a SUBMIT file.

Ending an Editing Session

The QUIT command (described in Chapter 3) ends the editing session in one of two ways:

- It replaces the old version of the file with the updated version.
- It ignores any changes and leaves the old file unchanged.

How you end an editing session depends on which QUIT subcommand you choose. The QUIT Exit, Update, and Write commands store the updated version of your file. The QUIT Init and Abort commands ignore all changes made to your file. See Chapter 3, QUIT Command, for a complete explanation of QUIT and its options.

File Backup

When you edit an existing file, ALTER never changes the file until you execute the QUIT Exit or QUIT Update command. Thus, you can abandon an editing session with QUIT Abort and nothing is changed. When you end an editing session with a QUIT Exit or QUIT Update command, ALTER creates a copy of the original file on the same disk, with the same filename and an extension of .BAK. Thus, the original version of a file is always available after ending an editing session. You can turn this option off using the SET BAK file command described in Chapter 3. However, the BAK file protects against accidental loss or destruction of your file. Therefore, turning this option off can be dangerous.

CHAPTER 2
THE EDITOR BASICS

Before reading Chapters 3 and 4, you should understand the following editor basics:

- Screen-mode editing
- ALTER display/menu format
- Keyboard
- Cursor
- Beep warning
- Lines and line terminators
- Printing and non-printing characters
- End-Of-File (EOF) Marker
- Tags
- Repeat function (count)
- Buffer

Screen-Mode Editing

ALTER's greatest advantage is its ability to display and verify changes to text as you make them. While moving through a file, you can quickly make changes, insertions, and deletions, verifying them as you go. You can examine a screenful of text, locate the text you wish to change, change it with a function key, and review the next screenful of text. A portion of text is always displayed on the screen.

ALTER Display/Menu Format

The Series III system has a 25-line, 80-column display screen (columns are numbered from 0 to 79). The cursor, a blinking underline, is the reference point for all operations: insert, find, delete, etc. The screen is divided into three sections: the prompt line, the message line, and the text area. See figure 2-1.

Prompt Line

The prompt line is the bottom line of the display. There are three types of prompts: menu prompts, line-edited prompts (i.e., a prompt that requires the user to type more than one character), and yes-no prompts.

When ALTER is first invoked, the menu prompt is displayed and the editor is at main command level. Menu prompts are a partial list of up to eight words that indicate which commands are available. The word "more" indicates that pressing the TAB key will display the next line of prompts. Figure 2-2 shows the three prompt lines available at main command level. (The prompt line disappears when scrolling at main command level, but reappears as soon as scrolling is completed.)

To select the desired command, type the first character of the prompt word. The prompt for a command does not have to be visible if you invoke it with the initial letter of that command; that is, the prompt may be on one of the prompt lines indicated by the word "more."

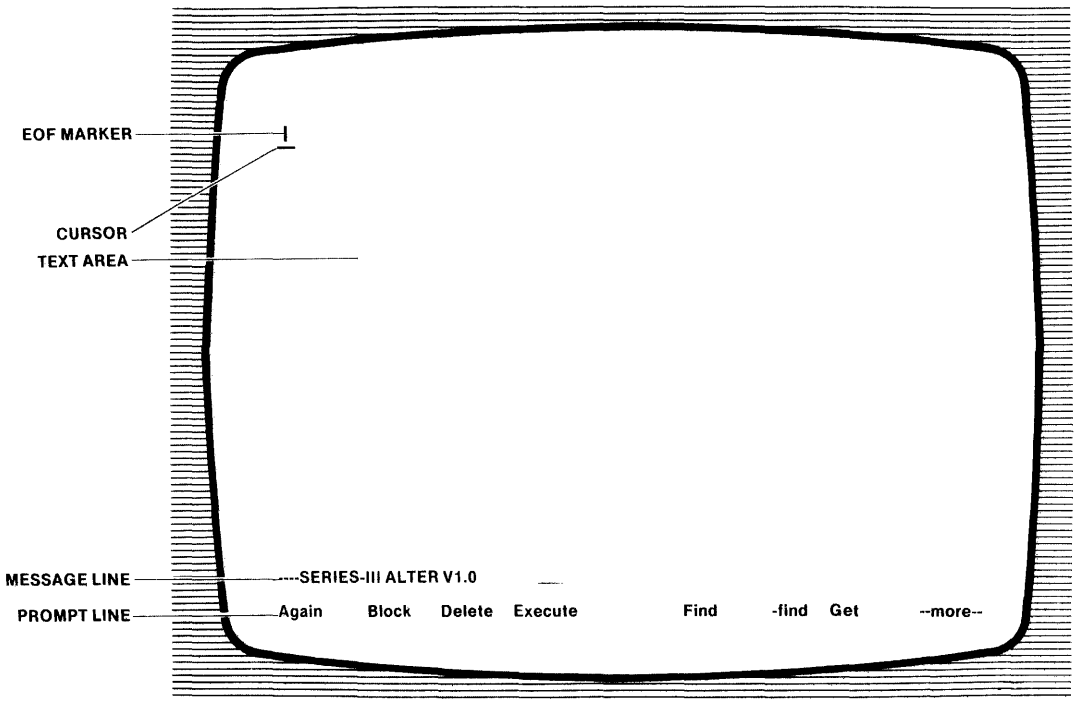

Figure 2-1. ALTER Display

121956-1

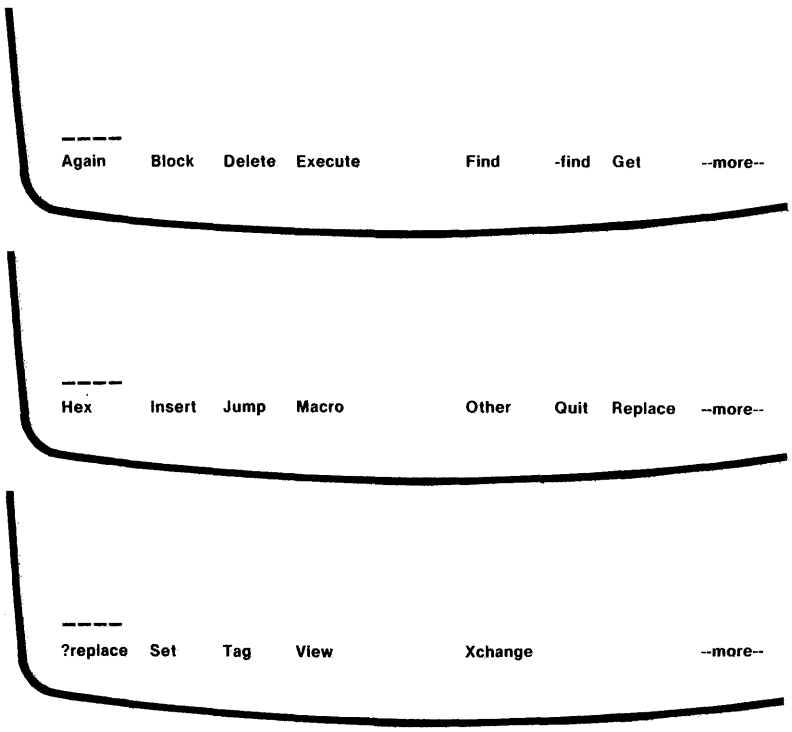

Figure 2-2. Menu Prompt Lines

121956-2

Line-edited prompts ask for information (such as a filename) that requires more than a single character user response. The response can be up to 58 characters.

Yes-no prompts are of the form "prompt" ? (y or [n]), where brackets surround the default. For example:

```
all changes lost? (y or [n])
```

Any response other than y (or Y) is considered a negative response.

Message Line

The message line (directly above the prompt line) is used to display status messages or to indicate command modes. The message line contains a blank in the first column followed by four minus signs. The word "Macro" or "Other," or both, may follow the minus signs to indicate that a macro is being defined or that the secondary buffer is being displayed. This part of the message line never changes unless OTHER or MACRO mode is changed. The remainder of the message line is changeable because it is used for status messages or to indicate the count (repeat function) of a command.

Text Area

The text area comprises the rest of the screen; i.e., the top 23 lines.

Keyboard

The keyboard, shown in figure 2-3, is your interface with the editor. It is a typewriter-style, electronic keyboard that supports the ASCII character set (see Appendix C).

Every keyboard character can be considered a command because every key causes something to happen. Most keys are self-explanatory. Some, however, perform functions rather than enter characters and thus are called function keys.

