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

Order Number: 9800902-02

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

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

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

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

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

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

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

BXP	Intelevision	Multibus
CREDIT	Intellec	Multimodule
i	iRMX	Plug-A-Bubble
ICE	iSBC	PROMPT
iCS	iSBX	Promware
i_m	Library Manager	RMX/80
Insite	MCS	System 2000
Intel	Megachassis	UPI
int_el	Micromap	μScope

and the combination of ICE, iCS, iRMX, iSBC, iSBX, MCS, or RMX and a numerical suffix.

PRINTED IN USA/A350/481/15K CP

PREFACE

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

The User's Guide is organized into five chapters and five appendices:

- Chapter 1. Introduction, a description of concepts essential to using CREDIT, and a tutorial section.
- Chapter 2. Starting and Ending an Editing Session, a description of invoking a file for editing, renaming it, and leaving the editor.
- Chapter 3. Screen Editing, a description of editing functions that act directly on text shown on the CRT screen.
- Chapter 4. Command Line Editing, a description of editing functions that are typed into a line of commands and then executed.
- Chapter 5. Advanced Editing Commands, a description of editing functions that support macros, multiple iteration of commands, conditional execution, command and side files, and commands to alter CREDIT to fit particular applications.
- Appendix A. Configuring CREDIT for Non-Intel Terminals, a description of the process of configuring the editor for different terminals and listings of tested configurations for specific terminals.
- Appendix B. CREDIT Editing Command Summary, a listing of the CREDIT commands with their syntax.
- Appendix C. CREDIT File Usage, a description of the ISIS-II files used by CREDIT.
- Appendix D. CREDIT Error Messages, a listing of the error messages issued by CREDIT and a description of the causes.
- Appendix E. ASCII Codes, a listing of the ASCII codes with their hexadecimal and decimal values.
- Appendix F. Changes in CREDIT, V2.0, a description of the enhancements and other changes made in Version 2.0 of CREDIT.

Notation Conventions

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

BOLDFACE indicates command text given to CREDIT.

ITALIC indicates arguments used with commands, or emphasis.

Brackets ([ ]) indicate optional syntactic elements.

The vertical slash (|) indicates choice of syntactic elements on either side.

Braces ({ }) indicate a mandatory choice from two or more syntactic elements.

Related Publications

The ISIS-II CREDIT (CRT-Based Editor) User's Guide assumes a working familiarity with the *ISIS-II User's Guide*, 9800306.

CONTENTS

CONTENTS (Cont'd.)

TABLES

ILLUSTRATIONS

CHAPTER 1
INTRODUCTION

CREDIT is a screen-oriented editor for use with ISIS-II on an Intellec Microcomputer Development System or an Intellec Series II Microcomputer Development System with 64K bytes of memory. CREDIT lets you display a file, move the cursor to any point in the text, make insertions, deletions, or other corrections, and see the results of the changes immediately. You can page forward or backward through the file, and correct misspellings by simply positioning the cursor at the incorrect character and typing the correct one.

To add text to the middle of a file, you move the cursor to the point of insertion, type Control-A (the Add Text command character), enter the new text, and type Control-A again. Deleting text, whether individual characters or several lines, is equally simple. In all cases, you see the changes as you make them.

CREDIT also has a set of commands for command-mode editing. These include the more complex editing functions such as move, copy, command iterations, macro definition, and external file operations. Unless you request that the modified line or lines be displayed, you do not see the results of command-mode editing commands.

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

This chapter contains two major sections: an introduction to the basics of the editor, and a tutorial session to introduce you to the basic commands and give you a little experience before going into the complete command set in the following chapters.

The Editor Basics

Before starting to use the editor you should understand certain concepts:

- Screen-mode editing
- Command-mode editing
- The CREDIT display
- Lines and line terminators
- Data, controls, and the literalizing character
- Printing and non-printing characters
- The keyboard
- The pointer, which marks a position in the file
- The cursor, which marks a location on the screen
- Tags

Screen-Mode Editing

Much of the power of CREDIT comes from its ability to display changes to the text as you make them. You can examine a screenful of text, locate the text you wish to change, change it with a function key, and review the next screenful of text. By moving through the file, you can quickly make changes, insertions, and deletions, verifying them as you go.

If you need more powerful editing functions, you can enter the command line editing mode.

Command-Mode Editing

CREDIT includes a set of commands for more powerful and complex editing functions. In addition to moving and copying text from one place in the file to another, you can design your own set of commands by defining *macros* that consist of other CREDIT commands.

You can perform all the editing functions (replacement, insertion, and deletion) with command-mode functions, plus many more. The results are not displayed, however; in command mode, you must tell CREDIT to display text (with the Print command).

The CREDIT Display

The Intel terminals have a 25-line display screen. Other terminals you may use might have more or fewer lines. While CREDIT is running, the screen is divided into two parts. In screen mode, the bottom 20 lines of the screen, called the *text area*, display text from the file. In the remaining lines, called the *command area*, the commands entered in command mode and the responses to the commands are displayed. It is also the area where CREDIT displays error messages. The two areas are separated by a line of five dashes. As commands are entered into the command area, the area grows and moves into the text area; the line of dashes and any data displayed in the text are erased from the screen.

The text area is restored to a full 20 lines when you switch to screen mode. Figure 1-1 shows the CREDIT display in screen mode at the start of an editing session.

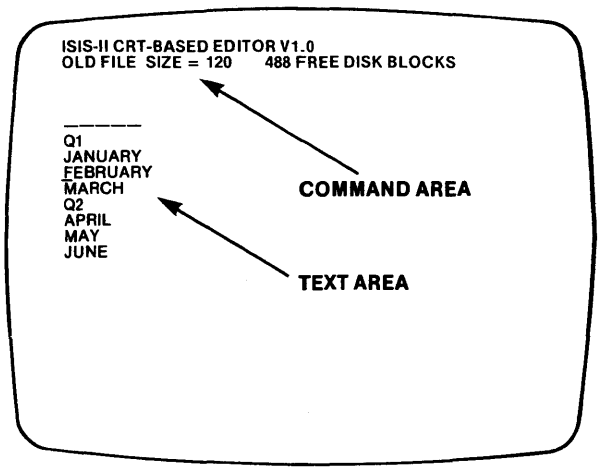


Figure 1-1. The CREDIT™ Display 902-1

Data, Controls, and the Literalizing Character

All characters typed on the keyboard go to the editor. Certain characters have special meaning to the editor. The special characters are control characters, the literalizing character, and the Escape (ESC) character. A control character is any character entered while the Control key (CNTL) is depressed. Whether a character is interpreted as data depends on which mode you are in. In screen mode, any special character that is not interpreted as a command or terminal function causes the warning beep to sound; all other characters are accepted as data. When the warning beep sounds, it means the character was not accepted. In command mode, every character

is accepted as data, a command, or a control. A character accepted as data or a command is displayed on the screen. A character accepted as a control causes an action on the screen. For example, the Control-H is the backspace character, which when pressed, causes the cursor to move one position to the left. The Control-A, on the other hand, has no special meaning in command mode and displays on the screen as "↑A."

Literalizing Characters

You can enter any character (control character or otherwise) as data by preceding it with a reverse slash (\), the *literalizing character*. The first character that follows the literalizing character is accepted as data. You can literalize any character except the ESC character, even those that don't need it. For example, to enter a Control-A into the file, type "\↑A". The reverse slash, itself, is a special character and must be literalized to be entered as data. That is, to enter a reverse slash, you must type the character twice (\ \).

In screen mode, the literalizing character is overlayed by the character being literalized. That is, the reverse slash is displayed when it is typed but the cursor does not move to the next character position. When the character being literalized is typed, the appropriate graphic replaces the reverse slash. In command mode, the reverse slash is displayed on the screen and the literalized character follows it.

Remember, the literalizing character literalizes only one character. If you literalize RETURN (which generates a carriage return-linefeed combination) only the first character, the carriage return, is literalized; the second is treated as a lone linefeed. To enter a lone carriage return or lone linefeed, you can use the hexadecimal entry facility described later in this chapter.

Lines and Line Terminators

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 when you type RETURN. The lines need not be limited to 80 characters (the width of the display), but it is generally easier to work with a file if each line fits on a display line.

The line terminator is displayed as an up arrow (↑) on the display. It is treated as a single character by CREDIT.

NOTE

You can change the character displayed to represent the line terminator with the Alter command. See "Chapter 5: Advanced Editing Techniques."

Printing and Non-Printing Characters

The screen mode, a code that has a graphic symbol, is displayed on the screen. A character that has no graphic symbol displays as an up arrow (↑). You can point to a character displayed as an up arrow with the cursor and replace or delete it like any other character.

In command mode, all codes that have a graphic symbol are displayed on the screen. A control character displays as an up arrow (↑) followed by the printing character. For example, Control-A displays as "↑A". Codes that do not have a graphic symbol associated with them are displayed in command mode as "↑#".

The Keyboard

Figure 1-2 shows the Intel keyboard. The editor supports the ASCII character set, which is shown in Appendix E.

Most of the keys are self-explanatory. Some, however, perform functions rather than enter characters:

- The CNTL key changes the function of some keys on the keyboard. To change the function of a key, hold down the CNTL key and press the other key. For example, to enter Control-A (the Add Text command in screen mode), hold down the CNTL key and type A.
- In screen mode, the HOME key switches CREDIT to command mode. In command mode, it performs no valid function.
- In screen mode, the cursor control keys (↑←→↓) move the cursor in the direction indicated by the arrow. In command mode, the cursor control keys perform no valid function.
- In either mode, the ESC key aborts commands. When you press ESC, <BREAK> displays in the command area and the command terminates.
- The RUBOUT key performs the normal RUBOUT function (delete the previous character) in command mode and during text insertion (↑A) in screen mode. Otherwise in screen mode, the RUBOUT key performs the same function as the left cursor key: it moves the cursor one position to the left.

Hexadecimal Entry

You can enter characters by typing their hexadecimal code bracketed by Control-B (↑B) characters. For example, to enter an ASCII NUL character (hexadecimal 00) enter "↑B00↑B". The editor interprets this string as a single character. Multiple characters can be entered:

```
↑B414243444546↑B
```

The preceding sequence enters the string "ABCDEF".

You can insert spaces or carriage returns (RETURN) between hexadecimal codes for ease of reading:

```
↑B 41 42 43 44<RETURN>
45 46 ↑B
```

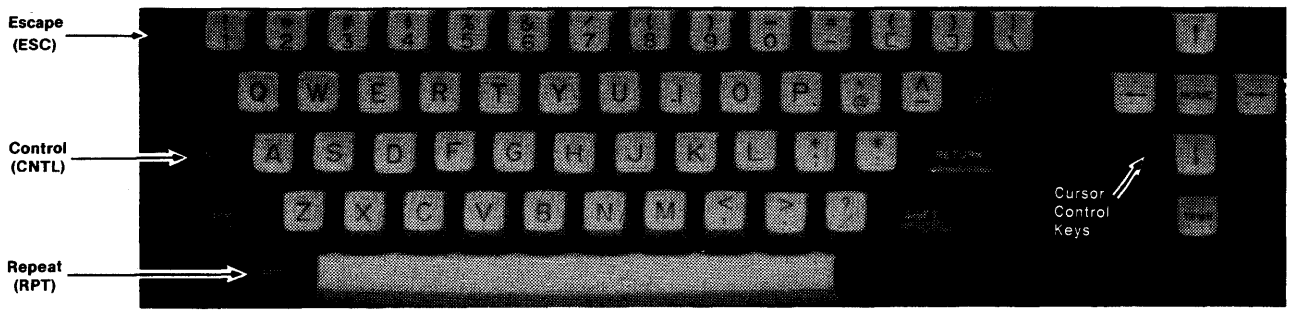


Figure 1-2. The Intel® Keyboard 902-2

results in the same data being entered as with:

```
↑B41424344454 6↑B
```

The blank or carriage return is assumed to be the breaking point between characters. That is, the first two digits following a blank are assumed to be a hexadecimal character. A blank may leave a single digit in the preceding substring or may precede a single digit. An isolated single digit is assumed to have a leading zero. For example, "↑BA↑B" is treated as "↑B0A↑B" and "↑B 414 243 ↑B" is treated as "↑B 41 04 24 03 ↑B", not "↑B 41 42 43 ↑B".

