intel®

ISIS CREDIT™ CRT-BASED TEXT EDITOR USER'S GUIDE

Order Number: 9800902-03

ISIS CREDIT™ CRT-BASED TEXT EDITOR USER'S GUIDE

Order Number: 9800902-03

A905/283/10K/J

REV.	REVISION HISTORY	DATE
-01	Original Issue	
-02	Revised to cover changes caused by CREDIT V2.0. Refer to Appendix F.	
-03	Revised to cover changes caused by CREDIT V2.1. Revised to show applicability to other Intel Development Systems	10/82

PREFACE

The ISIS CREDIT (CRT-Based Text Editor) User's Guide describes the operation and use of CREDIT. Since CREDIT runs under the Intel ISIS operating system, this manual assumes familiarity with ISIS.

- Chapter 1 "Introduction", outlines what text editing is, lists CREDIT's features, and discusses CREDIT's two editing modes.
- Chapter 2 "Getting Started", describes the screen mode editing functions, and includes a tutorial session to familiarize the user with CREDIT basics.
- Chapter 3 "Intermediate Editing", covers the command line mode commands. The tutorial session shows how to edit a text file from command line mode.
- Chapter 4 "Advanced Techniques", describes the editing functions supporting macros, multiple iteration of commands, conditional execution, command and side files, and commands to alter CREDIT to fit particular applications. The tutorial sessions in this chapter cover most of these editing functions.
- Appendix A, "Configuring CREDIT for Non-Intel Terminals", describes the process of configuring the editor for different terminals and gives tested configurations for specific terminals.
- Appendix B "CREDIT Editing Command Summary", lists the CREDIT commands and syntax.
- Appendix C, "CREDIT File Usage", describes the ISIS files used by CREDIT.
- Appendix D "CREDIT Error Messages", lists the error messages issued by CREDIT and describes the causes.
- Appendix E, "ASCII Codes", lists the ASCII codes with their hexadecimal and decimal values.
- Appendix F, "Changes in CREDIT V2.1", describes the enhancements and changes made in Version 2.1 of CREDIT

Notation Conventions

The following notation is used to describe the command language of CREDIT.

BOLDFACE	indicates user entry to CREDIT.
UPPERCASE	Characters shown in upper case must be entered exactly as shown. Uppercase is used to denote command keywords. For example: CREDIT pathname1 TO pathname2
<class name>	angle brackets denote general terms that must be replaced by a specific member of the class referenced. For example, filename is replaced by a valid ISIS filename.

[<option>]	Brackets enclose optional material that may or may not be included on the command line. For example, TO pathname2 is an optional item that may be appended to the CREDIT command if certain actions are desired.
$\left\{\begin{matrix}\text{<item>}\\\text{<item>}\end{matrix}\right\}$	Braces indicate that one and only one of the enclosed entries must be selected. If the items are also enclosed by brackets, they are optional and no choice is required. For example, $\left\{\begin{matrix}\text{Y}\\\text{N}\end{matrix}\right\}$
punctuation	Punctuation other than braces and brackets must be entered as shown. For example, the commas and parentheses in the following command must be entered:

```
SUBMIT<command name>(<parm1>,<parm2>,<parm3>)
```

Related Publications

For operating system information pertaining to your system, see the following manuals:

For the Intellec Microcomputer Development system 800 or Series II, see the *ISIS-II User's Guide,* order number 9800306.

For the Intellec Series III, see the *Intellec Series III Microcomputer Development System Console Operating Instructions,* order number 121609.

For the Intel Personal Development System (iPDS), see the *Intel Personal Development System User's Guide,* order number 162606.

CONTENTS

CONTENTS (continued)

FIGURES

TABLES

CHAPTER 1 INTRODUCTION

CREDIT is a text editor that runs under the control of the ISIS operating system on any Intellec Microcomputer Development System with 64K bytes of memory. CREDIT lets the user display a file, move the cursor to any point in the text, make insertions, deletions, or other corrections, and see the results immediately. CREDIT also has a set of commands for the more complex editing functions such as move, copy, command iterations, macro definition, and external file operations. With these commands the modified line or lines are not displayed unless requested.

This chapter describes the operation of CREDIT. A text editor is a program that aids in creating and modifying text files. Text files are files containing alphanumeric characters, i.e., each byte in the file is interpreted as a character according to the ASCII code. The byte values and corresponding characters for ASCII codes are in Appendix C.

The CREDIT text editor provides a set of subcommands, called editing commands, that allow the user to enter text by typing characters at the keyboard. The text is stored in a file that can later be modified by CREDIT editing commands or can be processed by other commands. For example, if the text file contains the source code for a program, a language translator can process it to create machine code.

CREDIT also includes a HELP command that displays the format and a brief functional description of each command. The HELP feature reduces the need to turn to this manual for routine information.

Text Editing and The Development Task

Text editing plays an important role in the software development effort. Source programs for compilers and assemblers are initially created and later modified as text files. The text file containing the source program, referred to as the source file, is then processed by the appropriate translator.

An editor can also be used to create and maintain other documents for the development project, such as memos or engineering specifications.

CREDIT Features

The CREDIT text editor offers the following features:

- Display of text file on the CRT screen
- Entry of characters into the text file
- Insertion, deletion, and replacement of characters in the text
- Immediate update of screen display to reflect all changes
- Search for and replace characters throughout the text file
- Block move and block copy of text
- Advanced editing techniques through commands and macros

The CREDIT text editor operates in two different modes: screen mode and command line mode. In screen mode, CREDIT takes advantage of the capabilities of the CRT screen to offer CRT-based editing. Text is displayed on the screen as it appears in the file and can be directly edited as described in the following section. The screen display is updated as changes are made to the text. In command line mode, text is edited indirectly through commands which provide advanced editing functions. The text is not displayed on the screen as changes are made.

Chapter 2 describes beginning editing techniques. Intermediate editing techniques are described in Chapter 3. Advanced editing techniques including macro definitions and examples are discussed in Chapter 4 . Each chapter contains tutorial sessions illustrating the use of CREDIT.

CHAPTER 2 GETTING STARTED

In this chapter, all of the screen mode editing functions (except macros which are covered in Chapter 4) are described as well as a simplified ISIS command line to invoke CREDIT. The commands described in this chapter are sufficient for many editing applications. Later chapters describe the complete command line with additional details on the use of CREDIT.

Simplified Command Format

```
CREDIT <pathname1>
```

where

<pathname1>	specifies a valid ISIS disk file using the format described in the users manual. If the specified file already exists, it contains the source text (the text to be edited). If the specified file does not already exist, CREDIT creates a new file and allows text to be entered.

Once the CREDIT command line is entered, screen editing commands can be used as described in later chapters.

Examples

To create a new file named MOD3.PLM on logical drive :F2:, enter:

```
CREDIT :F2:MOD3.PLM
```

Since MOD3.PLM is a new file, the editor responds by clearing the screen and then displaying the following sign-on message at the top of the screen:

```
ISIS-II CRT-BASED TEXT EDITOR Vx.y
NEW FILE
```

The version number of the editor appears instead of x.y as shown above.

To edit an existing file named MOD2.PLM on the system default disk, enter:

```
CREDIT MOD2.PLM
```

Since MOD2.PLM already exists, the editor responds by clearing the screen and then displaying the following sign-on message at the top of the screen:

```
ISIS-II CRT-BASED TEXT EDITOR Vx.y
OLD FILE SIZE=n CHARACTERS
```

The n is replaced by the number of characters (bytes) in the file. If an error is made in typing the command, an error message appears instead of the sign-on message. Error messages are explained in the section "Error Messages" in Appendix C.

Screen Editing Basics

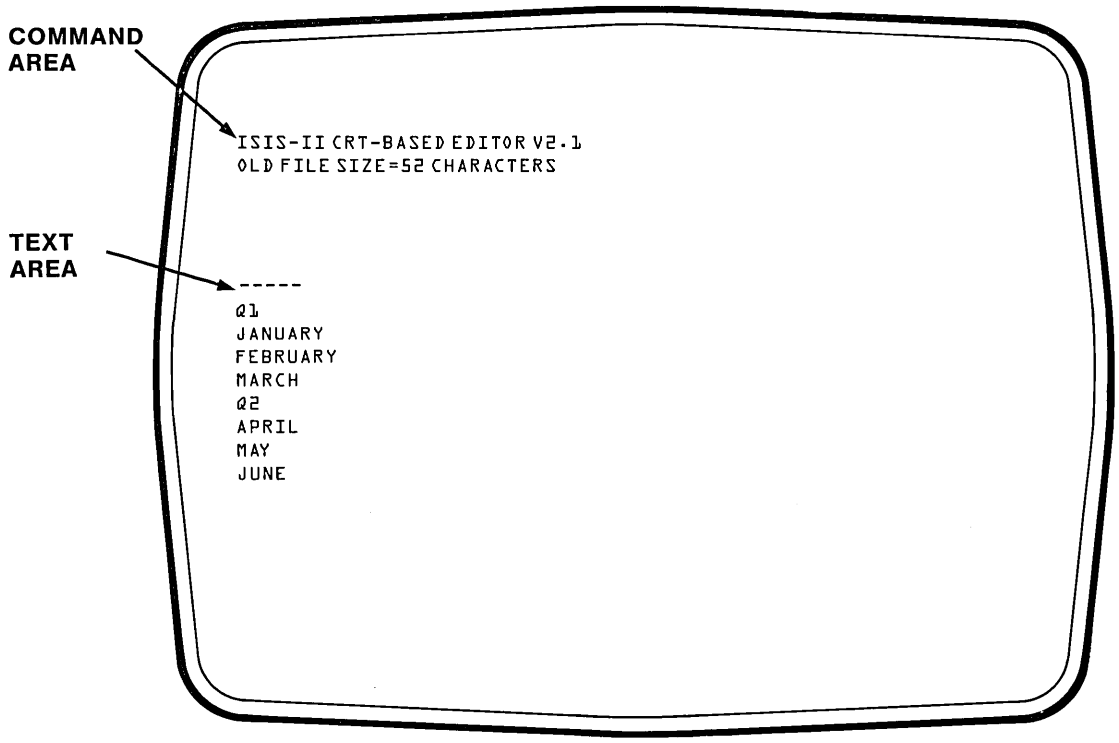
The following sections describe some of the basic features of screen editing. Then, the screen editing functions are described in detail.

The CREDIT Display

When the command line has been entered, CREDIT clears the screen and divides it into two parts as illustrated in figure 2-1. In screen mode, the bottom 20 lines, called the text area, display text from the file. In the remaining lines at the top, the sign-on message, error messages, and status messages are displayed. In screen mode, all operations are performed in the text area; the message area at the top of the screen is not accessed by the user. The two areas are separated by a line of five dashes.

In the screen mode, any ASCII code with an associated character is displayed on the screen as that character. A code with no associated character is displayed as an up arrow (↑). Codes displayed as an up arrow can be pointed to with the cursor and replaced or deleted like any other character.

The end of the file is displayed as a vertical bar (|).



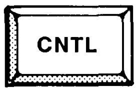
```
ISIS-II CRT-BASED EDITOR V2.1
OLD FILE SIZE=52 CHARACTERS



-----
Q1
JANUARY
FEBRUARY
MARCH
Q2
APRIL
MAY
JUNE
```

Figure 2-1 The CREDIT™ Display

The Keyboard

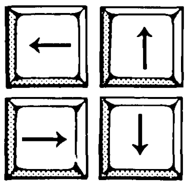
All characters typed at the keyboard are read by the editor. However, some of these characters have a special meaning; they do not simply represent text data, and they are not written to the text file. When editing in screen mode, text is entered through the development system keyboard to be saved in a disk file. Commands are also entered through the keyboard, but are not saved in the file.

Figure 2-2 shows the Series II/III development system keyboard. Be aware of the keyboard differences of your particular system. Character codes generated by the keys are interpreted as ASCII codes by the editor. The keys that perform special functions and are not normally entered as data into the file are:

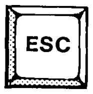


The CNTL key is used for entering control characters. Control characters are entered by pressing a character while holding down the CNTL key similar to the way SHIFTed characters are entered. Many screen mode functions are entered as control characters. For example, in screen mode, the insert text command is CNTL-A.

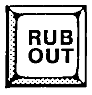
The HOME key switches CREDIT to command mode when it is in screen mode.



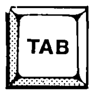
In screen mode, the cursor control keys (↑→ ↓ ←) move the cursor in the direction indicated by the arrow. See the following description of the cursor.



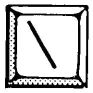
In either mode, the ESC key terminates commands. When ESC is pressed, <BREAK> is displayed in the message area of the screen.



The RUBOUT key deletes the previously entered character when inserting text in screen mode. Otherwise, the RUBOUT key moves the cursor one position to the left without deleting the character.



The TAB key positions the cursor to the next tab set on the line. Its operation is similar to a typewriter tab. The default for tab settings is every 8 characters. Tab settings can be changed using the Alter command described in Chapter 4, Advanced Editing Techniques.



The backslash (\ ) is the default literalizing character. It allows characters that normally perform some function to be entered into the file as data (literalized) instead. The character following the backslash is taken as data. The backslash itself can be entered as data in a file by typing two backslashes in a row. The second one is literalized and is entered as data in the file. The literalizing character can be changed from the backslash to any other character by using the Alter command. See the section on "Alter Commands" in Chapter 4.

All other characters are accepted as data and are entered in the text file, or they are invalid. Any invalid character causes a warning beeper to sound and no action to occur. An example of an invalid character is a command character entered in the middle of an insert or delete command.

The RETURN key is accepted as data and also acts as a line terminator. It is entered into the file as a pair of characters. A line of text consists of a character string terminated by a carriage return-linefeed. This pair of characters, called the line terminator, is entered in the file as a 0DH and 0AH when the RETURN key is pressed.

Lines are not limited to 80 characters (the width of the display), but it is generally easier to work with a file if each text line fits on a display line.

The line terminator is displayed as one character on the screen, the up arrow (↑). Most screen editing functions treat the terminator as one character.

The Cursor

The CREDIT editor maintains a pointer that marks a character in the text file. Changes are made relative to this pointer. For example, deleting a character erases the character designated by the pointer; insertions are made immediately preceding the pointer.

In the screen editing mode, the cursor, reflects the current position of the pointer.

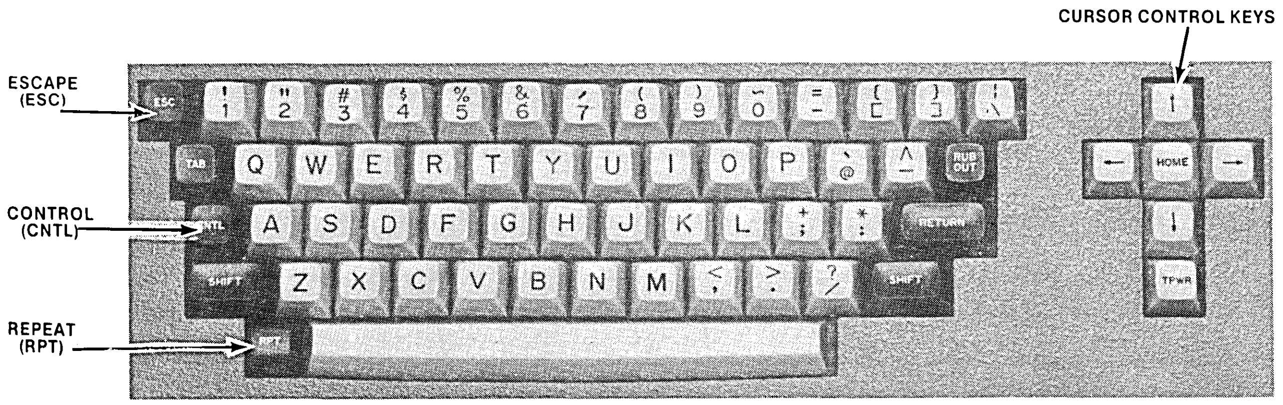


Figure 2-2 The Keyboard

In screen mode, when the cursor is pointing to an area of the screen that does not contain any characters, no edit commands are accepted. The warning beeper sounds if an attempt is made to enter commands with the cursor pointing to an area containing no characters. However, the CNTL-Z command to delete characters can be completed with the cursor pointing to an area containing no characters.

The area of the screen between the line terminator and the next line does not contain any characters; nor does the area beyond the end of file marker.

Screen Editing Functions

In the following sections, screen editing commands are described. With these commands, most routine editing work can be done. The screen mode provides five major editing functions:

- Text entry, to enter text into the file
- Text replacement, to replace text on a character-by-character basis
- Text insertion, to add new text to the file
- Text deletion, to remove text from the file
- Text display, to control the text that is displayed

Each of these command categories is discussed and illustrated in the following sections. The command area at the top of the screen is not used in screen editing except for display of error and status messages.

All of the screen editing functions follow the same general operating sequence. First, text to be edited is displayed on the screen in the text area; second, the cursor is positioned at the place where a change is desired; and, third, the change is made.

Text Entry

Text can be entered into the file in two steps as follows:

1. Move the cursor using the cursor control keys to the end of file mark (|), or to a line terminator (↑).
2. Type the characters on the keyboard. They will appear on the display screen as they are entered into the file.

If the cursor is at the end of file mark, the characters are added at the end of the file. If the cursor is at a line terminator, the characters are added to that line.

When CREDIT is used to create a new file, the cursor points to the end of file mark as soon as the file is created.

There is a wraparound feature, so the next character is automatically displayed on the line below the first character. The carriage return/linefeed need not be entered into the file. However, when the file is printed using the COPY command under ISIS, all the characters between carriage return/linefeeds will be printed on a single line.

Some keyboards have an auto repeat feature, present in the editor as well. If a key is pressed and then held down, it is automatically repeated. This feature is convenient for entering a line of the same character, for example, a line of asterisks (*).

Replacement

Replacement occurs on a character-by-character basis; a single character cannot be replaced by two characters; two characters cannot be replaced by a single character. The Insert and Delete functions provide for other cases besides one-to-one replacement.

To replace a character, position the cursor at the character to be replaced. Use the cursor control keys for this operation. Press any printing character key on the keyboard. The character pointed to is replaced with the character typed.

The two exceptions are when the cursor is pointed to a line terminator (up arrow) or the end of file mark (vertical bar). Line terminators and the end of file marks are never directly replaced. Either one is moved one position to the right, and the typed character is inserted ahead of it. To indirectly replace the line terminator, position the cursor under the terminator; type the text that replaces the terminator; delete the terminator with the Delete function (CNTL-D). The end of file mark cannot be deleted or replaced.

To enter characters as data in the file when they are normally given special meaning by the editor, precede the special character with the literalizing character. The default literalizing character is the backslash (\ ). Examples of characters with special meaning to the editor are control characters. Many control characters are interpreted as commands and are entered by pressing a character while holding down the CNTL key.

To change a single character of data in the file to CNTL-A, line up the cursor with the character to be replaced. Type backslash (\ ). Then, type the letter A while holding down the CNTL key. A control character is displayed as an up arrow.

This feature is useful in creating CREDIT macros and ISIS SUBMIT files as described in Chapter 4.

A backslash can be entered as data in the file by typing two backslashes.

Insertion

There are two insert functions: one to add any number of characters and one to add a single character. After either command, the new text appears immediately to the left of the cursor; the cursor points to the same character after the insert as it did before the insert. There is no need to insert characters at the end of the file or at the end of a line. See the section on "Text Entry" for instructions on adding text to the end of a line or at the end of a file.

CNTL A

Add Text

The format of the command is:

CNTL-A<text>CNTL-A

where

CNTL-A	is the Add Text command character.
<text>	can be any number of characters, including new lines. Control characters or non-printing characters must be preceded by the literalizing character (default of backslash).

The Add Text command inserts any number of characters before the character pointed to by the cursor.

To use the command move the cursor to the point of insertion. Press CNTL-A and type the text to be inserted. End the insertion with another CNTL-A.

The first CNTL-A clears the text area from the point of insertion to the end of the screen. The final CNTL-A fills the screen again.

During insertion, the whole text area rolls up one line each time the bottom line of the screen is filled with text.

Only text can be entered after the first CNTL-A. The RUBOUT key can be used to backspace but, other control characters and the cursor control keys do not have any effect during an insert.

CNTL C

Add Character

The format of the Add Character command is:

CNTL-C<x>

where

CNTL-C	is the Add Character command character.
<x>	is any character. Control characters and non-printing characters must be preceded by the literalizing character (default of backslash).

The Add Character command inserts a single character before the character pointed to by the cursor.

To use the command, move the cursor to the point of insertion. Press CNTL-C and type the character to be inserted; no terminator is required.

Deletion

There are two delete functions: one to delete any number of characters and one to delete a single character. After either command, the changed text is displayed immediately.

CNTL Z

Delete Text

The format of the Delete Text command is:

```
  CNTL-Z<move cursor>CNTL-Z
```

where

CNTL-Z	is the Delete Text command character.
<move cursor>	represents the movement of the cursor with the cursor control keys.

The Delete Text command deletes any number of characters.

To use the command, move the cursor to the first character to be deleted and press CNTL-Z. The editor replaces this character with the at sign (@). Move the cursor to the character following the last character to be deleted; if the last character to be deleted is a line terminator, move the cursor to the first character of the next line.

As the cursor moves over a character, the at sign (@) is displayed for that character. The original character returns when the cursor is moved. The at sign (@) is also displayed to the right of line terminators or beyond the end-of-file character where there is no data.

When the cursor is positioned one character beyond the last character to be deleted, press CNTL-Z again. All data beginning with the first at sign (@) and ending with the character preceding the last at sign (@) is deleted. The deleted characters are removed from the screen and the remaining characters after the deleted material are moved up in the file. After the command, the cursor points to the first character following the deleted material.

The location of the second CNTL-Z can be ahead of the first in which case, the second CNTL-Z entered is interpreted as the first and the first one entered is interpreted as the second. The second CNTL-Z may be to the right of the line terminator. It may also be to the right of the end-of-file marker or on a line below the end-of-file marker.

Once a delete is started, it can be canceled before the second CNTL-Z is pressed by pressing the ESC key or by pressing the second CNTL-Z with the cursor at the first CNTL-Z (the first @ sign). If the ESC key is used, the cursor is reset to the location of the first CNTL-Z and the message <BREAK> is displayed at the top of the screen in the message area. Two CNTL-Z's in a row without moving the cursor in between results in a null delete.

No other screen editing functions besides ESC are allowed between the two CNTL-Z's.

The boundaries of the two CNTL-Z characters are limited to the text displayed on a single screen. The text on subsequent or previous screens cannot be deleted with CNTL-Z.

CNTL D

Delete Character

The format of the Delete Character command is:

```
CNTL-D
```

where

CNTL-D	is the Delete Character command character.

The Delete Character command deletes a single character.

To use the command, move the cursor to the character to be deleted and press CNTL-D. The character is deleted and remaining characters in the line move one position to the left. After the command, the cursor points to the character following the deleted character.

If the cursor points to an area not containing any characters, the warning beeper sounds.

Display

There are three display functions that rewrite the screen:

- View Page displays text relative to the current pointer position.
- Next Page displays the screenful of text that follows the current display.
- Previous Page displays the screenful of text that precedes the current display.

CNTL V

View Page

The format of the View Page command is:

CNTL-V

where

CNTL-V	is the View Page command character.

In screen mode, the View Page command rewrites the screen with 20 lines from the text file.

The line containing the cursor will be the third line of the text area when the screen is rewritten. The following 17 lines of the file appear on the screen also. The position of the cursor determines which lines will reappear when the screen is rewritten. The cursor points to the same character after the CNTL-V as before the CNTL-V.

CNTL N

Next Page

The format of the Next Page command is:

CNTL-N

where

CNTL-N	is the Next Page command character.

The Next Page command displays the screenful (20 lines) of text that follows the current display.

There is a two-line overlap in the display; the last two lines of the display before the command become the first two lines of the display after the command. The cursor points to the first character of the third line in the text area. If there are fewer than three lines remaining in the file, the last three lines are displayed and the cursor points to the beginning of the last line.

CNTL P

Previous Page

The format of the Previous Page command is:

```
CNTL-P
```

where

CNTL-P	is the Previous Page command character.

The Previous Page command displays the last screenful (20 lines) of text that precedes the current display.

There is a two-line overlap in the display; the first two lines of the display before the command become the last two lines of the display after the command. If there is less than a screenful (20 lines) of text preceding the current display, the first 20 lines of the file are displayed. The cursor points to the first character of the third line in the text area.

Exiting From The Editor

Once all changes have been made, the editing session is ended and control is returned to ISIS. There are three ways to end an editing session:

- Replace the old version of the file with the updated version.
- Store the updated version with a different name, saving both the old version and the updated version.
- Ignore any changes and leave the old file unchanged.

The first two ways are accomplished with the Exit (EX) command; the third way with the Quit (EQ) command.

For either command, it is also necessary to switch from screen mode to command line mode. An editing session can only be ended in the command line mode of editing.

HOME

Shift To Command Line Mode

The editor is in screen mode when an editing session first begins after the CREDIT command line is entered.

To shift from screen mode to command mode, press the key labeled:

HOME

The cursor moves to the top of the screen where messages appear, and an asterisk (*) prompt is displayed to indicate that the editor is ready to accept a command.

The editor must be in command mode before the editing session can be ended.

EX

Exit Command

The format of the Exit command is:

```
EX [<filename>]
```

where

EX	is the Exit command name.
<filename>	is the name of the updated version of the file. If no filename is specified, the old version is replaced by the updated version.

The Exit command ends the editing session and stores the updated version of the file. The new version can either replace the old version or it can be saved with a different name.

NOTE

Be sure to type only EX as the command name and not EXIT. Typing EXIT causes display of the the error message IMPROPER OPERAND. The editing session is not ended and the updated version of the file is not stored.

CAUTION

If the device name given is not a valid pathname, ISIS may return a file access error.

To end the editing session and replace the old version of the file with the updated version, enter:

```
EX
```

To end the editing session and save the updated version of the file with the name MOD2.ASM on the disk in drive 1, enter:

```
EX :F1:MOD2.ASM
```

EQ

Quit Command

The format of the Quit command is:

```
EQ
```

where

EQ	is the command name.

The Quit command ends the editing session without updating any files on the disk. This command is used if a major error was made in editing the file. The file is restored to the state it was in prior to editing. All editing changes are lost.

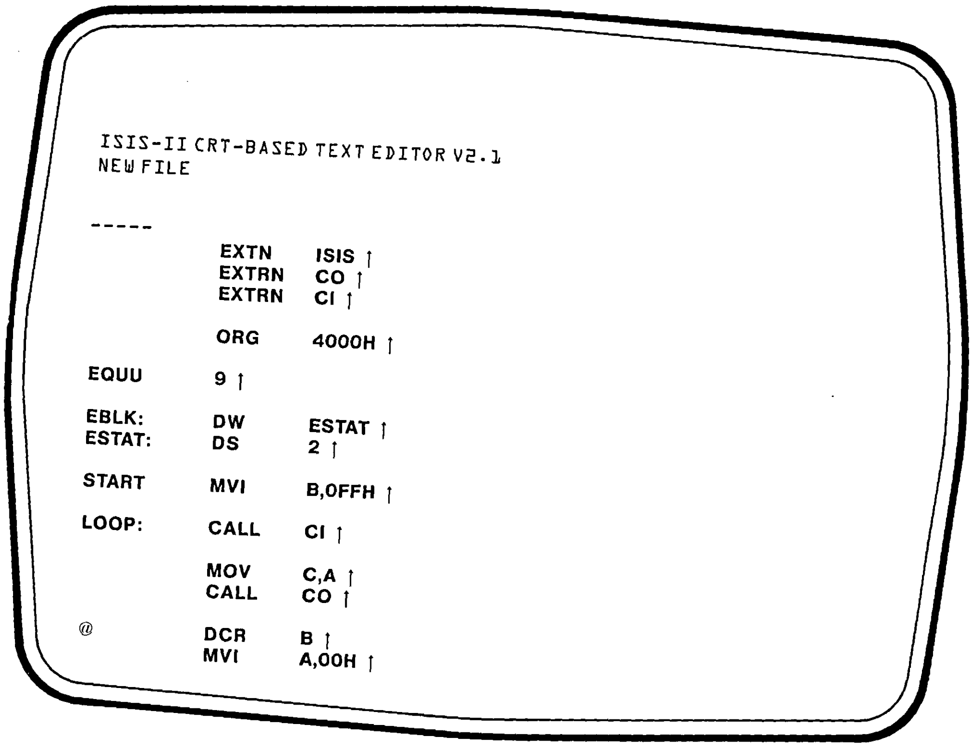
To keep from accidentally losing data, the editor prompts before returning to the operating system. For example,

```
EQ
QUIT?
```

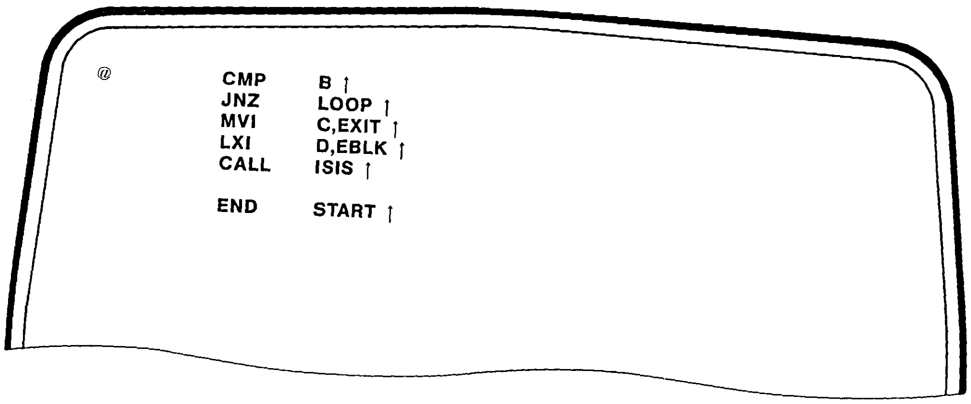
Type Y or y (for yes) to verify the exit; any other character causes the editing to continue. The RETURN key is not required after the Y or y entry.