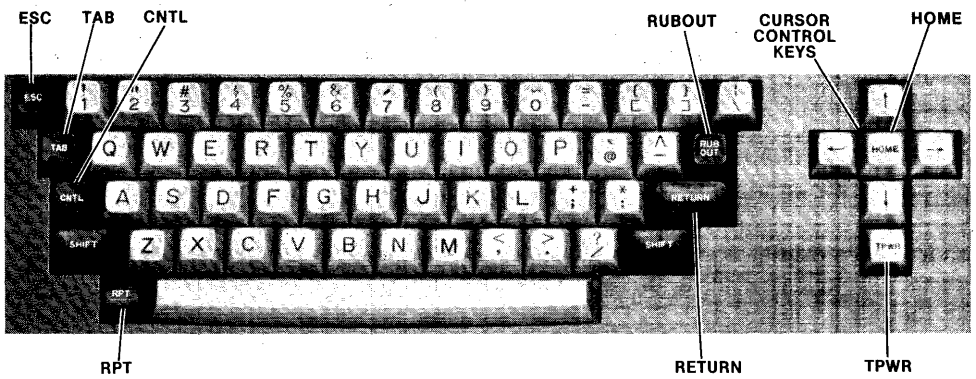

Figure 2-3. Series III Keyboard 121956-3

ARROWS	The four keys labeled with directional arrows are the cursor control keys.
CNTL	The CNTL key changes the function of some keys on the keyboard. For example, to change the function of X, hold down the CNTL key and press X. Control X is the DELETE LEFT command and deletes all text to the left of the cursor in that line.
CNTL C	Control C aborts commands and returns the editor to main command level.
ESC	The ESC (escape) key ends commands and returns the editor to main command level.
HOME	The HOME key works in conjunction with the arrow keys to page backward or forward through a file or to move rapidly to the beginning or end of a line.
RETURN	The RETURN key moves the cursor to the beginning of the next line. The carriage return/line feed character is not displayed on screen. The use of the RETURN key in examples of input lines is indicated by <cr>.
RPT	The RPT (Repeat) key provides multiple entry of other keys. When RPT and a second key are held down, the function of the second key is repeated until the RPT key is released. The RPT key functions with all keys except the CNTL, SHIFT, HOME, and TPWR keys.
RUBOUT	The RUBOUT key deletes the character to the left of the cursor.
TAB	The TAB key rotates the menu prompt line to display the next line of commands.
TPWR	The TPWR key provides lowercase entry (latched position) or uppercase entry (unlatched position) of alphabetic characters. This key functions with the alphabetic keys only.

Cursor

The cursor refers both to the cursor displayed on the screen as a blinking underline (the physical cursor) and to the current file location (the character under the cursor—or logical cursor). The physical and logical cursor diverge only when the physical cursor leaves the text area for message or prompt lines or when the UP or DOWN command (see Chapter 3) moves the cursor to a screen location past the end of a line. When the cursor is located past the end of a line, all commands except the UP and DOWN commands (including TAB or an illegal command) can move it back to the end of the line before executing.

Where the cursor addresses a location in text rather than a character (in INSERT mode, for example), it refers to the space between the character on which it is positioned and the character to the left. Thus, characters are inserted immediately before the character under the cursor.

Beep Warning

The editor beeps when you attempt to do something illegal such as

- Execute an illegal QUIT subcommand
- Type an invalid character during INSERT or XCHANGE mode

Lines and Line Terminators

A line of text consists of a character string terminated by a carriage return/line feed. This pair of characters, called the line terminator, is entered in the file when you press the RETURN key. (RETURN is displayed on screen as a blank.)

If a line is over 80 characters long, an exclamation point (!) is displayed in the last column on the screen and the portion of the line that does not fit is not displayed. To view the portion that does not fit, use the SET Leftcol command (see Chapter 3).

Printing and Non-Printing Characters

All characters are displayed on the screen except illegal characters (hex values less than 20H and RUBOUT), which print as '?'. Carriage return and tab print as blanks.

End-Of-File (EOF) Marker

The EOF marker--a vertical bar (**|**)—indicates the end of a file. If the end of a file is on the screen, the EOF marker is displayed immediately after the last character of the file.

Tags

Tags identify locations within a file. You can specify four locations, A through D, with the TAG command and use them as destinations for the JUMP command.

Repeat Function (Count)

A repeat count is an optional decimal repetition factor in the range 1 to 65535. Count indicates the number of times to repeat a command (some commands ignore count or limit count to 32). You may input a count prior to entering a command letter. A forward slash (/) is accepted as a count and means repeat forever. The default count is one.

Buffer

ALTER has three buffers: the primary buffer, the secondary buffer, and the BLOCK buffer.

The primary buffer is the text area at startup.

The secondary buffer allows you to examine, search, or borrow from another file while editing the primary buffer. The secondary buffer is accessed with the OTHER command (see Chapter 3, "OTHER COMMAND") and has a fixed maximum size of 7k.

The BLOCK buffer is the storage area for text that you move, copy, or delete, using the BLOCK command (see Chapter 3, "BLOCK COMMAND").

CHAPTER 3
EDITING COMMANDS

This chapter describes ALTER's editing commands. Commands are presented so that no command is mentioned before it is described.

Several commands prompt for line-edited input; that is, input that requires you to enter more than one character. For example, the GET command will prompt for an input filename. You must terminate line-edited input by pressing the ESC key or the RETURN key. Exceptions are the FIND, -FIND, REPLACE, and ?REPLACE commands, and INSERT and XCHANGE modes, which you terminate with the ESC key only.

Cursor Movement Commands

The cursor movement commands control cursor movement within a file. LEFT, RIGHT, UP, and DOWN refer to the keys labeled with directional arrows (←, →, ↑, ↓); HOME refers to the key labeled HOME. The following descriptions assume a default count of one.

LEFT

The LEFT command moves the cursor one character to the left.

- If the cursor is on the first character of the file, the command is ignored.
- If the cursor is at the top of the screen at the beginning of the line, the screen is rewritten to display previous lines of text.
- If the cursor is at the beginning of a line, the cursor moves to the last character of the previous line.

This command accepts any count where count multiplies the distance moved.

RIGHT

The RIGHT command moves the cursor one character to the right.

- If the cursor is on the EOF marker, the command is ignored.
- If the cursor is on the last character of the last line on the screen, the screen scrolls.
- If the cursor is on the last character of a line, it moves to the beginning of the next line.

This command accepts any count where count multiplies the distance moved.

UP

The UP command moves the cursor up one line within the same column.

- If the cursor is on the top line of the file, the command is ignored.
- If the cursor is in the top line of the screen, the screen is rewritten to display previous lines of text.

This command accepts any count where count multiplies the distance moved.

DOWN

The DOWN command moves the cursor down one line within the same column.

- If the cursor is on the bottom line of the file, the command is ignored.
- If the cursor is in the last line of text on the screen, the screen scrolls.

This command accepts any count where count multiplies the distance moved.

HOME

The HOME command allows you to page forward or backward through a file or to move rapidly to the end or beginning of a line, depending upon the prior command.

- If DOWN was the prior command, the next page of text is displayed leaving the cursor in the same position as at the start of the command. (You only need to press DOWN once for multiple paging; e.g., DOWN HOME HOME HOME will page three times.)
- If UP was the prior command, the previous page of text is displayed leaving the cursor in the same line as at the start of the command. (You only need to press UP once for multiple paging.)
- If LEFT was the prior command, the cursor moves to the beginning of the line.
- If RIGHT was the prior command, the cursor moves to the end of the line.

Count is only significant in the case of the up and down HOMEs, where it multiplies the distance moved.

Carriage Return Command

The RETURN key moves the cursor to the beginning of the next line.

- If the cursor is on the bottom line of the file, carriage return moves the cursor to the end of the file.
- If the auto indent option is on, carriage return moves the cursor to the first non-blank character of the next line.

This command accepts any count where count multiplies the distance moved.

Delete Commands

The following five commands allow you to delete text. RUBOUT refers to a specifically labeled key; CHAR DELETE, CLEAR LINE, DELETE LEFT and DELETE RIGHT refer to control commands.

CHAR DELETE

The CHAR DELETE command, which you execute by typing CNTL F, deletes the character under the cursor. If a carriage return is deleted, the following line feed (if any) is also deleted because the carriage return/line feed pair counts as a single character.

This command limits count to 32 to prevent accidental destruction of the file.

CLEAR LINE

The CLEAR LINE command, which you execute by typing CNTL Z, deletes the entire line on which the cursor is positioned. The cursor is left in the same column on the line below.

RUBOUT

The RUBOUT key deletes the preceding character. If a line feed is deleted, the preceding carriage return is deleted also.

DELETE LEFT

The DELETE LEFT command, which you execute by typing CNTL X, deletes all characters to the left of the cursor on the line on which the cursor is positioned.

DELETE RIGHT

The DELETE RIGHT command, which you execute by typing CNTL A, deletes all characters to the right of the cursor on the line on which the cursor is positioned.

