DataGeneral

CRT/EDIT

Display Terminal Text Editor

055-005-00

Data General Corporation, Westboro, Massachusetts 01581

For the latest enhancements, cautions, documentation changes and other information on this product, please see the release notice supplied with the software.

****
TABLE OF CONTENTS CRT/EDIT
****

PREFACE

This manual is written for the programmer who is engaged in creating, modifying, or maintaining source programs for DGC Commercial Systems.

The manual describes the use and operations of CRT/EDIT, a visual display program editor. The editor is designed specifically for use with DGC's Commercial Systems software and display terminals.

Chapter 1 presents an overview of the editing process, and describes some of the functions of CRT/EDIT. The novice user should become familiar with this chapter before reading further or using the editor. Chapter 2 describes the basic editor commands used in creating and maintaining program source files. Chapter 3 describes the advanced editor commands and command file execution facilities in detail. Chapter 4 contains a brief summary of all editor commands and symbols.

TYPOGRAPHICAL CONVENTIONS

Throughout this manual the following conventions are followed:

* The names of special keyboard characters are enclosed in angle brackets: for example the key with "DEL" written on it is represented by <DEL>.

* The <CTRL> key on the DASHER D2 keyboard is equivalent to the <CMD> key on the DASHER D3, D4, and D5 keyboards. In this manual, we refer only to <CTRL>. Combinations involving the <CRTL> key are represented with angle brackets: for example, <CTRL-L> stands for the character entered by holding down the <CTRL> key and typing the letter "L".

* The function keys are denoted with the prefix "f": for example, <f8> denotes function key #8. The DASHER D3, D4, and D5 function keys are marked in this way. The DASHER D2 function keys are unmarked.

* CRT/EDIT echoes the <ESC> key as an underscored dollar sign. This manual always uses <ESC> to represent this character.

* Names enclosed in tildes (~) denote variables: for instance, ~string~ and ~filename~. Any place where a numeric argument or numeric argument expression may be used is denoted by <arg>.

INTRODUCTION

CRT/EDIT is a display terminal oriented version of Data General's "TEXT EDITOR" program. CRT/EDIT runs under control of the operating system, and is executed through a CLI command. During an editing session a single source file may be edited in one or more passes, or several files may be edited in sequence. The source program files created by CRT/EDIT may be input to the Interactive COBOL compiler.

CRT/EDIT is capable of all the basic editing functions (e.g. delete, insert, display, and search). It also has many sophisticated functions for the more advanced user. There are facilities for:

* Cut and paste (removing a block of program text from one place and inserting it elsewhere)

* Documentation and text justification (line width may be set and running text justified on a paragraph basis)

* Executing editor command files (often used command sequences can be stored in files and executed with a single command)

* Search and replace (locate a group of characters and substitute another group)

* Conditional execution of commands (perform an edit function depending on some variable)

CRT/EDIT enables programmers to create or modify source programs rapidly and efficiently with a minimum of keying errors in the resulting source files. The reason for this is the visual feedback supplied by CRT/EDIT. Whenever a command is executed, the screen is redrawn to show exactly what is contained in the portion of the file being edited. This feature allows the programmer to "see" what the lines of program code look like as they appear in the file. It also enables the user to backtrack and correct any mistakes before the file is written. This means fewer compiler runs as most typographical, and other types of errors can be eliminated before an attempt is made to compile a source file.

CRT/EDIT has a feature which lets the user edit the command string being entered. This is most helpful in correcting errors due to mistyped commands, and provides the additional benefit of being able to modify commands as necessary.

The line width and paragraph justify functions provide the capability to create documentation (such as this manual). Running text can be entered in the same way as program text and then justified on a full word basis (words are not broken at the end of a line) to fit a specified line length.

The command file execution feature is a most powerful development tool. It can be used to invoke often used short sequences of commands (such as those used to enter a basic COBOL program Identification Division) thus saving time and keystrokes. This feature allows the programmer to develop complex sequences of search and replace and/or insert and delete commands as well as other editing functions while eliminating the need to rekey complicated commands.

CRT/EDIT accepts numeric arguments for commands in the form of numbers or arithmetic expressions which are evaluated by the editor. There are special argument characters which return specific values and logical operators which can be used to form simple or complex expressions.

CHAPTER 1 AN OVERVIEW OF CRT/EDIT

The material in this chapter provides an overview of CRT/EDIT and its functions.