Sample Screen Editing

In this section, the screen editing commands are used to create a source program and edit it. To begin, enter: CREDIT PROGA.SRC(Return).



```
ISIS-II CRT-BASED TEXT EDITOR V2.1
NEW FILE

-----
                EXTN      ISIS ↑
                EXTRN     CO ↑
                EXTRN     CI ↑

                ORG       4000H ↑
EQUU            9 ↑

EBLK:           DW        ESTAT ↑
ESTAT:          DS        2 ↑

START           MVI       B,0FFH ↑

LOOP:           CALL      CI ↑

                MOV       C,A ↑
                CALL      CO ↑

@               DCR       B ↑
                MVI       A,00H ↑
```



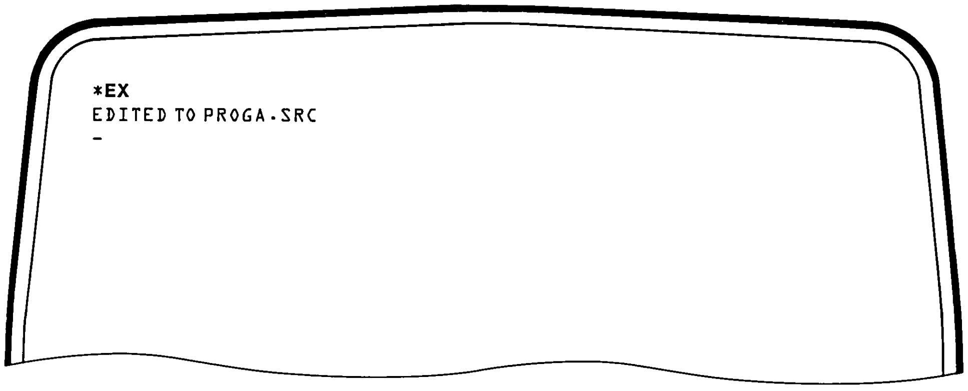
```
@               CMP       B ↑
                JNZ       LOOP ↑
                MVI       C,EXIT ↑
                LXI       D,EBLK ↑
                CALL      ISIS ↑

                END       START ↑
```

Comments

Enter the program as shown above. Use TABS instead of spaces between words. Notice that when the last line is entered, the top line scrolls off the screen and the whole screen moves up one line to allow another line to be entered at the bottom.

Key-in Sequence	Comments
CNTL P ↑	Type CNTL-P to return to the beginning of the file. In this example, a character was omitted from EXTRN on the first line of the file. Use the cursor control keys to move the cursor to the N following the missing character.
CNTL C R	Type CNTL-C followed by the missing character, R, to perform the single character insertion.
↓ ←	In this example, more than one character was omitted. Use the cursor control keys to move the cursor to the E of EQUU 9.
CNTL A EXIT TAB	When the CNTL-A is entered, the screen is cleared at the point of the insert. Type the characters to be inserted.
CNTL A	Type a second CNTL-A to complete the insertion. The open area on the screen is then closed up again.
→	In this example, an extra character was typed. Use the right arrow cursor control key to move the cursor to one of the extra U's in EQUU.
CNTL D	Type CNTL-D to delete the single extra character.
↓ →	The program that has just been entered allows the user to type characters at the keyboard. Then, it displays the characters on the CRT screen. As written, the program allows 255 characters to be entered (as determined the value of 0FFH that is moved into the B register). Suppose that the program is to be rewritten to allow characters to be entered until the character 1AH (the ASCII code for a CNTL-Z) is typed. This modification would require changing the 0FFH to 1AH and deleting the DCR B and MV1A,00H instructions. Use the cursor control keys to move the cursor to the 0 of 0FFH.
1AH CNTL D	Type 1AH to replace the 0FF and then enter a CNTL-D to delete the extra H.

↓ ←	Use the cursor control keys to move the cursor to the first character on the line with the DCR B instruction. Then, enter CNTL-V to display the next page of text from the file.
CNTL Z	Type CNTL-Z. An @ character will appear at the position of the cursor. This @ sign marks the beginning of the text to be deleted.
↓	Use the down arrow cursor control key to move the @ marker to the first character of the CMP B line. The first @ will remain at the DCR B line. The two @'s mark the bounds of the text to be deleted.
CNTL Z	Type CNTL-Z again to finish the deletion.


```
*EX
EDITED TO PROGA.SRC
-
```

Key-in Sequence	Comments
HOME	Press the HOME key to switch to the command line mode. This mode will be described in the next section. However, it is necessary to switch in order to exit from the editor. The CREDIT prompt will appear in the command area of the screen (above the line).
EX RETURN	Enter the EX command to return to the operating system.
	The screen is cleared. It appears as shown above after exiting from the editor. Do not delete the file PROGRA.SRC; it will be used in later examples.

CHAPTER 3
INTERMEDIATE EDITING

In this chapter, all the commands covered are entered in the command line mode of editing. However, not all command line commands are covered. The commands to end an editing session are covered in the previous section. Also, some advanced commands are covered in the section on "Advanced Editing Techniques." This chapter begins with an intermediate form of the CREDIT command that can be entered under ISIS. Then, basic features of the command line mode of editing are described. Last, the commands themselves are described.

Intermediate Command Format

```
CREDIT <pathname1> [TO <pathname2>]
```

where

<pathname1>	specifies a valid ISIS file or device pathname using the form described in the users manual. The source text (the text to be edited) is contained in the file or device specified. More details are given below.
TO <pathname2>	is an optional parameter that specifies a valid ISIS file or device pathname. The destination file or device will contain the text after editing.

The intermediate form of the command can be used when the input text is not on a disk file. For example, the source text could come from the serial input device. The source text could also come from the console input device. However, receiving input from the console device using the intermediate command line format is the same as using the simplified command format to create a new disk file for editing.

With this form of the CREDIT command line, two copies of the file are saved. Both the version prior to editing and the version after editing are saved. However, it is not necessary to use the intermediate version of the command to maintain backups, because backups are automatically maintained when the simplified version of the command is used.

When editing an existing file (as opposed to creating a new file for edit) using the simplified command format, CREDIT automatically saves the most recent version of the file prior to editing as well as the updated version. The updated version of the file is saved with the identifier <pathname1>, and the version of the file prior to editing is saved on the same disk drive with the same <filename> as <pathname1> but with the extension of .BAK.

CAUTION

To edit or recover a .BAK file, rename or edit the .BAK file to some other filename. The edited .BAK file is deleted when exiting the editor if no file name is specified.

The automatic backup feature of CREDIT is overridden if a second pathname is specified on the command line. Using the intermediate form of the command, the backup can be saved under a different filename or on a different disk drive from the source file. The updated file is saved as <pathname2> while the old file is saved as <pathname1>.

Examples

To receive text from the serial input device and save it as a disk file on drive 2, enter:

```
CREDIT :TI: TO :F2:MOD2.PLM
```

To edit the text in the disk file MOD2.PLM and save it in the file MOD3.PLM, enter:

```
CREDIT :F1:MOD2.PLM TO :F2:MOD3.PLM
```

Command Line Editing Basics

The following sections describe some of the basic features of command line editing. Then, the commands are described in detail.

The CREDIT Display

When the command line is first entered under ISIS, the screen is cleared and divided into two parts as shown previously in figure 2-1. CREDIT initially enters screen mode; pressing the HOME key switches to command line mode.

In command line mode, the top area of the screen, called the command area, is the only area accessed by the user. In fact, the text area is erased as commands are entered. The asterisk prompt is displayed in the command area indicating that a command can be entered.

When the command line mode is first entered, the text area contains the residual display of the file left over from previous screen editing operations. As commands are entered at the keyboard, they are displayed in the command area. As soon as the command area exceeds the top three lines, the entire text area is erased allowing commands to fill the screen. Once the screen is full of commands, it scrolls up one line at a time as new commands are entered.

In the command line mode, ASCII codes with an associated graphics character are displayed as an up arrow (↑). The up arrow character is displayed as two up arrows (↑↑) to distinguish it from codes with no associated graphics character.

The text area is not used in command line mode.

The Keyboard

When editing in command line mode, commands are entered at the development system keyboard to indirectly modify the text in a file. Data to be added to the text file is entered as a parameter to a command. It is not directly entered into the file as in screen mode.

Figure 2-2 shows the keyboard. Some of the keys perform special functions in the command line mode of editing as listed below.

CNTL

The CNTL key is used for entering control characters. Control characters are entered by pressing a key while holding down the CNTL key. Some commands are entered using control keys. For example, CNTL-V switches from command line editing to screen editing.

ESC

In either mode, the ESC key aborts commands. When ESC is pressed, <BREAK> is displayed in the command area of the screen.

RUB OUT

The RUBOUT key deletes the previous character when in command line mode.

CNTL B

Many of the commands in command line mode require a string of characters as a parameter. The string of characters must be delimited by a valid delimiter character. CNTL-B is a special delimiter character that causes the string to be interpreted as hexadecimal values rather than as ASCII codes. This character is discussed in the section on entering commands.

&

The ampersand is used as a continuation character for command lines. This character is discussed in the section on entering commands.

;

The semicolon is used to separate multiple commands entered on a single command line. This character is discussed in the section on entering commands.

The HOME key, the cursor control keys, the TAB key, and the backslash key do not perform any special command line editing function.

In command line mode as in the screen mode, the RETURN key is entered as two characters in the file (carriage return, 0DH, and linefeed, 0AH). Most commands treat the RETURN as two characters.

The Pointer

The CREDIT editor maintains a pointer that marks a character in the text file. Changes are made relative to this pointer. For example, deleting a character erases the character designated by the pointer; insertions are made immediately preceding the pointer.

In command mode, there are several commands to move the pointer.

In command mode, the cursor has no association with the pointer. It merely indicates where command lines are displayed as they are entered. See figure 3-1.

Tags

In addition to the pointer which marks the current location in the text file, the CREDIT editor also marks four other locations in the file. The markers used are called system tags. They point to the beginning and end of the file and the beginning and end of the portion of the file located in the system memory. The entire file may not fit in user memory at one time.

In addition to the system tags maintained by the editor, the user can specify ten tags in command mode to mark locations of interest in the text file. See figure 3-1.

The ten user-defined tags are T0 though T9; the four system tags are:

- TT - Marks the beginning of the file
- TE - Marks the end of the file
- TB - Marks the beginning of the portion of the file currently resident in memory
- TZ - Marks the end of the portion of the file in memory

Tags are used by several of the commands to identify the location in the text affected by the command.

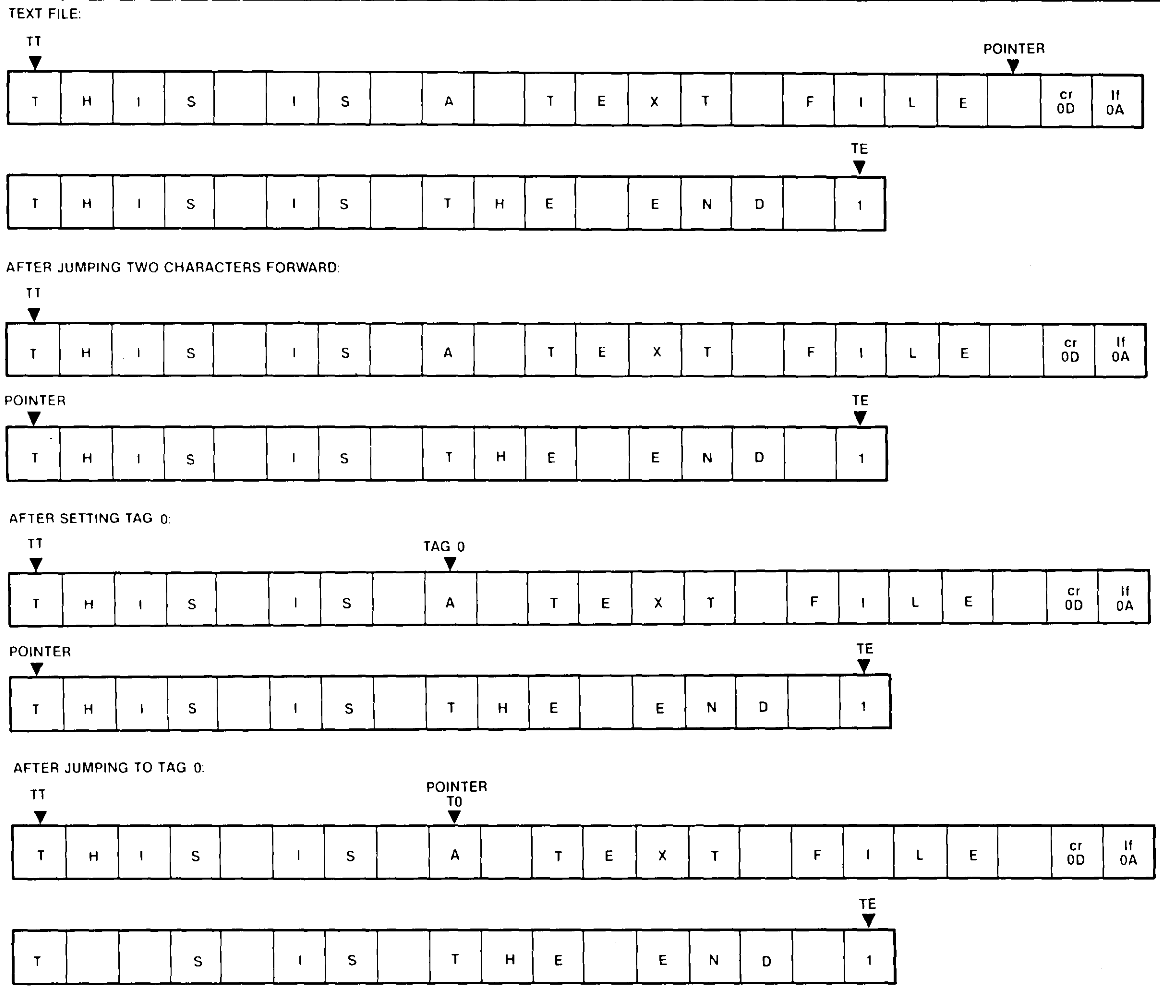


Figure 3-1 Pointers and Tags in the Text File

Disk File Use

The CREDIT editor stores text in disk files and loads the text into memory for editing. Usually, only a part of the file is loaded into memory at a given time, since the entire file usually will not fit in memory. Often, files in addition to the one containing the text are needed during an editing session. These files are temporary files created by the editor, backup files created by the editor, and files used by different CREDIT commands. See figure 3-2.

Temporary Files

In addition to the old edit file containing the source text data, the following temporary files are created by CREDIT during an editing session:

- An output file called CREDT1.TMP contains the modified text data during the editing session. If the session is ended with the EXIT command, CREDT1.TMP is renamed. If no name was supplied on the CREDIT command line (as part of the TO clause), the old file is renamed with the extension of .BAK, and CREDT1.TMP is renamed to the old edit file. If the TO clause is supplied, CREDT1.TMP is renamed to the file specified as part of the TO clause on the command line.
- A temporary file called CREDT2.TMP is created only if a part of the file no longer resident in memory is edited.
- A temporary file called CREDT3.TMP may also be created to store the modified text data during an editing session.

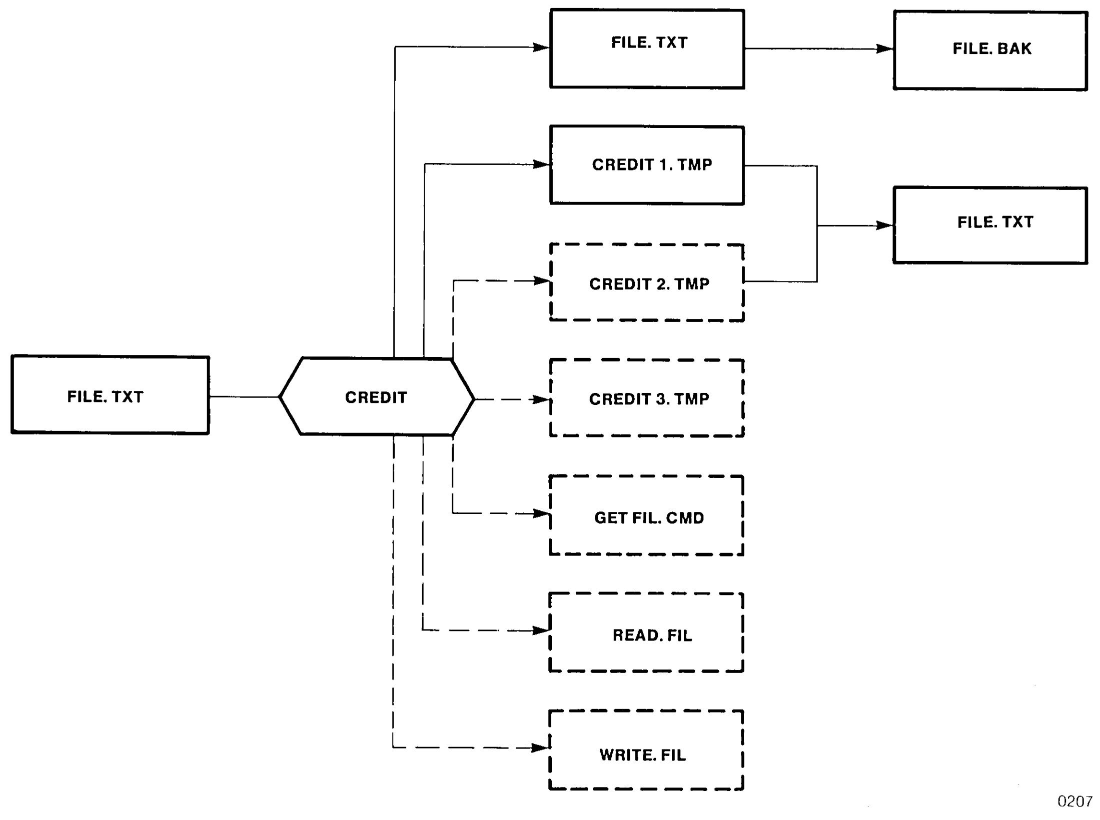


Figure 3-2 Disk File Use

CREDT1.TMP, CREDT2.TMP, and CREDT3.TMP are reserved filenames and should not be assigned to files by the user. If the TO clause is used on the CREDIT command line, they are created on the same drive as the file specified in the TO clause. Otherwise, they are created on the same drive as the old file being edited. None of the temporary files appear in the directory unless the operating system is reloaded (for example, if RESET is pressed) before the editing session is terminated. However, the temporary files can be viewed in the directory on a dual processing system if the user runs the DIR command on one processor while the editing session is in progress on the other processor. See figure 3-2.

Backup Files

When an existing file is edited and no TO clause is specified, it is renamed with the same filename and an extension of .BAK when the editing session is ended with the EX command. Thus, after changes are made, the previous version of the file is still available.

If a backup file already exists from previous editing, it is automatically deleted and replaced by the version of the file prior to the current editing session. See figure 3-2.

Several rules must be followed to successfully use the backup feature:

- The .BAK version of the file should not be deleted.
- The .BAK version of the file should not be edited.
- The .BAK version of the file should not be write protected; don't set the write attribute to 1.

If the .BAK version is deleted, no backup will be available.

If the .BAK version is edited, the changes made will not be reflected in the original; the original version is copied to the backup version, not vice versa. The first time the original is accessed through the editor, the .BAK version will be replaced by the current version, wiping out any changes made in the backup file.

If the .BAK version is write protected, the editing session cannot be ended with the EX command unless a filename other than the source file for the output is specified. Only EQ or EX with a filename parameter is accepted. If EQ is used, all changes from the editing session will be lost.

Files Used by CREDIT Commands

In command line mode, the XC and XM commands use a temporary file named CREDT3.TMP. Some of the advanced CREDIT commands use additional files. The section on "Advanced Editing Techniques" describes file use in more detail. See figure 3-2 for an illustration of disk file use by the CREDIT editor.

Limits on Disk File Use

ISIS allows a maximum of six files to be open at any one time. This leaves three files for user applications after allowing for the three files that the CREDIT editor can open. Normally, this number will not be exceeded; however, the user should exercise judgment in opening files for access. Files should be closed when not being accessed.

Editing under the control of the SUBMIT program further limits the number of available user files by one. SUBMIT file requirements must be considered when using CREDIT with SUBMIT. If more than six files are opened at a time, a fatal error occurs, and the operating system is reinitialized.

Performance and File Size

The size for CREDIT files is limited only by the storage device. There must be enough space available on the diskette or bubble to hold the file, the backup file, and the temporary files that the editor uses. The free space on the disk must be two times as great as the size of the file being edited.

CREDIT works best if files are restricted to 20K bytes or less (the size of the text buffer in memory). Files less than 20K bytes can be loaded into memory, and all editing functions can be performed in memory with a minimum of disk accesses. A file with 20K bytes is about four 8 1/2 x 11 pages.

Entering Commands

Commands can be entered from the keyboard whenever the asterisk (*) prompt is displayed. Commands must end with a RETURN key before they are executed. More than one command may be entered on a single screen line as long as commands are separated by semicolons (;) as shown in the following example.

```
*L15;TS1;L15;TS2;L15;TS3 RETURN
```

The CNTL-V command to switch to screen mode must be entered as the first command on a screen line or the first command after a semicolon.

An ampersand (&) immediately preceding the RETURN key and following the last semicolon on a screen line continues the command line after the RETURN key is pressed. The prompt character for the continuation line is two asterisks (**) instead of one. In the following example, the sequence of commands is not executed until the second RETURN is entered.

```
*L15;TS1;L15;TS2;L15;TS3;& RETURN
**L15;TS4;L15;TS5 RETURN
```

Correcting Commands

The command line can be corrected with the RUBOUT key prior to pressing RETURN. The RUBOUT key backspaces through the current command line one character at a time erasing each character it passes. The correct characters can then be typed.

Delimiters

Many of the CREDIT commands require strings of characters as parameters. These strings must be delimited, so that the editor can distinguish text from commands. A valid delimiter is any character that is not used within the string except space, RETURN, linefeed, the literalizing character (default of backslash), or escape. The delimiter character is used before and after the string. In most of the examples in this section, the double quote character is used as a delimiter. For example:

```
I "LOOP: MOV A,M;SAVE THE VALUE RETURN
" RETURN
```

In this example, the program statement, all the characters between the double quotes including the RETURN at the end of the line, is inserted at the current location of the cursor in the file.

Hexadecimal Entry with CNTL-B Delimiter

A special delimiter character allows the entry of hexadecimal values instead of ASCII codes. Hexadecimal values are entered as parameters for commands by using a CNTL-B as the delimiter character. For example, to enter the hexadecimal value 00, type:

```
I CNTL-B 00 CNTL-B RETURN
```

as part of the command line. The editor interprets this string as a single hexadecimal byte. Multiple characters can also be entered:

```
I CNTL-B414243444546CNTL-B RETURN
```

This sequence enters the characters "ABCDEF".

The space bar or the RETURN key can be used to separate the hexadecimal codes:

```
I CNTL-B 41 42 43 44 RETURN
45 46 CNTL-B RETURN
```

results in the same data being entered as in the preceding example.

Hexadecimal values are interpreted as pairs of digits. However, a single digit may be entered, or a single digit may be set off by spaces or RETURNs. In either case, an isolated digit is assumed to be preceded by a zero. For example,

```
I CNTL-B 5 CNTL-B RETURN
```

is treated the same as:

```
I CNTL-B 05 CNTL-B RETURN
```

This example enters a CNTL-E (05H) into the file. CNTL-E is often used with the SUBMIT command. Any character other than a valid hexadecimal digit (0-9 and A-F) results in a syntax error.

General Command Format

The general format of an editing command is:

```
<command name> [<parameter>]
```

where

<command name>	specifies which command to execute. All command names are either one or two characters. The command name is shown in capital letters in the format line for each command.

<parameter>	can be either one, two, or three items required by the command. In the format line for a specific command, the parameter may or may not be in brackets. The brackets mean that the parameter is optional. While some parameters are numeric values, other parameters are characters. Text parameters must be set off by delimiters as described previously, so the editor can distinguish the parameter from the command.

Some examples of commands are:

```
L3
```

to move the pointer forward three lines, and

```
XC T1,T2
```

to copy all text between tags one and two to the current pointer location.

Command Mode Editing Functions

Command mode editing provides enhancements to the screen editing mode. In the command line mode, CREDIT commands are entered in the command area of the screen (above the dashed line); the text area is not used. The command area expands to fill the entire screen and then scrolls up as further commands are entered. CREDIT commands duplicate screen functions with the exception of the display functions and supply additional functions that are not available in screen mode.

- The Help command displays a summary of CREDIT commands and screen mode functions.
- Pointer commands move the pointer.
- Tag commands set and reset auxiliary pointers called tags.
- Text commands replace, insert, delete and display text, duplicating some of the screen mode features.
- Block move and block copy commands allow sections of text to be moved or copied.
- Search commands locate strings of characters within the text and can also substitute new strings for old ones.

Each of these categories is discussed and illustrated in the following sections.

HELP

The HELP Command

The format of the Help command is:

```
H
```

where

H	is the command name.

The Help Command displays a summary of the formats of CREDIT commands and operations.

No parameters are allowed with this command. The Help information is kept in the file CREDIT.HLP, which must be on the same disk containing the CREDIT program.

The Help information is displayed in three parts. The first screen lists the screen editing commands; the second lists the command mode commands, and the third lists the advanced editing commands. The screens are illustrated in the following example.

At the end of each screen, a prompt is displayed so that the next screen can be displayed or skipped.