UNDO Command

The UNDO command, which you execute by typing CNTL U, restores up to 100 characters deleted by the last CLEAR LINE, DELETE LEFT, or DELETE RIGHT command at the current cursor position. Consecutive Control U's will repeat the restoration.

INSERT Mode

INSERT mode allows you to enter text. To insert text, you must first enter INSERT mode by pressing I. To exit INSERT mode and return to main command level, press the ESC key.

INSERT mode is modified if it is preceded by a forward slash (/). All text past the cursor in that line is moved down to avoid distracting flashing on the screen (which results from text getting pushed out as more text is inserted). The text is restored prior to any delete or move subcommand (except RUBOUT) or when the insertion is complete.

Description

Press I; the menu prompts:

```
[insert]
```

Begin typing. The cursor movement commands LEFT, RIGHT, UP, DOWN, and HOME, and the delete commands CHAR DELETE, CLEAR LINE, DELETE LEFT, DELETE RIGHT, and RUBOUT all function as at main command level; a repeat count is not an option in INSERT mode, however.

The ESC key causes the editor to leave INSERT mode and return to main command level.

Control C deletes all text inserted since the beginning of INSERT mode (or the last move command) and returns the editor to main command level.

XCHANGE Mode

XCHANGE mode allows you to type over text. To exchange text, you must first enter XCHANGE mode by pressing X. To exit XCHANGE mode, press the ESC key.

Description

Press X; the menu prompts:

```
[exchange]
```

Move the cursor to any location in text and begin typing; characters are replaced on a one-for-one basis (except carriage return, which is never replaced; instead, the line is extended).

The cursor movement commands, CLEAR LINE, DELETE LEFT, DELETE RIGHT, and CHAR DELETE, work as at main command level.

The RUBOUT key replaces the character to the left of the cursor with the original character (if it has been exchanged) with the following exceptions:

- If the line has been extended, RUBOUT works as at main command level and deletes the character immediately to the left of the cursor.
- If the cursor is at the original replacement location, RUBOUT moves one character to the left, but does not delete the character.

The ESC key returns the editor to main command level.

Control C restores original text (text before it was exchanged) and returns the editor to main command level.

Once you have exchanged text and pressed the ESC key, the RETURN key, or any of the cursor movement commands (LEFT, RIGHT, UP, DOWN, or HOME), changes cannot be revoked. The RETURN key and the cursor movement commands restart XCHANGE mode at a new location. The ESC key exits XCHANGE mode and returns the editor to main command level.

XCHANGE mode has a limit of 100 characters. This limit is the maximum number of characters of original text that can be restored by pressing the RUBOUT key. Therefore, any attempt to replace over 100 characters (without somehow restarting XCHANGE mode) will result in the following error message:

```
Xchange limit is 100
```

FIND Command

The FIND command searches forward from the current cursor position for a string of characters and positions the cursor after the next occurrence of the string. To find a string of characters, press F, type in the target string, and then press the ESC key.

Description

Press F; the menu prompts:

`Find "old string"`

At first, the target string of the previous FIND command (the "*old string*") is displayed within the quotes. Hitting the ESC key before typing a new target string repeats the last find.

To find a new string, simply type over the old target string and press the ESC key.

Note that pressing the RETURN key when specifying a target string inserts a carriage return/line feed into the target string and adds the RETURN key symbol, <cr>, to the prompt line. You must press the ESC key to complete the string specification and execute the FIND command.

The cursor is placed immediately after the next occurrence of the target string. If the string is not found, the message "not found *string*" is displayed in the message line.

The FIND command accepts any count where count indicates the number of times to search for a target string. The search stops after finding the last occurrence of the target string.

-FIND Command

The -FIND command, which you invoke by pressing -, is identical to the FIND command except that this command searches backward from the current cursor position. Also, with the -FIND command, the showfind option is ignored and the cursor is left on the first character of the matched string.

REPLACE Command

The REPLACE command, which you invoke by pressing R, is very similar to the FIND command except that it allows you to replace the old target string with a new string. The REPLACE command also allows you to delete a target string.

Description

Press R; the menu prompts:

`Replace "old string"`

Type in the string to be replaced (if other than the "*old string*") and press the ESC key.

The menu prompts:

`Replace "old string" with "new string"`

Type in the new string and press the ESC key. The next occurrence of the old string is replaced with the new string.

To delete a target string from text, type a space (to replace the "*old string*"), press RUBOUT to remove the space, and then press ESC to terminate the string.

For example:

```
Replace "old string" <ESC> with "<space><RUBOUT>" <ESC>
```

The REPLACE command accepts any count where count indicates the number of times to replace a target string. The replacement stops after replacing the last occurrence of the target string.

?REPLACE Command

The ?REPLACE command, which you invoke by pressing ?, works exactly the same as the REPLACE command except that you are prompted on each find:

```
ok to replace? (y or [n])
```

TAG Command

The TAG command, which you invoke by pressing T, allows you to specify a location within a file and, with the JUMP command (see "JUMP COMMAND" which follows), move the cursor at any time to this location. The editor automatically returns to main command level.

Description

You can set four tags: A, B, C, and D.

The cursor's current position determines the tag location. Position the cursor.

Press T; the menu prompts:

```
A tag     B tag     C tag     D tag
```

Set tag(s) A through D with the corresponding letter.

JUMP Command

The JUMP command, which you invoke by pressing J, moves the cursor to a specified location in text. The editor automatically returns to main command level.

Description

Press J; the menu prompts:

```
Start  End  Line  Position  A tag  B tag  C tag  D tag
```

Start

The Start subcommand moves the cursor to the start of the file.

End

The End subcommand moves the cursor to the end of the file.

Line

The Line subcommand prompts for a decimal value:

```
line:
```

The cursor jumps to the start of the designated line. (The first line of a file is number one.)

Position

The Position subcommand prompts for a decimal value:

```
column:
```

The cursor jumps to the designated column. (Columns start at 0.)

- If the current line is too short, the cursor jumps to the last column.
- If there is no character at the specified column, the cursor jumps to the next character.

A tag B tag C tag D tag

The cursor jumps to the specified tag, which you set with the TAG command.

BLOCK Command

The BLOCK command, which you invoke by pressing B, allows you to delimit a section of text that can then be deleted, moved, or copied, depending on which subcommand you specify. The Buffer subcommand copies the delimited section to the block buffer. The Delete subcommand deletes the delimited section and places it in the block buffer. The Put subcommand copies the section to an external file. Find, -find, Jump, and the cursor movement commands work as at main command level. Text saved in the block buffer can be retrieved with the GET command, which copies the contents of the buffer (or external file) at the current cursor position in your file. (See "GET COMMAND" section later in this chapter.)

Description

To delimit a section of text, move the cursor (as at main command level) to the first character of the block. Press B; the "at" sign (@), which serves as the delimiter, will replace the character, marking one endpoint of the block. (Endpoints can be set in either order, even though this example sets the beginning endpoint first.)

The menu prompts:

```
Buffer     Delete     Find     -find     Jump     Put
```

Move the cursor to the character immediately after the block to be delimited. The character under the first "at" sign (@) is included in the block, but the character under the second is not.

Now specify one of the following subcommands by pressing the initial letter of that subcommand.

Buffer

The Buffer subcommand, which you execute by pressing B, copies text to the block buffer. The text is unchanged, the "at" signs (@) are removed, and the delimited section is copied to the buffer.

Delete

The Delete subcommand, which you execute by pressing D, deletes the section and moves it to the block buffer. If the deleted text does not fit in the portion of the block buffer that is in memory, the menu prompts:

```
cannot save in memory--save anyway? (y or [n])
```

If you specify y, the text that does not fit is written to an unnamed file. (See "BLOCK BUFFER" section later in this chapter.)

Find

The Find subcommand, which you execute by pressing F, works as at main command level.

-find

The -find subcommand, which you execute by pressing -, works as at main command level.

Jump

The Jump subcommand, which you execute by pressing J, works as at main command level.

Put

The Put subcommand, which you execute by pressing P, allows you to copy a block of text to a named output file.

Press P; the menu prompts:

```
Output file:
```

Type in the output filename and press the RETURN key or the ESC key to copy the range to the specified file.

Block Buffer

The block buffer has a fixed maximum size of 2k. If the block is over 2k, the remainder is written to an unnamed (:WORK:) file. The contents of the buffer remain unchanged until you execute another BLOCK command, in which case the new block is inserted in the buffer.

DELETE Command

The DELETE command, which you invoke by pressing D, works exactly the same as the BLOCK command. The subcommands and the manner of execution are identical. The DELETE command exists so that blocks can be deleted by typing D at both endpoints.

GET Command

The GET command, which you execute by pressing G, retrieves the contents of the block buffer (or an external file) and places it at the current cursor position in your file.