STARTING AND ENDING A CRT/EDIT SESSION

CRT/EDIT runs at the system's master terminal or concurrent operation console. Entering "CRTEDIT <CR>" at the CLI's "R" prompt starts the editor. An editing session is ended, and control returned to the CLI, with CRT/EDIT's "H" command. See Chapter 2 for further details.

FILE REPRESENTATION

CRT/EDIT is a character-oriented and string-oriented editor. A string is any contiguous set of characters which can be manipulated as a unit. CRT/EDIT treats an input or output file as one continuous string of characters. The <CR> that terminates a line of text is treated just like any other text character. The first character of the next line immediately follows the terminating <CR> -- CRT/EDIT does not add trailing spaces to "fill out" a line of text.

Breaking Source Files into Pages

CRT/EDIT maintains a variable size "edit buffer" in memory. Before CRT/EDIT can modify a section of text, it must first read the text from the input file into the edit buffer. Since the edit buffer may not be large enough to hold all of a large input file, it may be necessary to break up the input file into "logical pages".

Logical pages are delimited by the character <CTRL-L> (form feed). When a CRT/EDIT file is printed, <CTRL-L> causes the printer to skip to the top of the next page. During an editing session, a <CTRL-L> in the input file marks the endpoint of a page to be read by the "read next page" command. Thus, a physical page of a printed copy of the file does not correspond exactly to a logical page.

CRT/EDIT is a "pipeline" editor. This means that editing starts at the beginning of a file and proceeds forward, one logical page at a time. If a file is only one page long, the user may move backward and forward freely. If a file is more than one page long, the user can move freely only within a particular page. Once a page has been written, it can no longer be accessed directly. To return to a previous page, the file must be closed and then re-opened for editing.

CRT/EDIT cannot display text across page boundaries. That is, when editing the last few lines on a page, the first few lines of the next page cannot be displayed. For this reason, it is often convenient to make pages large enough to contain a logical unit of program-text, so that the entire unit may be freely edited in a single pass.

NOTE: For purposes of obtaining cross-referenced compiler listings, source files should not contain more than 99 lines of source code per logical page. For details on inserting page breaks in text files, see the insertion/deletion commands in Chapter 2.

OPERATIONS

This section summarizes the overall flow of CRT/EDIT operations, and explains the format of the commands that control these operations.

An Edit Pass

CRT/EDIT permits the user to create and/or modify source files. CRT/EDIT operates on two separate files, an input file and an output file. The user has the choice of storing editing changes (1) under the original input filename, deleting the previous version of the file, or (2) under another filename, presrving the original input file unchanged.

An edit pass may be aborted at any time: the output file is deleted, and the original input file remains unaltered.

Only one file at a time may be open for editing. Several files may be edited in a single session by opening and closing them one after another. The sequence of editing operations on a single file may be termed an "edit pass". In general, an edit pass includes these steps:

1A. OPEN AN EXISTING FILE	or	1B. DEFINE A NEW FILE
This process opens an existing file for input and creates a "scratch" output file (filename.SC).		This process creates an new file and assigns it as the output file to receive newly entered or copied text.

2. EDIT THE TEXT

A variety of editing commands may be performed. These include displaying, inserting, deleting, changing, and copying text.

3. WRITE AND CLOSE THE FILE

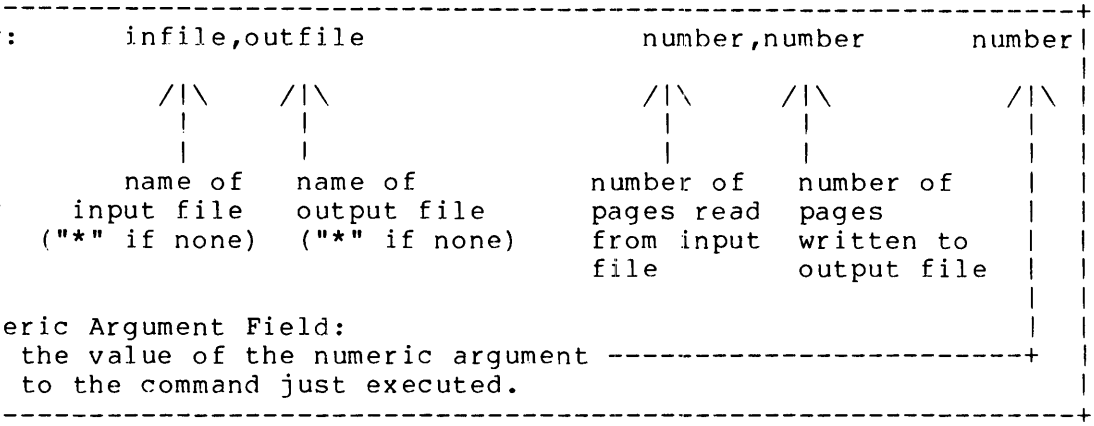
The editing changes may be stored under the original input filename (if any) or under a new filename. In the latter case, the original input file (if any) remains intact and may be used as a backup.