Any character other than valid hexadecimal digits (0-9, A-F) or blanks that are entered between the Control-Bs results in a syntax error.

The Control-B can be used in any command mode command that uses strings set off by delimiters; it cannot be used in screen mode. The Alter command expects hexadecimal entry but does not use the Control-B. Use of the Control-B with the Alter command causes an error.

The Pointer

CREDIT maintains a pointer that marks a character in the file. All changes are made relative to the position of the pointer. Deleting a single character, for example, erases the character where the pointer is; insertions are made immediately *preceding* the pointer. Several commands move the pointer.

The Cursor

The *cursor* is a blinking underline (_) character. In screen mode, if the cursor points to a character, it represents the position of the pointer.

NOTE

When the cursor is not pointing to a visible character or the line terminator, you cannot enter any command that changes text.

In command mode, the cursor has no association with the pointer. It merely indicates where command lines are displayed as you enter them.

Tags

CREDIT marks the beginning and end of the file and the beginning and end of the portion of the file in memory. The marker's CREDIT uses are called *tags*. The tags marking the beginning and end of the file and the portion in memory are permanent.

You can specify locations for ten additional tags with the Tag Set command, and use them as destinations for pointer-manipulation commands or range boundaries in other commands.

The ten user-defined tags are named T0 through T9; the four permanent tags have the the following names:

- TT — Top of Text (beginning of the file)
- TE — End of file
- TB — Beginning of portion of file in memory
- TZ — End of portion of file in memory

Configuring the Editor for Non-Intel Terminals

CREDIT runs properly, without reconfiguration, on an Intel terminal or any terminal that uses the same codes for terminal functions, such as cursor movement, screen erase, etc. CREDIT also has facilities that let you redefine all the terminal control codes, so you can use the editor on a variety of terminals. Appendix A describes the procedure of reconfiguring the editor. It also contains tested configuration data for several popular terminals. The reconfiguration is accomplished with the Alter command, which is described in "Chapter 5 Advanced Editing Techniques".

A CREDIT Tutorial

This section is a short tutorial to illustrate the use of the CREDIT commands you use most of the time. The functions covered are:

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

The purpose of this section is to get you started, not to teach you the details of the editor. Remember, for a brief description of the commands, enter H (the Help command) in command mode. Each command is described in detail in chapters 2, 3, and 4.

To use this section, you should be at an Intellec system. The system disk containing CREDIT should be loaded and ISIS-II should be in control of the system. See the *ISIS-II User's Guide* for additional ISIS-II information.

Activate the editor with the CREDIT command. This command either edits an existing file or creates a new file. The command is the word CREDIT followed by a space and the file name. For this tutorial, enter the following command to create a new file (change the file name if you already have a file named TEST.CRD on the system disk):

```
CREDIT TEST.CRD
```

The editor responds by displaying the following message at the top of the screen:

```
ISIS-II CRT-BASED TEXT EDITOR Vx.y
NEW FILE nnnn
FREE DISK FILES
```

The x.y in this example represents the version of the editor. The nnnn in the second line represents the number of unused blocks left on the disk.

The vertical bar (|) in the text area (below the line of dashes) represents the end-of-file character.

Under the end-of-file character is the cursor. You can move the cursor around the screen with the cursor control keys. The cursor moves in the direction of the arrow on the key top. If you try to move the cursor beyond its limits, the beep sounds and the cursor does not move. The limits are the right side, left side, and bottom of the screen. The top limit is the line below the five dashes.

The Add Text command (Control-A) tells the editor to accept text typed at the keyboard and enter it into the file. Move the cursor to the end-of-file character and type Control-A (hold the CTRL key down and type A). The beep sounds if you didn't position the cursor under the end-of-file character.

Type a line of text (a line of source code, a sentence, etc.). At the end of the line, type RETURN. This causes the character that represents the line terminator to be displayed.

You now have a line of text on the screen. The cursor is blinking at the start of the line directly beneath your text. Type in another word, but misspell it. To correct the error, simply press the RUBOUT key. Each time you press RUBOUT, the cursor backs up one column and erases the character there. When the erroneous character is erased, simply type the correct character(s). Type Control-A to complete the insertion.

To create a file of test data, we'll delete the lines you just entered, then enter a series of short lines. First, position the cursor at the beginning of the first line and press Control-Z (the Delete Text command character). The first character changes to an at-sign (@). Use the down arrow key to move the cursor to the end of file marker (note that each character except the line terminator is displayed as an at-sign when the cursor is on it). Now type Control-Z again to complete the deletion. The file is now empty, but the end-of-file character is still displayed.

Enter the following lines, ending each with RETURN. Several of the words are deliberately misspelled; enter them as shown:

```
Q1
January
Febuary
March
Q2
April
Mya
June
Q3
Augudt
September
September
Q4
October
Novemeber
December
```

Complete the insertion by typing Control-A again.

August is misspelled (Augudt). To correct it, position the cursor at the d and type "s". To correct May (Mya), position the cursor at the y and type "ay".

To insert the missing r in February (Febuary), position the cursor at the u, type Control-C (the Add Character command character), and type "r". To delete the extra e from November (Novemeber), position the cursor at the extra (second) e and type Control-D (the Delete Character command character).

For a brief tour of command mode, press the HOME key. Now the cursor is in the top line of the display immediately following the asterisk (*). You can no longer move the cursor with the cursor control keys. (Try it, but erase the resulting symbols with RUBOUT. If you don't, CREDIT displays an error message).

NOTE

Each command line must end with the RETURN key. This is not shown in the sample commands here, but the command isn't sent to CREDIT until you type RETURN.

Move the cursor to the beginning of this short file:

```
JTT
```

This is the Jump command (J), followed by the permanent tag that marks the beginning of the file (TT, the Top of Text). Note that nothing is changed in the text area, and the cursor is now in the second line of the command area, waiting for another command.

Display the entire file:

```
P16
```

This is the Print command (P), which specifies the screen to display (print) 16 lines. Now the command area takes up most of the screen. The pointer is still at the beginning of the file, because the Print command doesn't change the location of the pointer. The cursor is in the line following the last line of the file that the Print command displayed, waiting for another command.

Although you can see the entire file now, you still can't change any of the text by pointing with the cursor because you're still in command mode.

Move the pointer to the line that contains Q2:

```
L4
```

This is the Line command (L), which moves the pointer a specified number of lines (4 in this case) and positions the pointer at the first character of the line. Now display the line with the Print command:

```
P
```

Let's change Q2 to SECOND QUARTER:

```
S/Q2/SECOND QUARTER/
```

This is the Substitute command (S). It says to find the first occurrence of Q2, delete it, and insert SECOND QUARTER in its place. The pointer points to the first character following the insertion (in this case, the line terminator of that line).

Now let's return to screen mode by typing Control-V:

```
↑V
```

This is the View Page command. In command mode, it switches to screen mode. It must be entered at the beginning of a command line. The screen is rewritten, the bottom 20 lines again display the file, and the cursor is in the third line of the text area, pointing to the character that represents the line terminator following SECOND QUARTER.

July is missing. To add it in its proper place, position the cursor at the A in August, type Control-A (the Add Text command character), type "July", and type Control-A again. Now the line reads JulyAugust because we didn't end the line with the line terminator; the cursor is still positioned at the A, so type Control-C (the Add Character command character) and RETURN.

September appears twice. To delete the extra one, position the cursor at the S in the second September, type Control-Z, move the cursor down to the next line and type Control-Z again.

These commands should cover most of the editing you do. Chapter 3 describes the screen-mode commands in detail, and Chapter 4 describes the command-mode editing commands. Chapter 5 describes some of the advanced features of CREDIT, such as macros and executing the editor with the ISIS-II submit-file feature.

To end an editing session, you must switch from screen mode to command mode. Press the HOME key; the cursor moves to the command area, and an asterisk is displayed to indicate that CREDIT is ready to accept commands.

To save this file, enter EX (the Exit command). To ignore the changes made in the editing session, enter EQ (the Quit command); CREDIT prompts you before actually returning to ISIS-II to reduce the possibility of inadvertant loss of work.

In this instance, you probably don't want to keep this file, so enter EQ. CREDIT prompts you to be sure you don't want the changes:

```
QUIT?
```

If you reply anything other than "Y" or "y", CREDIT ignores the Quit command and continues the editing session. Because the file didn't exist on disk before the editing session, it won't be on the disk after the Quit command is executed.

CHAPTER 2
STARTING AND ENDING AN EDITING SESSION

To start an editing session, enter the CREDIT command from ISIS-II. CREDIT loads the file and creates a backup copy or, if the specified file doesn't exist, creates a file with that name.

When you're through editing the file, you can either replace the old file with the new version or save the new version with a new name. If you make an error in editing that damages the data in the file, you can abort the editing session without affecting the file on disk.

By specifying a device instead of a disk file name, you can read the input file from a non-disk source (such as a paper-tape reader) and save the file on disk.

File Backup

When you edit an existing file, CREDIT creates a copy of the file on the same disk with the same filename and an extension of .BAK. Thus, the original version of a file is always available after an editing session.

If a backup file already exists when you enter the CREDIT command, the backup file is deleted and replaced by the current version.

To be safe, you shouldn't create a new file with the extension .BAK; you will never have a backup copy of the previous version. For the same reason, you shouldn't edit the backup (.BAK) copy of a file. If you make changes in the backup copy, they will not be included in the original copy of the file and will be lost if the original is accessed by the editor.

You should never write-protect a ".BAK" file (don't set the write attribute to 1).

Performance and File Size

The only size limitation imposed on a CREDIT file is from the storage device. There must be enough space available to hold the file, the backup file, and the temporary files that CREDIT uses.

However, if you keep your files to 20K bytes or less you will obtain the maximum performance from CREDIT. 20K bytes is the approximate size of the text buffer.

Starting an Editor Session (CREDIT Command)

The CREDIT command starts an editing session. You must specify a valid ISIS-II file or device name with the command. If the file exists, CREDIT loads it into memory for editing. If the file does not exist, CREDIT creates a new file with the specified name.

As an option you can specify that the modified file be stored under a different name. You can also specify that a command file be loaded and executed when the editor is invoked. A command file is an ISIS-II file that

contains CREDIT commands; it is described in detail in "Chapter 5: Advanced Editing Techniques". The normal uses for a command file are to define a set of CREDIT macros and to reconfigure CREDIT for non-Intel terminals.