Description

Move the cursor to the point in your file where you want the contents of the buffer (or external file) to be placed.

Press G; the menu prompts:

```
Input file:
```

To retrieve the contents of the buffer, press the RETURN key or the ESC key. Otherwise, specify a filename and the file will be copied at the current cursor location.

This command accepts any count.

VIEW Command

The VIEW command, which you execute by pressing V, rewrites the screen leaving the cursor in the viewrow—the row on which you wish the cursor to be positioned by the VIEW command (5 is the default).

OTHER Command

ALTER has two distinct buffers; only one is available for display at a given time. The text area used at startup is called the primary buffer; the other is called the secondary buffer. The OTHER command, which you execute by pressing O, allows you to switch between text buffers and view the new one.

Description

Press O; the word "Other" is displayed at the start of the message line whenever the secondary text is displayed:

```
---- Other
Again  Block  Delete  Execute  Find  -find  Get  --more--
```

This secondary buffer allows you to examine, search, or borrow from another file while editing the primary buffer. Because the secondary buffer has a fixed maximum size of 7k, text that you have already scanned is thrown away in order to display new text.

If text size is greater than 7k and text is lost, the message "some text lost" appears in the message line and the output file is changed to :BB: to avoid inadvertent loss of the file with a QUIT Exit. (:BB: stands for byte bucket--a pseudo-device designated for input or output data that is to be discarded.)

For the same reason, any attempt to edit a file larger than 7k in the secondary buffer results in the message "text does not fit." If the file is too large, it cannot be resaved with a QUIT Exit or QUIT Update command.

AGAIN Command

The AGAIN command, which you execute by pressing A, repeats the last command.

If AGAIN is used after a FIND, -FIND, REPLACE, or ?REPLACE command, the entire command, including arguments, is repeated.

This command accepts any count, which is the count for the repeated command.

SET Command

The SET command, which you invoke by pressing S, allows you to set tabs, set the viewrow (the row on which the cursor is positioned after a VIEW command), and set the left column to display lines over 80 characters long.

The SET command also allows you to change the switches that control several options in the Editor's environment. (A switch is an option that has only two states: yes or no.) Brackets indicate the initial state of each option. This state is changed by typing the alternative that is not set currently.

Description

Press S; the menu prompts:

```
Autocr  BAK file  Case  Indent  Leftcol  Notab  Showfind  --more--
Tabs    Viewrow                                          --more--
```

To specify an option, press the initial letter of that option.

Autocr

The Autocr option prompts:

```
insert cr,lf automatically? (y or [n])
```

- If y, carriage return/line feed is inserted in the last column on the screen whenever an attempt is made to insert a character in that column. Trailing blanks and tabs are deleted and the carriage return/line feed is inserted between words or after a special character.
- If n, the option is turned off.

BAK File

The BAK file option prompts:

```
create .BAK files? ([y] or n)
```

- If y, the file you are editing is renamed file.BAK before the QUIT Exit or QUIT Update replaces it.
- If n, this option is turned off.

Turning this option off can be dangerous because the backup file protects against accidental loss of, or damage to, your file.

Case

The Case option prompts:

```
ignore case of Find target? ([y] or n)
```

- If y, you can type the target string in uppercase, lowercase, or a combination of both, but the case is ignored.
- If n, you can type the target string in uppercase, lowercase, or a combination of both and the case is significant.

For example:

If you type 'tHe' and the state is y, ALTER will find tHe, the, THE, etc.

If you type 'tHe' and the state is n, ALTER will find tHe only.

Indent

The Indent option prompts:

```
automatically indent during insertion? (y or [n])
```

- If y, inserted carriage returns are followed automatically by the same combination of blanks and tabs as the preceding line. This is useful when using a structured language such as PL/M or Pascal. ALTER will delete copied blanks and tabs from null lines.
- If n, this option is turned off.

Leftcol

The Leftcol option prompts:

```
first column:
```

This command allows you to view lines over 80 characters long on the screen and accepts any number from 0 to 80 (columns start at zero). The number input indicates the number of characters at the start of a line that should not be displayed.

For example, if a line is 90 characters long, set the left column to 10 and the screen will display the line from column 10 to column 89.

An exclamation point (!) is printed in column 0 if any characters are not displayed.

Notab

The Notab option prompts:

```
insert blanks for tabs? (y or [n])
```

- If y, blanks are inserted instead of tabs whenever you press the TAB key while in INSERT mode.
- If n, this option is turned off.

Showfind

The Showfind option prompts:

```
list lines on multiple finds? (y or [n])
```

- If y, when you execute a FIND or REPLACE command, all of the lines of text in which any string is found or replaced are listed on the screen.
- If n, the FIND and REPLACE commands execute as usual, but the lines in which any string is found or replaced are not listed on the screen separately.

Tabs

The Tabs subcommand prompts:

```
Tabs:
```

The message line lists the current tab settings. If you only want to inspect the tab settings, type CNTL C to return to main command level.

- If you type in numbers increasing in the range 0 to 158, tabs are set as indicated.
- If the last column is less than 159, the difference between the last two stops is repeated.

For example: 4 sets tabs at 4, 8, 12, 16...

6, 10 sets tabs at 6, 10, 14, 18

NOTE

Columns start at zero—not one. Therefore, FORTRAN tabs should be 6, 10—not 7, 11.

Viewrow

The Viewrow subcommand prompts:

```
row for View:
```

Type the number of the line on which you wish the cursor to be positioned by the VIEW command. To center the cursor on the middle lines, set viewrow to 10 or 11. Legal viewrows are row numbers 1-22.

HEX Command

The HEX command, which you execute by pressing H, allows you to insert the ASCII equivalents of hexadecimal values in the text. This command also displays the hexadecimal values in the message line.

Description

Press H; the menu prompts:

```
Input Output
```

To specify a subcommand, press the initial letter of that subcommand.

Input

Press I; the menu prompts:

```
Hex value:
```

Type the hexadecimal values of the characters you wish to insert in the text at the current cursor location.

Output

Press O.

The count that preceded the Hex command indicates the number of bytes you wish to display in hexadecimal format. Up to 10 bytes of hexadecimal values can be displayed in the message line. If more bytes are to be displayed, the message line states:

```
hit space to continue
```

Examples

Example 1

To add the form feed character (hexadecimal value 0C) to the current location, type:

H(ex) I(input) C <cr>

Example 2

To add the digits 1, 2, and 3 to the text, type:

H(ex) I(nput) 313233 <cr>

Now, back up the cursor (the cursor should cover the one) and type 3 H O to display the characters in the message line.

QUIT Command

The QUIT command, which you invoke by pressing Q, ends the editing session. The QUIT command has several subcommands: Abort, Exit, Init, Update, and Write. QUIT has two prompts, depending on whether or not the file you are editing has a filename already specified.

Filename Specified

Assume you are editing a file with filename already specified.

Press Q; the menu prompts:

```
---- Editing file [to file]
Abort        Exit        Init        Update        Write
```

To specify a subcommand, press the initial letter of that subcommand.

Abort

The Abort subcommand returns control to the operating system and all changes (if any) are lost. However, if any changes have been made, the menu will prompt: "all changes lost? (y or [n])" to avoid inadvertent loss of text. Any response other than y returns the editor to main command level.

Exit

The Exit subcommand replaces the existing file with the newly edited text and ALTER returns control to the operating system.

Init

The Init subcommand allows you to start another editing session without returning to the operating system. If any changes have been made, the menu prompts: "all changes lost? (y or [n])" before initializing the new editing session.

Press I; the menu prompts:

```
enter [file [TO file]]
```

The editing session is restarted with the new file.

Update

The Update subcommand updates your file without returning to the operating system.

Write

The Write subcommand prompts for an output filename and the entire contents of your file are written to the named file without returning to the operating system.

The QUIT prompt is always reissued after an Update or Write subcommand. Press the ESC key or type CNTL C to return to main command level.

Filename Not Specified

If you are editing a new file and have not yet specified a filename, the menu prompt is altered:

```
---- no input file
Abort              Init              Write
```

The subcommands work the same. If you Abort or Init a new file, your current text will be lost.

Other Buffer

Whenever you type a QUIT Abort or QUIT Exit command, the editor checks the other buffer for any changes. If there are changes, the other buffer is displayed and the menu prompts:

```
all changes lost? (y or [n])
```

CHAPTER 4
MACRO COMMANDS

ALTER macros are sequences of ALTER commands (sequences of keystrokes) that have been given a name. They are stored in a separate file. When you call a macro file, all of the macros within the file are then available for execution in your current text file. (See "MACRO FILES" section later in this chapter.)