After the first screen, the program prompts:

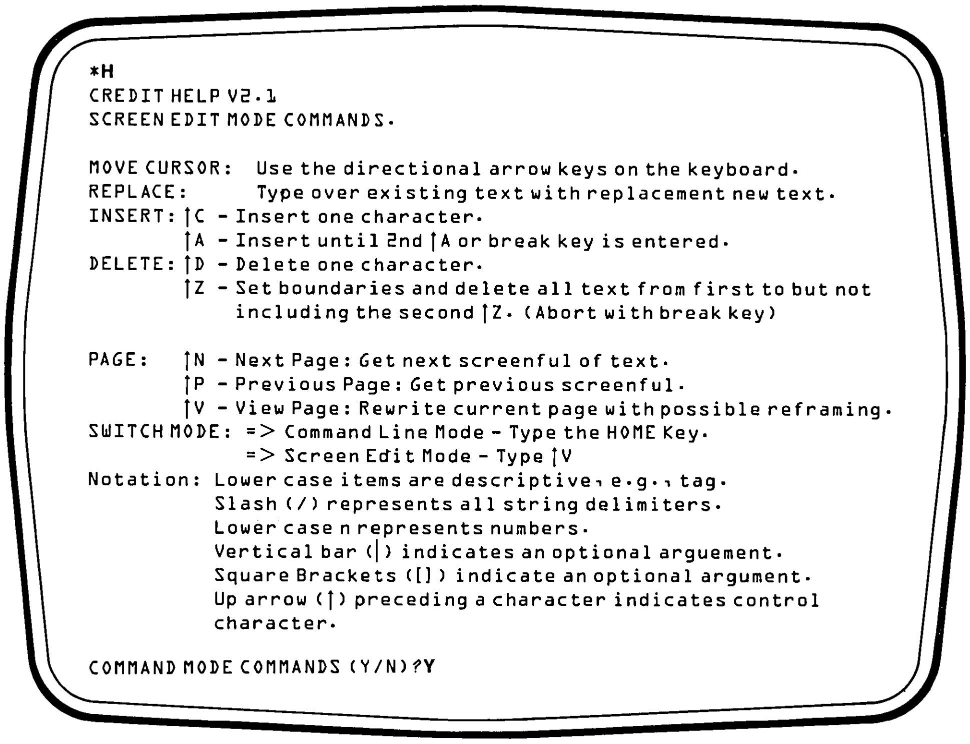
```
COMMAND MODE COMMANDS (Y OR N)
```

Typing Y displays the command mode help information followed by a prompt. Typing N skips the command mode help screen and displays the next prompt:

```
ADVANCED EDITING COMMANDS (Y OR N)
```

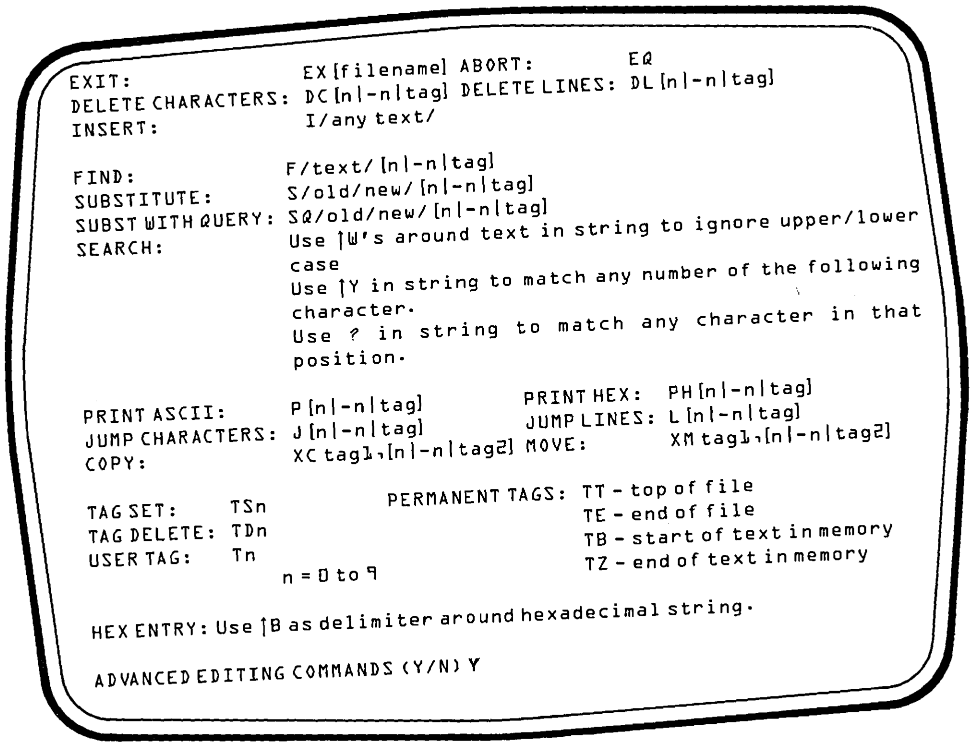
Typing Y displays the advanced editing help information. Typing N skips the advanced help information and ends the Help command.

The following screen displays illustrate the use of the Help command.



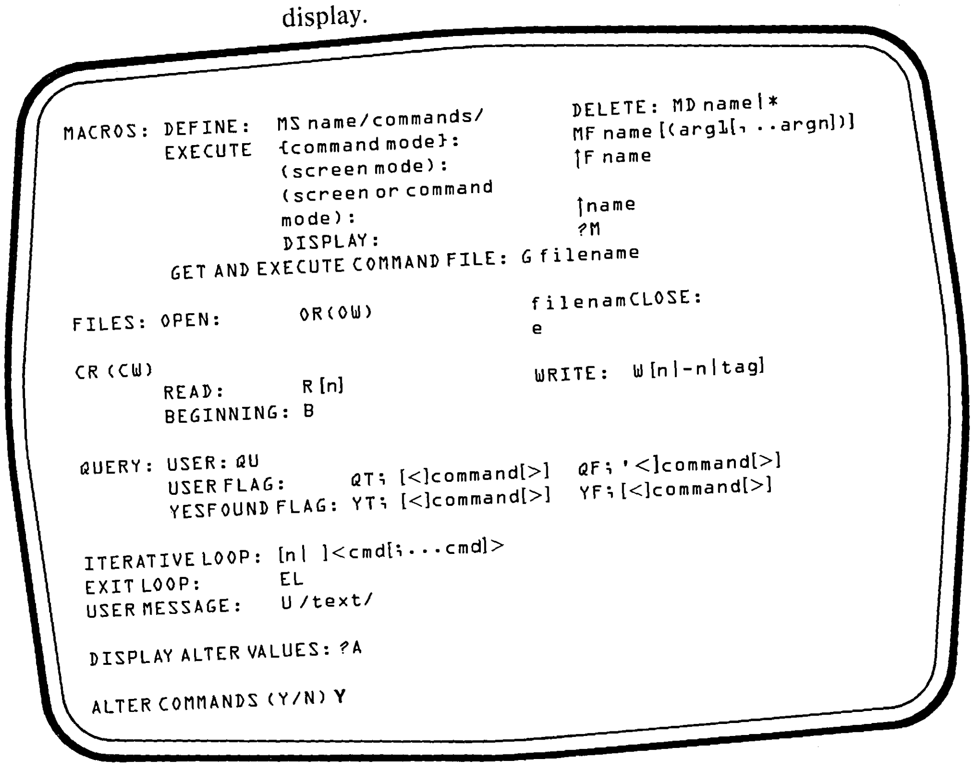
```
*H
CREDIT HELP V2.1
SCREEN EDIT MODE COMMANDS.

MOVE CURSOR:  Use the directional arrow keys on the keyboard.
REPLACE:      Type over existing text with replacement new text.
INSERT: ↑C - Insert one character.
        ↑A - Insert until 2nd ↑A or break key is entered.
DELETE: ↑D - Delete one character.
        ↑Z - Set boundaries and delete all text from first to but not
             including the second ↑Z. (Abort with break key)

PAGE:   ↑N - Next Page: Get next screenful of text.
        ↑P - Previous Page: Get previous screenful.
        ↑V - View Page: Rewrite current page with possible reframing.
SWITCH MODE: => Command Line Mode - Type the HOME Key.
             => Screen Edit Mode - Type ↑V
Notation: Lower case items are descriptive, e.g., tag.
          Slash (/) represents all string delimiters.
          Lower case n represents numbers.
          Vertical bar (|) indicates an optional arguement.
          Square Brackets ([]) indicate an optional argument.
          Up arrow (↑) preceding a character indicates control
          character.

COMMAND MODE COMMANDS (Y/N)?Y
```

Key-in Sequence	Comments
CREDIT ANYFL.TXT RETURN	The next series of examples illustrate the CREDIT Help command. Enter this command line to invoke CREDIT so that the Help command can be run. A new file is created as a result of the invocation. CREDIT is orginally in screen mode.
HOME **H** RETURN	The Help command is entered in command line mode, so press the HOME key to switch to command line mode. Then, enter the H command. The file CREDIT.HLP must be on the disk before the Help command is successful.
Y	The first screen of Help describes screen mode editing functions. Type Y in response to the prompt to continue the display.



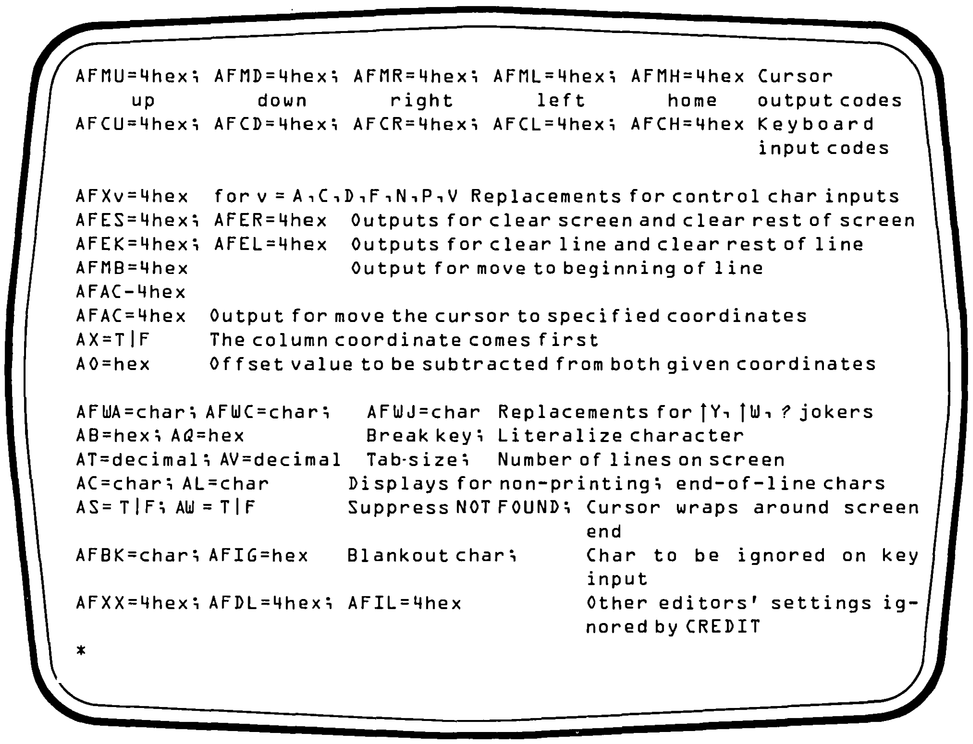
```
EXIT:                EX[filename] ABORT:          EQ
DELETE CHARACTERS:   DC[n|-n|tag] DELETE LINES:  DL[n|-n|tag]
INSERT:              I/any text/

FIND:                F/text/[n|-n|tag]
SUBSTITUTE:          S/old/new/[n|-n|tag]
SUBST WITH QUERY:    SQ/old/new/[n|-n|tag]
SEARCH:              Use ↑W's around text in string to ignore upper/lower
                     case
                     Use ↑Y in string to match any number of the following
                     character.
                     Use ? in string to match any character in that
                     position.

PRINT ASCII:         P[n|-n|tag]          PRINT HEX:   PH[n|-n|tag]
JUMP CHARACTERS:     J[n|-n|tag]          JUMP LINES:  L[n|-n|tag]
COPY:                XC tag1,[n|-n|tag2]  MOVE:        XM tag1,[n|-n|tag2]

TAG SET:      TSn           PERMANENT TAGS:  TT - top of file
TAG DELETE:   TDn                            TE - end of file
USER TAG:     Tn                             TB - start of text in memory
                    n = 0 to 9               TZ - end of text in memory

HEX ENTRY: Use ↑B as delimiter around hexadecimal string.

ADVANCED EDITING COMMANDS (Y/N) Y
```

Key-in Sequence	Comments
Y	The second Help screen contains information on command line editing features. Press Y to continue the Help display.



```
                                                  DELETE: MD name|*
MACROS: DEFINE:  MS name/commands/                MF name[(arg1[, ..argn])]
        EXECUTE  {command mode}:                  ↑F name
                 (screen mode):
                 (screen or command
                 mode):                           ↑name
                 DISPLAY:                         ?M
         GET AND EXECUTE COMMAND FILE: G filename

                                                 filenamCLOSE:
FILES: OPEN:          OR(OW)                     e

CR (CW)                                          WRITE:  W[n|-n|tag]
        READ:         R[n]
        BEGINNING:    B

QUERY:  USER: QU
        USER FLAG:          QT; [<]command[>]   QF; '<]command[>]
        YESFOUND FLAG:  YT; [<]command[>]   YF; [<]command[>]

ITERATIVE LOOP:  [n| ]<cmd[;...cmd]>
EXIT LOOP:       EL
USER MESSAGE:    U /text/

DISPLAY ALTER VALUES: ?A

ALTER COMMANDS (Y/N) Y
```

Key-in Sequence	Comments
Y	The third screen contains Help on advanced editing features. Press Y to continue the Help display.



```
AFMU=4hex; AFMD=4hex; AFMR=4hex; AFML=4hex; AFMH=4hex Cursor
    up         down        right       left       home    output codes
AFCU=4hex; AFCD=4hex; AFCR=4hex; AFCL=4hex; AFCH=4hex Keyboard
                                                      input codes

AFXv=4hex  for v = A,C,D,F,N,P,V Replacements for control char inputs
AFES=4hex; AFER=4hex  Outputs for clear screen and clear rest of screen
AFEK=4hex; AFEL=4hex  Outputs for clear line and clear rest of line
AFMB=4hex             Output for move to beginning of line
AFAC-4hex
AFAC=4hex  Output for move the cursor to specified coordinates
AX=T|F     The column coordinate comes first
AO=hex     Offset value to be subtracted from both given coordinates

AFWA=char; AFWC=char;  AFWJ=char  Replacements for ↑Y, ↑W, ? jokers
AB=hex; AQ=hex         Break key; Literalize character
AT=decimal; AV=decimal Tab-size;  Number of lines on screen
AC=char; AL=char      Displays for non-printing; end-of-line chars
AS= T|F; AW = T|F     Suppress NOT FOUND; Cursor wraps around screen
                                          end
AFBK=char; AFIG=hex   Blankout char;      Char to be ignored on key
                                          input
AFXX=4hex; AFDL=4hex; AFIL=4hex           Other editors' settings ig-
                                          nored by CREDIT
*
```

Comments
The fourth and final screen contains information on Alter commands. The Credit prompt is returned at the end of the Help display so other commands can be entered.

CNTL V

Switch To Screen Mode

The format of the CNTL-V command is:

CNTL-V

In command line mode, the CNTL-V command changes from command line mode to screen editing mode.

The CNTL-V should be the first and only command on a screen line or the last command after the last semicolon on a screen line of multiple commands.

Pointer Commands

The editor keeps track of its position in a text file with a marker called a pointer. The pointer always points to some character or to a special end-of-file marker that follows the last character of the file.

In screen mode, the cursor represents the pointer when it is over a character, and the cursor control keys move the pointer through the file.

In command mode, there is no visual representation of the pointer on the screen. There are two commands which explicitly move the pointer through the text file: the line command and the jump command. Other editing commands such as the Search group of commands also affect the pointer.

L

Line Command

The format of the line command is:

```
L[<number>]
```

where

L	is the command name.
<number>	specifies.the number of lines the pointer should move.

The Line command moves the pointer a specified number of lines forward or backward in the file and positions it at the first character in the line.

<number>	Values of the Parameter
omitted	If <number> is omitted, the editor assumes a value of one and moves the pointer to the beginning of the next line.
negative	A minus character preceding the number indicates that the number is negative, and the editor moves the pointer backwards the number of lines specified.
positive	If the number is positive, no sign is needed, and the pointer is moved forward the specified number of lines.
zero	If the number is zero, the pointer is moved to the first character of the current line.

To move the pointer five lines forward, enter:

```
L5
```

To move the pointer to the beginning of the current line, enter:

```
L0
```

To move the pointer five lines backward, enter:

```
L-5
```

J

Jump Character Command

The format of the Jump character command is:

$$J \left[\begin{Bmatrix} \text{<number>} \\ \text{<tag>} \end{Bmatrix} \right]$$

where

J	is the command name.
<number>	specifies the number of characters the pointer moves.
<tag>	specifies any defined tag, either a system tag or a user defined tag.

The Jump command moves the pointer a specified number of characters forward or backward in the file or to a specified tag. The RETURN character counts as two characters for the Jump command. The parameters <number> and <tag> cannot both be specified.

<number>	Values of the Parameter
omitted	If <number> is omitted and no <tag> is specified, the editor assumes the number one and moves the pointer one character forward.
negative	A minus character preceding the number indicates that the number is negative, and the editor moves the pointer backwards the number of characters specified.
positivel	If the number is positive, no sign is needed, and the pointer is moved forward the specified number of characters.
zero	If the number is zero, the pointer is not moved.

To move the pointer 18 characters forward, enter:

```
J18
```

To move the pointer 10 characters backward, enter:

```
J-10
```

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

```
JTT
```

If the pointer marks the character preceding the RETURN, the command:

```
J3
```

is required to move the pointer to the beginning of the next line. In screen mode, the carriage return/linefeed combination is displayed as a single character even though it is stored in the file as two characters and counts as two characters for commands.

The command:

```
J0
```

is a null command.

TAG COMMANDS

Besides the pointer, the editor provides up to 14 additional markers in the file called tags. There are four system tags in each file that cannot be deleted or changed by the user. The system tags are:

- TT - Beginning of file
- TE - End of file
- TB - Beginning of text currently in system memory; a large file may not fit into the available memory
- TZ - End of text currently in system memory; a large file may not fit into the available memory

In addition to the four permanent tags, the Tag commands create and delete up to 10 user-defined markers, called T0 through T9.

TS

Tag Set Command

The format of the Tag Set command is:

```
TS<n>
```

where

TS	is the command name.
<n>	is the tag number. It can be any digit from 0 to 9.

The Tag Set command associates one of the 10 user tags with the character at the current pointer location.

Once the tag is set at the current pointer location, the pointer can be moved, but the tag remains at that character. The character can then be recalled by commands using the tag.

Defining a tag with a number that is already in use results in the deletion of the existing tag and the creation of the new one. Deleting the character to which a tag points does not delete the tag, but rather moves it to the next character in the file. Use the Tag Delete command (TD) to delete the tag.
The user defined tags are only in effect during an editing session or until they are re-defined. They are not saved after the EX command.

To set tag T3 to the current pointer location:

```
TS3
```

TD

Tag Delete Command

The format of the Tag Delete command is:

```
TD<n>
```

where

TD	is the command name.
<n>	is the number of the tag to be deleted. It can be any digit from 0 to 9.

The Tag Delete command deletes an existing tag. This command cannot delete any of the four permanent tags (TB, TE, TT, or TZ). If a tag number that has not been created with the Tag Set command is deleted, no action is taken; the delete command is ignored.

To delete tag T3, enter:

```
TD3
```

Text Commands

The text commands affect the text in a file by printing existing text, inserting new text, moving text, and copying text.

The text commands are:

- Print Command (P)
- Print Hexadecimal Command (PH)
- Insert Command (I)
- Delete Line Command (DL)
- Delete Character Command (DC)
- Move Command (XM)
- Copy Command (XC)

P

Print Command

The format of the Print command is:

$$P\left[\begin{Bmatrix}\text{<number>}\\ \text{<tag>}\end{Bmatrix}\right]$$

where

P	is the command name.
<number>	is the number of lines to be printed.

<tag> indicates that all lines from the current pointer to the tag specified are to be printed.

The Print command displays one or more lines of text on the screen. The parameters <number> and <tag> cannot both be specified.

<number>	Values of the Parameter
omitted	If <number> is omitted and no <tag> is specified, the editor prints the entire current line (the line containing the pointer) from beginning to end.
negative	A minus character preceding the number indicates that the number is negative, and the editor prints the number of lines specified preceding the pointer.
positive	If the number is positive, no sign is needed and the editor prints the specified number of lines from the pointer forward. If a value of 1 is specified, the editor prints from the character pointed to through the end of the line.
zero	If the number is zero, the current line is printed from the first character down to but not including the character pointed to.

To print three lines, beginning with the current line, enter:

```
P3
```

To print the 3 lines preceding the current line, enter:

```
P-3
```

To print from the current line to the end of the file, enter:

```
PTE
```

To print from the current line to tag 3, enter:

```
PT3
```

PH

Print Hexadecimal Command

The format of the Print Hexadecimal command is:

$$\mathrm{PH}\left[\begin{Bmatrix}\text{<number>}\\ \text{<tag>}\end{Bmatrix}\right]$$

where

PH	is the command name
<number>	is the number of bytes to be printed.
<tag>	indicates that all bytes from the current pointer to the tag specified are to be printed.

The Print Hexadecimal command displays one or more bytes as hexadecimal values for the ASCII character. A listing of the hexadecimal values for the ASCII character set is in Appendix C. The parameters <number> and <tag> cannot both be specified.

With the Print command, data is displayed, and special codes such as RETURNs perform their functions; with the Print Hexadecimal command, all codes including RETURNs and linefeeds are displayed as hexadecimal bytes with single spaces between the codes.

<number>	Values of the Parameter
omitted	If <number> is omitted and no <tag> is specified, the editor assumes a value of one and prints the character currently pointed to.
negative	A minus character preceding the number indicates that the number is negative, and the editor prints the number of characters specified preceding the character pointed to but not including the character pointed to.
positive	If the number is positive, no sign is needed and the editor prints the specified number of characters from the character pointed to forward. The character pointed to is included.
zero	If the number is zero, nothing is printed.

For the text "A line of hex.", the Print command displays:

```
A line of hex.
```

The Print Hexadecimal command (PH16) displays:

```
41 20 6C 69 6E 65 20 6F 66 20 68 65 78 2E 0D 0A
```

The RETURN key is displayed as 0D 0A rather than as a single character that performs the function of carriage return/linefeed.

To print three bytes, beginning with the current location of the pointer, enter:

```
PH3
```

To print the three bytes that precede the pointer, enter:

```
PH-3
```

To print from the pointer location to, but not including, the byte marked by tag 4, enter:

```
PHT4
```

I

Insert Command

The format of the Insert command is:

```
I<delimiter><text><delimiter>
```

where

I	is the command name.
<delimiter>	is any character except space, carriage return, linefeed, the literalizing character (default of backslash), escape, or a character appearing in <text>. Both occurrences of the <delimiter> must be the same.
<text>	is one or more characters to be entered into the file. The <text> must be delimited. The <text> can be ASCII characters entered from the keyboard or hexadecimal values if CNTL-B is used as the delimiter.

The Insert command inserts text to the left of the character currently pointed to. The pointer is not moved by the command.

If no starting delimiter is typed, the first character after the I is interpreted as the delimiter, and succeeding characters are interpreted as text until a match for the first character is typed. Unexpected results can occur.

If no ending delimiter is typed, the same thing can happen. A RETURN will not terminate the insert string, and the editor continues searching the text for the next occurrence of the initial delimiter character. A maximum of 2000 characters can be typed for the text; the command buffer fills and the editor terminates the command automatically.

Every character entered between delimiters is stored in the file as text, including carriage returns, linefeeds, control characters, cursor control codes, and so on.

To insert the line

```
LOOP: MOV A,M ;SAVE THE VALUE
```

into a file, first position the pointer at the character following the point of insertion. This can be done with the Line and Jump commands. Then, enter the Insert command as follows:

```
I"LOOP: MOV A,M ;SAVE THE VALUE[RETURN]
"[RETURN]
```

Be sure to enter the RETURN at the end of the line before the closing delimiter.

DL

Delete Line Command

The format of the Delete Line command is:

$$\mathrm{DL}\left[\begin{Bmatrix}\text{<number>}\\ \text{<tag>}\end{Bmatrix}\right]$$

where

DL	is the command name
<number>	specifies the number of lines to delete from the line where the pointer is currently.
<tag>	specifies the end of a block of characters to delete. DL<-tag> is equivalent to DC<tag>. The beginning of the block is the character at the current pointer.

The Delete Line Command removes one or more lines of text from the file.

<number>	Values of the Parameter
omitted	If <number> is omitted, the entire line containing the pointer is deleted. The pointer is moved to the first character of the following line.
negative	A minus character preceding the number indicates that the number is negative, and the deletion begins at the character immediately preceding the pointer and goes backward. The pointer is not changed.
positive	If the number is positive, no sign is needed, and the deletion begins at the character pointed to and goes forward. The first line counted is from the pointer to the end of the line. The pointer is moved to the first character following the deleted text.
zero	If the number is zero, the current line up to but not including the pointer is deleted. The pointer is not changed.

To delete three lines beginning at the pointer, enter:

```
DL3
```

To delete the three lines preceding the pointer, enter:

```
DL-3
```

To delete text from the beginning of the line up to the pointer, enter:

```
DL0
```

To delete text from the character following the pointer to the line terminator, enter:

```
DL1
```

DC

Delete Character Command

The format of the Delete Character command is:

$$\text{DC}\left[\begin{Bmatrix}\text{<number>}\\\text{<tag>}\end{Bmatrix}\right]$$

where

DC	is the command name.
<number>	specifies the number of characters to delete from the character at the current pointer.
<tag>	can be either a permanent or a user-defined tag. The tag specifies one boundary of a block of characters to delete. The other boundary of the block is the character at the current pointer.

The Delete Character command removes one or more characters from a file. After a deletion, the pointer is positioned immediately following the last character deleted. The parameters <number> and <tag> cannot both be specified.

<number>	Values of the Parameter
omitted	If <number> is omitted and no <tag> is specified, the editor assumes the number one and deletes the character pointed to. The pointer is moved to the next character.
negative	A minus character preceding the number indicates that the number is negative, and the editor deletes the <number> of characters preceding the pointer. The character pointed to remains.
positive	If the number is positive, no sign is needed, and the specified number of characters starting with the character pointed to are deleted. The pointer is moved to the first character after the deleted text.
zero	If the number is zero, no characters are deleted.

To delete 10 characters beginning with the character at the pointer, enter:

```
DC10
```

To delete 15 characters preceding the pointer, enter:

```
DC-15
```

To delete all the characters from the pointer to the end of the file, enter:

```
DCTE
```

To delete the block of characters from the pointer to tag 9, enter:

```
DCT9
```

XM

Move Command

The format of the Move command is:

$$\text{XM} <\text{tag1}>,\ \left\{\begin{array}{c} <\text{number}> \\ <\text{tag2}> \end{array}\right\}$$

where

XM	is the command name.
<tag1>	specifies the beginning of the block of text to be moved. The character at <tag1> is included in the move.
<tag2>	specifies the end of the block of text to be moved. The character at the second tag is not included in the move. When all the text involved with the move is in memory, the <tag1>,<tag2> form is faster than the <tag1>,<number> form of the command.
<number>	specifies the number of lines to be moved relative to <tag1>

The Move command deletes a block of text from its current location and inserts it immediately to the left of the current pointer location. The parameters <number> and <tag2> cannot both be specified.

The text block specified is moved to the character preceding the current pointer, i.e., in front of the character currently pointed to. Tags associated with the data being moved (either the first character of the block or within the block) are moved with the data. For example, if T3 is set for a character within the block of text being moved, T3 will mark the same character after the move in its new location.

After the move, the pointer marks the same location as it did before the move, i.e., the pointer is located at the first character following the moved text. The text is no longer at its old location.

The permanent tags TT, TE, TZ, and TB can be specified as tags in the XM command. However, the results on the file are unpredictable, because tags do not move with the block.

NOTE

> The pointer must not be located within the data being moved. However, CREDIT does not issue any error or warning if the text to be moved includes the current pointer.

If the second tag specified precedes the first tag in the file, the text from the second tag to the first tag is moved, not including the character at the first tag. The tag occurring first in the file is treated as <tag1> even if it is entered second on the command line.

<number>	Values of the Parameter
omitted	If <number> is omitted and no <tag> is specified, the editor returns an error message. It is not valid to omit <number> and <tag>.
negative	A minus character preceding the number indicates that the number is negative, and the <number> of lines preceding <tag1> are moved. The character marked by <tag1> is not included.
positive	If the number is positive, no sign is needed, and the <number> of lines specified, beginning with the character marked by <tag1>, are moved.
zero	Zero is not a valid entry. An error is given if zero is entered.