The format of the CREDIT command is:

CREDIT *filename1* [**TO** *filename2*] [**MACRO** | (*commandfile*)] | **NOMACRO**]

where:

CREDIT is the command name.

filename1 specifies a valid ISIS-II file or device. The format of a *filename* is:

[:*device*:][*name*[.*extension*]]

:*device*: is any valid ISIS-II device code. If :*device*: is not a disk, *name* and *extension* are ignored. If :*device*: is not specified, it defaults to :F0: See the *ISIS-II User's Guide* for the valid device designators.

name is a 1- to 6-character name. *name* must be specified when :*device*: specifies a disk. It and *extension* are ignored in all other cases.

extension is an optional 1- to 3-character extension. It must be preceded by a period (.).

TO *filename2* specifies that the updated version of the file is to be stored with the name specified by *filename2*; the old version (specified by *filename1*) remains on the disk and is not changed. If TO *filename2* is omitted, the old copy is renamed with the extension .BAK and the updated copy is stored under the name *filename1*.

MACRO [(*commandfile*)] is an optional parameter specifying that a command file be loaded and executed. *commandfile* specifies which command file to load. *commandfile* is a valid ISIS-II file name enclosed in parentheses. If a file name is not included with MACRO, the default command file, CREDIT.MAC, is loaded if it exists. The editor looks for CREDIT.MAC on the same disk on which the editor itself resides. If CREDIT.MAC is not found and MACRO is explicitly specified, an error is flagged. MACRO is the default condition, i.e., if neither MACRO nor NOMACRO is specified, CREDIT.MAC is loaded if it exists. If CREDIT.MAC does not exist and MACRO is implied by default, no command file is loaded and no error is flagged.

NOMACRO specifies that no command file be loaded. NOMACRO is coded when CREDIT.MAC exists, but you do not want to load it.

At the beginning of the editing session, CREDIT displays a sign-on message:

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

If the file named in the command is an existing file, the sign-on message is followed by:

```
OLD FILE SIZE = mm nnnn FREE DISK BLOCKS
```

mm is the size of the file in blocks; nnnn is the number of unused blocks on the disk.

If it's a new file, the sign-on message is followed by:

```
NEW FILE  nnnn FREE DISK BLOCKS
```

nnnn is the number of unused blocks on the disk.

An error in the CREDIT command sequence (or in loading the command file) results in an error message and returns you to ISIS-II. The error messages are listed and explained in Appendix D.

CREDIT Command Examples

To edit a file named MOD2.PLM on the system disk and not load a command file:

```
CREDIT MOD2.PLM NOMACRO
```

To edit the file in the previous example and load a command file named PLM.MAC with it:

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

To edit the file in the previous example, store the updated version as MOD3.PLM, and load the default command file (CREDIT.MAC):

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

or

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

To read a file from the paper tape reader, store it on disk with the name MOD3.SRC and load the default command file, CREDIT.MAC:

```
CREDIT :HR: TO MOD3.SRC
```

Ending an Editing Session

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.
- Ignore any changes and leave the old file unchanged.

Exit Command (EX)

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

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 you don't specify *filename*, the old version is replaced by the updated version.

NOTE

If you type EXIT, CREDIT stores the updated version on drive 0 as file IT (:F0:IT). You still have the original file with the old *filename*.

Exit Command Examples

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

```
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:

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

Quit Command (EQ)

The Quit command ends the editing session without updating any file on disk. To reduce the possibility of inadvertant loss of changes, CREDIT prompts you before returning to ISIS-II:

```
QUIT?
```

If you reply anything other than Y or y, CREDIT continues the editing session.

The format of the Quit command is:

EQ

EQ is the Quit command name.

CHAPTER 3
SCREEN EDITING

Screen editing is making direct changes to the data displayed on the screen. You move the cursor to a point on the screen and make a change, insertion, or deletion. The screen editing functions cover most of the work you routinely do when editing a file.

CREDIT begins an editing session in screen mode. To shift from screen mode to command mode, press the HOME key. To return to screen mode from command mode, press Control-V (↑V), the View command.

Warning Beep

The electronic beep sounds in two circumstances:

- When the screen displays a character 10 spaces from the right margin, just as a typewriter margin bell warns of the right margin (Intellec Series II only)
- When you try to enter a command that CREDIT can't accept (for example, to change text when the cursor isn't pointing to text)

Screen Editing Functions

Four types of editing functions are available in screen mode:

- Replacement, which replaces text on a character-by-character basis
- Insertion, which adds new text to the file
- Deletion, which removes text from the file
- Display, which controls what text is displayed

The Control-B hexadecimal entry function is not available in any of the screen mode functions. To enter a character not on the keyboard, you must use the Insert command in command mode.

Replacement

If you locate the cursor on any displayed character and press any printing character key on the keyboard, the character pointed to is replaced with the character typed. The one exception is when the cursor is located at a line terminator character. The line terminator is never replaced; it is moved one position to the right and the typed character is inserted ahead of it. To replace the line terminator, you must enter the new text in front of the terminator and then delete the terminator with the Delete Character (↑D) command.

Change occurs on a character-by-character basis; you cannot replace one character with two or two characters with one. You must use the insert or delete functions to do anything other than a one-for-one replacement.

You can replace an existing character with a character that has special meaning to CREDIT by preceding the special character with the literalizing character: the reverse slash (\). The characters that have special meaning to CREDIT are the control characters, including but not limited to the commands, and the first character of

the cursor control sequences currently defined for the terminal in use. For example, to change a character to Control-A (↑A), enter \↑A. The reverse slash is displayed, then the cursor backs up positioned for insertion of the character. If the character is a non-printing character, an up arrow (↑) is displayed.

To enter a reverse slash itself, you must precede it with a reverse slash.

Insertion

There are two insertion 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 it did before the insertion.

Add Text (↑A)

The Add Text command inserts any number of characters.

The format of the Add Text command is:

↑A*text*↑A

where:

↑A is the Add Text command character.

text can be any number of characters, including new lines. Control or non-printing characters must be preceded by the literalizing character (\).

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

When you enter the first ↑A, the text area clears from the point of insertion to the end of the screen. When you enter the final ↑A, the screen fills again.

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

Add Character (↑C)

The Add Character command inserts a single character.

The format of the Add Character command is:

↑C*x*

where:

↑C is the Add Character command character.

x is any character. Control and non-printing characters must be preceded by the literalizing character (\).

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

Deletion

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

Delete Text (↑Z)

The Delete Text command (↑Z) deletes any number of characters.

The format of the Delete Text command is:

↑Z <*move cursor*> ↑Z

where:

↑Z is the Delete Text command Character.

<*move cursor*> represents movement of the cursor with the cursor control keys.

To use the command, move the cursor to the first character to be deleted and press ↑Z. CREDIT replaces this character with a commercial at sign (@). Move the cursor to the position *beyond* the last character to be deleted; if that character is a line terminator, move the cursor to the first character of the next line. As the cursor moves over a character, a commercial at sign is displayed. The original character returns when the cursor moves on. If you move the cursor over the first commercial sign, the original character is restored. When the cursor is at the first position beyond the last character to be deleted, press ↑Z again. All data beginning with the first commercial at sign and ending with the character before the last commercial at sign is deleted. The deleted characters are removed from the screen and the characters following the deleted material are moved up.

The location of the second ↑Z can be ahead of the first. The second ↑Z may be to the right of the line marker. It may also be to the right of the end of file marker, but it cannot be on a line following the end of file marker. You can cancel a deletion after the first location is set by pressing the ESCAPE key.

No other screen mode command is valid between the first and second ↑Zs.

Delete Character (↑D)

The Delete Character command deletes a single character.

The format of the Delete Character command is:

↑D

↑D is the Delete Character command character.

To use the command, move the cursor to the character to be deleted and press ↑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.

Display

The display functions rewrite the screen. There are functions to display the following:

- Text relative to the current pointer position
- The screenful of text that follows the current display
- The screenful of text that precedes the current display

View Page (↑V)

The View Page command rewrites the screen with the cursor in the third line.

The format of the View Page command is:

↑V

↑V is the View Page command character.

↑V rewrites the screen with the line containing the cursor as the third line of the text area. The position of the cursor determines which lines will reappear when the screen is rewritten. The cursor stays on the same character after the screen rewrite.

If entered at the beginning of a command line, ↑V shifts CREDIT to screen mode. The screen clears and the text area fills, with the cursor in the third line. CREDIT is now ready to accept screen-editing commands. ↑V only works as the first character in a command line in command mode.

Next Page (↑N)

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

The format of the Next Page command is:

↑N

↑N is the Next Page command character.

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 cursor points to the beginning of the last text line).

Previous Page (↑P)

The Previous Page command displays the last 18 lines of text that precede the current display.

The format of the Previous Page command is:

↑P

↑P is the Previous Page command character.

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. The cursor points to the first character of the third line in the text area.

CHAPTER 4
COMMAND LINE EDITING

Command line editing is the process of entering commands for execution in the command area of the screen. The commands do not use the cursor or the data displayed in the display area of the screen. (The command area can expand into the display area causing the line of dashes and the display area to be erased.) The commands provide functions that are not available in screen mode, and can duplicate the screen functions with the exception of the display functions. The results of these commands are not reflected in the text shown in the display window. As you enter editing commands they fill the screen and replace the text area.

Classes of Editing Commands

The editing commands fall into the following categories:

- Pointer and tag commands, which move the pointer and establish auxiliary position markers called tags
- Text commands, which affect the text in the file by printing, inserting, deleting or replacing text
- Search Commands, which locate target text strings and can substitute new strings

How to Enter Commands

You can enter editing commands from the keyboard whenever the asterisk (*) prompt is displayed. Commands must end with a carriage return, and they will not execute unless one is entered. You can enter more than one editing command on a line by separating them with semicolons (;):

*command;command;command <CR>

An ampersand (&) immediately preceding the carriage return will continue the command line after the carriage return, regardless of the context. When the ampersand precedes the carriage return, CREDIT begins the next line with two asterisks (**):

*command;command;command;command; &<CR>

**command;command <CR>

You can correct the current line by pressing RUBOUT for each incorrect character, and reentering the correct characters.

Command Syntax

The general format of an editing command is:

* *command name* [*argument*]

where:

* is the editing command prompt displayed by CREDIT (don't type it). *command name* specifies what command is to be executed. The *command name* is either one or two characters. In the individual command format statements, the *command name* is shown in capital letters.

argument can be one, two, or three items required by the command. The brackets [ ] mean the *argument* is optional. For example:

```
L3
```

means move the pointer forward three lines.