Macros are typically used for long command sequences that are executed often. Instead of entering a series of commands, you can call the predefined macro, which will then execute the commands automatically. The macro facility speeds your work and reduces the typing errors associated with long command sequences.

MACRO Command

The MACRO command, which you invoke by pressing M, allows you to define macros, retrieve them from memory, and list the names of all currently defined macros in the message line.

Macro definitions are a series of commands written in macro form. Macros can be defined in two ways: interactively—using the MACRO Create command, or directly—by writing macros to your macro file using the MACRO Insert command (this command inserts text in macro form automatically). To save interactively defined macros, you must write them to a separate macro file, in macro form, using the MACRO Save command. Macro files are processed with the MACRO Get command, making the macros within the file available for execution.

Description

Press M; the menu prompts:

```
Create        Get        Insert        List        Save
```

To specify a subcommand, press the initial letter of that subcommand.

Create

The Create subcommand, which you execute by pressing C, "creates" a macro interactively by remembering a sequence of keystrokes.

Press C; the menu prompts:

```
Macro name:
```

Macro names can be either single characters (whose names do not conflict with ALTER commands) or character strings. Macros are executed with the EXECUTE command (see "EXECUTE COMMAND" section later in this chapter); however, single character macros can be executed by simply typing that character.

Type in the macro name followed by ESC or RETURN. The word macro is now displayed at the message line and remains there until the macro definition is complete.

After you type in the macro name, ALTER returns to main command level, but all subsequent keystrokes are "remembered" or "trapped" by the editor. The trapped keystrokes constitute the macro definition, which is terminated by one of the following commands:

Control C	Terminates macro mode without defining the macro.
MACRO command	Terminates the macro definition.

Only 1k of memory is available. If macros exceed this limit, the message "no more room for macros" appears in the message line and the definition is terminated.

Get

The Get subcommand, which you invoke by pressing G, processes macro files, making the macros within the file available for execution.

(ALTER.MAC is the default name for which ALTER will search automatically and process upon invocation; that is, you do not have to "get" ALTER.MAC in order to execute the macros within it. However, you must "get" a macro file with any other filename before you can execute the macros within it.)

Press G; the menu prompts:

```
Macro file:
```

- If you do not type in a filename but only hit the RETURN key, the current text is treated like a macro file.
- If you do type in a filename, the named file is read as a macro file.

Insert

The Insert subcommand, which you execute by pressing I, causes all subsequent input (including RUBOUT) to be inserted in text in macro form (see "MACRO FILES" section later in this chapter). This command is terminated when you type CNTL C.

Press I; the menu prompts:

```
Control C to stop
```

Type in the following line, which defines Control L to mean jump to start of line. (Remember that what you type will not execute, but will be inserted in macro form.)

```
M(acro) <CONTROL> L <ESC> <RIGHT> <LEFT>
<HOME> \EM (end macro) <cr>
```

The following macro definition will be added at the current text location:

```
M\00C\BR\CR\CL\CH\EM
```

Now type CNTL C to terminate MACRO Insert mode.

List

The List subcommand, which you execute by pressing L, lists the names of all currently defined macros in the message line. If the message line is not large enough to

accommodate a list of all the defined macros, the message "hit space" appears on the line. Pressing the space bar will continue the list. Any other command returns the editor to main command level.

Save

The Save subcommand, which you invoke by pressing S, translates interactively defined macros to macro form, so that they can be saved in macro files.

Press S; the menu prompts:

```
Macro name:
```

Type in the macro name followed by ESC or RETURN. If the macro exists, it is placed in the text at the current cursor location in macro form.

Thus, to "save" an interactively created macro, you invoke the macro file using the QUIT Initial command (specifying the macro filename when prompted), translate the macro into macro form using the MACRO Save command, then resave the macro file using the QUIT Update command.

For example, the following sequence of commands will save an interactively defined macro named '*'.

Q(UIT) I(nitial) ALTER.MAC<cr>	Obtains macro file (ALTER.MAC).
M(ACRO) S(ave)	Invokes the MACRO Save command, which prompts for the macro name.
* <cr>	Executes the MACRO Save command, which then translates the macro named '*' to macro form.
Q(UIT) U(pdate)	Updates the macro file (ALTER.MAC), which now includes the macro named '*'.

If the macro does not exist, the message "no such macro" appears in the message line and the editor returns to main command level.

EXECUTE Command

The EXECUTE command, which you invoke by pressing E, calls a macro by name and executes it. Execution means taking all console input (except responses to ?Replace and the "all changes lost" prompts) from the macro text instead of from the console.

Description

Press E; the menu prompts:

```
Macro name:
```

Type in the macro name followed by ESC or RETURN. If the macro exists, it is executed. If it does not exist, the message "no such macro" appears on the message line.

The macro terminates when:

- It has been executed the specified number of times.
- An attempt is made to move forward (RIGHT, DOWN, HOME, or carriage return) at the end of the file.
- An attempt is made to move backwards (UP, LEFT, or HOME) at the start of the file.
- A FIND, -FIND, REPLACE, or ?REPLACE command fails to find the target.

The last three terminations return the editor to main command level. Normal macro termination leaves ALTER in the current mode; for example, if macros are nested (i.e., if a macro contains a macro), only the current macro ends.

Control C terminates all nested macros and returns the editor to main command level.

Macro Files

Macro files may consist of SET commands and macro definitions.

SET commands, when included in a macro file, must be terminated by a semi-colon or carriage return. (SET commands are described in Chapter 3.) When executed, the same error messages are issued as when the SET command is used.

Macro definitions are a series of commands written in macro form; that is, a series of commands written using the following representations of control characters and control codes:

Name	Represents
\BR	ESC
\CU	UP
\CD	DOWN
\CR	RIGHT
\CL	LEFT
\CH	HOME
\XA	DELETE RIGHT
\XF	CHAR DELETE
\XX	DELETE LEFT
\XZ	CLEAR LINE
\NL	CARRIAGE RETURN
\RB	RUBOUT
\0h	Hex value of a character
\EM	End of macro definition

Macros are defined with the following format:

`M` *macro name* `\BR` *characters in macro* `\EM {<cr>,;}`

where

M	declares that a macro definition follows.
macro name	is any name given to the macro being defined.

\BR	stands for <ESC>.
characters in macro	become macro contents.
\EM	signals end of macro.
<cr> or ;	signals end of definition.

Macro Examples

Example 1

When writing a block-structured language, it is often necessary to move a block of code to the right or left to adjust the indentation. The following macros will adjust text where tabs are used for indentation:

```
M>\BR\CUR\NL <TAB> \BR\NL <TAB> <TAB> \BR\CD\EM <cr>

M<\BR\CUR\NL <TAB> \BR\NL\BR\CD\EM <cr>
```

These macros can also be defined interactively using the MACRO Create command. For example, the first macro can be created by entering the commands:

```
M(acro)
C(reate)
>  <cr>
<UP>
R(eplace) "<cr><TAB>" <ESC> with "<cr><TAB><TAB>" <ESC>
<DOWN>
M(to terminate macro definition)
```

After you define these macros, typing a right angle bracket (>) at main command level will add a tab to the next line that contains tabs; typing a left angle bracket (<) will delete a tab from the next line that contains tabs; typing '10>' will indent the next 10 lines.

Example 2

The following macros, named '.' and ',' will move the cursor eight characters to the right and eight characters to the left, respectively.

```
M.\BR8\CR\EM <cr>

M,\BR8\CL\EM <cr>
```

These macros can also be defined interactively using the MACRO Create command:

```
M(acro)
C(reate)
.  or  ,  <cr>
8→(move cursor eight times to the right) or  8←(move cursor eight times to the left)
M(to terminate macro definition)
```

After you define these macros, typing a period (.) at main command level will move the cursor eight characters to the right; typing a comma (,) will move the cursor eight characters to the left.

Example 3

It is often desirable to use visual breakpoints in programs. For example, you can use comment lines filled with hyphens to separate procedures within a language. The following macro will create a break line:

```
M@\BRi(*-------------------------*)\NL\BR\EM <cr>
```

You can also create this macro interactively with the following commands:

```
M(acro)
C(reate)
@  <cr>
I  (*-----------------------------------------*)
M(to terminate macro)
```

After you define this macro, typing '@' at main command level will insert the break line at the current cursor position in text.

APPENDIX A
ALTER COMMAND SUMMARY

This appendix lists the ALTER commands and their format and description. (Within the alphabetic list, the up arrow (↑) represents the CNTL key.) Angle brackets indicate a specifically labeled key outside of the normal keyboard; e.g., <RUBOUT> refers to the key labeled RUBOUT.

Function Keys

The following commands are executed by pressing the specifically labeled key on the keyboard or by typing the indicated control commands.