To move the 11 lines of text starting at tag 8 to the current location of the pointer, enter:

```
XM T8,11
```

To move the text bounded by tag 3 and tag 6 to the current location of the pointer, enter:

```
XM T3,T6
```

If T3 points to a character that follows T2 in the file, the command:

```
XM T3, T2
```

is the same as the command:

```
XM T2,T3
```

XC

Copy Command

The format of the Copy command is:

$$\text{XC}<\text{tag1}>,\ \left\{\begin{matrix}<\text{number}> \\ <\text{tag2}>\end{matrix}\right\}$$

where

XC	is the command name.
<tag1>	specifies the beginning of the block of text to be copied.
<tag2>	specifies the end of the block of text to be copied. The character at the second tag is not included in the copy. When all the text involved with the copy is in memory, the <tag1>,<tag2> form is faster than the <tag1>,<number> form of the command.
<number>	specifies the number of lines to be copied relative to the pointer.

The Copy command duplicates the specified block of text at the current location of the pointer in the file. The parameters <number> and <tag2> cannot both be specified.

The text block specified is copied to the character preceding the current pointer, i.e., in front of the character currently pointed to. Tags associated with the data are not copied with the data. For example, if T3 is set for a character within the block of text being moved, T3 will mark the same character after the copy. It will not mark a new location at the copied text.

After the copy, the pointer marks the same location as it did before the copy, i.e., the pointer is located at the first character following the copied text. The text remains at its old location.

NOTE

The pointer must not be located within the data being copied. However, CREDIT does not issue any error or warning if the text to be moved includes the current pointer.

If the second tag specified precedes the first tag in the file, the text from the second tag to the first tag is copied. The tag occurring first in the file is treated as <tag1> even if it is entered second on the command line.

<number>	Values of the Parameter
omitted	If <number> is omitted and no <tag> is specified, the editor returns an error. It is not valid to omit both <number> and <tag>.
negative	A minus character preceding the number indicates that the number is negative, and the editor copies the <number> of lines preceding the character marked by <tag1>, but not including the character marked by <tag1>.
positive	If the number is positive, no sign is needed, and the number of lines specified, beginning with the character marked by <tag1>, are copied.
zero	Zero is not a valid entry. An error is returned if zero is entered.

To copy the 20 lines of text starting at tag 7 to the current location of the pointer, enter:

```
XCT7,20
```

To copy the text bounded by tag 4 and tag 9 to the current location of the pointer, enter:

```
XCT4,T9
```

Search Commands

There are two search commands available: the Find command and the Substitute command. Both commands accept ranges of text and wildcard characters.

A range is an area of text specified as the boundary for a search. The editor only searches for the text within the boundary specified. A range is defined relative to the current pointer. It extends a specified number of lines from the pointer or from the pointer to a specified tag.

There are three wildcard characters which represent other characters in search commands.

? means match any character. For example,

```
ABC?E
```

matches ABCDE, ABCPE, ABC8E, ABC*E, or any string with A, B, C, and E in positions 1, 2, 3, and 5 and with any character in position 4.

CNTL Y followed by a character matches any number of the character which follows it. For example,

```
B CNTL-Y AD
```

matches BD, BAD, BAAD, BAAAD, or any string with B in the first position, D in the last position, and any number of A's in between, including zero A's.

CNTL W encloses a sequence of characters and matches either upper or lower case for the characters enclosed. For example,

```
CNTL-W mhz CNTL-W
```

matches mhz, Mhz, mHz, mhZ, MHz, MhZ, mHZ, or MHZ.

F

Find Command

The format of the Find command is:

$$\text{F<delimiter><string><delimiter>}\left[\begin{Bmatrix}\text{<number>}\\ \text{<tag>}\end{Bmatrix}\right]$$

where

F	is the command name.
<delimiter>	is any character except space, carriage return, linefeed, the literalizing character (default of backslash), escape, or a character appearing in <string>. Both occurrences of the <delimiter> must be the same.
<string>	specifies the character string to be found. The specified string must be enclosed within a valid delimiter. The string can be any ASCII character or a hexadecimal value enclosed by CNTL-B.
<tag>	is optional and specifies a valid range for the search. If <tag> is specified, the editor searches from the current pointer position to <tag>. If <tag> precedes the current pointer, the search is from the <tag> to the current pointer. If <tag> is not found, the editor searches to the end of the file.
<number>	is also optional and specifies a valid range for the search as a <number> of lines relative to the pointer.

The Find command searches for a character string. If the string is found, the pointer is moved to the character immediately following the string. Both <number> and <tag> cannot be specified.

If the string cannot be found, the editor displays a NOT FOUND message and the pointer remains where it was preceding the command. The NOT FOUND message can be suppressed, so it will not be displayed. See the Alter command described in Advanced Editing Techniques for information on suppressing the display of messages.

The Find command can be used in loops as described in Advanced Editing Techniques.

<number>	Values of the Parameter
omitted	If <number> is omitted and no <tag> is specified, the range of the search is between the pointer and the end of the file, including the character pointed to.
negative	A minus character preceding the number indicates that the number is negative, and the range for the search is the <number> of lines preceding the line containing the pointer.
positive	If the number is positive, no sign is needed, and the range is the number of lines following the line containing the pointer.
zero	If the number is zero, the editor searches from the beginning of the current line to the pointer. The character pointed to is not included in the range.

To find the string "LOOP4" located somewhere between the pointer and the end of the file, enter:

```
F"LOOP4"
```

To find the same string located somewhere between the pointer and tag 7, enter:

```
F"LOOP4"T7
```

To find the same string located somewhere in the 23 lines preceding the pointer, enter:

```
F"LOOP4"-23
```

S and SQ

Substitute Commands

The format of the Substitute command is:

```
                                                        [ {<number>} ]
S[Q]<delimiter><oldtext><delimiter><newtext><delimiter> [ {        } ]
                                                        [ { <tag>  } ]
```

where

S[Q]	is the command name. S causes the substitution to be made if <oldtext> is found. SQ causes the editor to prompt when the old text is found before the substitution is made. When prompted, Y or y (for yes) causes the substitution to be made. Any other response cancels the command.
<delimiter>	is any character except space, carriage return, linefeed, the literalizing character (default of backslash), escape, or a character appearing in <oldtext>. All three occurrences of the <delimiter> must be the same.
<oldtext>	specifies the character string to be replaced if found. The <oldtext> must be enclosed within a valid delimiter. The <oldtext> can be any ASCII character or a hexadecimal value enclosed by CNTL-B and can be any length except zero.
<newtext>	specifies the replacement character string. The <newtext> can be any length including zero. When the <newtext> is a null string (length of zero), the <oldtext> is deleted. The <newtext> must be followed by a valid delimiter character even if it is a null string.
<tag>	is optional and specifies a valid range for the search. If <tag> is specified, the editor searches from the current pointer position to the <tag>. If <tag> is not found, the editor searches to the end of the file.
<number>	is also optional and specifies a valid range for the search as the <number> of lines relative to the line containing the pointer.

The Substitute command searches for a character string and substitutes a new string for the old one. There are two forms of the command: S and SQ. If the string is found, the pointer is moved to the character immediately following the string. Both <number> and <tag> cannot be specified. Wildcard characters can be specified with the Substitute command. The Substitute command can be used in loops as described in Advanced Editing Techniques.

If the string cannot be found, the editor displays a NOT FOUND message and the pointer remains where it was preceding the command. The NOT FOUND message can be suppressed. See the Alter command described in the "Advanced Editing Techniques" section for information on suppressing the display of messages.

If the ESC key is pressed while the editor is making a substitution, searching stops but any substitution in progress is completed and any substitutions already made remain.

<number>	Values of the Parameter
omitted	If <number> is omitted and no <tag> is specified, the range of the search is between the pointer and the end of the file. The character pointed to is not included in the range.
negative	A minus character preceding the number indicates that the number is negative, and the range for the search is the <number> of lines preceding the line containing the pointer.
positive	If the number is positive, no sign is needed, and the range is the number of lines following the line containing the pointer.
zero	If the number is zero, the editor searches from the beginning of the current line to the pointer. The character pointed to is not included in the range.

To change the first occurrence of "THEN:" to "NEXT:", enter:

```
S"THEN:"NEXT:"
```

To make the same change with the query option, enter:

```
SQ"THEN:"NEXT:"
```

The editor prompts by printing the line containing the string and a question mark:

```
THEN: A ?
```

Y or y causes the substitution; any other reply cancels the substitution.

To make the same change if the string is in the next 18 lines of text, enter:

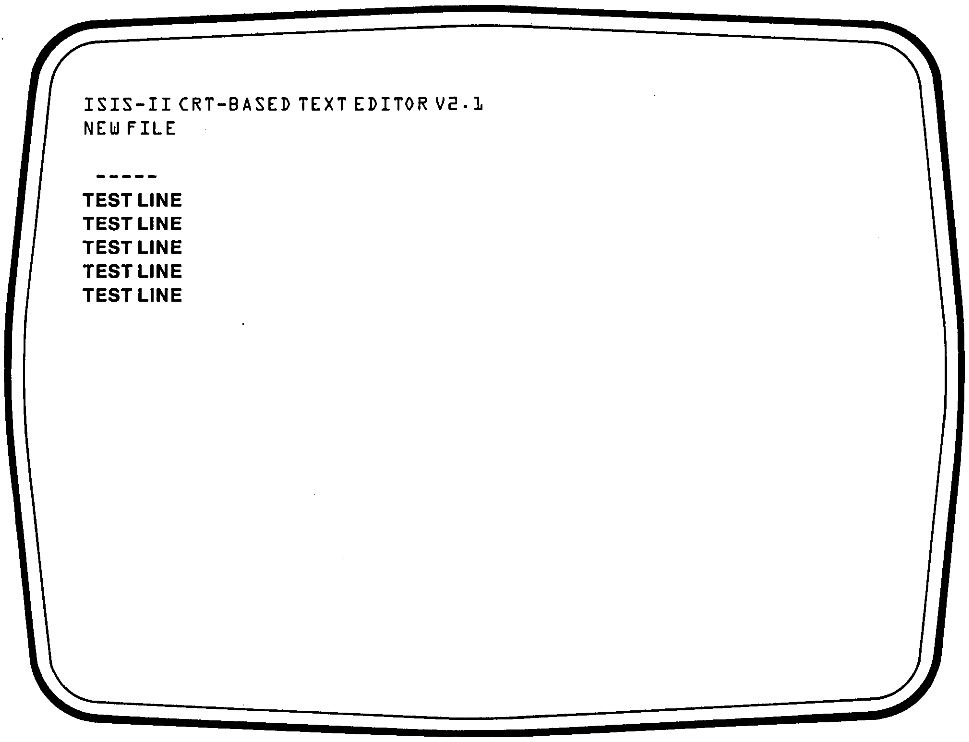
```
S"THEN:"NEXT:"18
```

To make the same change if the string is between the pointer and tag 6, enter:

```
S"THEN:"NEXT:"T6
```

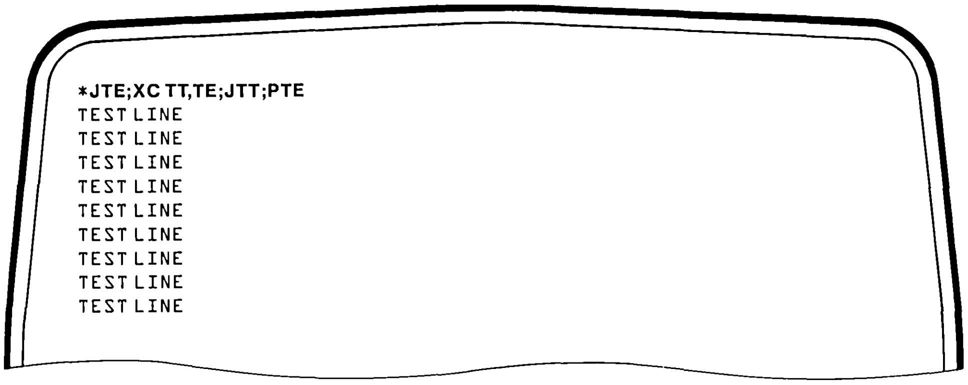
Sample Command Line Editing

In this section, the editing commands are used to edit a new source program . Enter: CREDIT TEST.TXT(Return).



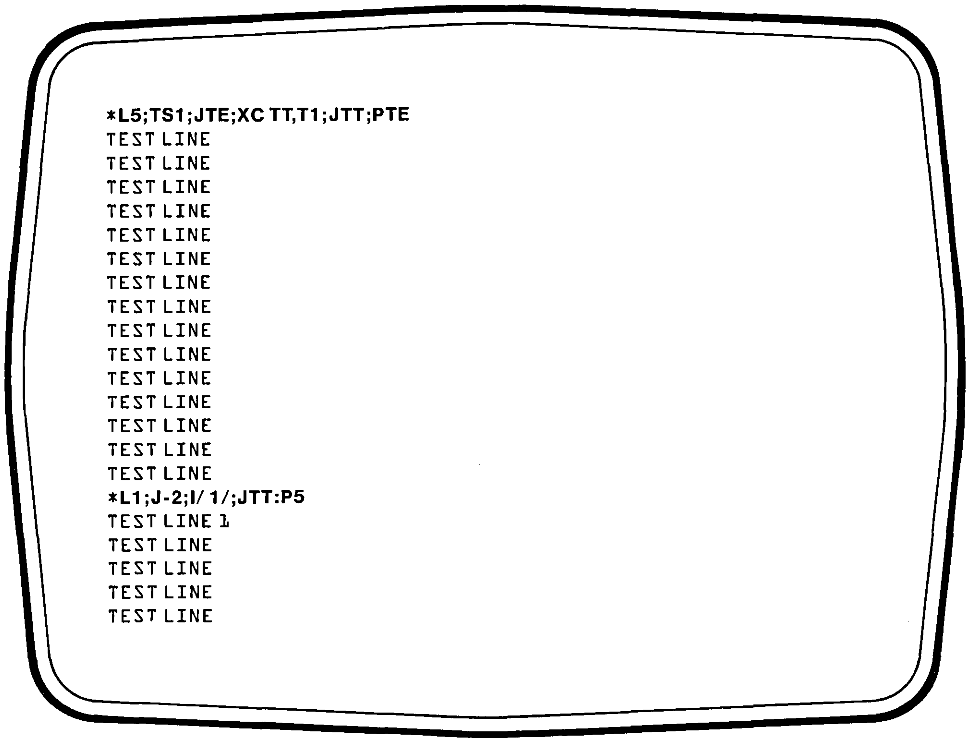
```
ISIS-II CRT-BASED TEXT EDITOR V2.1
NEW FILE

 -----
TEST LINE
TEST LINE
TEST LINE
TEST LINE
TEST LINE
```

Key-in Sequence	Comments
CREDIT TEST. TXT	This command creates a new file to be edited in the CREDIT command line mode. CREDIT commands are used primarily in macros and submit files as illustrated in the following section on Advanced Editing Techniques. However, the next series examples show how to exercise these command lines in a test file.
TEST LINE **TEST LINE** **TEST LINE** **TEST LINE** **TEST LINE**	Enter the line TEST LINE five times as shown above.



```
*JTE;XC TT,TE;JTT;PTE
TEST LINE
TEST LINE
TEST LINE
TEST LINE
TEST LINE
TEST LINE
TEST LINE
TEST LINE
TEST LINE
```

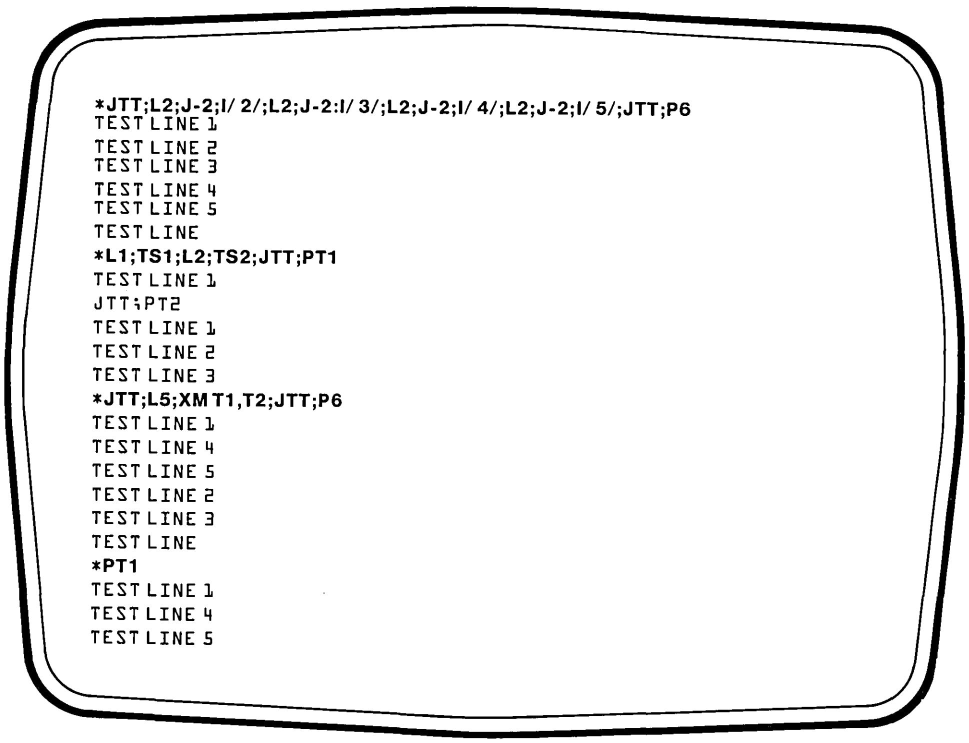
Key-in Sequence	Comments
HOME **JTE;**	Then, switch to command line mode by pressing HOME key. Enter the command JTE. This command moves the file pointer to the end of the file. Notice that the text from the screen mode of editing remains on the screen as a residual display.
XC TT, TE;	The XC command makes a copy of the text between the beginning of file tag (TT) and the end of file tag (TE at the position of the file pointer (moved to the end of the file by the previous command). The text displayed in the text area is still a residual display and is not updated while in command line mode.
JTT; **PTE** RETURN	To see the effects of the copy without switching to screen mode, first, enter the JTT command. This moves the file pointer to the beginning of the file (TT). The PTE command displays the lines of the file from the current file pointer (just set to TT) until the end of the file (TE). As shown above, there are ten TEST LINE's. Notice that the residual text from the screen mode disappears as the commands scroll down.



```
*L5;TS1;JTE;XC TT,T1;JTT;PTE
TEST LINE
TEST LINE
TEST LINE
TEST LINE
TEST LINE
TEST LINE
TEST LINE
TEST LINE
TEST LINE
TEST LINE
TEST LINE
TEST LINE
TEST LINE
TEST LINE
TEST LINE
*L1;J-2;I/ 1/;JTT:P5
TEST LINE 1
TEST LINE
TEST LINE
TEST LINE
TEST LINE
```

Key-in Sequence	Comments
L5; TS1;	To add five more TEST LINEs to the file, first, move the file pointer five lines with the L5 command. Then, set a user-defined tag at this line TS1.
JTE; XC TT, T1;	Move the file pointer to the end of the file with the JTE command. Then make a copy of the text between the beginning of the file (TT) and tag 1 (T1) to the current file pointer (set at the end of the file). Now, the file should contain 15 lines of TEST LINE.
JTT; PTE RETURN	To see the results, enter the JTT command to move the file pointer to the beginning of the file (TT). Then display from the file pointer to the end of the file (TE) with the PTE command.
L1; J-2; I/*SPACE*1/;	This command sequence shows how to insert a character at the end of the first line. The L1 command moves the file pointer one line ahead. Since the file pointer is at the beginning of the file, the L1 command results in moving the pointer to the beginning of the second line. The J-2 command moves the pointer to the end of the previous line by jumping back two characters. Since the line terminator character is actually two bytes (0AH and 0DS), this command results in moving the file pointer just before the line terminator characters of the previous line. The insert command (I/ 1) inserts the character 1 at this point.

JTT; **P5** RETURN	To see the results of the insert, use the JTT command to jump to the beginning of the file, and, then, print five lines with the P5 command.



```
*JTT;L2;J-2;I/ 2/;L2;J-2:I/ 3/;L2;J-2;I/ 4/;L2;J-2;I/ 5/;JTT;P6
TEST LINE 1
TEST LINE 2
TEST LINE 3
TEST LINE 4
TEST LINE 5
TEST LINE
*L1;TS1;L2;TS2;JTT;PT1
TEST LINE 1
JTT;PT2
TEST LINE 1
TEST LINE 2
TEST LINE 3
*JTT;L5;XM T1,T2;JTT;P6
TEST LINE 1
TEST LINE 4
TEST LINE 5
TEST LINE 2
TEST LINE 3
TEST LINE
*PT1
TEST LINE 1
TEST LINE 4
TEST LINE 5
```

Key-in Sequence	Comments
JTT;L2;J-2;I/*SPACE*2/;	This sequence of commands inserts the 2 at the end of the second line.
L2;J-2;I/*SPACE*3/; **L2;J-2;I/*SPACE*4/;** **L2;J-2;I/*SPACE*5/;**	The sequence of commands shown here inserts a digit at the end of lines 3, 4, and 5.
JTT; **P6** RETURN	To verify these insertions, jump to the beginning of the file with the JTT command and print the first six lines with the P6 command.

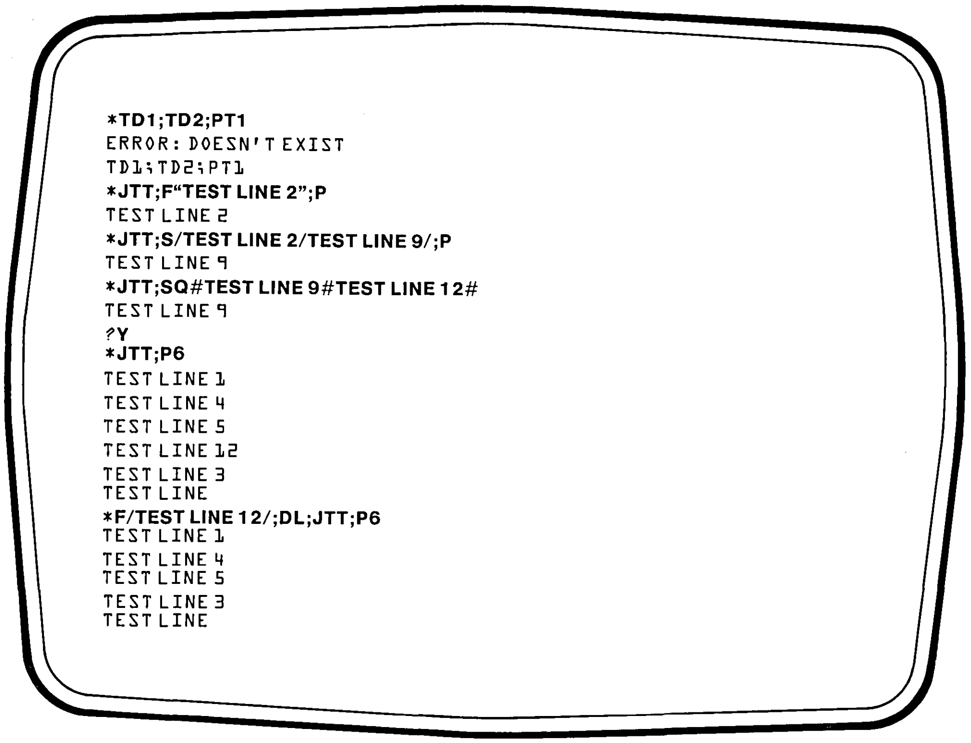
L1;TS1;
L2;TS2;
	In this example the file is being set up to move TEST LINE 2 and TEST LINE 3 into the middle of the file. First, a tag is set at the start of TEST LINE 2 by moving the file pointer one line ahead from the beginning of the file and setting tag 1. Then, the pointer is moved an additional two lines ahead, to the beginning of TEST LINE 2 and tag 2 is set.

JTT;PT1

The next step in moving TEST LINE 2 and TEST LINE 3 is to move the file ponter to the destination. The L5 command accomplishes this. Then, perform the move with XM T1, T2. This command moves the text between tag 1 and tag 2 to the destination.

JTT;P6 RETURN PT1

To verify the move, jump to the beginning of the file and print six lines. The PT1 command prints down to tag 1. This command verfies that the tag was moved along with the text.



```
*TD1;TD2;PT1
ERROR: DOESN'T EXIST
TD1;TD2;PT1
*JTT;F"TEST LINE 2";P
TEST LINE 2
*JTT;S/TEST LINE 2/TEST LINE 9/;P
TEST LINE 9
*JTT;SQ#TEST LINE 9#TEST LINE 12#
TEST LINE 9
?Y
*JTT;P6
TEST LINE 1
TEST LINE 4
TEST LINE 5
TEST LINE 12
TEST LINE 3
TEST LINE
*F/TEST LINE 12/;DL;JTT;P6
TEST LINE 1
TEST LINE 4
TEST LINE 5
TEST LINE 3
TEST LINE
```

Key-in Sequence **Comments**

TD1;TD2;PT1 RETURN

To delete the two tags, use the TD1 and TD2 command. to verify that the tags were deleted, try to print down to tag 1 again. An error message is displayed.

JTT;F"TEST LINE 2 "; P RETURN

Move the pointer to the beginning of the file and search for the string TEST LINE 2. Double quotes are used for the string delimiters. The pointer is moved to the character following this string which is the first character of TEST LINE 3. Thus, the P command displays TEST LINE 3.

JTT;S/TEST LINE / TEST LINE 9/; P RETURN

This command sequence illustrates the substitute command. The string TEST LINE 9 is substituted for the string TEST LINE 2. Then, the line is displayed with the print command.

JTT;SQ#TEST LINE 9# TEST LINE 12# RETURN

The Substitute command also operates in a query mode so that the user is prompted when the string is found before the substitution is made. Here, the # character is used as a delimiter.

Y RETURN

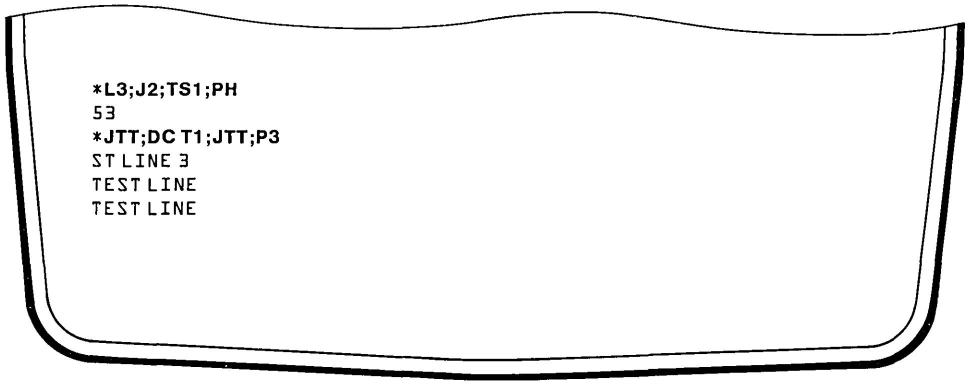
JTT;P6 RETURN

The editor prints the line found and prompts with a question mark. Type Y to make the substitution. To verify the change, jump to the beginning of the file and print the first six lines.

F/TEST LINE 12/; DL;

JTT;P6 RETURN

This example shows how to delete a line in command line mode. First, find the line, and then, use the DL command to delete it. To verify the deletion, jump to the beginning of the file and print the first six lines.



```
*L3;J2;TS1;PH
53
*JTT;DC T1;JTT;P3
ST LINE 3
TEST LINE
TEST LINE
```

Key-in Sequence **Comments**

L3;J2;TS1;PH RETURN

This command sequence moves the file pointer down to the third character of the third line of the file and then sets a tag at that character. Then, the ASCII value of the character is printed. In this case, the character is S, and the ASCII value is 53H.

JTT;DC T1;

JTT;P3 RETURN

This example illustrates the DC command. First, the file pointer is moved to the beginning of the file. Then, characters are deleted from the file pointer down to tag 1, set at the third character of the third line. To verify the deletion, move the file pointer to the beginning of the file and print the first three lines. This ends the demonstration of command line commands. Their actual use in macros is shown in the following chapter.

CHAPTER 4 ADVANCED TECHNIQUES

This chapter describes how to perform advanced editing operations using the CREDIT text editor. An advanced form of the ISIS command line to run CREDIT is also given.

Advanced Command Format

```
                                       [ { MACRO[(<command file>)] } ]
CREDIT <pathname1> [TO <pathname2>]    [ {                         } ]
                                       [ { NOMACRO                 } ]
```

where

<pathname1>	specifies a valid ISIS file or device pathname . The source text (the text to be edited) is contained in the file or device specified. More details on this part of the command line are given in previous sections.
TO <pathname2>	specifies the destination file which will contain the text after editing. More details on this part of the command line are given in previous chapters.