```
XC T1, T2
```

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

You can also use text strings as parameters with some commands. In this case, you must set them off with delimiters so CREDIT can tell the strings from the commands. When specifying a string, CREDIT interprets the first nonblank character encountered after the command as a delimiter. You can use any ASCII character as a delimiter, except a space, carriage return, line feed, backslash, or escape.

Help Command

The Help command displays a syntax summary of the CREDIT commands. The command supplies three different screen displays. The first lists the screen mode commands, the second lists the command mode commands, and the third lists the advanced editing commands. The text for the Help command displays is kept in the file CREDIT.HLP, which must reside on the same disk that holds CREDIT.

At the end of each screen, you are given the option of displaying or skipping the next screen with the following prompts:

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

after the first screen or

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

after the second screen.

Figure 4-1 shows a Help command display.

The format of the Help command is:

H

where:

H is the command name. No parameters are allowed with the command.

Help Example. To display the first screen of the Help command:

```
H
```

Pointers and Tags

CREDIT keeps track of its position in your file with a pointer. In screen mode the cursor, when pointing to a character, represents the position of the pointer. This pointer always points to some character or to a special end-of-file marker that follows the last character of the file. Additionally, you can set, change, and delete up to ten auxiliary markers in the file. These auxiliary markers are called tags. There

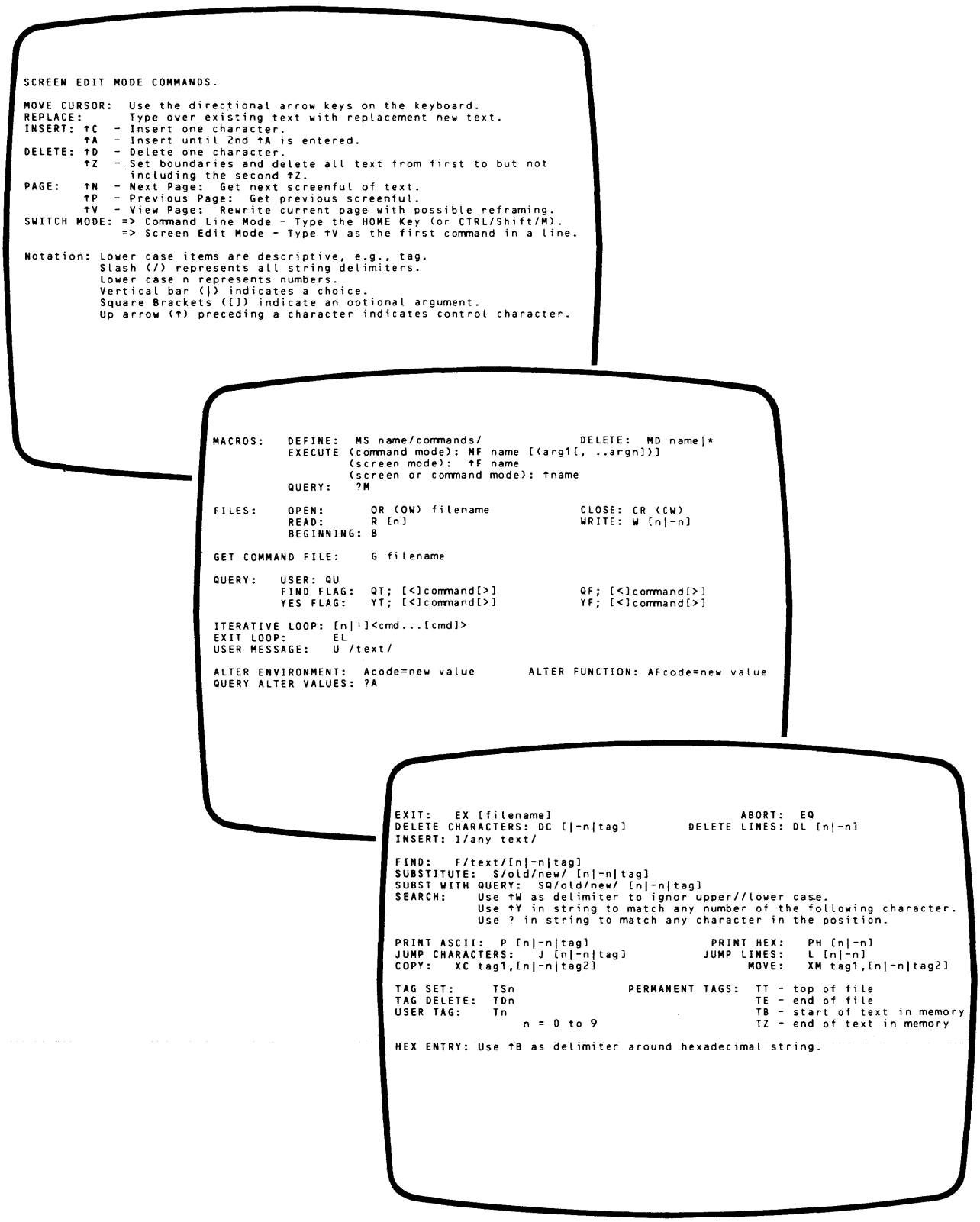


Figure 4-1. Help Command Display

902-3

are four tags in each file that cannot be deleted or changed. These permanent tags mark the beginning and end of the file and the beginning and end of that portion of the file currently in system memory. (A large file may not fit into the available memory.)

Pointer Commands

The pointer commands move the pointer any number of characters or lines forward or backward in the file, or to tags.

Line (L)

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

The format of the Line command is:

L [*number*]

where:

L is the command name.

number is a parameter that specifies the number of lines the pointer should move. If *number* is omitted, the pointer moves to the beginning of the next line. If *number* is positive, the pointer moves toward the end of the file. If *number* is negative, the pointer moves toward the beginning of the file. If *number* is zero, the pointer moves to the first character of the current line.

Line Examples. To move the pointer five lines forward:

```
L5
```

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

```
L0
```

To move the pointer five lines backward:

```
L-5
```

Jump (J)

The Jump command moves the pointer a specified number of characters forward or backward in the file, or to a specified tag location.

The format of the Jump command is:

J[*number*|*tag*]

where:

J is the command name.

number is an parameter that specifies the number of characters the pointer moves. If *number* is positive the pointer moves toward the end of the file. If *number* is negative the pointer moves toward the beginning of the file.

tag is an optional parameter that specifies any defined tag. If *number* is omitted and tag is not entered, CREDIT assumes a *number* of 1; i.e., the pointer moves one character forward.

Jump Examples. To move the pointer 18 characters forward:

```
J18
```

To move the pointer to the beginning of the file (the permanent tag TT):

```
JTT
```

To move the pointer to tag T9:

```
JT9
```

Tag Commands

The Tag commands create or delete a marker and associate it with the character at the current pointer location. This marker can be used with many command mode commands. You can define up to ten tags; each tag has a number from zero through nine such as T0, T5, etc. Defining a tag with a number that is already in use results in the deletion of the existing tag and 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.

You cannot change or delete the four permanent tags listed below; they are always present in your edited file.

- TT — Beginning of edit file
- TE — End of edit file
- TB — Beginning of text currently in system memory
- TZ — End of text currently in system memory

A tag is assigned to the character at the pointer at the time you define it. If the character is moved (with the Move command) the tag goes with it.

Tag Set (TS)

The Tag Set command defines a tag pointing to the character at the current pointer location. If a tag with the same name already exists, it is deleted; no message is displayed to indicate the deletion.

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 through 9.

Tag Set Examples. To set tag T3 to the current pointer location:

```
TS3
```

Tag Delete (TD)

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

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. *n* is any digit from 0 through 9.

Tag Delete Examples. To delete tag T3:

```
TD3
```

Text Commands

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

The text modifying commands covered in this section are:

- Print (PH)
- Insert (I)
- Delete (DL, DC)
- Move (XM)
- Copy (XC)

Print (P)

The Print command displays one or more lines of text.

The format of the Print command is:

P[*n*|*tag*]

where:

P is the command name.

n is the number of lines to be printed. A positive *n* specifies that *n* lines, beginning with the line containing the pointer and toward the end of the file are to be printed. A negative *n* causes *n* lines preceding the pointer to be printed.

Print Examples. To print three lines, beginning with the current line:

```
P3
```

To print the current line from the beginning to the location of the pointer:

```
P0
```

Print Hexadecimal (PH)

The Print Hexadecimal command displays one or more bytes in hexadecimal codes. A listing of the hexadecimal codes for the ASCII character set is in Appendix E.

The format of the Print Hexadecimal command is:

PH [*n*|*tag*]

where:

PH is the command name.

n is the number of bytes to be printed. A positive *n* specifies that *n* bytes, beginning with the byte at which the pointer is located and toward the end of the file are to be printed. A negative *n* causes *n* bytes preceding the pointer location to be printed.

The Print command displays data as it is entered:

```
A line of hex.
```

The Print Hexadecimal command displays sixteen hexadecimal bytes to a display line with single spaces between the codes:

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

With the Print command, codes such as carriage returns and line feeds perform their function on the screen, that is, they cause a carriage return and a line feed. With the Print Hexadecimal command, the codes themselves are displayed, i.e., 0D for carriage return and 0A for line feed.

Print Hexadecimal Examples. To print three bytes, beginning with the current location of the pointer:

```
PH 3
```

To print the three bytes that precede the pointer:

```
PH -3
```

To print from the pointer location to (but not including) the byte indicated by tag 4:

```
PH T4
```

Insert (I)

The Insert command inserts text anywhere in a file. The text is inserted immediately preceding (to the left of) the pointer. The pointer is unchanged.

The format of the Insert command is:

I/*text*/

where:

I is the command name.

text is one or more characters to be entered into the file. *text* can be ASCII characters entered from the keyboard or hexadecimal codes surrounded by Control-Bs (↑B). The first nonblank character after I is taken to be the delimiter. If you forget the intended delimiter, CREDIT will interpret text as part of the string until it finds a match for whatever character appears first in *text*. It's possible for CREDIT to interpret this unexpected text as commands, or to return a syntax error. If you forget the closing delimiter, the same thing can happen; a carriage return won't terminate the insert string, and CREDIT continues searching for the next occurrence of the first character in *text*. If you insert more than 2000 characters, the command buffer fills and CREDIT terminates the command automatically.

Every character entered between the delimiters is stored in the file as text. This includes carriage returns, line feeds, control characters, cursor control codes, etc.

Insert Examples. To insert the line LOOP: MOV A,M ;SAVE THE PARM into a file, you must first position the pointer at the character *following* the point of insertion (The pointer can positioned with any of the pointer manipulation commands or with the pointer controls in screen edit mode). Then enter the Insert command:

```
I/LOOP: MOV A,M ;SAVE THE PARM<CR>
/
```

You should enter a carriage return-line feed at the end of the command. If it was not entered, the next line would immediately follow PARM.

Delete (DL, DC)

The Delete commands remove text from the edited file. There are two Delete commands, one to delete lines of text and one to delete individual characters. With both commands the deletion begins at the pointer and removes the specified number of lines or characters. A positive number deletes text from the pointer forward (toward the end of the file). A negative number deletes text from the pointer backward (toward the beginning of the file).

DL — Delete Line

The Delete Line command removes one or more lines of text from a file.

The format of the Delete Line command is:

DL[*number*]

where:

DL is the command name.

number is a parameter that specifies the number of lines to delete. If *number* is omitted, the entire line containing the pointer is deleted, regardless of where the pointer is in the line. If you specify *number*, deletion begins at the pointer (positive number) or immediately preceding the pointer (negative number). If *number* is zero, the current line up to but not including the pointer is deleted.

Delete Line Examples. To delete three lines, beginning at the pointer:

```
DL3
```

To delete the three lines preceding the pointer:

```
DL-3
```

To delete the text that preceeds the pointer location in the current line:

```
DL0
```

DC — Delete Character

The Delete Character command deletes characters from the file.

The format of the Delete Character command is:

DC [*number*|*tag*]

where:

DC is the command name.

number is a parameter that specifies the number of characters to delete. Deletion begins at the pointer. If *number* is positive, the number of specified characters are deleted, beginning at the pointer. If the number parameter is negative, the number of characters preceding the pointer are deleted but the character at the pointer remains. If *number* or *tag* is omitted, the character at the pointer is deleted. If *number* is specified, *tag* cannot be.

tag is an optional parameter that specifies the end of a block of characters to delete. You can use any permanent or user-defined tag. If you specify *tag*, you cannot specify *number*.

After a deletion the pointer appears immediately following the last character deleted.

Delete Character Command Examples. To delete 10 characters beginning with the character at the pointer:

```
DC10
```

To delete the 15 characters that precede the pointer:

```
DC-15
```

Move (XM)

The Move command deletes a block of data from its current location and inserts it immediately to the left of the current pointer location. The pointer must not be located within the data being moved. Tags associated with the data are moved with the data. The pointer is not affected by the move; that is, it points to the same character after the move as it did before the move.

The format of the Move command is:

XM *tag*, { *tag*|*number*|*-number* }

where:

XM is the command name.

tag specifies the beginning of the block of text to be moved. A second *tag*, or *number* of lines, specifies the end of the text block to be moved. The second *tag* location is not included in the move, that is, it indicates the first position beyond the block to be moved. When all the text involved with the move resides in memory, the two *tag* form of the command is faster than the *tag-number* form. The first tag must precede the second in the file.

If you specify *number*, you move the next n lines ahead of the tag with a positive n, or the previous n lines behind the tag with a negative n. The destination of the text block is the current pointer position when XM executes.

After the move, the pointer is located at the first character following the moved text, and the text no longer appears at the old file location.

Move Examples. To move the 11 lines of text from tag T8 to the current location of the pointer:

```
XM T8,11
```

To move the text bounded by tags T3 and T6 to the current location of the pointer:

```
XM T3,T6
```

Copy (XC)

The Copy command duplicates a block of data. The data being copied is not changed. The block is inserted immediately to the left of the pointer. The pointer must not be located within the data being copied. Tags associated with the data being copied are not copied with the data. The pointer is not affected by the copy; that is, it points to the same character after the copy as it did before the copy.

The format of the Copy command is:

XC *tag*, { *tag*|*number*|*-number* }

where:

XC is the command name.

tag specifies the beginning of the block of text to be copied. A second *tag*, or *number* of lines, specifies the end of the text block to be copied. The second *tag* location is not included in the copy, that is, it indicates the first position beyond the block to be copied. When all the text involved with the copy resides in memory, the two *tag* form of the command is faster than the *tag-number* form. The first tag must precede the second in the file.

If you specify *number*, you move the next n lines ahead of the tag with a positive n, or the previous n lines behind the tag with a negative n. The destination of the text block is the current pointer position when XC executes.

After the move, the pointer is located at the first character following the copied text.

Copy Examples. To copy the 20 lines of text starting at tag 7:

```
XC T7,20
```

To copy the text block bounded by tags T4 and T9:

```
XC T4,T9
```

Search Commands

There are two search commands available in CREDIT. The Find Command searches text for a character string. If it finds the string, the pointer moves to the first character following the string. The Substitute Command searches a specified area for a character string like the Find Command; if it finds the string, it substitutes a new string for the old one, and the pointer is on the first character following the string. If the string isn't found, the pointer stays at the starting search position.

Ranges

When you are searching for a character string, you can specify the area of text searched. This area is called a range, and is defined to extend from the current pointer a specified number of lines forward or backward, or to a tag.

Wildcard Characters

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

? means match any character. For example:

```
ABC?E
```

matches ABCDE, ABCPE, ABC8E, or ABC*E, or any string with A,B,C,and E and any ASCII character in the ? position.

↑Y matches any number of the character which follows it. For example:

```
B↑Y AD
```

matches BD, BAD, or BAAD.

↑W matches either upper case or lower case characters enclosed within. For example:

```
F↑Wmhz↑W
```

finds mhz, Mhz, mHz, MHz, or MHZ.

Find (F)

The Find command searches for a character string and positions the pointer to the character immediately following the string. It displays a NOT FOUND message if the string is not found. This message can be suppressed with the Alter command described in Chapter 5. When the Find command is used in iterative loops, the loop is exited if the string is not found. The NOT FOUND message is issued only if the search fails on the first pass through the loop.

The format of the Find command is:

F/*string*/ [*tag*|*number*]

where:

F is the command name.

string specifies the character string to be found. *string* must be enclosed in delimiters. *string* can be entered as ASCII characters or as hexadecimal codes enclosed by Control-Bs (↑B).

The range, specified by [*tag*|*number*], is optional. *tag* may be any defined tag. When you specify *tag*, CREDIT searches from the current pointer position to *tag*. If *tag* is not found CREDIT searches to the end of the file. If you specify *tag*, you can't specify *number*.

number specifies the number of lines to search. A positive value specifies a search starting at the cursor for *number* lines. A negative value specifies a search of *number* lines immeditely precedeing the cursor. If you enter zero for *number*, CREDIT searches from the beginning of the line up to the pointer. If you specify *number*, you can't specify *tag*.

Find Examples. To find the character string "LOOP4" between the pointer and the end of the file:

```
F/LOOP4/
```

To find the same character string between the pointer and tag T7:

```
F/LOOP4/T7
```

To find the same character string in the 23 lines following the pointer:

```
F/LOOP4/23
```

Substitute (S, SQ)

The Substitute command replaces a character string with another character string. It displays a NOT FOUND message if the character string does not exist. This message can be suppressed with the Alter command, described in Chapter 5. When the Substitute command is used in iterative loops (described in Chapter 5), the loop is exited if the string is not found. The NOT FOUND message is issued only if the search fails on the first pass through the loop.

There are two forms of the command: one makes the replacement when the string is found, and the other displays the line containing the string and prompts for a yes response before making the change.

The format of the Substitute command is:

[S|SQ]/*oldtext*/*newtext*/ [*number*|*tag*]

where:

S and SQ are the command names. If you specify S substitution is made if *oldtext* is found. If you specify SQ, CREDIT prompts for a response before the substitution. A Y or y response allows the substitution; any other response cancels it.

oldtext is the character string to be replaced if found. If it is not found a NOT FOUND message is displayed. *oldtext* can be any length except zero, and it must be surrounded by delimiter characters.

newtext is the replacement character string. *newtext* can be any length, including zero. When a null string is specified for *newtext*, *oldtext* is deleted. *newtext* must be surrounded by delimiter characters, even if it is a null string.

The range, specified by [*tag*|*number*], is optional. *number* specifies the number of lines to search for *oldtext*. If *number* is zero, the search is from the beginning of the line the pointer is in up to the pointer.

tag is any defined tag. When you specify *tag*, CREDIT searches from the current pointer position to *tag*. If *tag* is undefined or precedes the pointer, CREDIT searches to the end of the file.

If you press the ESC key while CREDIT is doing a substitution, searching stops but any substitution in process or completed remains. Wildcard characters can be used in the search string, as in the Find Command.

Substitute Command Examples. To change the first occurrence of the string "THEN:" to "NEXT:":

```
S/THEN:/NEXT:/
```

To make the previous change with the query option:

```
SQ/THEN:/NEXT:/
```

CREDIT responds by printing the line containing the search string and a question mark:

```
25 THEN: A ?
```

If you type Y or y, CREDIT makes the substitution; any other reply cancels substitution.

To make the previous change if the string occurs within the next 18 lines:

```
S/THEN:/NEXT:/18
```

To make the same change but only if the string occurs between the pointer and tag T6:

```
S/THEN:/NEXT:/T6
```

CHAPTER 5
ADVANCED EDITING TECHNIQUES

This chapter describes how to use editor macros, command iteration, conditional execution, command files, side files, the Alter commands, and the SUBMIT facility:

- A macro is a named series of CREDIT commands that are executed when you call the macro by name.
- Command iteration lets you repeat a command a given number of times, or until the end of the file.
- Conditional execution lets you leave an iterative loop when desired.
- Command files are files containing CREDIT commands loaded when CREDIT is invoked, or in command mode. Command files execute immediately.
- Side files are ISIS-II files other than the edited file opened for reading or writing.
- Alter commands allow you to change CREDIT characteristics or functions.
- The ISIS-II SUBMIT facility allows you to execute an entire editing session with a single command.

Macro Facilities

CREDIT macros are sequences of CREDIT commands that have been given a name. They are stored in memory, where they can be called by name, and executed. To define a macro and give it a single character name, use the Macro Set (MS) command. To execute it, call it with the Macro Function (MF or ↑F) command (or simply by name in special cases). To delete a macro, use the Macro Delete (MD) command. To display a list of all the currently defined macros, use the Query Macro (?M) command.

Macros are typically used for long command sequences that are executed often. Rather than enter a long series of CREDIT commands, you call the predefined macro with a single command. The macro facility speeds your work and reduces the typing errors usually associated with a long string of commands.

A use for a macro might involve going to the beginning of the file, searching for a string, and inserting a line of code following every occurrence of the found string. The macro definition naming the macro Q looks like:

```
MSQ$JTT;!<F@old code line@;L;I@new code line@>$
```

(The MS command and !<...> structure are described later in this chapter.)

You can pass parameters to macros. Macros that accept parameters can be executed only from command mode. The preceding macro example is more useful if is defined:

```
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 invoke this macro if it is defined as having the name "Q", type:

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

The parameters are entered with the MF command, enclosed in parentheses, and separated with a comma. (A comma in a parameter must be literalized or it will be assumed to be a separator.) The parameters replace the percent signs in the macro; thus the following command string is executed:

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

There must be a parameter for every percent sign in the macro definition; an ARGUMENT MISTMATCH error occurs if you pass too few or too many parameters.

Another common 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 would be:

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

Macros can execute in screen mode or command mode as long as the commands being called are for the current mode. For example, just as you cannot execute a screen mode command in command mode, you cannot execute a macro consisting of screen mode commands in command mode. However, you can do anything within a macro that you can do in normal operation. This means, among other things, you can change modes within macros. A HOME code encountered in a macro causes the editor to change from screen- to command mode. Likewise, a ↑V causes the editor to change from command to screen mode. (A ↑V must be the first character in the macro or it must be preceded by a line terminator (carriage return-line feed).)

You can execute macros from either mode. The Macro Function command is MF in command mode and ↑F in screen mode. Any macro can be initiated with both forms of the command. An UNRECOGNIZED COMMAND error occurs if a screen mode command is executed in command mode. If a command mode command is executed in screen mode, the macro text (the command string) replaces the existing data beginning at the pointer location. (This is exactly the same thing that would happen if you attempted to enter the individual commands in screen mode.)

If necessary, you can write macros that can be executed in either mode. Such macros must begin with the View Page command (↑V) which is the only command acceptable in both modes. Following the View Page command, your macro can execute screen mode commands or can issue a HOME code to go to command mode.

Be careful writing macros that change modes. Your macros should end in the same mode from which they were initiated. If the Macro function command is in the midst of a string of commands (a macro or entered at terminal) and the macro changes the mode, the command that follows Macro Function command will cause an UNRECOGNIZED COMMAND error. If you have a macro that changes modes and does not change back, do not call the macro in a string of commands.

Macros can call other macros. You should avoid the condition where macro A calls macro B which in turn calls macro A, it puts the system into an endless loop. If either of these macros generate data or commands, the file could be filled or the maximum command length exceeded, causing an error. If you find yourself in such a loop, press the Escape key (ESC) to cancel the macro.

The Macro Commands

Macro Set (MS)

The MS command defines a group of editor commands as a macro. If another macro by the same name exists, an error message prints and the new macro is not defined.