<CNTL> A	Deletes all characters to the right of the cursor.
<CNTL> C	Aborts command in progress and returns editor to main command level.
<CNTL> F	Deletes the character under the cursor.
<CNTL> X	Deletes all characters to the left of the cursor.
<CNTL> Z	Deletes the entire line on which the cursor is positioned.
<ESC>	Terminates commands and returns editor to main command level.
<RETURN>	Moves to start of next line.
<RUBOUT>	Deletes character to left of cursor.
<TAB>	Rotates the menu prompt line to display the next line of commands.
To move cursor:	Use keys labeled with directional arrows on keyboard. Use HOME key in conjunction with arrow keys for fast cursor movement.

ALTER Editing Commands

The following is an alphabetic list of the ALTER editing commands and their format and description.

Command	Format	Function
AGAIN	A	Repeats the last command.
BLOCK	B	Delimits section of text that can then be deleted, moved, or copied.
Buffer	B	Copies text to the block buffer.
Delete	D	Moves text to the block buffer.

Find	F	Searches forward from current cursor position for string. Moves cursor if found.
-find	-	Searches backward from current cursor position for string. Moves cursor if found.
Jump	J	Moves cursor to specified location in text.
Put	P	Copies text to named output file.
CHAR DELETE	↑F	Deletes the character under the cursor.
CLEAR LINE	↑Z	Deletes the entire line on which the cursor is positioned.
DELETE	D	Delimits section of text that can then be deleted, moved, or copied. (Same as BLOCK command.)
DELETE LEFT	↑X	Deletes all characters to the left of the cursor.
DELETE RIGHT	↑A	Deletes all characters to the right of the cursor.
FIND	F	Searches forward from current cursor position for string. Moves cursor if found.
-FIND	-	Searches backward from current cursor position for string. Moves cursor if found.
GET	G	Retrieves contents of block buffer or external file; places contents at current cursor position.
HEX	H	
Input	I	Inserts ASCII equivalent of hexadecimal values in text.
Output	O	Displays hexadecimal values of ASCII characters in message line.
INSERT	I	Begins INSERT mode: inserts text at cursor.
JUMP	J	Moves cursor to a specified location in text.
Start	S	Moves cursor to the start of the file.
End	E	Moves cursor to the end of the file.
Line	L	Moves cursor to the start of the designated line.
Position	P	Moves cursor to the designated column.
A tag	A	Moves cursor to A tag.
B tag	B	Moves cursor to B tag.
C tag	C	Moves cursor to C tag.
D tag	D	Moves cursor to D tag.
OTHER	O	Switches between primary and secondary buffers.

QUIT	Q	Ends editing session.
Abort	A	Returns to operating system; all changes are lost.
Exit	E	Returns to operating system; the file is updated.
Init	I	Restarts editing session; all changes are lost; initializes new file without returning to operating system.
Update	U	Updates file without returning to operating system.
Write	W	Writes file to a new named output file without returning to operating system.
REPLACE	R	Searches for target string; replaces it with new string if found.
?REPLACE	?	Queries user on each find before replacing string.
SET	S	Sets tabs, viewrow, and left column. Controls several yes/no options in editor's environment. (Note: The word default, which appears in parenthesis in the following descriptions, indicates the initial state of an option.)
Autocr	A	While in INSERT mode, inserts carriage return in text automatically when line is full.
BAK file	B	Creates backup file of file you are currently editing before QUIT Update or QUIT Exit command replaces it. (Default)
Case	C	Tells editor to ignore case of strings during search and replace commands. (Default)
Indent	I	Indents text automatically.
Notab	N	Inserts blanks in place of tabs.
Showfind	S	Lists lines of multiple finds.
Tabs	T	Sets tabs.
Leftcol	L	Sets left column.
Viewrow	V	Sets row to which cursor will move on VIEW command.
TAG	T	Specifies locations within a file.
UNDO	↑U	Restores characters deleted by last CLEAR LINE, DELETE LEFT, or DELETE RIGHT command at current cursor position.
VIEW	V	Rewrites screen leaving cursor in viewrow (5 is default).
XCHANGE	X	Enters XCHANGE mode: replaces characters on a one-for-one basis.

ALTER Macro Commands

The following is an alphabetic list of the ALTER macro commands and their format and description.

EXECUTE	E	Executes specified macro.
MACRO	M	Creates macros, retrieves them from memory, and inserts them in text.
Create	C	Creates macros interactively by remembering sequence of keystrokes.
Get	G	Retrieves macros from external file.
Insert	I	Inserts subsequent input in text in macro form.
List	L	Lists names of all currently defined macros in message line.
Save	S	Saves macros in form suitable for macro files.

APPENDIX B
ALTER ERROR MESSAGES

This appendix lists the error messages reported by ALTER when a problem is encountered in a command.

Editing Command Errors

Message	Explanation
bad tabs	Attempt to set bad tabs; e.g., 4,2. Editor returns to main command level.
bad Leftcol	Attempt to set bad left column; e.g., x. Editor returns to main command level.
bad Viewrow	Attempt to set bad viewrow; e.g., 45. Editor returns to main command level.
cannot delete more than 32	Attempt to use count greater than 32 with CHAR DELETE command. Editor returns to main command level.
illegal command	Attempt to enter illegal command. Editor ignores command.
invalid hex value	Attempt to input an invalid hex value. Editor returns to main command level.
no such macro	Attempt to execute macro that does not exist. Editor returns to main command level.
not found: "*target string*"	Target string not found. Editor returns to main command level.
some text lost	Attempt to increase contents of secondary buffer past 7k. Output file changed to :BB: to avoid inadvertent loss of file with a QUIT Exit.
text does not fit	Attempt to edit a file larger than 7k in secondary buffer. Editor returns to main command level.
filename <error message supplied by operating system>	An error occurs during a QUIT Exit, QUIT Update, GET, or BLOCK Put command. Editor returns to main command level.
Xchange limit is 100	Attempt to replace over 100 characters without restarting XCHANGE mode. Editor remains in XCHANGE mode.

Macro File Errors

If any error is found in a macro file, one of the following messages is printed (where *nnn* is the line of the macro file that contains the error) and macro file processing continues.

Message	Explanation
Error In *nnn* no more room for macros	Attempt to create a macro when macro buffer is full. Macro definition is terminated.
Error in *nnn* bad hex value	Macro definition contains bad hex value; e.g., 3G.
Error in *nnn* bad '\' code	Backslash (\) is not followed by a valid value.
Error in *nnn* missing ';'	SET command not terminated with a semicolon or carriage return.
Error in *nnn* bad AV value bad AX value bad AF value	Configuration command contains bad AV, AX, or AF value.
Error in *nnn* bad command	Macro definition contains a bad command—unknown control code or character; i.e., not M or S.
Error in *nnn* no macro name	Macro definition does not include macro name.

Invocation Errors

File Does Not Exist: *filename*	Attempt to specify *output file* while *input file* does not exist. Editor returns to main command level.
Insufficient Memory	ALTER does not have a large enough RAM partition. Editor exits to operating system.
Illegal Invocation Line	Invocation line contains unknown option. This error is not fatal (i.e., editor does not exit to operating system).

APPENDIX C
ASCII CHARACTER SET

ASCII CHARACTER	HEX	ASCII CHARACTER	HEX
NUL	00	@	40
SOH	01	A	41
STX	02	B	42
ETX	03	C	43
EOT	04	D	44
ENQ	05	E	45
ACK	06	F	46
BEL	07	G	47
BS	08	H	48
HT	09	I	49
LF	0A	J	4A
VT	0B	K	4B
FF	0C	L	4C
CR	0D	M	4D
SO	0E	N	4E
SI	0F	O	4F
DLE	10	P	50
DCI	11	Q	51
DC2	12	R	52
DC3	13	S	53
DC4	14	T	54
NAK	15	U	55
SYN	16	V	56
ETB	17	W	57
CAN	18	X	58
EM	19	Y	59
SUB	1A	Z	5A
ESC	1B	[	5B
FS	1C	\	5C
GS	1D	]	5D
RS	1E	^ (↑)	5E
US	1F	_	5F
space	20	`	60
!	21	a	61
"	22	b	62
#	23	c	63
$	24	d	64
%	25	e	65
&	26	f	66
'	27	g	67
(	28	h	68
)	29	i	69
*	2A	j	6A
+	2B	k	6B
,	2C	l	6C
-	2D	m	6D
.	2E	n	6E
/	2F	o	6F
0	30	p	70
1	31	q	71
2	32	r	72
3	33	s	73
4	34	t	74
5	35	u	75
6	36	v	76
7	37	w	77
8	38	x	78
9	39	y	79
:	3A	z	7A
;	3B	{	7B
<	3C	|	7C
=	3D	}	7D

APPENDIX D
CONFIGURING ALTER FOR NON-INTEL TERMINALS

Configuration Commands

ALTER is designed to run on an Intellec Series III development system. The codes expected by the editor from the terminal or sent to the terminal are those used by the Intel terminals.