DISPLAY SCREEN FORMAT

The display screen is divided into three areas, each of which is updated at the start of every command cycle.

The Editor Status Line

The first line of the screen displays information regarding the current directory and the current input and output files.



The numeric argument field is useful for using CRT/EDIT as a desk calculator. When a numeric expression is entered alone, CRT/EDIT displays the value of the expression in the numeric argument field. CRT/EDIT offers a full range of algebraic operators, and can work with different number bases (radices), including binary, octal, decimal, and hexadecimal. Thus, this feature can be used as a powerful calculating tool.

The Text Window

The main portion of the display screen is a continually-updated window into the part of the file being edited. When it is not executing a command, CRT/EDIT's attention is focused at one character in the edit buffer. The position of this character is called the "cursor location". CRT/EDIT's text window displays a portion of text from the edit buffer surrounding the cursor location. This location is marked in the text window by a flashing underscore, the "cursor".

At the start of every command cycle, the display is automatically adjusted to include eight lines on either side of the line containing the cursor. (This number may be modified -- see "Miscellaneous Commands" in Chapter 3.) If the cursor location is near the beginning or the end or the edit buffer, a full screen of text is always displayed, but the cursor may not be centered in the window.

OVERFLOW LINES. Normally, a line of text -- all characters between successive <CR>s -- is displayed on one line of the display screen. The position to the right of the last visible character is occupied by the <CR> that terminates the line. If a line is too long to be displayed across the screen, it is broken with a "-" in column 80 and continued on the next line. If the text window includes more than one such line, information may disappear below the bottom of the text window. When editing a file consisting of many long lines, the user should reduce the screen window size ("TF" command) or enter line-truncate mode ("TT" command). Information that disappears in line-truncate mode is not deleted from the edit buffer -- it merely is omitted from the display screen.

The Command Window

The four lines at the bottom of the screen display CRT/EDIT commands as they are typed. This display scrolls up and down as lines are inserted or deleted from the command. CRT/EDIT displays a tilde (~) at the end of the command so that trailing <SPACE> or <TAB> characters may be detected. During execution of a command, CRT/EDIT displays a blinking "#" to indicate that it is "busy".

When CRT/EDIT starts a new command cycle, the previous command string (if any) is redisplayed in the command window. If <ESC> is pressed, this command is reexecuted and is once again redisplayed. This facility makes it possible to repeatedly execute a command string at different locations in the file.

The command window is always in line-truncate mode -- only the first 79 characters of each line are displayed.

The Error Display

If an error occurs during the execution of a command string, the screen is erased and an error message is displayed on the top line. Up to the first eight lines of the command which caused the error are displayed just below the error message. The error condition is cleared and normal display restored by typing <CR> or <NEWLINE>.

EDITOR COMMAND CYCLE

The cycle of CRT/EDIT command execution is as follows:

1. Display the text surrounding the current cursor location.
2. Accept a command string from the keyboard.
3. Execute the command string.
4. Go to step 1.

Note that the text window is updated automatically after the execution of every command string. This provides immediate visual verification of the action of the commands just executed.

EDITOR COMMAND FORMAT

Commands may be typed in either uppercase or lowercase. Commands lines are executed strictly from left to right. See, however, "Branching and Labels" and "Error Suppression" in Chapter 3.

The command area displays only the first 79 characters of each command line. Therefore, the user should avoid long command strings unbroken by <CR>s (such as lengthy text insertions).

THE CRT/EDIT command format is illustrated below. An individual command may not require all three elements shown. Execution of a command begins when two consecutive <ESC>s are pressed.

```
  <arg>   command character(s)   character-string<ESC>
```

<arg>	The optional numeric argument usually specifies a repetition factor. The argument may be a single number, a value stored earlier by the user, a value maintained by CRT/EDIT, or an expression involving several such values.
command character(s)	Some commands are invoked with a single command character -- others use a sequence of two or more characters. Command that address storage registers (for text or numerical values) also include a register label.
character-string<ESC>	Search/replace commands and text insertion commands include one or more string arguments, each terminated by <ESC>. Several commands include string arguments that name files.