The macro name is a single character, not necessarily a printing character.

The format of the MS command is:

MS *name @text@*

where:

MS is the command name.

name is a single character you call the macro with. You can use any character, printing or non-printing, except for carriage return, line feed, escape, space, backslash, or asterisk. You can duplicate editor command characters, but a single specified character can call only one macro.

text is the sequence of editor commands executed when the macro is called. The sequence of commands must be enclosed in string delimiters.

When you define a macro, you can specify that a parameter will be passed by entering percent signs (%), one for each parameter. When the macro executes a parameter must be passed for each percent sign, in parentheses and separated by commas. Entering commas with no text within specifies a null parameter.An error occurs if an extra parameter is passed to a macro or if no parameter is passed when one is expected. Examples of parameter passing are shown in the MF command description. If you use a percent sign as text, not as a variable, it must be preceded by two reverse slashes.

Macro Set Examples

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

```
MSB@JTT;P10@
```

To define a macro named S that will make a substitution with text supplied in the macro invocation command and then print the modified line:

```
MSS@S/%/%/;P@
```

To define a macro named M that moves the pointer to the beginning of the file and then enters CRT mode:

```
MSM@JTT;
↑V@
```

Note that Control-V to enter CRT mode was preceded by a carriage return. The macro definition will fail unless the commands are entered exactly as they would be in a command line. Remember that you cannot enter a Control-V to change mode anywhere except at the beginning of a command line.

To define a screen mode macro named ↑Y (cursor right key) that moves cursor to the extreme right side of the screen:

```
MS↑Y@↑T↑T↑T↑T↑T...↑T@
```

Note that the macro definition must contain 79 cursor right character sequences. Macros of this type for all four cursor directions provide a fast method of moving the cursor around the display area. This macro is much faster than pressing the cursor key with the repeat (RPT) key.

Invoking Macros (MF or ↑F)

The Macro Function commands execute a macro. If the macro does not exist, a DOESN'T EXIST error message is displayed.

There are two explicit and two implicit forms of the command. The explicit forms are the MF and ↑F command names used in command and screen mode, respectively. The implicit forms apply only to a macro with a control character name, such as ↑Y. The control character name cannot have any special meaning to CREDIT. For example, you cannot use the screen mode command names as macro names. In command mode you can execute the ↑Y macro by entering: ↑Y immediately following the CREDIT prompt (*). This implicit invocation does not work if the cursor is anywhere except at the prompt. In screen mode you can execute the ↑Y macro at anytime by entering ↑Y.

You can pass parameters to macros invoked explicitly from command mode with the MF command. No other forms of macro invocation can call macros that take parameters. Parameters must be passed in the same sequence they are defined in the macro. They are passed following the macro name, enclosed in parentheses and separated by commas.

The formats of the Macro Function commands are:

MF*name*[(*parm*[,...,*parm*])]

↑F*name*

name

where:

MF is the command name for invoking macros in command mode. **↑F** is the screen mode command for invoking macros.

name is the single character macro name. In the implicit form of the command, *name* must be a control character, other than those that have meaning to CREDIT.

parm is a parameter passed to the macro. All parameters must be enclosed in parentheses and separated by commas. Null parameters can be passed by entering the surrouning commas but no text. To omit the last parameter, end the list with a comma. To enter a comma in a parameter, precede it with the literalizing character (\).

Macro Function Examples

To execute the B macro, defined earlier, which requires no parameters:

```
MFB
```

To execute the S macro which requires two parameters:

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

To execute the S macro as in the preceding example, but to set the second parameter passed to a null value, thereby deleting the specified string:

```
MFS(1978,)
```

To execute the ↑Y macro:

```
↑Y
```

Screen Macro Function Examples

To execute the T macro which enters command mode, move the cursor to the top of the file, and return to screen mode:

```
↑FT
```

To execute the ↑K macro:

```
↑K
```

Macro Delete (MD)

The MD command deletes a macro specified by name, or all macros if you specify an asterisk.

The format of the MD command is:

MD { *name*|* }

where:

MD is the command name.

name specifies the name of a single macro to be deleted. * deletes all currently defined macros. If both * and *name* are omitted, an error message is displayed and no macros are deleted.

Macro Delete Examples

To delete a macro named V:

```
MDV
```

To delete all macros:

```
MD*
```

Query Macro (?M)

The ?M command displays the name and text of all macros. The display is in the following format:

```
B=jtt;p10

S=S/%/%/;P0

M=jtt;
↑V
```

The format of the ?M command is:

?M

Command Iteration

CREDIT provides a number of useful functions that let you create flexible and powerful commands that repeat themselves for a specified number of times or until a given condition is satisfied, and then halt.

By themselves, angle brackets specify that a group of commands execute together. You can convert any command string into a repeating sequence of commands, called an *iterative loop*, by enclosing the string in angle brackets and preceding the left bracket with the number of times you wish to execute the commands:

```
10<S@data@text@>
```

In this example, the string "text" is substituted for "data" 10 times. If there are no occurrences of "text", the command string is terminated with a NOT FOUND message. If the iteration factor is omitted, the command is executed once. If an exclamation point (!) is specified as the iteration factor, the command string is executed forever or, as in the case of the example, until the end of file is reached. Angle brackets can be nested up to five deep.

Conditional Execution Commands

CREDIT provides several commands that let you establish conditional command flows in iterative command loops and in macros. The commands are:

- Query User (QU), which lets you set a flag.
- Query True (QT) and Query False (QF), which test the flag set by the Query User command.
- Yes Flag True (YT) and Yes Flag False (YF), which test the condition of the Yes Flag, which is set by a successful (string found) FIND or Substitute command and reset by an unsuccessful one.
- Exit Loop (EL), which causes an unconditional exit from an iterative loop.

Another command, User Message (U), is useful for displaying prompts for the Query User command or supplying the user of a macro with information.

Query User (QU) Command

The Query User command provides the means to set or reset a flag that can be subsequently tested with the Query True or False commands. This command prompts for input with a question mark (?). A "Y" or "y" response sets the flag to the true condition; any other response resets it to the false condition.

See the description of the Query True and Query False commands for some applications where decisions are made on the basis of the response to the Query User command. Alone, this command causes a pause allowing you to read the display before proceeding (by pressing any key).

See the User Message command described later in this section for a means of creating more meaningful prompts for this command.

The format of the Query User command is:

QU

where:

QU is the command name.

Query True (QT) and Query False (QF) Commands

The Query True and False commands test the Query Flag that was set with the Query User command. The command determines whether the next command or bracketed group of commands are to be executed. If the flag is set to true, the Query True command causes the next command or commands to be executed; if the flag is set to false, the Query False command causes the next command or commands to be executed. The following table shows the possible conditions:

Flag Status	Next Command Executed? QT	QF
true	yes	no
false	no	yes

In either case, the command following those controlled by the Query True or False is executed.

The formats of the Query True and False commands are:

QT;[<]*command*[>]
QF;[<]*command*[>]

where:

QT and QF are the command names.

command is any CREDIT command mode command. Multiple commands can be put under control of the Query command by enclosing them in angle brackets (< >). An iteration factor can also be specified.

Query Command Examples

The following command string gives you the option of deleting all lines that contain the specified string. After the string is found, the line is displayed, and the user is prompted for a decision whether to delete the line.

```
!<F@XXX@;P;QU;QT;DL>
```

The line is deleted only if the response to the prompt is a "Y" or "y".

Yes Flag True (YT) and Yes Flag False (YF) Commands

The search commands (Find and Substitute) set a flag, called the Yes Flag, to true if the search results in a match. The Yes Flag commands test the status of this flag to determine whether the command or commands enclosed in angle brackets are to be executed. The following table shows the effect of the flag on command execution.

Flag Status	Next Command Executed? YT	YF
true	yes	no
false	no	yes

In either case, the command following those controlled by the Yes Flag True or False is executed.

The format of the Yes Flag True and False commands are:

YT;[<]*command*[>]
YF;[<]*command*[>]

where:

YT and **YF** are the command names.

command is any CREDIT command. A group of commands can be put under control of the Yes Flag commands by enclosing them in angle brackets (< >).

Yes Flag Command Examples

To search for the string "Version 4" and exit the file if the string is not found:

```
<F@Version 4@>;YF;EX
```

To search for the string "QQ357" and, if it is found, jump to the beginning of the file and print the first 10 lines:

```
<F@QQ357@>;YT;<JTT;P10>
```

In the preceding example, note that the Yes Flag command has control over a group of commands enclosed in angle brackets, rather than over a single command.

Exit Loop (EL) Command

The Exit Loop command causes an unconditional exit from an iterative command loop. The command is used with the Query or the Yes Flag commands. The Exit Loop command is meaningless unless it is within an iterative loop; outside of a loop construct, it is ignored.

The format of the Exit Loop command is:

EL

where:

EL is the command name.

Exit Loop Examples

The following command string searches the whole file (specified by !) for the string "xyz" and exits the loop if the line also contains the string "abc":

```
!<F/xyz/;<F/abc/1>;YT;EL>
```

Note that inner Find command must be enclosed in angle brackets because the Find command exits a loop if the search string is not found. If the inner Find was not enclosed in brackets, failure to find the string "abc" would cause an exit from the whole iterative loop.

User Message Command

The User Message command writes a message, contained in the command, to the display. The command is useful for giving directions or information to the user and for creating more meaningful prompts for the Query User command. The message must be enclosed in delimiters. The first non-blank character after the command name is assumed to be the delimiter.

The format of the User Message command is:

U/*text*/

where:

U is the command name.

text is a string of one or more characters that are to be displayed.

The first non-blank character after the U is assumed to be the delimiter. Every character up to the next occurrence of the delimiter is taken as the message.

User Message Examples

To search for the string "crt" and display the message "FOUND:" plus the line containing the string when the string is found:

```
<<F/crt/>;YT;<U/FOUND: /;P>>
```

Note that the Yes Flag command controls the execution of the User Message and the Print command because they are enclosed in angle brackets.

Command Files

A command file is an ISIS-II file that contains CREDIT commands. When the file is loaded by CREDIT, the commands are executed. A major use of the command file is to define macros that will be needed throughout an editing session.

A command file can be created with the editor. It is nothing more than a series of CREDIT commands entered as data. For example, the following command file defines the macros F, B, and G:

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

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

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

There are two means of loading a command file during an editing session:

- Naming the file with the MACRO option in the CREDIT command (or loading the default command file, CREDIT.MAC)
- Issuing a Get command specifying the name of a command file.

The use of the MACRO option is described in "Chapter 2: Starting and Ending an Editing Session." The Get command is described in the following section.

Get (G) Command

The Get command loads a command file and executes the CREDIT commands in it. The Get command can be issued anytime CREDIT is in command mode. A command file can contain Get commands that load other files. 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 is kept.

The format of the Get command is:

G *filename*

where:

G is the command name.

filename is the name of an ISIS-II file that contains CREDIT commands.

Get Examples

To load a command file named OPSYS.MAC that resides on the :F2: disk:

```
G :F2:OPSYS.MAC
```

Side Files

A side file is any ISIS-II file to which CREDIT has either read or write access. Before a file can be used, it must be opened. Opening a file is the process of establishing a logical connection to it. When you're finished with a file, it can be closed. Closing a file breaks the logical connection and renders the file inaccessible to the editor until it is again opened. Opening and closing files is similar to opening and closing a file drawer. While the drawer is open you can get information out or put it in. When the drawer is closed, you can no longer get information in or out.