You can, however, configure ALTER to operate with other terminals. Configuration files are needed when using a non-standard or non-Intel terminal with characteristics different from those of the Series III screen. Configuration files allow you to identify characteristics of your particular terminal by setting various parameters and specifying control sequences by which various screen functions can be performed. Configuration files are not needed when using a Series III terminal.

Configuration commands should be inserted in a macro file, e.g., ALTER.MAC, so that they are automatically executed when ALTER is invoked. The configuration commands enable you to modify certain keyboard and CRT codes.

To create an ALTER configuration file, compare your terminal's behavior to the actions expected by ALTER. Refer to your user manual for the codes that your terminal expects and generates. (See table D-1 for a list of the ALTER configuration commands, their default values, and meaning.)

ALTER requires that the terminal meet the following conditions:

- ASCII codes 20H through 7EH display some symbol that requires one column space. Carriage return (0DH) and linefeed (0AH) perform their usual functions.
- The following cursor functions have cursor key output codes and CRT cursor output codes: down, home, left, right, and up. Output codes for clear screen, clear rest of screen, clear line, clear rest of line, and direct cursor addressing are desirable, but not required. The default codes, shown in table D-1, can be changed with the configuration commands.
- The terminal accepts a blankout code that blanks out the former contents of the screen location to which it is output. The default, 20H, can be changed with the configuration commands.
- The CRT has 22 to 25 lines. The default, 25 lines, can be changed with the configuration commands.
- ALTER automatically generates a linefeed each time a carriage return is entered. Your terminal should not generate a linefeed with a carriage return. In some terminals, this feature can be switched on and off.

When configuring to execute on a non-Intel terminal, you may have to change some or all of the codes assigned to the following configuration commands:

- The cursor key output codes expected by the editor — AFCH, AFCU, AFCD, AFCR, and AFCL.
- The editor generated cursor movement codes sent to the CRT — AFMH, AFMU, AFMD, AFMR, AFML.
- The erase screen code — AFES.

- The blankout code — AFBK.
- The screen size code — AV.
- The BREAK character code — AB.
- The codes expected by the editor for the screen mode commands — AFXA, AFXF, AFXX, AFXU, and AFXZ. You may want to change these codes to match function keys or other convenient keys on the terminal keyboard.

Table D-1 lists the configuration commands, their default values, and their meaning.

Table D-1. Configuration Commands

Command	Series III Default	Meaning
AV=n	25	Sets the number of lines of the display.
AB=hhhh	1BH	Sets ESC.
AR=hhhh	7FH	Sets RUBOUT.
AFXA=hhhh	1H	Sets DELETE RIGHT. CONTROL A.
AFXF=hhhh	6H	Sets CHAR DELETE. CONTROL F.
AFXU=hhhh	15H	Sets UNDO. CONTROL U.
AFXX=hhhh	18H	Sets DELETE LEFT. CONTROL X.
AFXZ=hhhh	1AH	Sets CLEAR LINE. CONTROL Z.
AFCD=hhhh	1CH	Sets DOWN.
AFCH=hhhh	1DH	Sets HOME.
AFCL=hhhh	1FH	Sets LEFT.
AFCR=hhhh	14H	Sets RIGHT.
AFCU=hhhh	1EH	Sets UP.
AFIG=h		This character will be ignored if input. This character is needed on terminals, such as the Hazeltine 1510, which have multiple character key codes for UP and DOWN. AFIG should be set to the lead in (tilde) and UP and DOWN should be set to the second letter of the cursor up or down key code. This avoids problems caused by the lack of a typeahead buffer.
AFMB=hhhh	0DH	Moves cursor to start of line.
AFMD=hhhh	1CH	Moves cursor down.
AFMH=hhhh	1DH	Moves cursor home.
AFML=hhhh	1FH	Moves cursor left.
AFMR=hhhh	14H	Moves cursor right.
AFMU=hhhh	1EH	Moves cursor up.
AFES=hhhh	1B45H	Erases entire screen.

Table D-1. Configuration Commands (Cont'd.)

Command	Series III Default	Meaning
AFER=hhhh	1B4AH	Erases rest of screen.
AFEK=hhhh	1B4BH	Erases entire line.
AFEL=hhhh		Erases rest of line.
AFAC=hhhh		Addresses cursor lead in. When used, code will be followed by column number (0 to 79) and row number (0 to 24).
AO=h	0H	Offset to add both row and column number with address cursor command.
AX=T or F	T	True if X (column) precedes Y (row) in address cursor command.
AW=T or F	T	Allows user to indicate that terminal wraps when character is printed in column 80.
AFIL=hhhh		Inserts line code. Used in line 0 for reverse scrolling.
AFDL=hhhh		Deletes line code. Used to speed up display on the Hazeltine 1510 and similar terminals.
AFBK=h	20H	Blankout character. <BLANK> on most terminals.

n must be 22, 23, 24, or 25.

h is a one byte hex number.

hhhh is a one to four byte hex number. A null value indicates that the function is not available.

T is 'T' or 't' indicating true.

F is 'F' or 'f' indicating false.

The above commands must be terminated by a semicolon (;) or a carriage return.

Tested Configurations

This appendix contains tested configurations for several non-Intel terminals. The terminals presented here are not the only ones on which you can use ALTER; they are merely the ones that have been tested. The following sections list the configuration functions and values required to run ALTER on the Intel tested terminals. The terminals are:

- ADDS Regent 200
- Beehive Mini-Bee
- DEC VT52
- DEC VT100
- Hazeltine 1510
- Lear Seigler ADM-3A

The commands to configure ALTER for the tested terminals are included, on disk, with the ALTER program. The name of the file is included in each table.

ADDS Regent Model 200

This ADDS model has a 24 line CRT display with 80 characters per line. Each character is formed in an 8x8 dot matrix as a dark character on a light background. The 25th line of the screen displays the operating condition of the terminal.

Function Code	Hexadecimal Value	Graphic or ASCII Name
CD	0A	Line Feed
CH	01	SOH
CL	15	NAK or BS
CR	06	ACK
CU	1A	SUB
MD	0A	Line Feed
MH	1B 59 20 20	
ML	15	NAK or BS
MR	06	ACK
MU	1A	SUB
AC	1B 59	ESC Y
EK	not available	
ER	1B 6B	ESC K
ES	0C	FF
XA	14	DC4
XU	16	SYN
AO	20	SP
AX	F	
XF	1B 45	ESC E
XZ	1B 6C	ESC I
AB	5C	\
AV		24

Command file: ADDS.MAC

```
AFCD=0A; AFCL=15; AFCR=06; AFCU=1A; AFCH=01;
AFMD=0A; AFML=15; AFMR=06; AFMU=1A; AFMH=1B 59 20 20;
AFEK=; AFER=1B 6B; AV=24;
AFXA=14; AFES=0C; AFXU=16;
AFER=1B6B; AFAC=1B 59; AO=20; AX=F;
AFXF=1B 45; AFXZ=1B 6C; AB=5C;
```

NOTE: DEL CHAR must be typed instead of Control F for delete character. DEL LINE must be typed instead of Control Z for delete line. Control T must be typed instead of Control A for delete right. Backslash (\) must be typed instead of escape. Control V must be typed instead of Control U for Undo.

Beehive Mini-Bee

This Beehive terminal can be formatted to display either 12 or 25 lines of 80 characters per line. Only the 25 character format is usable with ALTER. Each character is generated in a 5x7 dot matrix. The maximum transmission rate for this terminal is 9600 baud. Note that the ESCAPE character has to be changed so that the default ESCAPE code can be used; the choice of the ↑K is totally personal preference.

Function Code	Hexadecimal Value	Graphic or ASCII Name
CD	1B 42	ESC B
CH	1B 48	ESC H
CL	1B 44	ESC D
CR	1B 43	ESC C
CU	1B 41	ESC A
MD	1B 42	ESC B
MH	1B 48	ESC H
ML	1B 44	ESC D
MR	1B 43	ESC C-
MU	1B 41	ESC A
EL	1B 4B	ESC K
ER	1B 4A	ESC J
B	0B	↑K
AV		24

Command file: MICROB.MAC

```
AFCU=1B 41; AFCD=1B 42; AFCR=1B 43; AFCL=1B 44; AFCH=1B 48;
AFMU=1B 41; AFMD=1B 42; AFMR=1B 43; AFML=1B 44; AFMH=1B 48;
AFEL=1B 4B; AFER=1B 4A;
AB=0B; AV=24;
```

NOTE: Control K must be typed instead of escape.