This form of the command line shows the optional parameter used for loading macros. Macros are described in more detail later in this chapter. A macro is a sequence of CREDIT commands that have been given a single name and can be executed by that single name.

Macros are executed from memory. They can be defined in memory using editing commands under CREDIT, or they can be loaded into memory from a disk file and executed using the optional parameter on the CREDIT command line. There are four possible ways this optional parameter can be entered.

<none>	Omitting the parameter specifies that the default macro command file is to be loaded and executed. Omitting the parameter is the same as typing MACRO without specifying the command file. However, no error results if the default command file (CREDIT.MAC) cannot be found.
MACRO	specifies that the default macro command file is to be loaded and executed. The default pathname of the command file is CREDIT.MAC on the same device as the CREDIT program. An error results if the file cannot be found, and CREDIT returns to ISIS.

MACRO(<commandfile>)	specifies that a macro command file with a valid ISIS pathname of <commandfile> is to be loaded and executed. The pathname of the command file must be enclosed in parentheses and must follow the parameter MACRO. An error results if the file cannot be found, and CREDIT returns to ISIS.
NOMACRO	specifies that no command file be loaded. NOMACRO is used when CREDIT.MAC exists but should not be loaded and executed.

Examples

To edit a file named MOD2.PLM on the system disk and not load a file of macros, enter:

```
CREDIT MOD2.PLM NOMACRO
```

This command is used if CREDIT.MAC exists but should not be loaded and executed.

To edit the file MOD2.PLM as well as load and execute a macro command file named PLM.MAC, enter:

```
CREDIT MOD2.PLM MACRO(PLM.MAC)
```

If PLM.MAC cannot be found, an error results and CREDIT returns control to the operating system.

To edit MOD2.PLM, store the updated version as MOD3.PLM, and use the default macro command file CREDIT.MAC, enter:

```
CREDIT MOD2.PLM TO MOD3.PLM MACRO
```

or

```
CREDIT MOD2.PLM TO MOD3.PLM
```

In the last example, no error results if CREDIT.MAC cannot be found.

Advanced Editing Basics

The advanced editing operations described in this section are performed in either the screen editing mode or the command line editing mode. The features of these two modes of operation are described in previous sections of this chapter. All the features, such as the keyboard and display features, disk file use, and command entry, apply to the advanced editing techniques described here.

Advanced Editing Functions

There are six major advanced editing functions:

- Macro facility to execute sequences of commands by a single name
- Command iteration to repetitively execute editing commands
- Conditional execution of editing commands
- Direct access of disk files from editor
- Compatibility with ISIS SUBMIT
- Ability to alter the environment in which the editor runs

Each of these topics is described in the following sections.

Macro Facility

A macro is a sequence of CREDIT commands that have been given a name. The macro text is the sequence of editing commands that make up the macro; the macro name is the single character name used to refer to the macro. Macros are stored in memory and are executed when they are called by name.

They are typically used for long command sequences that are executed often. Rather than enter the long series of CREDIT commands each time, the user can execute the predefined macro with a single command. The macro facility speeds the work and reduces typing errors that would occur in repeatedly typing a long series of commands.

A sample use for a macro might involve searching a file for a given string and inserting a line of code following that string. A macro to perform this function is shown in figure 4-1.

Four commands are associated with the CREDIT macro facility: Macro Set, Macro Function, Macro Delete, and Macro Display.

Macros are defined and named with the Macro Set command (MS). Macro are assigned single character names by the user.

Once a macro is defined, it can be executed either from the command mode of editing or from the screen mode with the Macro Function command. There are four forms of the Macro Function command.

To delete a macro, use the Macro Delete (MD) command.

To display a list of all currently defined macros, use the Display Macro (?M) command.

CREDIT Commands Within Macros

Macros can contain both screen editing functions and command editing commands. However, the commands must be executed in the correct mode. For example, a macro consisting of screen mode commands cannot be run in command mode. The first command in a macro of screen mode commands that is to be run from command mode should be a CNTL-V to switch to screen mode.

If a screen mode operation is attempted in command mode, it is ignored.

If a command mode operation is attempted when in screen mode, the macro text (the commands in the macro) replaces the existing data starting at the pointer location. This is exactly what would happen if commands were entered when in screen mode.

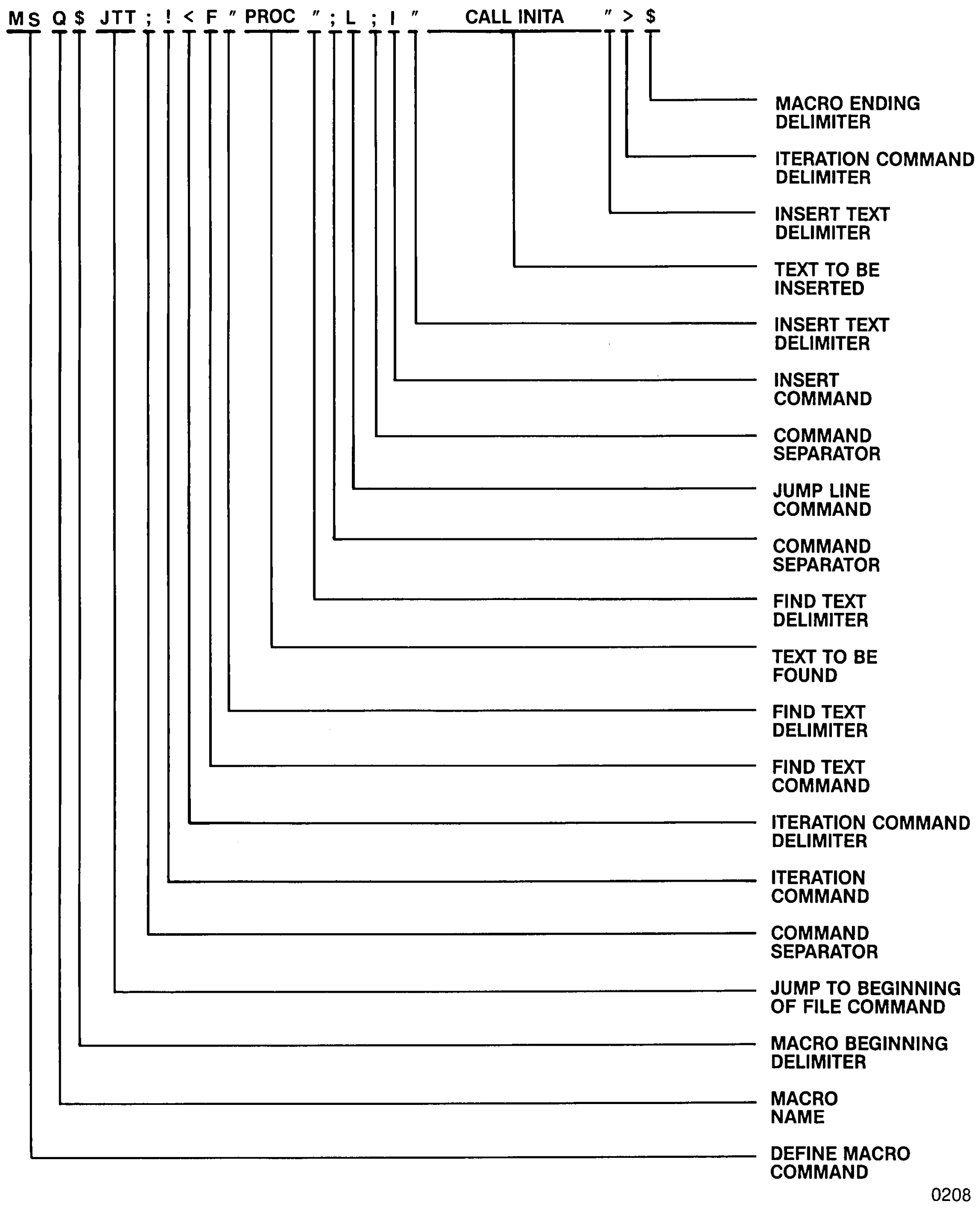


Figure 4-1 Sample Macro

Since any valid operation can be done within a macro, the editing mode can be switched from within the macro. The code for the HOME function causes the editor to change from screen mode to command mode. Likewise, a CNTL-V causes the editor to change from command mode to screen mode.

To write a macro that executes in either mode, use CNTL-V as the first command. CNTL-V is the only command that is valid in either mode. Following the CNTL-V, the macro can either execute screen mode commands or can issue the code for the HOME function to go to command mode.

Macros should end in the same mode from which they were initiated. If the macro is called from another macro or is part of a string of commands and it changes the mode, the command following the macro call will generate an UNRECOGNIZED COMMAND error. If a macro changes modes and does not change back, do not call that macro from another macro or use it as part of a string of commands.

Parameters

Parameters can be passed only to macros that are executed from the command mode of editing. The macro in figure 4-1 can be re-defined as:

```
MSQ$JTT;!<F"%";L;I"%">$
```

The percent signs (%) are placeholders for data to be supplied when the macro is called with the MF command. To use percent signs within the text of macros instead of as a placeholder, precede the percent sign with two literalizing characters (default of backslash). To invoke this macro and pass parameters to it, enter:

```
MFQ(PROC,CALL INITA)
```

The parameters are separated from one another by a comma. They are enclosed in parentheses and are entered on the MF command line. The parameters replace the percent signs in the macro definition. Thus, the following commands are executed when the Q macro is called:

```
JTT;!<F"PROC";L;I"CALL INITA">
```

There must be a parameter for every percent sign in the macro definition; an ARGUMENT MISMATCH error occurs if there are too few or too many parameters passed.

Another command macro that accepts parameters is a global substitution that changes all occurrences of a string in a file to another string and displays the changed string. The command string is:

```
JTT;!<S"%"%";P>
```

Use the MS command to define and name this command string. For example, to name it M, enter:

```
MSM$JTT;!<S"%"%";P>$
```

To execute it, use the MF command and pass it two parameters, the old string to be replaced and the new string to replace it. For example, enter:

```
MFM(JNZ,JZ)
```

Nested Macros

Macros can call other macros. However, if macro A calls macro B which in turn calls macro A, the system enters an endless loop. If either of these macros generate data or commands, the file could be filled or the maximum length command exceeded, causing an error. This condition should be avoided, but pressing the ESC key cancels the macro. Use of ESC to cancel a macro does not nullify the effects of the part of the macro that has already executed. It stops the macro from completing execution.

MS

Macro Set Command

The format for the Macro Set command is:

```
MS<name><delimiter><text><delimiter>
```

where

MS	is the command name.
<name>	is a single character name assigned to the macro. Each character can call only one macro. Any character, printing or non-printing can be used except carriage return, linefeed, escape, space, the literalizing character (default of backslash), or asterisk. For example, besides single alphanumeric characters, control characters can also be used as a name. Control characters are stored as single bytes.
<delimiter>	is any character except space, carriage return, linefeed, the literalizing character (default of backslash), escape, or a character appearing in <test>. Both occurrences of the <delimiter> must be the same.
<text>	is the sequence of editor commands executed when the macro is called. The sequence of commands must be enclosed by a delimiter character.

The Macro Set command (MS) is executed in the command mode of editing. It defines a group of editor commands as a macro and assigns a single letter name to that macro.

A macro can only be defined in command line mode. However, it can be executed from either mode. See the MF/CNTL-F command for executing a macro. Macros that are defined in an editing session are only retained for that session. They are not saved after the EX command. However, macros can be saved in a text file. The commands for loading macros from a file are described in the section "Accessing Disk Files from the Editor" later in this chapter.

A single character can name only one macro. If a macro already exists with a given name, that name cannot be used again. If the name is reused in a subsequent MS command, an error message is displayed, and the new macro is not defined.

If the name of the macro is a control character (a character entered while the CNTL key is held down) the macro can be executed simply by typing that control character without the MF command. The control character used to name a macro cannot be one of the control characters with a special meaning to the editor; i.e., it cannot be one of the screen mode commands. See the MF command for details on how to execute macros.

The sequence of commands in a macro can contain parameters that will be passed when the macro is called. Parameters are specified in the MS command with a percent sign (%) in place of each value that will be passed when the macro is called. Then, in the Macro Function command (MF), the value for each percent sign is specified enclosed in parentheses and separated by commas.

The values passed with the MF command are substituted for the percent signs in order: the first value is substituted for the first percent sign, the second value for the second percent sign, and so on. An error occurs if an extra parameter is passed or if no parameter is passed when one is expected.

To pass a null value to a parameter, enter commas with no text to hold the place of that parameter. For example, to pass a null value for the third percent sign, enter the first value, a comma, the second value, a comma, another comma, the fourth value, and so on.

To use a percent sign as text in the MS command, precede it with two literalizing characters (default of backslash). Then, it will not be interpreted as a parameter.

To define a macro named B that moves the pointer to the beginning of the file and prints the first ten lines, enter:

```
MSB"JTT;P10"
```

To define a macro named S that substitutes text that is passed as a parameter when the macro is executed, enter:

```
MSS/S"%"%";P/
```

To define a macro named M that moves the pointer to the beginning of the file and then enters the screen editing mode, enter:

```
MSM"JTT;
CNTL-V"
```

Note that the macro definition must follow every rule that applies if the command were entered directly.

To define a macro to execute in screen mode named CNTL-Y that moves the cursor to 79 spaces to the right side of the screen, enter:

```
MS CNTL-Y"CNTL-T CNTL-T . . . CNTL-T"
```

Do not enter spaces between the CNTL-T's. Note that the macro definition must contain 79 cursor right character sequences (CNTL-T). Macros of this type for all cursor directions provide a fast method of moving the cursor around the display area. This macro is faster than pressing the cursor control key with the RPT (repeat) key.

MF and CNTL F

Macro Function Command

There are three forms of the macro function command as follows:

```
{ MF<name>[(<parameter1>[,<parameter2>,...,<parameter>])] }
{ CNTL-F<name>                                             }
{ <name>                                                   }
```

where

MF	is the command name for invoking macros in the command mode.
CNTL-F	is the command name for invoking macros in the screen mode.
<name>	is the single character macro name assigned with the MS command. If the name is a control character, the third format can be used to execute the command.
<parameter1> through <parameter>	are parameters passed to the macro. The parameters are enclosed in parentheses and separated by commas. Null parameters are passed by entering the surrounding commas but no text. To omit the last parameter, end the list with a comma. If one of the parameters contains a comma, literalize that comma by preceding it with the literalizing character (default of backslash).

The Macro Function command executes a macro.

If a macro does not exist, the message

```
DOESN'T EXIST
```

is displayed on the screen.

There are four forms of the command: two for command mode and two for screen mode. In command mode, either use MF, or, if the macro name is a control character, type the control character only. The cursor must be positioned at the beginning of a command line.

In screen mode, either use CNTL-F; or, if the macro name is a control character, type the control character only. The control character cannot be a control character used by the editor. The cursor may be located anywhere.

In either mode, if the macro name is a control character, that character cannot have any special meaning to the editor. For example, the macro name cannot be any of the screen editing commands.

In command mode, to execute a macro named B which requires no parameters, enter:

```
MFB
```

In command mode, to execute a macro named S which requires two parameters, enter:

```
MFS(1978,1979)
```

To execute the same macro, but setting the second parameter to a null value, enter:

```
MFS(1978,)
```

In command mode, to execute the CNTL-Y macro, enter:

```
CNTL-Y
```

In screen mode, to execute the T macro, enter:

```
CNTL-FT
```

In screen mode, to execute the CNTL-K macro, enter:

```
CNTL-K
```

MD

Macro Delete Command

The format of the Macro Delete command is:

$$\left\{ \text{MD} \quad \begin{matrix} \text{<name>} \\ * \end{matrix} \right\}$$

where

MD	is the command name.
<name>	specifies the name of a single macro to be deleted. Once deleted, the name of a macro can be reused. If asterisk is specified, <name> cannot be used.
*	deletes all currently defined macros. If <name> is specified, asterisk cannot be used.

The Macro Delete command deletes a macro specified by name or deletes all current macros if an asterisk is specified.

If neither name nor * is specified, an error message is displayed and no macros are deleted.

To delete a macro named V, enter:

```
MDV
```

To delete all macros, enter:

```
MD*
```

?M

Display Macro Command

The format of the Display Macro command is:

```
?M
```

where

?M	is the command name.

The Display Macro command displays the name and text of all macros.

The macros are displayed with the macro name followed by the text of the macro. The text of the macro is the sequence of editor commands, i.e., the part of the macro enclosed in delimiters in the MS command. If a macro takes up more than one line, succeeding lines are shown below the first line.

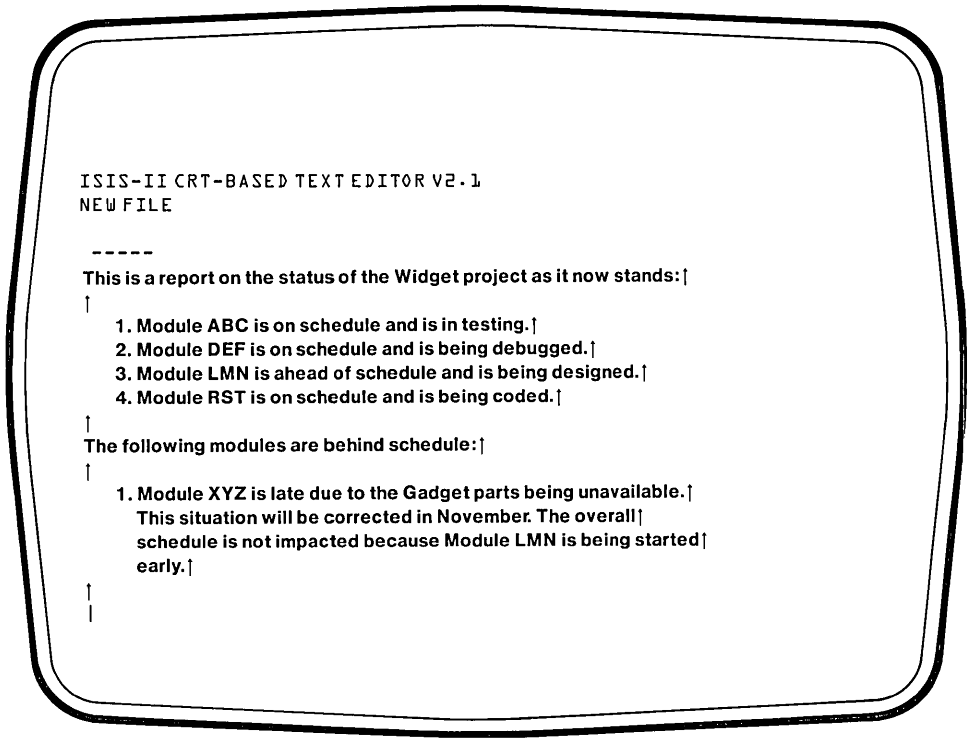
If the macro definition contains control characters, they are displayed as an uparrow followed by the character. CNTL-V would be displayed as ↑V.

If the macros B, S, and M were currently defined, the ?M command would display the following:

```
Bjtt;p10
SS"%"%";P0
Mjtt;
↑V
```

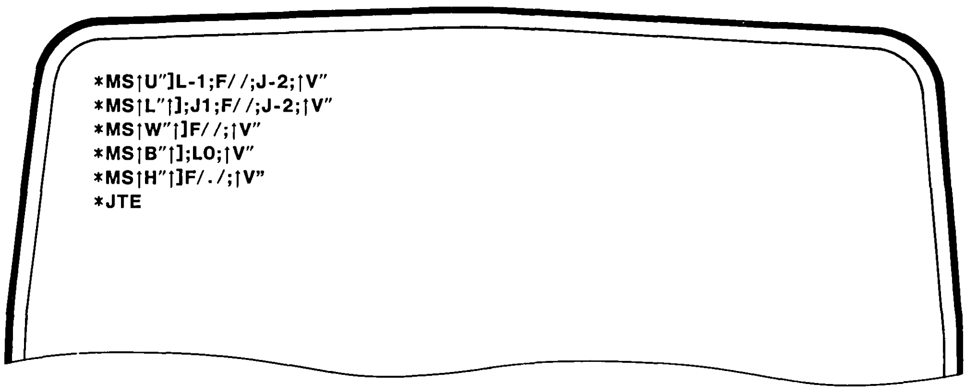
Sample Editing Using Macros

This section provides examples of defining and using several macros for text editing.



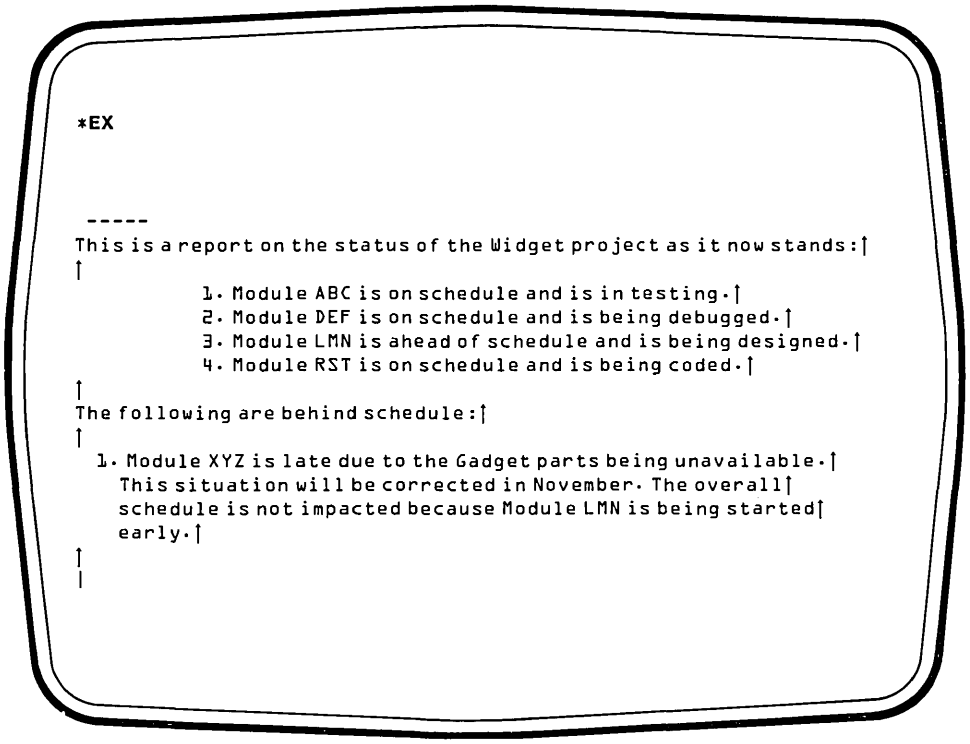
```
ISIS-II CRT-BASED TEXT EDITOR V2.1
NEW FILE

 -----
This is a report on the status of the Widget project as it now stands:↑
↑
	1. Module ABC is on schedule and is in testing.↑
	2. Module DEF is on schedule and is being debugged.↑
	3. Module LMN is ahead of schedule and is being designed.↑
	4. Module RST is on schedule and is being coded.↑
↑
The following modules are behind schedule:↑
↑
	1. Module XYZ is late due to the Gadget parts being unavailable.↑
	   This situation will be corrected in November. The overall↑
	   schedule is not impacted because Module LMN is being started↑
	   early.↑
↑
|
```

Key-in Sequence	Comments
CREDIT REPORT.TXT	This section of examples shows how to define and use CREDIT macros. First use the CREDIT command to create a new file named REPORT.TXT.
TPWR	TPWR switches to typewriter mode so that lower case characters can be entered. Enter the text as shown on the screen above. Use tabs and not spaces to indent the items 1-4 and 1.
TPWR HOME	Press TPWR and the HOME key to switch to the command line mode of editing.



```
*MS↑U"]L-1;F/ /;J-2;↑V"
*MS↑L"↑];J1;F/ /;J-2;↑V"
*MS↑W"↑]F/ /;↑V"
*MS↑B"↑];LO;↑V"
*MS↑H"↑]F/./;↑V"
*JTE
```

Key-in Sequence	Comments
MS CNTL U " HOME L-1; F/*SPACE*/; J-2; CNTL V" RETURN	The MS command defines a macro named CNTL-U that can be executed in screen mode. This macro moves the cursor to the end of the previous line. In entering the MS command, do not type any spaces except where shown. When control characters areentered. Theyare displayed as an up arrow followed by the character. The HOME key is displayed as an up arrow followed by the right bracket.
MS CNTL L" HOME ; J1;F/*SPACE/*; J-2; CNTL V" RETURN	This MS command defines a macro named CNTL-L that can be executed in screen mode. CNTL-L moves the cursor to the end of the current line. If there is no line terminator on the line, the CNTL-L macro prints the message NOT FOUND.
MS CNTL W" HOME F/*SPACE*/; CNTL V" RETURN	The macro named CNTL-W, which can be executed in screen mode, moves the cursor to the next space character in the file. Usually, this will correspond to the next word, so the macro can be used to jump from word to word.
MS CNTL B" HOME ; LO; CNTL V; RETURN	The CNTL-B macro, which can be run in screen mode, moves the cursor to the beginning of the current line.
MS CNTL H" HOME F/.*SPACE*; CNTL V" RETURN	The CNTL-H macro, which can be run from screen mode, moves the cursor to the next period. Usually, this corresponds to the end of a sentence, so the CNTL-H macro can be used to jump from sentence to sentence.
JTE RETURN CNTL V	Type JTE to move the file pointer to the end of the file. Then, type CNTL-V to switch to the screen mode of editing and try some of these macros. The last line of text in the file is displayed when screen mode is entered.



```
*EX


 -----
This is a report on the status of the Widget project as it now stands:↑
↑
          1. Module ABC is on schedule and is in testing.↑
          2. Module DEF is on schedule and is being debugged.↑
          3. Module LMN is ahead of schedule and is being designed.↑
          4. Module RST is on schedule and is being coded.↑
↑
The following are behind schedule:↑
↑
  1. Module XYZ is late due to the Gadget parts being unavailable.↑
    This situation will be corrected in November. The overall↑
    schedule is not impacted because Module LMN is being started↑
    early.↑
↑
|
```

Key-in Sequence	Comments
CNTL U	Type CNTL-U several times. Each time, the cursor is moved to the end of the preceding line. Continue until the cursor is at the end of the first line in the file. The cursor is shown in these examples as an underline.
CNTL B	Type CNTL-B to move the cursor to the beginning of the first line.
CNTL L	Type CNTL-L to move the cursor back to the end of the first line.
CNTL W	Type CNTL-W to move the cursor to the next word.
CNTL H	Type CNTL-H to move the cursor to the character following the next period and space. In this case, the next period follwed by a space follows the digit 2, so CNTL-H moves the cursor to the beginning of the next line.
CNTL H	Type CNTL-H again to move the cursor to the next period. These macros are very useful, but they are lost when the EX command is given to end the CREDIT session. However, there is a way ot save and retrieve them by entering the definitions into a CREDIT command file. See the section on "Command Files" later in this chapter.
EX RETURN	

Command Iteration

The format for the command iteration facility is:

$$\left\{\begin{array}{c} ! \\ \text{<number>} \\ \end{array}\right\} \quad \text{<delimiter1><commands><delimiter2>}$$

where

<number>	specifies the number of times the group of commands is to be repeated. If <number> is specified, an exclamation point (!) cannot be used. If neither <number> nor ! is specified, the group of commands is executed once. If the commands cannot be executed the number of times specified, the appropriate error message occurs.
!	specifies that the group of commands is to be repeated forever or until the end of file is reached. If ! is specified, number cannot be used.
<delimiter1>	is the left angle bracket (<). In this case, the left angle bracket normally used only to describe the command format must be entered on the command line.
<commands>	specifies a sequence of commands to be executed together as a group. The commands are enclosed in angle brackets.
<delimiter2>	is the right angle bracket (>). In this case, the right angle bracket normally used only to describe the command format must be entered on the command line.

A single command or a group of commands may be repeatedly executed with the command iteration facility.

The number or the exclamation point is called the iteration factor.

The command iteration facility can be used within macros. Also, iterated commands may be nested up to seven levels. For example, one of the commands to be executed in a group can itself be a group of commands.

If one of the search commands (Find or Substitute) is used within a loop, the editor exits from the loop if the specified string is not found. This action can be circumvented by placing an extra set of angle brackets around the search command.