Examples of single editor commands:

```
    4L              numeric argument = 4
                    command character = L
                    no string argument


    2Sabc<ESC>      numeric argument = 2
                    command character = S
                    string argument = abc    (CRT/EDIT echoes <ESC> as "$")


    #3TW            numeric argument = #3
                              (value stored in numeric storage register 3)
                    command characters = TW
                    no string argument
```

Multiple-Command Strings

CRT/EDIT commands may be strung together and executed as a unit. Execution of the command string does not begin until two consecutive <ESC>s are pressed.

Individual commands within a command string may be separated by a single <ESC>. This <ESC> is ~required~ only to mark the ends of string arguments.

Similarly, individual commands may be separated by <CR>s as a visual aid -- the effect is to place individual commands on different lines in the command window.

Examples of multiple-command strings:

The three command strings below are equivalent. They all execute the command "4L" followed by the command "Sabc<ESC>".

1. 4L<ESC>Sabc<ESC><ESC>

2. 4LSabc<ESC><ESC>

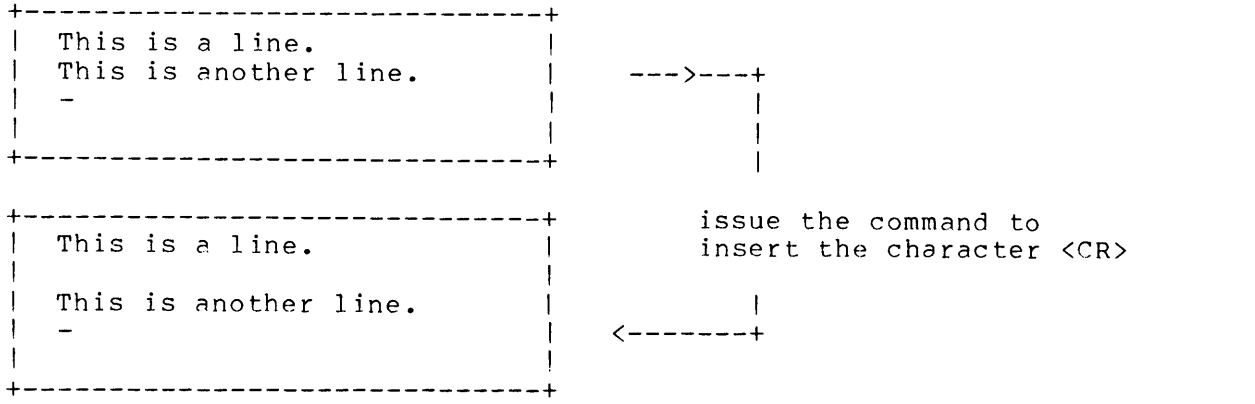
3. 4L<CR>
   Sabc<ESC><ESC>

SPECIAL KEYBOARD CHARACTERS

<CR> and <NEWLINE>

CRT/EDIT is fundamentally character-oriented, not line-oriented. A <CR> (carriage return) is not a control character, but a text character, just like any other. It is used to delimit lines of text, not to terminate editor commands.

CRT/EDIT displays a blank line when it encounters two consecutive <CR>s. Blank lines must be inserted with the insert command. For example:



The <NEWLINE> key is equivalent to <CR>. Both generate the same ASCII code (octal 15).

<TAB>

The TAB character is used to format text into columns, creating the ~effect~ of inserting blank spaces into the file. CRT/EDIT does ~not~ actually insert spaces, however. <TAB> is treated as a single character (octal 11).

The visual effect of inserting a <TAB> character depends on the column position. CRT/EDIT defines tab stops (predetermined and unalterable) every eight columns. When the editor displays a <TAB>, it moves the cursor to the next tab stop -- from one to eight columns, depending on the cursor location. For example:

```
+------------------------------+
|  Col 1:  Col 2.              |    <----- display of the character
|                              |           string:
|                              |
|                              |             <CR>Col 1:<TAB>Col 2:
+------------------------------+
```

Note: <TAB> appears to insert two spaces

```
+------------------------------+
|  Column 1:       Column 2:   |    <----- display of the character
|                              |           string:
|                              |
|                              |             <CR>Column 1:<TAB>Column 2:
+------------------------------+
```

Note: <TAB> appears to insert seven spaces

NOTE: The effect of <TAB> in a printed copy of the file varies from printer to printer.