With CREDIT, you can open a side file for read access or for write access but not both. That is, when you open a file for reading, you can read data from it but you cannot write to it or change it in any way. On the other hand, you can only write to a file open for writing. You cannot read back what you have written to it.

When you write to a file that is open for writing, all data written to it is inserted at the end.

The CREDIT side file commands cannot add data to an existing file. If an existing file is opened for write access, it is deleted and recreated, thus destroying the data in it. Opening a file for read access does not destroy the data in it. When reading from a file open for reading, records are read sequentially from the beginning. When a record is read, the side file pointer moves to the next record. There is a command to move the pointer to the beginning of the file.

CREDIT has seven commands supporting the use of side files:

- Open Read, which opens a file for reading
- Open Write, which opens a file for writing
- Read, which reads from the file open for reading
- Write, which writes to the file open for writing
- Begin File, which moves the side file pointer to the beginning of the file open for reading
- Close Read, which closes the file open for reading
- Close Write, which closes the file open for writing.

Open Read (OR)

The OR command opens a specified file so that it can be read from the editor. A file opened for reading cannot be written to, only read from. The file specified in the command cannot be opened for any purpose or an error will occur. The only exception to this is in the case of the console input file, :CI:, which is always open. When you open a file, either for reading or writing, the pointer is at the beginning.

The format of the OR command is:

OR *filename*

where:

OR is the command name.

filename is any valid ISIS-II filename that can be used for input. If the file cannot be used for input or is already open (except for :CI:), an error message is displayed.

Open Write (OW)

The OW command opens a specified file so that it can be written to by the editor. A file opened for writing cannot be read from, only written to. The file specified in the command cannot be opened for any other purpose or an error will occur. The only exception to this is in the case of the console output file, :CO:, which is always open. If the specified file already exists, it is deleted.

The format of the OW command is:

OW *filename*

where:

OW is the command name.

filename is any valid ISIS-II filename that can be used for output. If the file cannot be used for output or is already open (except for :CO:), an error message is displayed.

Begin File (B)

The B command moves the external file pointer to the beginning of the external read file. If no read file is open, the command has no effect.

The format of the B command is:

B

Read File (R)

The R command reads a specified number of lines from the currently open external read file and inserts them at the pointer in the edited file. The external read file pointer advances to the first character not read, so subsequent reads move through the file unless the file pointer is reset to the beginning of the file with the B command. The text in the external file is not destroyed.

The format of the R command is:

R *[n]*

where:

R is the command name.

n is an optional parameter that specifies how many lines to read. If *n* is omitted, one line is read. *n* cannot be a negative number. An *n* of zero makes the command a null command; i.e., nothing is read.

Write File (W)

The W command writes a specified number of lines from the location of the pointer in the edited file to the currently open external write file. There is no effect on the text or pointer in the file being edited.

The format of the W command is:

W *[n]*

where:

W is the command name.

n is an optional parameter that specifies the number of lines to be written to the external write file. If *n* is omitted, the entire current line is written. If *n* is a positive number, the specified number of lines, starting at the current pointer, are written to the external file. If *n* is negative, the specified number of lines preceding the current pointer and ending at the pointer are written to the external file. If *n* is zero, the characters from the beginning of the current line to the current pointer are written to the external file.

Close File (CR, CW)

The Close File commands, CR and CW, close the current external read or write file. If there is no open read or write file, the command is ignored.

The format of the C commands are:

CR

and

CW

Alter Commands

The Alter commands modify the editing environment and the communication link between CREDIT and the terminal attached to the system. There are two Alter commands: Alter Environment and Alter Function. The former allows you to modify the way data is presented on the screen. The latter allows you to modify the codes generated by the editor and expected by the editor. The Alter Function command makes it possible to configure the editor to run with non-Intel terminals. Appendix A lists tested configurations for several popular terminals.

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 Find and Substitute command error messages
- Wildcard characters
- Cursor movement codes
- Erase line and screen codes
- Screen mode command codes

The normal method of using the Alter commands is to put them into a command file and call the file with the MACRO option of the CREDIT command (the file CREDIT.MAC is automatically executed). This technique executes the commands before transferring control to the keyboard. In some cases of configuring CREDIT for a non-Intel terminal, it is possible for the editor to get hung in a situation from

which you cannot exit if control is transferred to the keyboard before the Alter commands are executed. This can occur if the terminal does not generate the codes that CREDIT is expecting.

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 various *codes* and their possible values are described in the following description of *value*.

value depends on which *code* is specified. Every character, including spaces, entered after the = sign is significant. The possible *values* for *code* are:

- L specifies that the line terminator (CRLF) is to be displayed as the specified single character, instead of as the up-arrow.
- V specifies that the number of lines on the screen be changed from 25 to the specified number. The possible values are 22, 23, 24, or 25. The smaller screens have a smaller command area; the text area remains the same at 20 lines.
- S specifies whether the Find and Substitute error message, NOT FOUND, is to be suppressed. The possible values are T, which means that the error message is suppressed, or F, which means it is not suppressed. The default setting is F, the error message is generated.
- T specifies the TAB setting. The value can be any integer in the range 0 through 79. The default setting is 8.
- B specifies which keyboard character is to be used to cause a BREAK. The BREAK character causes the command being executed to abort and *BREAK* to be displayed. The value must be entered as a single hexadecimal byte. The default is hexadecimal 1B (the ESCAPE character). This function is useful when using a terminal that requires the ESCAPE character for control sequences.
- C specifies which character is to represent all non-printing characters in screen mode. The value for C must be a printing character (including blank). The default is the up-arrow (↑).

Alter Function Command

The format of the Alter function command is:

AF *code* = *value*

where:

AF specifies that a function be changed. *code* is a two-character string specifying what function is to be changed. The possible values for *code* are:

- WA specifies the wildcard "match any number of" character. The default is Control-Y.
- WC specifies the wildcard "match either case" character. The default is Control-W.

- WJ specifies the wildcard "match any" character. The default is the question mark (?).
- BK specifies the code which blanks out a single screen location. The default is a space.

The following functions take a zero to four byte value in hexadecimal. The value is a control sequence, a series of values that have special meaning to CREDIT. The ↑B used for hexadecimal entry in text handling commands is not allowed in the Alter commands. If a function is unavailable on your terminal, specify a null value (zero bytes). Sequences of less than four bytes are padded with null characters (00H) by the editor. You cannot explicitly specify null characters in a control sequence.

The following five codes that begin with C specify the values generated by the terminal when the cursor control keys are pressed:

- CD specifies the cursor down code.
- CH specifies the cursor home code.
- CL specifies the cursor left code.
- CR specifies the cursor right code.
- CU specifies the cursor up code.

The following six codes that begin with M specify the values generated by CREDIT to move the cursor on the CRT display:

- MB specifies the move cursor to beginning of line code.
- MD specifies the move cursor down code.
- MH specifies the move cursor to home position code.
- ML specifies the move cursor to left code.
- MR specifies the move cursor to right code.
- MU specifies the move cursor up code.

The following four codes that begin with E specify the values generated by CREDIT to control the erasing of the CRT display:

- EK specifies the erase entire line code.
- EL specifies the erase rest of line code.
- ER specifies the erase rest of screen code.
- ES specifies the erase whole screen code.

The following eight codes that begin with X specify the values expected by CREDIT for the various screen mode commands. The capability to change these command codes is provided so you may make use of any function keys on your terminal or simply change the command codes for your personal preference. When the codes are changed, the old codes, ↑A, ↑V, etc., can be used as macro names, but the first character of the new code may not be used as a macro name.

- XA specifies a replacement code for the screen mode Insert command (↑A).
- XC specifies a replacement code for the screen mode Insert Character command (↑C).
- XD specifies a replacement code for the screen mode Delete Character command (↑D).
- XF specifies a replacement code for the screen mode Macro Function command (↑F).

- XN specifies a replacement code for the screen mode Next Page command (↑N).
- XP specifies a replacement code for the screen mode Previous Page command (↑P).
- XV specifies a replacement code for the screen mode View command (↑V).
- XZ specifies a replacement code for the screen mode Delete command (↑Z).

Alter Function Examples

To change the wildcard character from question mark (?) to asterisk (*):

```
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):

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

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

```
AFER=
```

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

```
AFMD=0A
```

To change screen mode View command (↑V) to the hexadecimal sequence 7EH 1CH (this is the CLEAR key on a Hazeltine 1510 terminal):

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

Query Alter Command

You can inspect the current settings of the alterable editing features with the Query Alter command. The syntax of the Query Alter command is:

```
?A
```

CREDIT prints a listing of the alterable editing features and their present values.

Alter Command Examples

There are two categories of changes possible with the Alter Command: changes made to editing parameters such as line length, tab settings, and so on, and changes made to ASCII characters used in non-Intel terminal I/O. Appendix A includes examples of changes needed to use CREDIT with nonstandard terminals.

Remember that blanks following the = sign are significant.

To change the tab setting from the default setting of 8 to 4, enter:

```
AT=4
```

To change the displayed representation of the line terminator to #, enter:

```
AL=#
```

Using CREDIT With SUBMIT

When invoked with SUBMIT, CREDIT operates only in command mode. Operation is basically the same with these exceptions:

- The Quit command (EQ) does not require a Y or y confirmation.
- CREDIT does not recognize ↑V.
- Non-printing characters, such as control characters, are not echoed to the screen as ↑<char> as they normally are.
- Only two external files may be open at a time because SUBMIT uses one file.

When CREDIT is running under SUBMIT, all interaction with the terminal, such as reading, line editing, and echoing of characters, is handled by SUBMIT. To enter commands, your SUBMIT file must contain a ↑E which turns control over to the terminal until the next ↑E is encountered. For further information about using SUBMIT, see the *ISIS-II User's Guide*.

APPENDIX A
CONFIGURING CREDIT™ FOR NON-INTEL TERMINALS

CREDIT is designed to run on an Intellec Series II development system or Model 800 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. However, you can configure CREDIT to operate with other terminals. The Alter commands allow you to modify certain keyboard and CRT codes. Additionally, it may be necessary to avoid the editing situations described below.

Table A-1. Intel® Terminal Control Codes

Cursor Function	Cursor Key Input Code	CRT Cursor Move Output Code
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 line	1BH,4BH
Blankout character	20H (ASCII blank)

This appendix contains tested configuratons for several non-Intel terminals. The terminals listed are not the only one you can use with the editor, they are the ones that have been tested with it.

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:

- ASCII codes 20H through 7EH display some symbol requiring one column space. 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 rest 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.

- A line containing more than 80 characters wraps around to the beginning of the next line. If your terminal doesn't do this, you must avoid entering lines longer than 80 characters and editing files that contain lines longer than 80 characters.
- The cursor, when moved left from the left margin, wraps around to the end of the previous line. If your terminal doesn't do this, you should avoid moving the cursor over the far righthand column during a deletion (↑Z) or starting a deletion on the far righthand column.
- 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 screens rolls up one line. If your terminal doesn't do this, avoid rolling the screen during text entry.
- CREDIT automatically generates a linefeed each time the carriage return is entered. Your terminal should not generate a linefeed with a carriage return. In some terminals, this functon can be switched on and off.

CREDIT Configuration Examples

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

- The cursor key output codes expected by the editor—AFCH, AFCU, AFCD, AFCR, and AFCL.
- The editor generated cursor movement codes sent to the CRT—AFMH, AFMU, AFMD, AFMR, AFML.
- The erase screen code—AFES.
- The 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—AFXA, AFXC, AFXD, AFXF, AFXN, AFXP, AFXV, and AFXZ. You 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. If you want to enter the Alter commands after invoking the editor, you will have to type the Intel-terminal HOME code (1DH) to enter command mode. The key that generates this code will vary from terminal to terminal. <Control-Shift-M> works on many terminals. Check the terminal manual for the means to generate this code with your terminal.

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-II 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.

ADDS Regent Model 200

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

Add text (↑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.

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	1B 59 20 20	
ML	15 or 08	NAK or BS
MR	06	ACK
MU	1A	SUB
EK	1B 4B	ESC K
ER	1B 6B	ESC k
ES	not used	
AV		24

Command file: ADDS.MAC