To substitute the string "text" for the string "data" 10 times, enter:

```
10<S"data"text">
```

If there are fewer than 10 occurrences of data in the file, the error message

```
NOT FOUND
```

is displayed.

To search for the string "cat" and then print the line where the string is found, enter:

```
!<F"cat";<F"dog">;P>
```

The Find command is nested in a pair of angle brackets so that if the string is not found the editor exits from this inner loop and not the main loop.

Conditional Execution Of Commands

The editor provides several commands for conditional editing in loops and macros. The commands associated with conditional execution are:

- Query User (QU) which sets the Query Flag
- Query True (QT) and Query False (QF) which execute a command based on the condition of the Query Flag set by the QU command
- Yes Flag True (YT) and Yes Flag False (YF) which execute a command based on the condition of the Yes Flag. The Yes Flag is set if a string is found by the Find or Substitute command and is reset if the string is not found.
- Exit Loop (EL) which causes an unconditional exit from an iterative loop
- User Message (U) which displays a prompt for the QU command or supplies the user with information about a macro

QU

Query User Command

The format of the Query User command is:

```
QU
```

where

QU	is the command name.

The Query User command sets or resets the Query Flag that can be tested by the Query True or Query False commands.

The QU command stops the display and prompts for input with a question mark (?). A "Y" or "y" response sets the flag to the true condition and continues processing. Any other response resets it to the false condition and continues. The RETURN key is not required after typing the response.

The QU command can be used alone or in combination with the Query True, the Query False, and the User Message commands. Used alone, the QU command stops the display from scrolling, so the screen can be read until the user responds to the prompt.

When used with the Query True and Query False commands, the QU command allows the user to conditionally execute a block of edit commands based on the condition of the Query Flag. The User Message command creates a message in addition to the question mark prompt to add meaning to the prompt for the user.

To set the Query Flag:

```
QU
? Y
```

To reset the Query Flag:

```
QU
? N
```

QT and QF

Query True and Query False Commands

The format of the Query True command is:

```
QT;<command>
```

where

QT	is the command name.
<command>	is the command or commands controlled by the Query Flag. It can be any editor command, including macros or a block of commands to be repeatedly executed. If an iterated block of commands is specified, it must be enclosed in angle brackets. It may be preceded by a number or by an exclamation point (!) to specify the iteration factor, the number of times the block of commands is to be executed.

The format of the QF command is:

```
QF;<command>
```

where

QF	is the command name.
<command>	is the same as above.

The Query True and Query False commands execute a command based on the condition of the Query Flag that was set by the QU command.

If the Query Flag is set to true, the Query True command causes the next command to be executed, and Query False causes the next command to be skipped. If the flag is set to false, the Query True command causes the next command to be skipped, and the Query False command causes the next command to be executed. The table 4-1 summarizes the possible conditions:

Table 4-1 Query Flag Conditions

Query Flag	Next Command Executed?	
Set by QU	QT	QF
Set True	YES	NO
Set False	NO	YES

To execute the Delete Line command only if the Query Flag is set true, enter:

```
QT;DL
```

To skip the Delete Line command when the Query Flag is set to true, enter:

```
QF;DL
```

To execute the Delete Line command when the Query Flag is set false, enter:

```
QF;DL
```

To skip the Delete Line command when the Query Flag is set false, enter:

```
QT;DL
```

The following command

```
QT;10<S"cat"dog";P>
```

executes the Substitute and Print command 10 times if the Query Flag is set true; if the Query Flag is set false the Substitute and Print commands are skipped.

YT and YF

Yes Flag True and Yes Flag False Commands

The format of the Yes Flag True command is:

```
YT;<command>
```

where

YT	is the command name.
<command>	is the command or commands controlled by the Yes Flag. It can be any editor command, including macros or a block of commands to be repeatedly executed. If an iterated block of commands is specified, it must be enclosed in angle brackets. It may be preceded by a number or by an exclamation point (!) to specify the iteration factor, the number of times the block of commands is to be executed.

The format of the Yes Flag False command is:

```
YF;command
```

where

YF	is the command name.
<command>	is the same as above.

The Yes Flag True and Yes Flag False commands execute a command based on the condition of the Yes Flag that was set by the Find and Substitute commands.

The search commands (Find and Substitute) set the Yes Flag. If the search is successful, the string is found, the Yes Flag is set true. If the search fails, the string is not found, the Yes Flag is set false.

If the Yes Flag is set to true, the Yes Flag True command causes the next command to be executed, and Yes Flag False causes the next command to be skipped. If the flag is set to false, the Yes Flag True command causes the next command to be skipped, and the Yes Flag False command causes the next command to be executed. The table 4-2 summarizes the possible conditions:

Table 4-2 Yes Flag Conditions

Yes Flag Set by S and F	Next Command Executed? YT	YF
Set True	YES	NO
Set False	NO	YES

To exit a file if the string "cat" is not found, enter:

```
F"cat";YF;EX
```

To jump to the beginning of the file and print 10 lines if the string "cat" is found, enter:

```
F"cat";YT;<JTT;P10>
```

EL

Exit Loop Command

The format of the Exit Loop command is:

```
EL
```

where

EL	is the command name.

The Exit Loop (EL) command performs an unconditional exit from a command loop.

Used with the Query commands and the Yes Flag commands, EL provides a means to conditionally exit from a loop. EL is ignored if it occurs outside of a command loop.

To set up a loop to search an entire file for the string "cat" and exit the loop if the line containing "cat" also contains "dog", enter:

```
!<F"cat";<F"dog"1>;YT;EL>
```

The inner Find command must be enclosed in angle brackets because, if a string is not found inside of a loop, the editor exits the loop. If the inner angle brackets were omitted, the editor would exit the outer loop if it did not find "dog". The inner brackets give the editor a dummy loop to exit when "dog" is not found, so it will continue to process the outer loop.

U

User Message Command

The format of the User Message command is:

```
U<delimiter><text><delimiter>
```

where

U	is the command name.
<delimiter>	is any character except space, carriage return, linefeed, the literalizing character (default of backslash), escape, or a character appearing in <text>. Both occurrences of the <delimiter> must be the same.
<text>	is a string of one or more characters to be displayed. The string must be enclosed by delimiters.

The User Message command (U) writes a message to the screen.

This command can precede a QU command to provide directions or other information to the user.

To search the file for a line containing the string "cat", print that line and delete it if the user desires, enter:

```
!<F"cat";YT;<P;U"DELETE";QU;QT;DL>>
```

In this example, if the string "cat" is found, the Yes Flag will be set true. If the Yes Flag is true, the inner block of commands is executed: the line is displayed; the prompt "DELETE" is displayed followed by the question mark prompt of the QU command; if the user sets the Query Flag by typing Y, the line is deleted by the DL command. The entire outer block is repeated throughout the file.

Accessing Disk Files From The Editor

Two types of disk files can be accessed through the editor: command files containing editor commands and data files which can be read or written with editor commands.

Command Files

A command file can contain any editor commands, including macros and command blocks. When a command file is accessed from the editor, the commands are loaded and executed. One application for command files is to define macros that will be needed throughout an editing session.

The command file can be created with the editor by entering a series of CREDIT commands as data. For example, the following lines in the file CREDIT.MAC define the macros F, B, and G:

```
MSF/<F"%">;YT;<U"FOUND:";P>/
MSB"JTT;P20"
MSG/!<<S"%"%">;YT;P>/
```

Each command in the file is terminated with either a carriage return or a semicolon. After the command file is loaded, the three macros F, B, and G are defined and can be used throughout the editing session.

The literalizing character (\ ) must be used to enter control characters into the command file.

If an invalid command is found in a command file, an error message is displayed and no further commands are executed. Also, if a command file attempts to define a macro that already exists, an error results and execution of the file stops.

There are two ways to load and execute a command file:

- Naming the file with the MACRO option in the CREDIT command line when the editor is first invoked
- Issuing a Get command specifying the name of the command file

Data Files

A data file is any ISIS file which the editor can open for reading or writing. To access the data in the file, follow these steps:

1) Open the file either for reading or writing.

2) Read from or write to the file.

3) Close the file.

This process is similar to getting information into or out of a file drawer. When the file drawer is open, files (information) can be taken out or put in. When the drawer is closed, the information cannot be accessed. Opening a file establishes a logical connection to the file; closing it breaks that connection.

With the editor, the file can be opened for reading or writing but not both at the same time. When a file is opened for reading, it cannot be written. When a file is opened for writing, no data can be read from that file. All data written to a file is inserted at the end of the file.

Only one file can be open for reading and one for writing at a time.

Data cannot be added to an existing file. If an existing file is opened for writing, it is deleted first and then recreated, destroying data in it. Existing files can be opened for reading without destroying the data in it.

When a file is opened for reading, it is read sequentially from the beginning character to the ending character. When a record is read, the pointer moves to the next record. There is a command to move the pointer to the beginning of the file.

Seven commands support the use of data files:

- Open Read opens a file for reading
- Open Write opens a file for writing
- Begin File moves the data pointer to the beginning of a file open for reading
- Read reads from a file already opened for reading
- Write writes to a file already opened for writing
- Close Read closes a file already open for reading
- Close Write closes a file already open for writing

The MACRO Option

The command line format is discussed in detail in the beginning of this chapter. The options MACRO and (command file) are used to access a file containing editor commands.

If MACRO is specified without a command filename, the command file CREDIT.MAC is loaded and executed if it can be found. An error results if it cannot be found.

If both MACRO and a command filename are specified, the command file named is loaded and executed if it can be found. An error results if it cannot be found.

If the MACRO parameter is omitted entirely, the command file CREDIT.MAC is loaded and executed if it can be found, but no error results if it cannot be found.

In all cases, the contents of the command file are loaded and executed before CREDIT allows the user to enter any commands at the keyboard. This facility allows the user to automatically configure CREDIT before interactive editing begins.

G

The Get Command

The format of the Get command is:

```
G <pathname>
```

where

G	is the command name.
<pathname>	specifies any ISIS file or device that contains editor commands.

The Get command loads a command file and executes it.

The Get command can be run anytime from the command mode of editing. A command file can contain another Get command to load another file. There is no limit to the depth of nesting Get commands within command files.

If the command file contains a macro definition with a name that is already defined, an error message is displayed, execution of the command file stops, and the old macro already defined is kept.

To load a command file named OPSYS.MAC on :F0:, enter:

```
G :F0:OPSYS.MAC
```

OR

Open Read Command

The format of the Open Read command is:

```
OR <pathname>
```

where

OR	is the command name.
<pathname>	specifies any valid ISIS file or device that can be opened for reading. If the file is already open (except the :CI: file) or cannot be open for reading, an error message is displayed.

The Open Read command opens a specified file so it can be read from the editor.

A file opened for reading cannot be written to. It also cannot be opened for any other purpose unless it is closed first. The only exception to this is the console input file, :CI:, which is always open. When a file is opened, the pointer is at the beginning of the file.

To open the file OLDFIL.TXT on :F0: for reading, enter:

```
OR :F0:OLDFIL.TXT
```

OW

Open Write Command

The format of the Open Write command is:

```
OW <pathname>
```

where

OW	is the command name.
<pathname>	specifies any valid ISIS file or device that can be used for output. If the file is already open (except for the file :CO:) or it cannot be used for output, an error message is displayed.

The Open Write command opens a specified file so it can be written to by the editor.

A file opened for writing cannot be read. The file specified in the command cannot be opened for any other purpose or an error occurs. The only exception to this is the console output file, :CO:, which is always open. If the specified file already exists, it is deleted.

To open the file NEWFIL.TXT on :F0: for writing, enter:

```
OW :F0:NEWFIL.TXT
```

B

Begin File Command

The format of the Begin File command is:

```
B
```

where

B	is the command name.

The Begin File command moves the read data file pointer to the beginning of the data file.

The Begin File command only works on files opened for reading with the OR command. If no read file is open, the command has no effect.

To move the data file pointer to the beginning of the data file, enter:

```
B
```

R

Read File Command

The format of the Read File command is:

```
R[<number>]
```

where

R	is the command name.
<number>	specifies how many lines to read. If no number is specified, one line is read. The number cannot be negative. If number is zero, nothing is read.

The Read File command reads a specified number of lines from the currently open data file and inserts the lines read into the file being edited.

The data file pointer is moved to the first character in the data file that was read. The text in the data file is not destroyed.

To read 10 lines of text from the file currently opened for reading, enter:

```
R10
```

W

Write File Command

The format of the Write command is:

$$\mathrm{W}\left[\begin{Bmatrix} \text{<number>} \\ \text{<tag>} \end{Bmatrix}\right]$$

where

W	is the command name.
<number>	is the number of lines to be written to the data file.
<tag>	specifies the end of a block of characters to be written. Any permanent or user defined tag can be used.

If no number or tag is specified, the entire current line in the file being edited is written.

If number is positive, the specified number of lines starting at the current pointer are written from the file being edited to the data file.

If number is negative (is preceded by a minus character) the specified number of lines preceding the pointer and ending at the pointer are written from the file being edited to the data file.

If number is zero, the characters from the beginning of the current line are written from the file being edited to the data file.

Both <number> and <tag> cannot be specified.

The Write command writes a specified number of lines from the file being edited to the file currently opened for writing.

There is no effect on the text or the pointer of the file being edited. If the file being written contains data, that data is written over by the Write command.

To write 10 lines starting at the cursor in the file currently being edited to the current data file, enter:

```
W10
```

To write the 10 lines preceding the cursor, enter:

```
W-10
```

CR and CW

Close File Commands

The formats of the Close File commands are:

$$\left\{\begin{matrix}\text{CR}\\\text{CW}\end{matrix}\right\}$$

where

CR	is the command name used to close the file opened for reading.
CW	is the command name of the file currently opened for writing.

The Close File commands close the current data files.

Both commands are ignored if no file is currently open.

The EX and EQ commands issued at the end of an editing session close any files that are currently open.

To close the file opened with the OR command, enter:

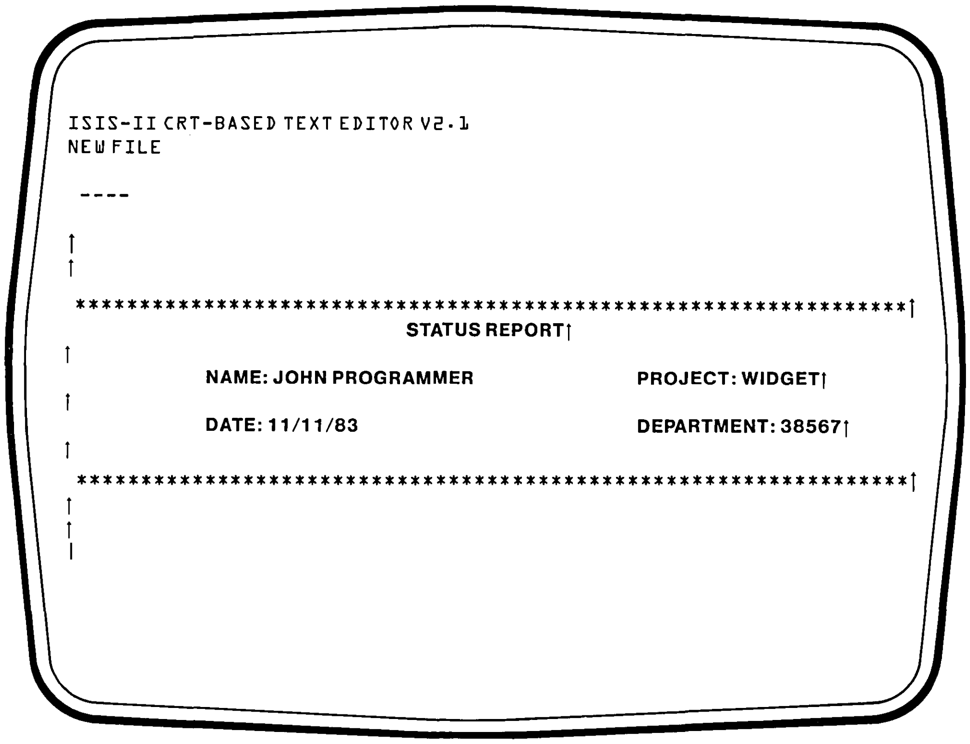
```
CR
```

To close the file opened with the OW command, enter:

```
CW
```

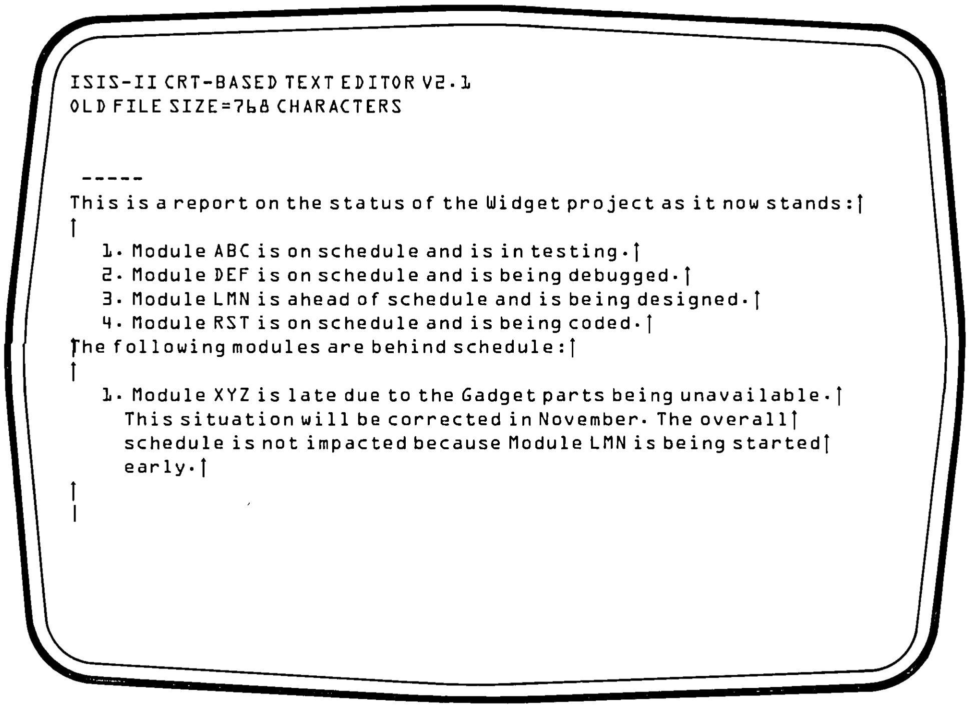
Sample Editing Using Data Files

The following example illustrates an application where data files are used in editing.



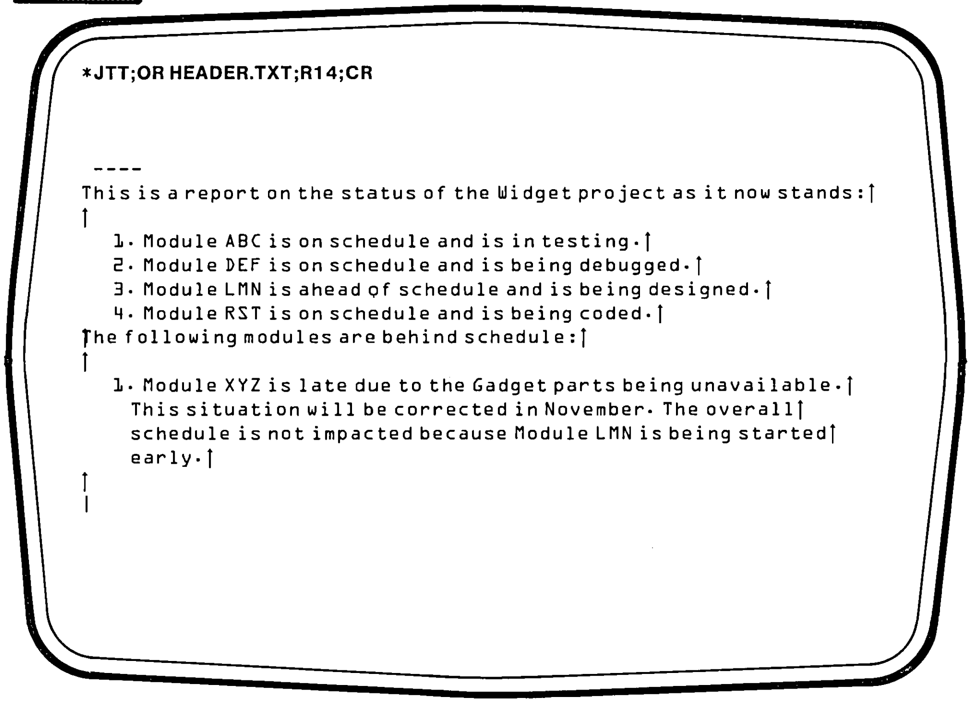
```
ISIS-II CRT-BASED TEXT EDITOR V2.1
NEW FILE

 ----

↑
↑
********************************************************************↑
                        STATUS REPORT↑
↑
          NAME: JOHN PROGRAMMER               PROJECT: WIDGET↑
↑
          DATE: 11/11/83                      DEPARTMENT: 38567↑
↑
********************************************************************↑
↑
↑
|
```

Key-in Sequence	Comments
CREDIT HEADER.TXT RETURN	This command line creates a new file named HEADER.TXT to be edited in the next series of examples. These examples show techniques for opening and reading files. The file, REPORT.TXT, created in a previous example is needed as well. Enter the data in the file as shown on the screen above using the screen editing functions.
HOME **EX** RETURN	There are several ways this header can be added to the text file REPORT.TXT. The method shown first illustrates how to open and read files. First, exit from HEADER.TXT.



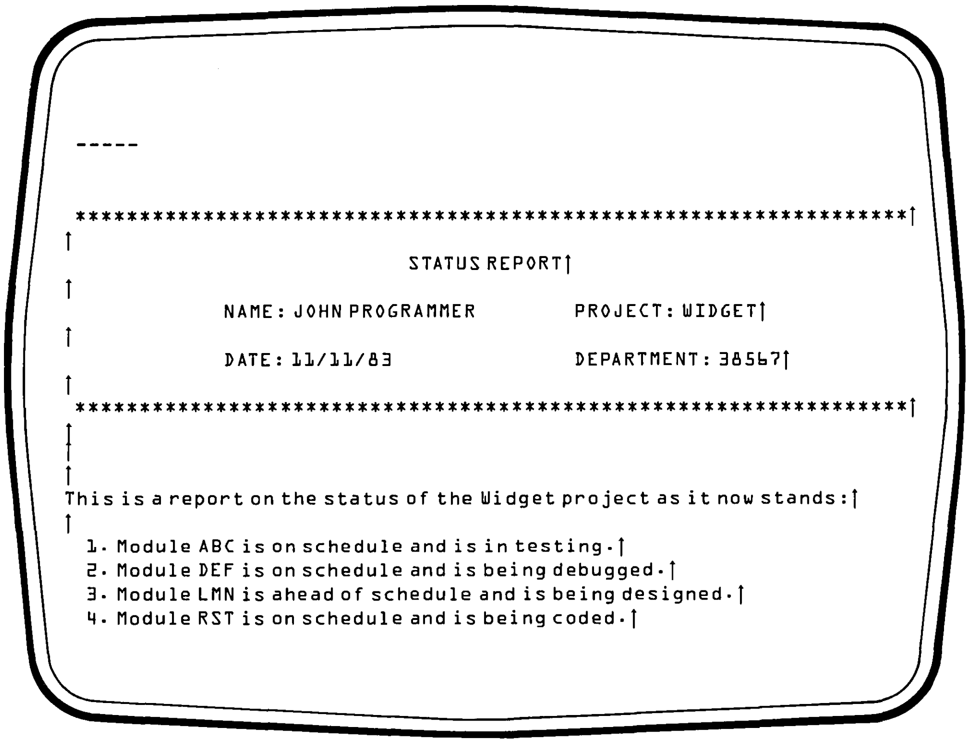
```
ISIS-II CRT-BASED TEXT EDITOR V2.1
OLD FILE SIZE=768 CHARACTERS


 -----
This is a report on the status of the Widget project as it now stands:↑
↑
   1. Module ABC is on schedule and is in testing.↑
   2. Module DEF is on schedule and is being debugged.↑
   3. Module LMN is ahead of schedule and is being designed.↑
   4. Module RST is on schedule and is being coded.↑
↑he following modules are behind schedule:↑
↑
   1. Module XYZ is late due to the Gadget parts being unavailable.↑
      This situation will be corrected in November. The overall↑
      schedule is not impacted because Module LMN is being started↑
      early.↑
↑
|
```

Key-in Sequence	Comments
CREDIT REPORT.TXT RETURN	Edit the file created in a previous example, REPORT.TXT.



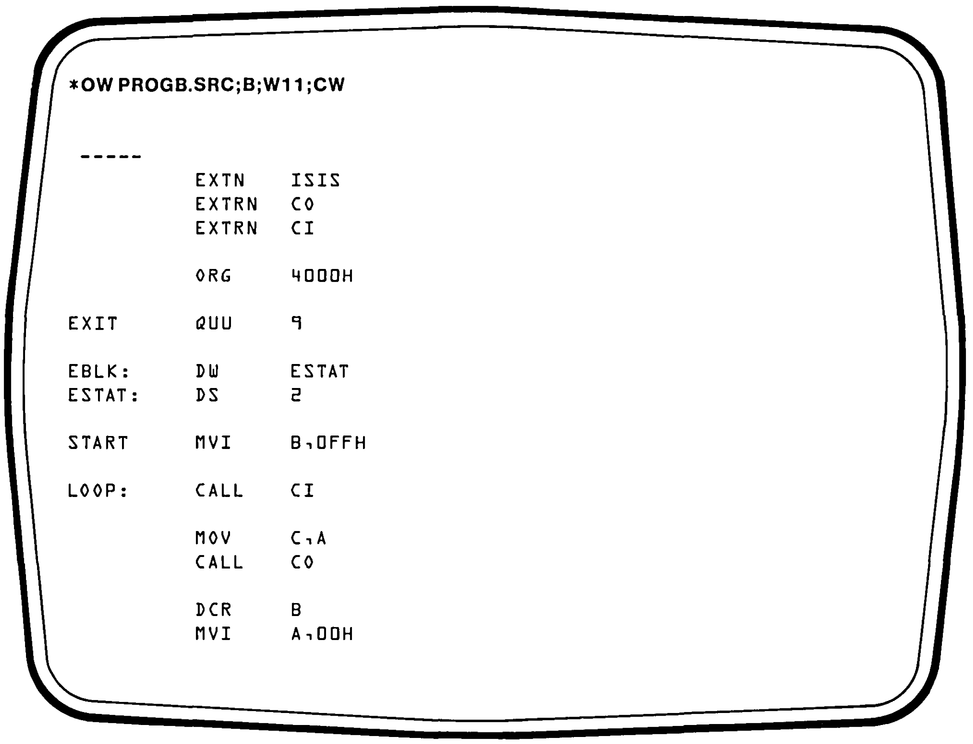
```
*JTT;OR HEADER.TXT;R14;CR


 ----
This is a report on the status of the Widget project as it now stands:↑
↑
   1. Module ABC is on schedule and is in testing.↑
   2. Module DEF is on schedule and is being debugged.↑
   3. Module LMN is ahead of schedule and is being designed.↑
   4. Module RST is on schedule and is being coded.↑
↑he following modules are behind schedule:↑
↑
   1. Module XYZ is late due to the Gadget parts being unavailable.↑
      This situation will be corrected in November. The overall↑
      schedule is not impacted because Module LMN is being started↑
      early.↑
↑
|
```

Key-in Sequence	Comments
HOME **JTT;OR HEADER.TXT; R14;CR** RETURN	Switch to command line mode and enter the commands as shown. The OR command opens the file containing the header. The command, R14, reads 14 lines from this file. Since the header is 14 lines, this command reads the entire header. Then the CR command closes the file HEADER.TXT.