<ESC>

<ESC> is CRT/EDIT's command separator and terminator. A single <ESC> may be placed between individual commands in a multiple-command string. CRT/EDIT begins command execution when two consecutive <ESC>s are pressed.

Cursor Movement Keys

Certain keys cause the cursor to move without affecting the command string. The use of these keys may be freely interspersed with pressing <ESC> to reexecute the currently displayed command string. The cursor movement keys are described in Chapter 2.

Operating System Control Characters

Certain control characters are intercepted by the operating system and have no effect in CRT/EDIT operations. They are: <CTRL-A>, <CTRL-C>, and <CTRL-F>.

The two operating system display controls, <CTRL-S> and <CTRL-Q>, are operative within CRT/EDIT: <CTRL-S> freezes the screen display (giving the appearance that CRT/EDIT has ceased to function) until <CTRL-Q> is pressed.

CHAPTER 2 THE BASIC EDITOR COMMANDS

CRT/EDIT supports a large number of commands, but only a few of these are necessary for basic editing operations; the less frequently used commands are included for the convenience of the experienced user and to support advanced editing requirements. The advanced commands are described in Chapter 3.

Pressing <f11> (function key 11) displays an abbreviated command summary on the screen; normal display is restored by pressing <CR> or <NEWLINE>.

The commands in this chapter are grouped by function as follows:

* Startup and Exit Commands
* File Input and Output Commands
* Cursor Movement Commands
* Cursor Movement Keys
* Insertion and Deletion Commands
* Search and Replacement Commands
* Paragraph Formatting Commands
* Command Editing Keys
* The HELP Key

STARTUP AND EXIT COMMANDS

Startup

To execute CRT/EDIT, type at the CLI "R" prompt:

```
        CRTEDIT <CR>
 or     CRTEDIT filename <CR>
```

If the argument ~filename~ is included, CRT/EDIT opens ~filename~ for editing as the input file. (This is equivalent to the command "Ofilename".)

Exit

Before ending a CRT/EDIT session, make sure that no input or output files are open. (These files may be closed with the "GX" command -- see File Input and Output commands below.) Then, issue the following command:

H — Terminate CRT/EDIT

Returns control to the Command Line Interpreter. An error occurs if an input file or output file is open.

FILE INPUT AND OUTPUT COMMANDS

Ofilename<ESC> — Open a file

This is the standard way to make a file available for editing. The command makes ~filename~ the input file and creates ~filename~.SC as a temporary output file. The first logical page of the input file is read into the edit buffer and the cursor is placed at the first character.

W<ESC> — Close a file

This is the standard way of terminating an edit session that was begun with the "O" command. This command stores the editing session under the the filename specified in the "O" command.

More generally, this command functions as follows:

* IF AN INPUT FILE EXISTS: Stores the editing session under the currently defined input filename -- as displayed on the editor status line -- and deletes the output file. The original input file is lost.

* IF AN INPUT FILE DOES NOT EXIST: Stores the editing session under the currently defined output filename -- as displayed on the editor status line.

NOTE: The <ESC> following this command is mandatory; otherwise, the characters following it will be interpreted as a file name (see next command).

Wfilename<ESC> — Close a file and rename

Copies the remainder of the input file (if any) to the output file and renames the output file to ~filename~. The input file remains intact with its original name. Thus, this command is useful for preserving a backup version of a file.

GWfilename<ESC> — Define an output file

Creates a new text file for editing. The output file is set to ~filename~ and the editor status line is updated. There is no input file. The "W<ESC>" command may be used as above to close the file.

GX — Abort an edit pass

Closes both the input and output files and deletes the output file. The input file remains unaltered. Any editing changes made to the input file are lost. When the part of the file already output should be retained, issue a "GCO" command (decribed in Chapter 3) to close the output file. This will prevent it from being deleted by the "GX" command.

J<ESC> — Start a new edit pass

Equivalent to "W<ESC>Oinputfilename<ESC>". This command stores the edit session under the input filename and starts a new edit session, setting the cursor back to the first character of the first page of the file. The original input file is lost.

NOTE: The <ESC> following this command is mandatory; otherwise, the characters following it will be interpreted as a file name (see next command).

Jnewfilename<ESC> — Rename and start a new edit pass

Equivalent to "Wnewfilename<ESC>Onewfilename<ESC>". This command stores the edit session under ~newfilename~ and starts a fresh edit pass on the renamed version of the modified file. The original input file remains unaltered. Thus, this command is useful for preserving a backup version of a file.