DEC VT52

This terminal displays 24 lines of 80 characters per line. The characters are generated in a 7x9 dot matrix. The maximum transmission rate is 19.2K baud. Note that the ESCAPE character has to be changed so that the default ESCAPE code can be used; the choice of Control K (↑K) is totally a personal preference. This terminal does not have a HOME key. The choice of Control O (↑O) for the HOME function is totally a personal preference.

Function Code	Hexadecimal Value	Graphic or ASCII Name
CD	1B 42	ESC B
CH	0F	↑O
CL	1B 44	ESC D
CR	1B 43	ESC C
CU	1B 41	ESC A
MD	1B 42	ESC B
MH	1B 48	ESC H
ML	1B 44	ESC D
MR	1B 43	ESC C
MU	1B 41	ESC A
AC	1B 59	ESC Y
W	F	
AO	20	SP
AX	F	
EL	1B 4B	ESC K
ER	1B 4A	ESC J
ES	not available	
EK	not available	
AV		24
B	0B	↑K

Command file: VT52.MAC

```
AFCU=1B 41; AFCD=1B 42; AFCR=1B 43; AFCL=1B 44; AFCH=0F;
AFMU=1B 41; AFMD=1B 42; AFMR=1B 43; AFML=1B 44; AFMH=1B 48;
AFES=; AFER=1B 4A; AFEL=1B 4B; AFEK=;
AB=0B; AV=24;
AFAC=1B 59; AO=20; AX=F; AW=F;
```

NOTE: Control K must be typed instead of escape. Control O must be typed instead of home.

DEC VT100

This terminal can be formatted with 14 lines of 132 characters per line or 24 lines of 80 characters per line. Only the 24 line format is compatible with ALTER. The characters are generated in a 7x9 dot matrix. The maximum transmission rate is 19.2K baud. You may choose between the DEC VT52 compatible and the ANSI standard (X3.41-1974, X3.64-1977) compatible terminal escape sequences for cursor control and screen erase functions. The ANSI codes are given in the following table. See the DEC VT52 description for the VT52 codes. Note that the ESCAPE character has to be changed so that the default ESCAPE code can be used; the choice of Control K (↑K) is totally a personal preference. This terminal does not have a HOME key. The choice of Control O (↑O) for the HOME function is totally a personal preference.

Function Code	Hexadecimal Value	Graphic or ASCII Name
CD	1B 42	ESC B
CH	0F	↑O
CL	1B 44	ESC D
CR	1B 43	ESC C
CU	1B 41	ESC A
MD	1B 5B 42	ESC [B
MH	1B 5B 48	ESC [H
ML	1B 5B 44	ESC [D
MR	1B 5B 43	ESC [C
MU	1B 5B 41	ESC [A
EK	1B 5B 30 4B	ESC [0 K
ER	1B 5B 30 4A	ESC [0 J
ES	not available	
EL	1B 5B 4B	
W	F	
AV		24
B	0B	↑K

Command file: VT100.MAC

```
AFCU=1B 41; AFCD=1B 42; AFCR=1B 43; AFCL=1B 44; AFCH=0F;
AFMU=1B 5B 41; AFMD=1B 5B 42; AFMR=1B 5B 43; AFML=1B 5B 44;
AFMH=1B 5B 48;
AFES = ; AFER=1B 5B 30 4A; AFEK=1B 5B 30 4B; AFEL=1B 5B 4B;
AB=0B; AV=24;
AW=F;
```

NOTE: Control K must be typed instead of escape. Control O must be typed instead of home.

Hazeltine 1510

This terminal displays 24 lines of 80 characters per line. The characters are generated in a 7x10 dot matrix. The maximum transmission rate is 19.2K baud. You may choose between the ESC or the tilde character (~) as the control sequence lead-in. It is advisable to the use the tilde; if you use the ESC, you must change the BREAK character.

Function Code	Hexadecimal Value (~ Lead-In)	Graphic or ASCII Name
CD	0B	~ VT
CH	12	~ DC2
CL	08	~ BS
CR	10	~ DLE
CU	0C	~ FF
MD	7E 0B	~ VT
MH	7E 12	~ DC2
ML	08	~ BS
MR	10	~ DLE
MU	7E 0C	~ FF
MB	0D	
AC	7E 11	~ DCI
EK	not available	
ER	7E 18	~ CAN
ES	not available	
EL	7E 0F	
XP	0F	SI
IL	7E 1A	~ SUB
DL	7E 13	~ DC3
AV		24

Command file: 1510T.MAC

```
AV=24; AFIG=7E;
AFCU=0C; AFCD= 0B; AFCR=10; AFCL=8; AFCH=12;
AFMU=7E0C; AFMD=7E0B; AFMR=10; AFML=8; AFMH=7E12;
AFMB=0D; AFES= ; AFER=7E18; AFEK= ; AFEL=7B0F;
AFAC=7E11; AFIL=7E1A; AFDL=7E13;
```

Function Code	Hexadecimal Value (ESC Lead-In)	Graphic or ASCII Name
CD	0B	ESC VT
CH	12	ESC DC2
CL	08	ESC BS
CR	10	ESC DLE
CU	0C	ESC FF
MD	1B 0B	ESC VT
MH	1B 12	ESC DC2
ML	08	ESC BS
MR	10	ESC DLE
MU	1B 0C	ESC FF
MB	0D	
EK	not available	
ER	1B 18	ESC CAN
ES	not available	
EL	1B 0F	
IL	1B 1A	ESC SUB
DL	1B 13	ESC DC3
XP	0F	SI
AC	1B 11	ESC DC1
AV		24
B	7E	~

Command file: 1510E.MAC

```
AV=24; AB=7E; AFIG=1B;
AFCU=0C; AFCD=0B; AFCR=10; AFCL=8; AFCH=12;
AFMU=1B0C; AFMD=1B0B; AFMR=10; AFML=8; AFMH=1B 12;
AFMB=0D; AFES= ; AFER=1B18; AFEK= ; AFEL=1B0F;
AFAC=1B11; AFIL=1B1A; AFDL=1B13;
```

NOTE: Tilde must be typed instead of escape.

Lear Siegler ADM-3A

This terminal displays 24 lines of 80 characters per line. The characters are generated in a 5x7 dot matrix. The maximum transmission rate is 19.2K baud.

Function Code	Hexadecimal Value	Graphic or ASCII Name
CD	0A	LF
CH	1E	RS
CL	08	BS
CR	0C	FF
CU	0B	VT
MD	0A	LF
MH	1E	RS
ML	08	BS
MR	0C	FF
MU	0B	VT
EK	not available	
ER	not available	
ES	1A	SUB
AX	F	
AO	20	SP
AC	1B 3D	ESC =
AV		24

Command file: LEAR.MAC

```
AFCU=0B; AFCD=0A; AFCR=0C; AFCL=08; AFCH=1E;
AFMU=0B; AFMD=0A; AFMR=0C; AFML=08; AFMH=1E;
AV=24; AFES=1A; AFER= ; AFEK= ; AFAC=1B 3D;
AX=F; AO=20;
```

INDEX

Within this index, *f* or *ff* after a page number means *and the following page* (or *pages*).

REQUEST FOR READER'S COMMENTS

Intel's Technical Publications Departments attempt to provide publications that meet the needs of all Intel prc users. This form lets you participate directly in the publication process. Your comments will help us correct improve our publications. Please take a few minutes to respond.

Please restrict your comments to the usability, accuracy, readability, organization, and completeness of this p cation. If you have any comments on the product that this publication describes, please contact your Intel re sentative. If you wish to order publications, contact the Intel Literature Department (see page ii of this manual)

1. Please describe any errors you found in this publication (include page number).

2. Does the publication cover the information you expected or required? Please make suggestions for impr ment.

3. Is this the right type of publication for your needs? Is it at the right level? What other types of publications needed?

4. Did you have any difficulty understanding descriptions or wording? Where?

5. Please rate this publication on a scale of 1 to 5 (5 being the best rating).

NAME _____ DATE _____

TITLE _____

COMPANY NAME/DEPARTMENT _____

ADDRESS _____

CITY _____ STATE _____ ZIP CODE _____

(COUNTRY)

Please check here if you require a written reply. ☐

'E'D LIKE YOUR COMMENTS ...

ıis document is one of a series describing Intel products. Your comments on the back of this form will ɜlp us produce better manuals. Each reply will be carefully reviewed by the responsible person. All ›mments and suggestions become the property of Intel Corporation.

NO POSTAGE
NECESSARY
IF MAILED
IN U.S.A.

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1040 SANTA CLARA, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Intel Corporation
Attn: Technical Publications M/S 6-2000
3065 Bowers Avenue
Santa Clara, CA 95051

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 (408) **987-8080**

Printed in U.S.A.