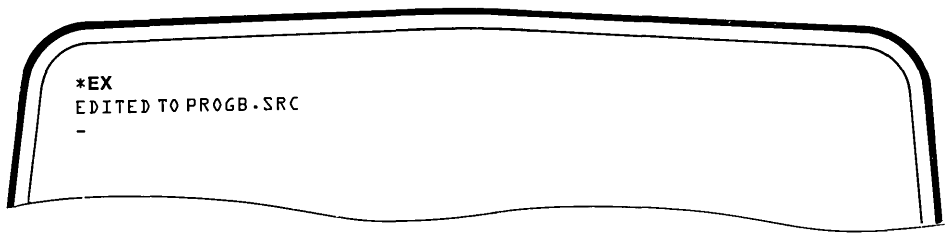
```
-----

 *******************************************************************↑
↑
                            STATUS REPORT↑
↑
           NAME: JOHN PROGRAMMER          PROJECT: WIDGET↑
↑
           DATE: 11/11/83                 DEPARTMENT: 38567↑
↑
 *******************************************************************↑
↑
↑
↑
This is a report on the status of the Widget project as it now stands:↑
↑
  1. Module ABC is on schedule and is in testing.↑
  2. Module DEF is on schedule and is being debugged.↑
  3. Module LMN is ahead of schedule and is being designed.↑
  4. Module RST is on schedule and is being coded.↑
```

Key-in Sequence	Comments
CNTL P CNTL Y	Enter the CNTL-V to switch back to screen mode and CNTL-P to display the first page fo the file. Notice that the header just read from the file HEADER.TXT is now at the beginning of REPORT.TXT. The file HEADER.TXT is not changed in any way by a read.
HOME EX RETURN	Exit from the file. Another way to add this header is to use a submit file as shown in a later section.



```
*OW PROGB.SRC;B;W11;CW

 -----
           EXTN    ISIS
           EXTRN   CO
           EXTRN   CI

           ORG     4000H

EXIT       QUU     9

EBLK:      DW      ESTAT
ESTAT:     DS      2

START      MVI     B,0FFH

LOOP:      CALL    CI

           MOV     C,A
           CALL    CO

           DCR     B
           MVI     A,00H
```

Key-in Sequence	Comments
CREDIT PROGRA.SRC RETURN	The next series of examples show how to write data to a file from the editor. This example shows how to copy a portion of one program that is to be used in another program. Use the file, PROGRA.SRC, created in a previous example. The first 11 lines, down to the beginning of the program body, will be copied to another file.
HOME **OW PROGB.SRC; B;W11;CW** RETURN	First, press home to enter the command line mode. The OW command opens the file specified, PROGRB.SRC, for writing. Type B to start at the beginning of the file. The specified file is opened as a new file, so any data already in the file will be lost. The command W11, writes the first 11 lines of PROGRA.SRC (the file being edited) to PROGRB.SRC. Type CW to close the file being written, PROGRB.SRC.

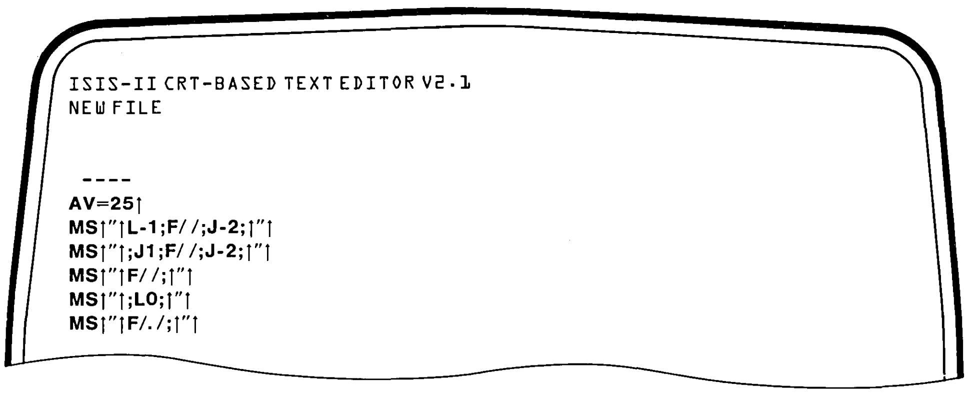


```
*EX
EDITED TO PROGB.SRC
-
```

Key-in Sequence	Comments
EX RETURN	Use the EX command to return to the operating system. Now, the file PROGB.SRC can be editied to add the remaining code.

Sample Editing Using Command Files

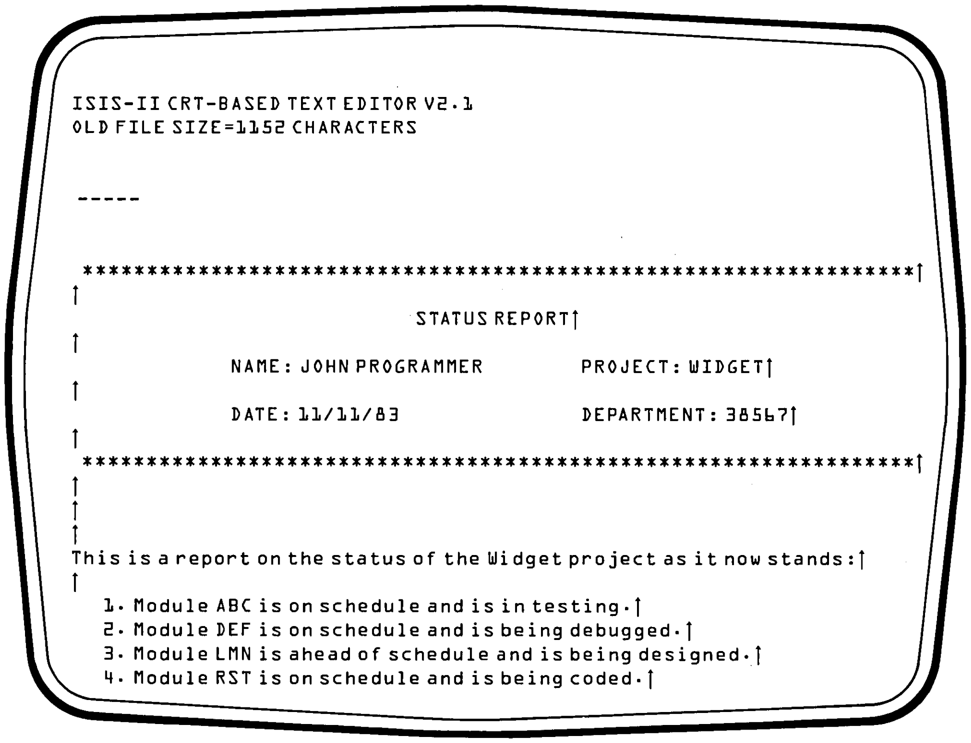
The following example illustrates an application where command files are used in editing.



```
ISIS-II CRT-BASED TEXT EDITOR V2.1
NEW FILE


 ----
AV=25↑
MS↑"↑L-1;F/ /;J-2;↑"↑
MS↑"↑;J1;F/ /;J-2;↑"↑
MS↑"↑F/ /;↑"↑
MS↑"↑;LO;↑"↑
MS↑"↑F/./;↑"↑
```

Key-in Sequence	Comments
CREDIT TMACRO RETURN	In addition to data files, CREDIT also accesses command files which contain CREDIT commands. A common use for command files is to contain a group of MS commands that define macros. Another use is to contain CREDIT environment commands which are described in a later section. The next series of examples illustrate the command file feature of CREDIT. First, the file TMACRO is created with CREDIT.
AV=25 RETURN	The TMACRO file will contain a group of MS commands that define text macros as well as one CREDIT environment command, AV=25. The AV=25 command changes the CREDIT screen size to twenty-five lines which is the size of the Series II screen.

Key-in Sequence	**Comments**
MS\ CNTL U"\ HOME L-1;F/*SPACE/*;J-2;\ CNTL V" RETURN	Enter the MS command as shown. Notice that to enter a control character or the HOME key, the backslash must be typed. The backslash does not appear on the screen. It acts as a literalizing character to allow the control characters to be entered. The backslash is not used when entering the macro definition from the command line. Each control character is displayed in the macro section of the manual.
MS\ CNTL L"\ HOME J1;F/*SPACE/;*J-2; \ CNTL V" RETURN	Continue entering MS commands as shown. CNTL-L is the name of a macro that moves the cursor to the end of the current line.
MS\ CNTL W"\ HOME F/*SPACE/*;\ CNTL V" RETURN	The macro named CNTL-W, defined in this example, moves the cursor to the next space. Usually, this will correspond to moving the cursor to the next word.
MS\ CNTL B"\ HOME ; LO;\ CNTL V" RETURN	The macro CNTL-B is defined here. It moves the cursor to the beginning of the current line.
MS\ CNTL H"\ HOME ; F/.*SPACE/*;\ CNTL V" RETURN	The macro named CNTL-H moves the cursor to the next period which usually corresponds to moving the cursor to the start of the next sentence.
HOME EX RETURN	Now press home and exit from the file TMACRO. This file can now be loaded into CREDIT in two ways; with the MACRO option on the command line and with the G command in the command line mode of CREDIT. In either case, all the CREDIT commands in the file are executed when loaded. Thus, the MS commands result in defining the macros involved. Once defined, these macros are available to the user while editing the file.



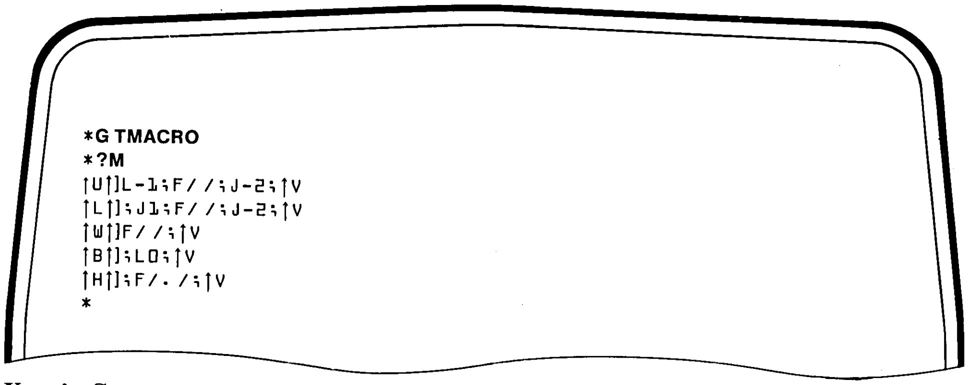
```
ISIS-II CRT-BASED TEXT EDITOR V2.1
OLD FILE SIZE=1152 CHARACTERS


-----


 ******************************************************************↑
↑
                              STATUS REPORT↑
↑
             NAME: JOHN PROGRAMMER          PROJECT: WIDGET↑
↑
             DATE: 11/11/83                 DEPARTMENT: 38567↑
↑
 ******************************************************************↑
↑
↑
↑
This is a report on the status of the Widget project as it now stands:↑
↑
   1. Module ABC is on schedule and is in testing.↑
   2. Module DEF is on schedule and is being debugged.↑
   3. Module LMN is ahead of schedule and is being designed.↑
   4. Module RST is on schedule and is being coded.↑
```

Key-in Sequence **Comments**

CREDIT REPORT.TXT

RETURN

This example shows how to load TMACRO using the G command. First, open a file for editing. Use the file REPORT.TXT created in a previous example.



```
*G TMACRO
*?M
↑U↑]L-1;F/ /;J-2;↑V
↑L↑];J1;F/ /;J-2;↑V
↑W↑]F/ /;↑V
↑B↑];L0;↑V
↑H↑];F/. /;↑V
*
```

Key-in Sequence **Comments**

HOME **G TMACRO** When the G command is entered, the TMACRO file is opened and all the commands are read in and executed.

RETURN

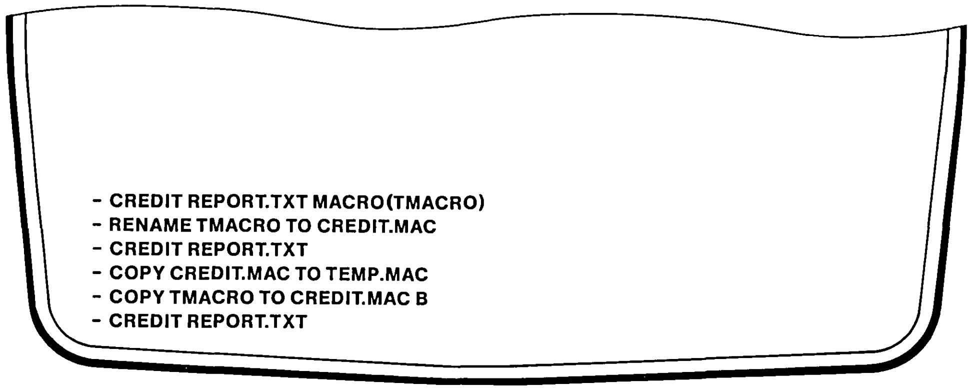
?M

The ?M command lists the macros currently defined. All the macros entered are run from screen mode, so use CNTL-V to switch to screen mode and try them out. They are described in the section on macros.

HOME

EX RETURN

Exit from the file and return to the operating system. The next example shows another way to use the file TMACRO by calling it in from the CREDIT command line.



```
- CREDIT REPORT.TXT MACRO(TMACRO)
- RENAME TMACRO TO CREDIT.MAC
- CREDIT REPORT.TXT
- COPY CREDIT.MAC TO TEMP.MAC
- COPY TMACRO TO CREDIT.MAC B
- CREDIT REPORT.TXT
```

Key-in Sequence	Comments
CREDIT REPORT.TXT MACRO (TMACRO) RETURN	This example shows a CREDIT command line to load the file TMACRO. In screen mode, enter the ?M command to verify that the MS commands were loaded.
RENAME TMACRO TO CREDIT.MAC RETURN **CREDIT REPORT.TEXT** RETURN	The macro file can also be loaded under the name CREDIT.MAC. Any file named CREDIT.MAC on the same disk as the CREDIT command is automatically loaded and executed unless the NOMACRO option is specified or the MACRO option is specified with another filename. If CREDIT.MAC does not already exist, the commands shown here will load the TMACRO file automatically under the default filename CREDIT.MAC.
COPY CREDIT MAC TO TEMP.MAC RETURN **COPY TMACRO TO CREDIT.MAC B** RETURN **CREDIT REPORT.TXT** RETURN	If the file CREDIT.MAC already exists, it should be saved first as shown in this example. Then, the commands in TMACRO can be loaded and executed automatically under the filename CREDIT.MAC. The file on the system disk named CMACRO contains all the macros defined in TMACRO as well as several others defined in the previous section. It can be added to CREDIT.MAC to auto load or can be loaded only when desired with G command. TMACRO can be deleted with the DELETE command since it is duplicated in CMACRO.

Using The Editor With Submit

When the CREDIT command is invoked from a SUBMIT file, the editor operates only in command mode. Command mode operations are the same as described in this chapter with the following exceptions:

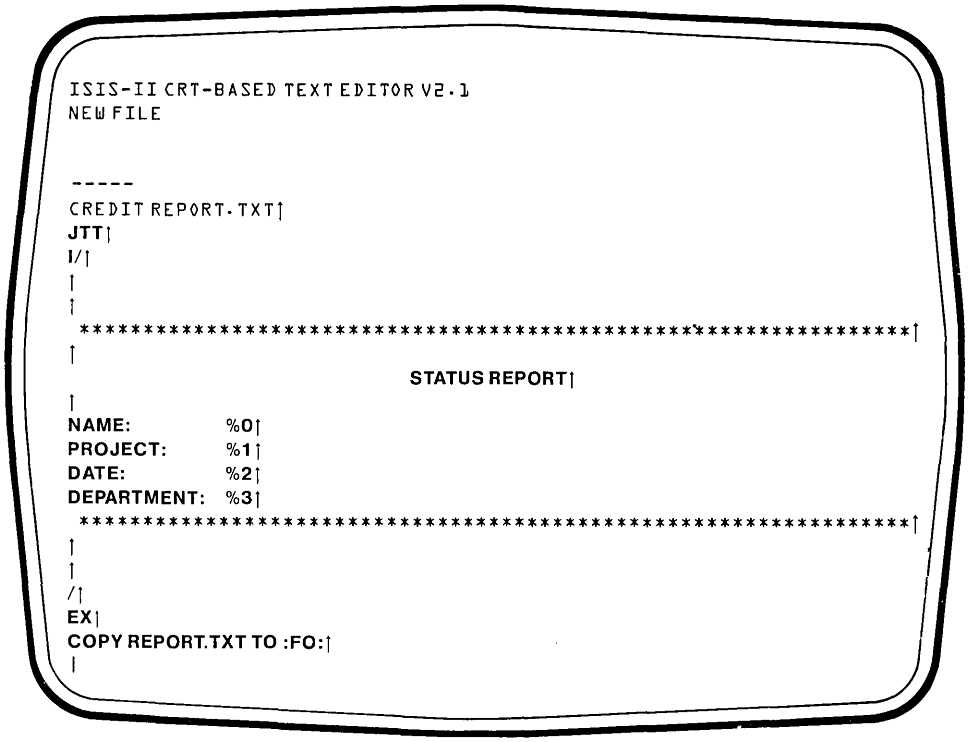
- The Quit command (EQ) does not require a Y or y confirmation.
- The editor does not recognize the CNTL-V command.
- Non-printing characters such as control characters are not echoed to the screen as Uparrow character as they normally are.
- Only two files may be opened from the editor at a time or the six file limit of ISIS will be exceeded.

When the editor runs under SUBMIT, there are four files open: the old edit file, two .TMP files, and the SUBMIT command file. This leaves only two additional files available. When using the Read, Write, and Get commands, the six file limit may be exceeded causing a fatal error and automatic reinitialization of the system.

When the editor is running under SUBMIT, all interactive processes such as display and editing are controlled by SUBMIT. To interactively enter commands, the SUBMIT file must contain a CNTL-E which turns control over to the terminal until the next CNTL-E is encountered. For further information on the SUBMIT command, see Chapters 4 and 5 of this manual.

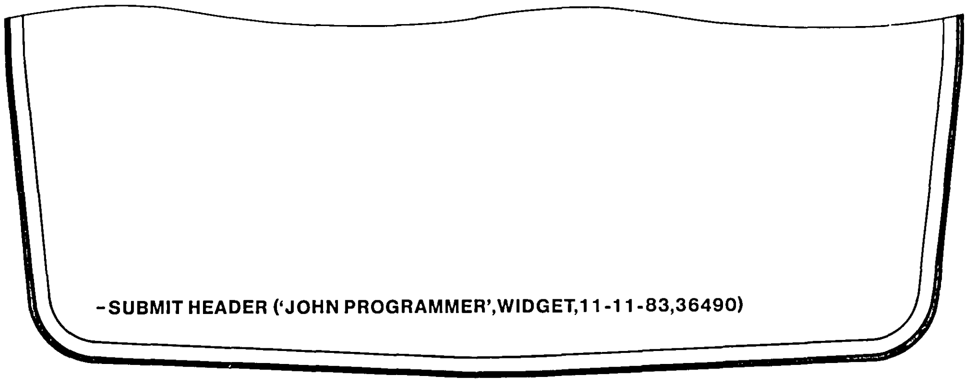
Sample Editing With Submit

The following example illustrates an application where editing was performed under the control of a SUBMIT file.



```
ISIS-II CRT-BASED TEXT EDITOR V2.1
NEW FILE


-----
CREDIT REPORT.TXT↑
JTT↑
I/↑
↑
↑
 *******************************************************************↑
↑
                                   STATUS REPORT↑
↑
NAME:          %0↑
PROJECT:       %1↑
DATE:          %2↑
DEPARTMENT:    %3↑
 *******************************************************************↑
↑
↑
/↑
EX↑
COPY REPORT.TXT TO :FO:↑
|
```

Key-in Sequence	Comments
CREDIT HEADER.CSD RETURN	The next series of examples show how to run CREDIT within a SUBMIT file. First, create a command file, HEADER.CSD, to be run under SUBMIT. Commands will be entered into HEADER.CSD to add a header to the file REPORT.TXT, created previously. Enter text into the file as shown on the screen above. Only commands from command line mode may be used. The %0, %1, %2, and %3 characters are SUBMIT parameters and may be substituted with different values each time the command line is run.
HOME **EX** RETURN	Switch to command line mode and exit to the operating system.



```
-SUBMIT HEADER ('JOHN PROGRAMMER',WIDGET,11-11-83,36490)
```

Key-in Sequence	Comments
SUBMIT HEADER ('JOHN PROGRAMMER', WIDGET, 11-11-83,36490) RETURN	Run the SUBMIT command as shown, specifying the parameters in order within the parentheses. The screen clears and CREDIT signs on. The JTT and Insert command are automatically run. Then, CREDIT is exited and the file is copied to the display screen. If HEADER.TXT is edited again, the new header will appear at the beginning of the file.

Alter Commands

The Alter commands modify the environment and the communication link between the editor and keyboard and CRT attached to the system.

There are three Alter commands: Alter Environment (see table 4-3), Alter Function (see tables 4-4 through 4-8), and Display Alter. The Alter Environment command modifies the way data is presented on the screen. The Alter Function command modifies the codes input or output through the editor and is used to configure the editor to run with non-Intel terminals (see Appendix A for details). The Display Alter command displays the current setting of alterable features.

The items that can be modified with the Alter commands are:

- Line terminator display character
- Number of lines in display screen area
- Tab settings
- Suppression of error message for the Find and Substitute commands
- Wildcard characters
- Cursor movement codes
- Erase line and screen codes
- Screen mode command codes

The Alter commands are often used in a command file that is called with the MACRO option when the editor is first invoked. In this way, the editor executes the commands setting up the proper environment before control is transferred to the keyboard.

This method of altering the environment and communications link is preferred because some non-Intel terminals do not generate the codes on their keyboard that the editor is expecting. If control is transferred to this type of keyboard, the editor may not function properly because it is expecting codes that the keyboard cannot generate.

A

Alter Environment Command

The format of the Alter Environment command is:

A<code>=<value>

where

A	is the command name.
<code>	is a single character specifying which parameter is to be changed. The valid codes and their values are listed in the table below.
<value>	depends on the code specified. The possible values for each code are given in the following charts. Every character after the equal sign is interpreted as part of the value, including spaces.

The Alter Environment command changes the way data is presented on the display screen.

Table 4-3 Alter Environment Values

CODE	DEFAULT	VALUE
B	1BH	Specifies which keyboard character is to be used to cause a BREAK. The BREAK character causes the command being executed to abort and the message <BREAK>to be displayed. The value following the equal sign must be entered as a single hexadecimal byte. Use the ASCII code table in Appendix E to determine the hexadecimal value for the key. The default is a hexadecimal 1B, the ESCAPE key. A new value should be assigned for terminals that require the ESCAPE key for control sequences.
C	↑	Specifies which character is to be displayed for all non-printing characters in screen mode. The value entered may be any printing character, including a blank.
L	↑	Specifies that the line terminator, carriage return and linefeed, is to be displayed as the character following the equal sign instead of the uparrow. The value can be any displayable ASCII character except ESC or &. Once the line terminator display character has been changed from uparrow, it can be redefined to uparrow by entering a command in the form A=<backslash><caret>. The <caret> character is the shifted underline.
O	20H	Specifies the offset value to be added to the row and column numbers following the cursor control sequence of the AFAC command. The value must be entered as a single hexadecimal byte. The setting has no effect until cursor addressing mode is set with the AFAC command.

Table 4-3 Alter Environment Values (continued)

CODE	DEFAULT	VALUE
Q	5CH (\)	Specifies which character to use as the literalizing character. The value must be entered as a single hexadecimal byte.
S	F	Specifies whether or not to suppress the NOT FOUND error message with the Find and Substitute commands. The possible values are T to suppress the message or F to display the message. Enter the characters T or F.
T	8	Specifies the TAB setting. The value can be any integer from 0 to 79. The default setting is 8, so TABS occur every 8 spaces. It is entered as a decimal value.
V	24	Specifies that the number of lines displayed on each screen be changed from 24 to the number following the equal sign. The possible values are 22, 23, 24, or 25. Screens smaller than 25 lines have a smaller command area; the text area remains the same at 20 lines. It is entered as a decimal value.
W	T	Specifies the status of the line wraparound feature of the editor. The two possible values are T to set the wraparound feature on and F to turn the wraparound feature off. With the line wrap feature, the editor automatically wraps around to the beginning of the next line when the last column of the current line is reached.
X	F	Specifies the addressing sequence used in the cursor addressing control mode. The cursor address is given as an ordered pair of (x,y) coordinates. The possible sequences for the cursor address are column first and row second (column major order) or row first and column second (row major order). The value T sets the sequence for column major order. The value F sets the sequence for row major order. The setting has no effect until cursor addressing mode is set with the AFAC command.

To change the terminator display character to a number sign (#), enter:

```
AL=#
```

To change the terminator character back, enter reverse slash followed by the caret:

```
A=\^
```

To change the number of lines on the screen to 22, enter:

```
AV=22
```

To suppress the display of the NOT FOUND error message, enter:

```
AS=T
```

To set tabs at every 10 spaces, enter:

```
AT=10
```

To change the keyboard BREAK character to CNTL-G (which has a hexadecimal value of 07H), enter:

```
AB=7
```

To change the display character for all non-printing characters to a space, enter:

```
AC=
```

AF

Alter Function Command

The format of the Alter Function command is:

AF<code>=<value>

where

AF	is the command name.
<code>	is a two character string specifying the function to be changed. Possible codes and their values are described in the table below.
<value>	depends on which code is specified. Possible values are described in the table below. Every character entered after the equal sign is interpreted as part of the value, including spaces.

The Alter Function command modifies the communication link between the editor and the terminal. It can be used to configure an external terminal connected through one of the system's I/O ports to be used with CREDIT.

The first group of Alter Function codes take on a single character as a value.

Table 4-4 Alter Function Values—Part 1

CODE	DEFAULT	VALUE
WA	CNTL-Y	Specifies the wildcard character for matching any number of characters.
WC	CNTL-W	Specifies the wildcard character for matching either upper or lower case characters.
WJ	?	Specifies the wildcard character for matching any single character.
AC	00H (null)	Specifies the code used as the cursor movement command by the terminal. When the command is given, the coordinates of the new cursor address follow the code. The coordinates are given in the order specified by the AX command and with the offset specified by the AO command applied to them.
BK	<space>	Specifies the code which blanks out a single screen location.
IG	00H	Specifies a byte to be ignored whenever it is received from the keyboard as input. If IG is set to 00H, all bytes are accepted from the keyboard. The setting is specified as a single hexadecimal value.

Table 4-4 Alter Function Values—Part 1 (continued)

CODE	DEFAULT	VALUE
IL	00H	Has no effect. The IL command accepts a 0 to 4 byte hexadecimal value. It is provided to maintain compatibility with other Intel editors.
DL	00H	Has no effect. The DL command accepts a 0 to 4 byte hexadecimal value. It is provided to maintain compatibility with other Intel editors.

The next group of Alter Function codes take a zero to four byte hexadecimal value. Do not use the CNTL-B, the hexadecimal entry code for text handling commands, to enter these hexadecimal values. The hexadecimal value entered should correspond to a control sequence on the terminal. If a function is not available on the terminal, enter a null value by entering zero bytes after the equal sign. Sequences of less than four bytes are padded with null characters (00H).

The five codes beginning with a C specify values generated when the cursor control keys are pressed.

Table 4-5 Alter Function Values—Part 2

CODE	DEFAULT	VALUE
CD	1CH	Specifies the cursor down code.
CH	1DH	Specifies the cursor home code.
CL	1FH	Specifies the cursor left code.
CR	14H	Specifies the cursor right code.
CU	1EH	Specifies the cursor up code.

The six codes beginning with M specify values generated to move the cursor on the CRT display screen.

Table 4-6 Alter Function Values—Part 3

CODE	DEFAULT	VALUE
MB	0DH	Specifies the code to move the cursor to beginning of line.
MD	1BH 42H	Specifies the code to move the cursor down.
MH	1BH 48H	Specifies the code to move the cursor to the home position.
ML	1BH 44H	Specifies the code to move the cursor to the left.
MR	1BH 43H	Specifies the code to move the cursor to the right.
MU	1BH 41H	Specifies the code to move the cursor up.

The four codes beginning with an E specify the values generated to control the erasing of the CRT display.

Table 4-7 Alter Function Values—Part 4

CODE	DEFAULT	VALUE
EK	1BH 4BH	Specifies the code to erase the entire line.
EL	<space>	Specifies the code to erase the rest of the line following the cursor.
ER	1BH 4AH	Specifies the code to erase the rest of the screen following the cursor.
ES	1BH 45H	Specifies the code to erase the whole screen.

The eight codes beginning with X specify the values expected by the editor for screen mode commands. These codes can be changed to use function keys on the terminal and to change the command names for personal preference. Once a code is changed, the old code is available for use as a macro name, but the new code may not be used.