```
AFCD=0A;AFCH=01;AFCL=15;AFCR=06;AFCU=1A;
AFMD=0A;AFMH=1B 59 20 20;AFML=15;AFMR=06;AFMU=1A;
AFEK=1B 4B;AFER=1B 6B;AV=24;AFXZ=0B;AFXF=05;
AFXA=14;AFES=;
```

Beehive Mini-Bee

This Behive 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.

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

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

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. The choice of the control-O (↑O) for the HOME function is totally a personal preference.

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

Command file: VT52.MAC

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

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 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. The choice of the control-O (↑O) for the HOME function is totally a personal preference.

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

Command file: VT100.MAC

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

Hazeltine 1510

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

To define a macro which issues a Hazeltine 1510 HOME code, you cannot use the HOME key because it generates a control-R code which echoes the command line. You must 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 the cursor-right key.

Function Code	Hexadecimal Value (~ Lead-In)	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
EK	0D 7E 0F	CR ~ SI
ER	7E 18	~ CAN
ES	7E 1C	~ FS
XP	0F	SI
V		24

Command file: 1510T.MAC

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

Function Code	Hexadecimal Value (ESC Lead-In)	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
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

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

Lear Siegler ADM-3A

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

Function Code	Hexadecimal Value	Graphic or ASCII Name
CD	0A	LF
CH	1E	RS
CL	08	BS
CR	0C	FF
CU	0B	VT
MD	0A	LF
MH	1E	RS
ML	08	BS
MR	0C	FF
MU	0B	VT
EK	not available	
ER	not available	
ES	1A	SUB
V		24

AFCD=0A;AFCH=1E;AFCL=08;AFCR=0C;AFCU=0B;
AFMD=0A;AFMH=1E;AFML=08;AFMR=0C;AFMU=0B;
AFEK=;AFER=;AFES=1A;AV=24;

APPENDIX B
CREDIT™ EDITING COMMAND SUMMARY

The following appendix gives you an alphabetical list of the features and facilities available to you in CREDIT.

CREDIT Screen Mode Commands

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

To move cursor:	Use the four directional arrow keys on the keyboard.
To modify text:	Type new character over old character.
To switch modes:	Press HOME key to enter command mode; press ↑V to enter screen mode.

CREDIT Command Mode Commands

Alter	A code = new definition	Change editing environment to new value
	?A	Display current Alter commands values.
Begin File	B	Go to beginning of Read file.
Close File	CR, CW	Close Read or Write side file.
Copy Text	XC tag, nl-nltag	Copy text from tag boundary to pointer.
Deletion	DC [nl-nltag]	Delete n characters, or text up to, but not including, tag.
	DL [nl-n]	Delete n lines forward or backward.
Exit Loop	EL	Exit current iteration loop.
Exit Quit	EQ	Exit; abandon editing changes to file.
Exit	EX [filename]	Exit; save editing changes to file.
Find	F/string/ [nl-nltag]	Find /string/; move pointer if found.
Get	G [filename]	Read text of filename into command line.
Help	H	Display menu of CREDIT commands.
Insert	I/text/	Insert /text/ in front of pointer.
Jump	J [nl-nltag]	Move pointer n characters or to tag.
Line Move	L [nl-n]	Move pointer n lines forward or backward.
Macros	MD <name>|*	Delete macro name or all macros *.
	MF <name> [arg[,...arg]]	Expand and execute macro contents.
	MS <name> /defining text/	Define macro name .
	?M	Print names and definitions of all macros.
Move Text	XM tag, nl-nltag	Move text from tag boundary to pointer.
Open Read	OR	Open side file for Read.
Open Write	OW	Open side file for Write.
Print	P [H][nl-nltag]	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 accordingly.
Read	R [n]	Insert n lines from side file at pointer.
Substitute	S/oldtext/newtext/ [nl-nltag]	/newtext/ replaces /oldtext/ if /oldtext/ is located.
	SQ/oldtext/newtext/ [nl-nltag]	Like S, but query user before change.
Tag	TD [n]	Delete tag. n=0 through 9.
	TS [n]	Set tag. n=0 through 9.
User	U/text/	Prints/text/when command line executes.
Write	W [n]	Write n lines to side file previously specified.
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™ FILE USAGE

The ISIS-II operating system allows only 6 open files at one time. Since CREDIT runs under ISIS-II, CREDIT can only have 6 open files at one time.

When you invoke CREDIT to edit an old file (that is, a file with text previously inserted) three files are opened:

- The old edit file
- An output file, called CREDT1.TMP. This file retains this name until you exit from CREDIT. At that point it is renamed to the old filename, or if you specify the TO <new filename> option, to the new filename.
- Another temporary file, called CREDT2.TMP, which opens only when the pointer backs up toward the beginning of the file into a part already written to the disk.

CREDT1.TMP and CREDT2.TMP are reserved filenames that you cannot use. The temporary files exist on the same drive as the old file, or on the same drive as the new file if the TO option is used. They will only appear in the directory if ISIS-II reboots while CREDIT is editing a file.

The following commands also use external files:

- G — Get uses your specified file.
- XC, XM — The Copy and Move commands both use CREDT3.TMP, a reserved filename.
- OR, OW — The Open file commands use your specified file, until closed.
- R, W — Read and Write use your files opened by OR or OW.

You don't have to worry about these open files during normal operation, since you can't use more than six files. Neither CREDIT itself or a command file invoked with CREDIT count toward these files.

However, when running CREDIT under the SUBMIT facility, it is possible to exceed the six file limit, which will cause a fatal ISIS-II error and automatic reboot. When CREDIT runs under SUBMIT, there are four background files open — the old file, the two .TMP files mentioned previously, and SUBMIT — which leaves only two available for user commands. If you open a Read or Write file, and issue a command which uses side files, you may exceed the six file limit.

APPENDIX D
CREDIT™ ERROR MESSAGES

CREDIT detects errors in both Screen Edit Mode and Command Mode. In general, different errors will appear in each mode, but three are not specific to either mode:

- INSUFFICIENT MEMORY
- CREDIT ERROR
- WARNING: DISK FULL

INSUFFICIENT MEMORY occurs when there is less than 64K bytes of memory available in the system CREDIT is running on. A CREDIT ERROR occurs when CREDIT detects an internal problem. The WARNING: DISK FULL message is not really an error, but a warning; it will be discussed separately below.

Screen Edit Mode Errors

When in the screen edit mode, the only printing error messages you will encounter are those associated with macros. In these cases, no ERROR flag will appear, nor will the erroneous text.

CRT errors are more likely to be illegal CRT commands. These cause "beeps":

- Attempting to move the cursor out of the display window with the cursor control arrow keys
- Attempting to modify a screen location (with insertion, deletion, etc.) that does not contain text
- Attempting to insert or modify an unliteralized invisible character via ↑A, or ↑C
- Pressing any key except cursor control arrow keys or the ↑Z when doing a ↑Z deletion
- Attempting to modify or delete the end-of-file character; ↑Z can pass the EOF character during a deletion, though

Syntax Errors

An error made during startup, or during command mode 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 the screen edit mode if you reference a faulty or undefined macro.

These are the possible errors:

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, or a second OR or OW command was issued without closing the file first.

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

IMPROPER OPERAND
The command argument is the wrong type for the command, or 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).

ITERATION ERROR
An illegal iteration quantifier was encountered, or there were too many or too few ending brackets.

ARGUMENT MISMATCH
During expansion of a macro, more percent signs were encountered than parameters in the invocation argument 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, G command, or when entering a command line. An error while entering a command line offers the option of executing the command line.

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

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

DOESN'T EXIST
The specified filename doesn't exist, or the specified tag or macro hasn't been defined.

ALREADY EXISTS
The filename specified for creation is already there, or the macro specified is already defined.

TAG POSITION
A tag specified for DC, XC, or XM 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.

WARNING: DISK FULL

The WARNING: DISK FULL message is a warning rather than an error. CREDIT is telling you that you are running out of disk space, and that there is only enough space left to exit and save your editing additions. You can also receive this warning if there is not enough space left to get out. This latter condition can occur on startup, if not enough space is available, or if you continue editing past the first warning.

This warning terminates operations which use side files, except for the G and H commands. This includes user-transparent file operations that organize memory internally.

The disk full condition is actually an approximation based on an internal count. It therefore makes worst-case estimates of factors it cannot know (such as how many more additions you will add) and notifies you before you run out of space. You can keep editing after receiving a disk full warning, but at your own risk.

When you exit the editor after receiving a DISK FULL message, you may find a relatively large amount of space available on the disk. Don't make the mistake of thinking you got the warning in error. CREDIT uses temporary files not on your directory listing because they are deleted when you exit the editor.

APPENDIX E
ASCII CODES

Table E-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
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	;

Table E-1. ASCII Code List (Cont'd)

Decimal	Octal	Hexadecimal	Character
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
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 F
CHANGES IN CREDIT™ V2.0

Version 2.0 of CREDIT has additions and changes that may alter the operation of previously defined command files and macros which worked under Version 1.0. These alterations are described below.

In thinking about ranges, it is fruitful to 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. Control-Z: Formerly this Delete Text command included the character under the final Control-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 Control-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 Control-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 the 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 also page 4-9.
4. Tag redefinition or deletion: Redefining an existing tag and deleting a non-existent 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 page 2-2 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 will automatically exit from the innermost iterative loop if 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 than 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 control-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 pages 5-13 through 5-15.

INDEX

REQUEST FOR READER'S COMMENTS

The Microcomputer Division Technical Publications Department attempts to provide documents that meet the needs of all Intel product users. This form lets you participate directly in the documentation process.

Please restrict your comments to the usability, accuracy, readability, organization, and completeness of this document.

1. Please specify by page any errors you found in this manual.

2. Does the document cover the information you expected or required? Please make suggestions for improvement.

3. Is this the right type of document for your needs? Is it at the right level? What other types of documents are needed?

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

5. Please rate this document on a scale of 1 to 10 with 10 being the best rating. _____

NAME _____ DATE _____

TITLE _____

COMPANY NAME/DEPARTMENT _____

ADDRESS _____

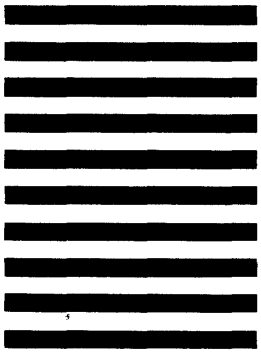
CITY _____ STATE _____ ZIP CODE _____

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

Printed in U.S.A.