Table 4-8 Alter Function Values—Part 5

CODE	DEFAULT	VALUE
XA	CNTL-A (01H)	Specifies the code for the screen mode Insert command, CNTL-A.
XC	CNTL-C (03H)	Specifies the code for the screen mode Insert command, CNTL-C.
XD	CNTL-D (04H)	Specifies the code for the screen mode Delete command, CNTL-C.
XF	CNTL-F (06H)	Specifies the code for the screen mode Macro Function command, CNTL-F.
XN	CNTL-N (0EH)	Specifies the code for the screen mode Next Page command, CNTL-N.
XP	CNTL-P (10H)	Specifies the code for the screen mode Previous Page command, CNTL-P.
XV	CNTL-V (16H)	Specifies the code for the screen mode View Page command, NCTL-V.
XX	CNTL-X (00H)	Has no effect. XX is provided to maintain compatibility with other Intel editors.
XZ	CNTL-Z (1AH)	Specifies the code for the screen mode Delete command, CNTL-Z.

To change the wildcard character from the question mark to the asterisk, enter:

```
AFWJ=*
```

To change the code that the editor expects when the cursor up key is pressed to a three character sequence of ESC, left bracket, and capital A (1BH 5BH 41H), enter:

```
AFCU=1B 5B 41
```

To specify that the terminal being used does not have an erase rest of screen code, enter:

```
AFER=
```

To specify that the code required to move the cursor down the screen is a single linefeed (0AH), enter:

```
AFMD=0A
```

To change the screen mode View Page command to the hexadecimal sequence 7EH 1CH (the CLEAR key on the Hazeltine 1510 terminal), enter:

```
AFXV=7E 1C
```

?A

Display Alter Command

The format of the Display Alter command is:

```
?A
```

where

?A	is the command name.

The Display Alter command displays the current settings of alterable features of the editor.

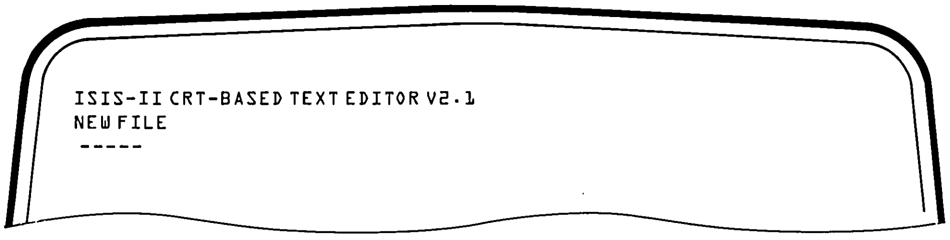
The editor displays a list of the alterable features and their current values.

To display the list, enter:

```
?A
```

Sample Alter Environment Session

The following session illustrates the Alter commands.



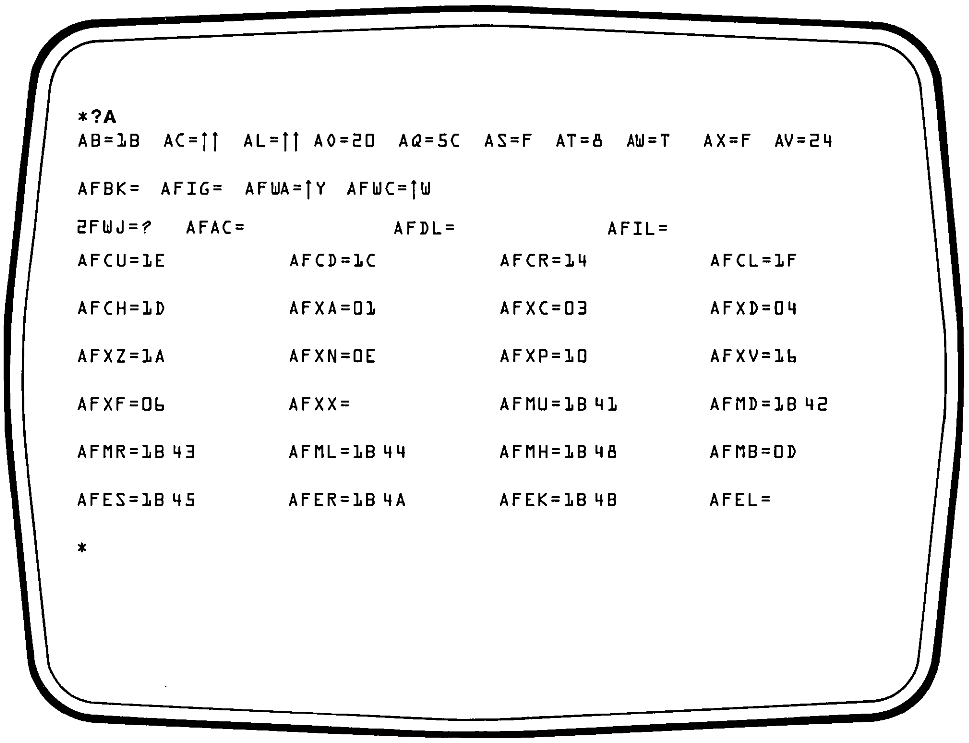
```
ISIS-II CRT-BASED TEXT EDITOR V2.1
NEW FILE
-----
```

Key-in Sequence

CREDIT NEWFIL.TXT
RETURN

Comments

Many of the parameters in the CREDIT environment can be changed. The examples in this section illustrate some of these varible features. Create a new file with the CREDIT command line as shown.



```
*?A
AB=1B  AC=↑↑  AL=↑↑ AO=20  AQ=5C  AS=F  AT=8  AW=T   AX=F  AV=24

AFBK=  AFIG=  AFWA=↑Y  AFWC=↑W

2FWJ=?   AFAC=            AFDL=             AFIL=
AFCU=1E          AFCD=1C          AFCR=14          AFCL=1F

AFCH=1D          AFXA=01          AFXC=03          AFXD=04

AFXZ=1A          AFXN=0E          AFXP=10          AFXV=16

AFXF=06          AFXX=            AFMU=1B 41       AFMD=1B 42

AFMR=1B 43       AFML=1B 44       AFMH=1B 48       AFMB=0D

AFES=1B 45       AFER=1B 4A       AFEK=1B 4B       AFEL=

*
```

Key-in Sequence

HOME

?A RETURN

Comments

To display the environment parameters, enter the command line mode by pressing the HOME key and enter the ?A command.

APPENDIX A CONFIGURING CREDIT™ FOR NON-INTEL TERMINALS

CREDIT is designed to run on any Intel microcomputer development system with an Intel terminal. The codes expected by the editor from the terminal or sent to the terminal are those used by the Intel terminals. Table A-1 is a list of the control codes used when CREDIT is running on an Intel terminal.

Using a non-Intel terminal with codes different from those of the Intel terminal, requires re-configuring CREDIT. The ALTER commands allow modification of certain keyboard and CRT codes. Additionally, it may be necessary to avoid the editing situations described in table A-1.

Table A-1 Intel Terminal Control Codes

Cursor Function	Code Produced Pressing Key	Code Sent to CRT to Move Cursor
Down	1CH,0	1BH,42H
Home	1DH,0	1BH,48H
Left	1FH,0	1BH,44H
Right	14H,0	1BH,43H
Up	1EH,0	1BH,41H

CRT Function	Input Code
Clear screen	1BH,45H
Clear rest of screen	1BH,4AH
Clear rest of line	not available
Clear line	1BH,4BH
Blankout character	20H(ASCII blank)

This appendix contains tested configurations for several non-Intel terminals. The terminals listed are the ones that have been tested. It is not a complete listing of all the terminals that can run with CREDIT.

To create a CREDIT configuration for a terminal not listed, compare the terminal's behavior in the situations described in the following list of actions expected by CREDIT. See the terminal's user manual for the codes expected and generated by the terminal.

CREDIT expects the following from a terminal:

- The ASCII codes 20H through 7EH display some symbol requiring a one column space. The carriage return (0DH), linefeed (0AH), and backspace (08H) perform their usual functions.
- There are cursor key output codes and CRT cursor move input codes for the cursor functions down, home, left, right, and up, and input codes for clear screen, clear test of screen, and clear line. The default codes, shown in Table A-1, can be changed with the Alter command.
- The home position is the upper left corner.

- 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 Alter command.

- The CRT has 22 to 25 lines. The default, 25 lines, can be changed with the Alter command.

- When the cursor is on the bottom line of the screen and RETURN is pressed, or wraps around from the right margin, the top screen line is deleted and the screen rolls up one line. If the terminal doesn't do this, avoid rolling the screen during text entry.

- CREDIT automatically generates a linefeed each time the carriage return is entered. The terminal should not generate a linefeed with a carriage return. In some terminals, this function can be switched on and off.

CREDIT Configuration Examples

When configuring the editor to execute with a non-Intel terminal, changing some or all of the codes assigned to the following Alter commands may be required:

- 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 erase rest of screen code—AFER.
- The erase line code—AFEK.
- The erase rest of line code—AFEL.
- 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 are—AFXA, AFXC, AFXD, AFXF, AFXN, AFXP. AFXV, and AFXZ. The user may want to change these codes to match function keys or other convenient keys on the terminal keyboard.

The most convenient way to configure CREDIT for a non-Intel terminal is to put the Alter commands into CREDIT.MAC. To enter the Alter commands after invoking the editor,type the Intel-terminal HOME code (1DH) to enter command mode. The key that generates this code varies from terminal to terminal. Check the terminal manual for the means to generate this code.

The following sections list the Alter functions and values required to run CREDIT on the Intel tested terminals. The terminals are:

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

The Alter commands to configure CREDIT for the tested terminals are included on disk, with the CREDIT program. The name of the file is included in each table. The best way to use the commands in the file for your terminal is to rename it CREDIT.MAC with the ISIS Rename command.

```
RENAME ADDS.MAC TO CREDIT.MAC
```

Each time you call the editor with the CREDIT command, the command file, CREDIT.MAC, will be loaded and executed. If you want to define macros for use with the editor, simply add the MS commands at the end of the file.

NOTE

Remember that the maximum file size of CREDIT.MAC is 2K bytes.

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 is used to display the operating condition of the terminal.

Because ↑A, ↑F, and ↑Z are cursor control keys on the ADDS 200, the screen functions for these keys are reassigned as follows:

Add test (↑A)	changed to (↑T)	(14)
Macro call (↑F)	changed to (↑E)	(05)
Zap text (↑Z)	changed to (↑K)	(0B)

CREDIT initially comes up in the command mode. Enter ↑V before entering any data. The Alter configuration required for the ADDS Regent Model 200 terminal is shown in Table A-2.

Table A-2 ADDS Regent Model 200 Alter Commands

Function Code	Hexadecimal Value	Graphic or ASCII Name
CD	0A	Line Feed
CH	01	SOH
CL	15 or 08	NAK or BS
CR	06	ACK
CU	1A	SUB
MD	0A	Line Feed
MH	none	
ML	15 or 08	NAK or BS
MR	06	ACK
MU	1A	SUB
AC	1B 59	ESC Y
EK	1B 4B	ESC K
ER	1B 6B	ESC k
ES	not used	
AV		24

Command file: ADDS.MAC

```
AFCU=1A;AFCD=0A;AFCR=06;AFCL=15;AFCH=01;
AFMU=1A;AFMD=0A;AFMR=06;AFML=15;AFAC=1B 59;
AFMH= ;AFES= ;AFER=1B 6B;AFEK=1B 4B; AV=24; AFXA=14;
AFXF=05;AFXZ=0B
```

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 CREDIT. 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. Table A-3 lists the Alter functions and values needed to run the CREDIT text editor on the Beehive Mini-Bee terminal.

Table 4-3 Beehive Mini-Bee Alter Commands

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

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
```

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 the control-K (↑K) is totally a personal preference. This terminal does not have a HOME key. Try using the control-O (↑O) for the HOME function. Table A-4 shows the Alter commands necessary for the DEC VT 52 terminal.

Table A-4 DEC VT 52 Alter Commands

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	
EL	1B 4B	ESC K
ER	1B 4A	ESC J
ES	not used	
V		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;AFER=1B 4A;AFEL=1B 4B;AFES= ;
AB=0B;AV=24
AFAC=1B 59;AW=F
```

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 CREDIT. 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 changed so that the default ESCAPE code can be used; the choice of the control K is arbitrary. This terminal does not have a HOME key. The choice of the control-O (↑O) for HOME function is arbitrary. The Alter commands used with the DEC VT 100 terminal are shown in Table A-5.

Table A-5 DEC VT 100 Alter Commands

Function Code	Hexadecimal Value	Graphic or ASCII Name
CD	1B 42	ESC B
CD	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 used	
W	F	
V		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
AB=0B;AV=24
AW=F
```

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. Choose between the ESC or the tilde character (~) as a control sequence lead-in. It is advisable to use the tilde; using the ESC means changing the BREAK character.

To define a macro which issues a Hazeltine 1510 HOME code, do not use the HOME key because it generates a control-R code which echoes the command line. Literalize the HOME codes by typing backslash-tilde-backslash-control-R.

The previous page screen command is changed from control-P to control-O to avoid a conflict with cursor-right key.

Table A-6 lists the Alter commands required for the Hazeltine 1510 terminal when using the tilde (~) character as the control sequence lead-in. When using the ESC key as the control-sequence lead-in refer to Table A-7.

Table A-6 Hazeltine 1510 Alter Commands (Tilde Control-Sequence Lead-in)

Function Code	Hexadecimal Value	Graphic or ASCII Name
CD	7E 0B	~VT
CH	7E 12	~DC2
CL	08	~BS
CR	10	~DLE
CU	7E 0C	~FF
MD	7E 0B	~VT
MH	7E 12	~DC2
ML	08	~BS
MR	10	~DLE
MU	7E 0C	~FF
AC	7E 11	~DC1
X	T	
O	0	
EK	0D 7E 0F	CR~SI
ER	7E 18	~CAN
ES	7D 1C	~FS
XP	0F	SI
V		24

Command file: 1510T.MAC

```
AFCU=7E 0C;AFCD=7E 0B;AFCR=10;AFCL=08;AFCH=7E 12
AFMU=7E 0C;AFMD=7E 0B;AFMR=10;AFML=08;AFMH=7E 12
AFER=7E 18;AFES=7E 1C;AFEK=0D 7E 0F
AV=24;AFXP=0F
AX=T;AO=0;AFAC=7E 11
```

Table A-7 Hazeltine 1500 Alter Commands (ESC Key Control-Sequence Lead-in)

Function Code	Hexadecimal Value	Graphic or ASCII Name
CD	1B 0B	ESC VT
CH	1B 12	ESC DC2
CL	08	ESC BS
CR	10	ESC DLE
CU	1B 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
AC	1B 11	ESC DC1
W	T	
O	0	
EK	1B 0F	ESC SI
ER	1B 18	ESC CAN
ES	1B 1C	ESC F5
XP	0F	SI
V		24
B	7E	~

Command file: 1510E.MAC

```
AFCU=1B 0C;AFCD=1B 0B;AFCR=10;AFCL=08;AFCH=1B 12
AFMU=1B 0C;AFMD=1B 0B;AFMR=10;AFML=08;AFMH=1B 12
AFER=1B 18;AFES=1B 1C;AFEK=0D 1B 0F
AB=7E;AV=24;AFXP=0F
AX=T;AO=0;AFAC=1B 11
```

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. Table A-8 catalogs the Alter functions and values needed to run the CREDIT text editor on the Lear Siegler ADM-3A terminal. Table A-8 catalogs the Alter functions and values needed to run the CREDIT text editor on the Lear Siegler ADM-3A terminal.

Table A-8 Lear Siegler ADM-3A Alter Commands

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
V		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=
```

APPENDIX B CREDIT^{TM} EDITING COMMAND SUMMARY

The following tables are an alphabetical list of the commands and features available in the CREDIT text editor.

Table B-1 Screen Editing Commands

Command	Function
CNTL-A	Insert until second CNTL-A is typed. Clear remainder of screen for insert.
CNTL-C	Insert single character at cursor, moving rest of line right.
CNTL-D	Delete single character at cursor, moving rest of line left.
CNTL-F	Expand and execute macros.
CNTL-N	Display next page of test.
CNTL-P	Display previous page of text.
CNTL-V	Display current page of text, with possible reformatting.
CNTL-Z	Delete text from first CNTL-Z up to but not including second CNTL-Z.

Table B-2 General Editing Functions

TO MOVE CURSOR:	Use the four directional arrow keys on keyboard.
TO MODIFY TEXT:	Type new character over old character.
TO SWITCH MODES:	Press HOME key to enter command mode; press CNTL-V to enter screen mode.

Table B-3 Command Editing Commands

NAME	FORMAT	FUNCTION
Alter	Acode=new value	Change editing environment to new value.
	?A	Display current Alter commands values.
Begin File	B	Go to beginning of Read file
Close File	CR or CW	Close Read or Write data file
Copy Text	XC Tag,n I -n I tag	Copy text from tag boundary to pointer.

Table B-3. Command Editing Commands (continued)

NAME	FORMAT	FUNCTION
Deletion	DC[n l -n l tag]	Delete n characters, or text up to, but not including tag.
	DL[n l -n l tag]	Delete n lines forward or backward.
Exit Loop	EL	Exit current iteration loop.
Exit Quit	EQ	Exit; disregard editing changes to file.
Exit	EX[filename]	Exit; save editing changes to file.
Find	F/text/[n l -n l tag]	Find /string/; move pointer if found.
Get	G[filename]	Read text of filename as command.
Help	H	Display menu of editor commands.
Insert	I/text/	Insert /text/ in front of pointer.
Jump	J[n l -n l tag]	Move pointer n characters or to tag
Line Move	L[n l -n]	Move pointer n lines forward or backward
Macros	MD<name>l *	Delete macro name or all macros.
	MF<name>[parms]	Expand and execute macro.
	MS<name>/macro/	Define macro name.
	?M	Print names and definitions of macros.
Move Text	XM tag,[n l -n l tag]	Move text from tag boundary to pointer.
Open Read	OR	Open data file for read.
Open Write	OW	Open data file for write.
Print	P[H][n l -n l tag]	From pointer, print n lines or up to tag
Query	QF;[command]	Do command only if Query Flag is false.
	QT;[command]	Do command only if Query Flag is true.
	QU	Query user. Set Query Flag.
Read	R[n]	Insert n lines from data file at pointer.
Substitute	S[Q]/old/new/	New text replaces old text if old text is found.
	[n l -n l tag]	With SQ form, user is queried before change.
Tag	TD[n]	Delete tag. n=0 to 9.
	TS[n]	Set tag. n=0 to 9.
User	U/text/	Displays text when command executes.
Write	W[n l -n l tag]	Write n lines to data file previously opened.
Yes Flag	YF;[command]	Do command only if Yes Flag is false.
	YT;[command]	Do command only if Yes Flag is true.

APPENDIX C
CREDIT™ ERROR MESSAGES

The editor detects errors in both screen mode and command mode. Most of the errors are different for each mode, but two messages occur in either mode:

- INSUFFICIENT MEMORY
- CREDIT ERROR

INSUFFICIENT MEMORY occurs when there is less than 64K bytes of memory available in the system. A CREDIT ERROR occurs when the editor detects an internal problem.

The user should ensure that the free disk space is greater than two times the size of the file being edited. If the disk is full, CREDIT exits with an ISIS fatal error 7, INSUFFICIENT DISK SPACE.

SCREEN EDITING ERRORS

When in screen editing mode, the only printing error messages are those associated with macros. Illegal commands cause the beeper to sound. Some typical error conditions in screen mode are:

- Attempting to move the cursor out of the display window
- Attempting to modify a screen location that does not contain text
- Attempting to insert or modify a non-literalized, invisible character using CNTL-A or CNTL-C
- Pressing any key except cursor control keys or CNTL-Z when deleting with the CNTL-Z function
- Attempting to modify or delete the end-of-file character

SYNTAX ERRORS

An error made during startup or during command editing produces an error message and a repeat of the command line up to the point where the error occurred. A syntax error can occur in screen editing if an invalid or undefined macro is referenced. The error messages are:

UNCLOSED STRING
: The string delimiter was not found before the end of the command line.

FILE ACCESS ERROR
: Attempt to access a file improperly, such as writing to a write protected file, reading from :LP:, or attempting to read or write an unopened file.

FILE IN USE
: The file specified is already open. A second OR or OW command was issued without closing the file first.

UNRECOGNIZED COMMAND
The command characters are not a valid editor command.

IMPROPER OPERAND
The command argument is the wrong type for the command, or it contains a syntax error.

MISSING OPERAND
The command lacks a required argument.

ILLEGAL VALUE
The number argument contains non-numeric characters, or is out of range (-32767 through 32767). Some commands restrict parameters to a limited range. For example, AV only accepts 22-25 as a value.

ITERATION ERROR
An illegal iteration quantifier was encountered, or there were too many or too few ending brackets, or the iteration was more than seven levels deep.

ARGUMENT MISMATCH
During expansion of a macro, more percent signs were encountered than parameters in the parameter list, or vice versa.

BUFFER FULL
An attempt was made to put more than 2000 characters in the command buffer. This could occur during the I command, macro expansion, the G command, or when entering a command line. An error while entering a command line offers the option of executing the line as entered.

TERMINATOR EXPECTED
A syntactically correct command was terminated by something besides a semicolon, RETURN, or a right bracket.

ILLEGAL NAME
An illegal filename, tag specifier, or macro name was used.

DOESN'T EXIST
The specified file does not exist, or the specified tag or macro has not yet been defined.

ALREADY EXISTS
The file specified for creation already exists, or the macro or tag specified is already defined.

TAG POSITION
A tag specified for the DC, XC, or XM command end boundaries is before the starting boundary.

UNKNOWN CURSOR KEY
A command character, that is the start of one of the input cursor control sequences, was received, but the following command characters did not complete any of the known cursor control sequences.

APPENDIX D
ASCII CODES

Table D-1 ASCII Code List

Decimal	Octal	Hexadecimal	Character
0	000	00	NUL
1	001	01	SOH
2	002	02	STX
3	003	03	ETX
4	004	04	EOT
5	005	05	ENQ
6	006	06	ACK
7	007	07	BEL
8	010	08	BS
9	011	09	HT
10	012	0A	LF
11	013	0B	VT
12	014	0C	FF
13	015	0D	CR
14	016	0E	SO
15	017	0F	SI
16	020	10	DLE
17	021	11	DC1
18	022	12	DC2
19	023	13	DC3
20	024	14	DC4
21	025	15	NAK
22	026	16	SYN
23	027	17	ETB
24	030	18	CAN
25	031	19	EM
26	032	1A	SUB
27	033	1B	ESC
28	034	1C	FS
29	035	1D	GS
30	036	1E	RS
31	037	1F	US
32	040	20	SP
33	041	21	!
34	042	22	“
35	043	23	#
36	044	24	$
37	045	25	%
38	046	26	&
39	047	27	’
40	050	28	(
41	051	29	)
42	052	2A	*
43	053	2B	+
44	054	2C	’
45	055	2D	—
46	056	2E	.
47	057	2F	/
48	060	30	0
49	061	31	1
50	062	32	2
51	063	33	3
52	064	34	4

Table D-1 ASCII Code List (continued)

Decimal	Octal	Hexadecimal	Character
53	065	35	5
54	066	36	6
55	067	37	7
56	070	38	8
57	071	39	9
58	072	3A	:
59	073	3B	;
60	074	3C	<
61	075	3D	=
62	076	3E	>
63	077	3F	?
64	100	40	@
65	101	41	A
66	102	42	B
67	103	43	C
68	104	44	D
69	105	45	E
70	106	46	F
71	107	47	G
72	110	48	H
73	111	49	I
74	112	4A	J
75	113	4B	K
76	114	4C	L
77	115	4D	M
78	116	4E	N
79	117	4F	O
80	120	50	P
81	121	51	Q
82	122	52	R
83	123	53	S
84	124	54	T
85	125	55	U
86	126	56	V
87	127	57	W
88	130	58	X
89	131	59	Y
90	132	5A	Z
91	133	5B	[
92	134	5C	\
93	135	5D	]
94	136	5E	∧
95	137	5F	—
96	140	60	'
97	141	61	a
98	142	62	b
99	143	63	c
100	144	64	d
101	145	65	e
102	146	66	f
103	147	67	g
104	150	68	h
105	151	69	i
106	152	6A	j
107	153	6B	k
108	154	6C	l
109	155	6D	m
110	156	6E	n
111	157	6F	o
112	160	70	p

Table D-1 ASCII Code List (continued)

Decimal	Octal	Hexadecimal	Character
113	161	71	q
114	162	72	r
115	163	73	s
116	164	74	t
117	165	75	u
118	166	76	v
119	167	77	w
120	170	78	x
121	171	79	y
122	172	7A	z
123	173	7B	{
124	174	7C	▮
125	175	7D	}
126	176	7E	~
127	177	7F	DEL

APPENDIX E CHANGES IN CREDIT™ VERSION 2.1

Version 2.1 of CREDIT has additions and changes that may alter the operation of previously defined command files and macros which worked under Versions 1.0 or 2.0. These alterations are described as follows:

1. The EX command now requires a space between EX and a filename if the file is a drive 0 file without :F0: given.

2. CREDIT detects when it is running on a newer model Intel development system and sets its default codes for that system.

3. New Alter commands have been added, including ones to take advantage of direct cursor addressing, to cope with no wraparound, and to change the literalizing character.

4. CREDIT formerly ignored the top (eighth) bit of all keyboard input bytes. Version 2.1 continues to do so unless any of the keyboard input control codes is set with a 1 in the top bit (i.e., with a byte greater than or equal to hexadecimal 80). In this case, the entire byte is accepted.

5. CREDIT does not attempt to detect a DISK FULL situation Always make sure there is more than twice as much free space on the disk as the size of the file being edited.

Earlier changes from Version 1.0 to 2.0 are summarized as follows.

In thinking about ranges, consider tags as existing between the character pointed at and the prior character. This fact applies to several of the discussions below. (See the index for more detail on each one.)

1. CNTL-Z: Formerly this Delete Text command included the character under the final CNTL-Z. This is no longer so: the range of the deletion ends with the prior character. One consequence is that a line can be deleted by positioning the second CNTL-Z at the beginning of the next line rather than at the end of the line to be deleted. It is also legal to position the second CNTL-Z above the first.

2. Closing tag: The character pointed at by the closing tag is not in the range affected by the command.

3. Tags in block moves: When a block of text is moved with the XM command, tags pointing to characters within the block are moved with block; i.e., each tag continues to point to the same character in the block that it did before the move.

   When the block is copied with the XC command, such tags do not move with the copied block, but remain in their original positions. See Chapter 3

4. Tag redefinition or deletion: Redefining an existing tag and deleting a nonexistent tag are no longer considered errors. Thus, such actions no longer cause an exit from a command string.

5. CREDIT.MAC: If it exists, this file is automatically read and executed as the first action after CREDIT is invoked. This action is suppressed if the command line invoking CREDIT explicitly stated NOMACRO. See Chapter 4 for further details on options and error conditions.

6. A macro whose name is a control key may be invoked simply by pressing the control key.

7. Failed Finds: The "F" and "S" commands automatically exit from the innermost iterative loop when the specified string is not found. This can be circumvented by placing an extra set of angle brackets around the command. A failed find outside of all iterations causes the command string to abort.

8. Non-Intel CRTs: If the CLEAR SCREEN output code has been changed, CREDIT starts out in command mode rather that screen editing mode, thus preserving the sign-on message.

9. Hexadecimal text can now be manipulated and printed out:

    a. Commands which accept delimited strings (i.e., I,F,S,SQ,MS,U) take hexadecimal input if the delimiter used is CNTL-B.

    b. The Print Hexadecimal command (PH) displays the hexadecimal codes beginning at the cursor or at a specified tag.

10. The Alter command has been expanded, and requires hexadecimal input in several cases where ASCII was formerly accepted. Macro files using such codes under Version 1.0 must be changed. See Chapter 4.

INDEX

intel®

REQUEST FOR READER'S COMMENTS

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

Please restrict your comments to the usability, accuracy, readability, organization, and completeness of this publication. If you have any comments on the product that this publication describes, please contact your Intel representative. 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 improvement.

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

WE'D LIKE YOUR COMMENTS ...

This document is one of a series describing Intel products. Your comments on the back of this form will help us produce better manuals. Each reply will be carefully reviewed by the responsible person. All comments 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®

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

Printed in U.S.A.