<arg>R — Read next page

default value of <arg>: 1

Performs a write/read cycle <arg> times: write the edit buffer to the output file and read the next page from the input file. The cursor is positioned to the first character of the new page. The input page number (@I) and output page number (@O) are incremented, as are the corresponding entries in the editor status line. (See "Numeric Arguments to Commands" in Chapter 3.)

If the end of the input file has been reached, or if no input file is open, an error occurs and the value of ?, the error flag, is set to -1. (See "Error Suppression" in Chapter 3.)

CURSOR MOVEMENT COMMANDS

B — Go to beginning

Places the cursor at the first character in the edit buffer. This character is numbered 0 (see usage of the numeric argument @C).

NOTE: Pressing <f15> on the DASHER D3 keyboard or the corresponding key (blank key below <TAB>) on the DASHER D2 keyboard is equivalent to issuing this command.

Z — Go to end

Places the cursor at the end of the edit buffer. This character is always a "null" (octal 0). It may not be changed or deleted.

NOTE: Pressing the <ERASE EOL> key is equivalent to issuing this command.

<arg>M — Move <arg> characters

default value of arg: 1

<arg> = 0 no action.

<arg> > 0 moves the cursor forward <arg> characters.

<arg> < 0 moves the cursor backward <arg> characters.

NOTE: Pressing the <right-arrow> key is equivalent to issuing the command "1M". Pressing the <left-arrow> key is equivalent to issuing the command "-1M".

CRT/EDIT treats <CR> and <TAB> as single characters. If the cursor is at the end of a line, moving the cursor to the right one character places it at the beginning of the next line. If the cursor is placed at a <TAB>, it jumps to the next tab stop. Tab stops are fixed at every eighth column.

If this command attempts to move the cursor past either end of the edit buffer, the value of ?, the error flag, is set to -1 (logical "true"). No error message is displayed.

<arg>L — Move <arg> lines

default value of <arg>: 1

<arg> = 0 moves the cursor to the beginning of the current line.

<arg> > 0 moves the cursor forward <arg> lines.

<arg> < 0 moves the cursor backward <arg> lines.

NOTE: Pressing the <down-arrow> key is equivalent to issuing the command "1L". Pressing the <up-arrow> key is equivalent to issuing the command "-1L".

If this command attempts to move the cursor past either end of the edit buffer, the value of ?, the error flag, is set to -1 (logical "true"). No error message is displayed.

Example:

```
+---------------------------+
|  ABCDEF                   |  If the cursor location is at "K", then:
|  GHIJKL                   |
|  MNOPQR                   |         0L    would move the cursor to "G"
|                           |    1L or L    would move the cursor to "M"
|                           |        -1L    would move the cursor to "A"
+---------------------------+
```

CURSOR MOVEMENT KEYS

Certain keys cause the cursor to move without affecting the command string. These are termed "cursor movement" keys. CRT/EDIT remembers the last command string executed -- pressing <ESC> causes reexecution of the command string. This facility, when used in conjuction with the cursor movement keys, allows a command string to be executed repeatedly at defferent locations in the edit buffer.

In the table below function keys are denoted with the prefix "f".

KEY	CURSOR MOVEMENT	EQUIVALENT COMMAND
<f1>	Back 4 lines	-4L
<f2>	Back 1 line	-1L
<f3>	Ahead 4 lines	1L
<f4>	Ahead 1 line	4L
<f5>	Back 4 characters	-4M
<f6>	Back 1 character	-1M
<f7>	Ahead 4 characters	1M
<f8>	Ahead 1 character	4M
<f15> or <BLANK KEY>	Beginning of edit buffer	B
<ERASE EOL>	End of edit buffer	Z
<HOME>	Other end of current line	1L-1M or 0L
<LEFT ARROW>	Back 1 character	-1M
<RIGHT ARROW>	Ahead 1 character	1M
<UP ARROW>	Back 1 line	-1L
<DOWN ARROW>	Ahead 1 line	1L

When a cursor-movement key, is used to move the cursor, CRT/EDIT does not redraw the text window unless the cursor is moved off the screen. When this happens, <f9> may be used to recenter the cursor (and blank the command string).

INSERTION AND DELETION COMMANDS

Istring<ESC> — Insert a String

Inserts ~string~ at the cursor location. The argument string may be anything from a single character to many lines of text. After the insertion, CRT/EDIT moves the cursor to the end of ~string~. The value of @S is updated with the negative of the length of ~string~. (See "Numeric Arguments to Commands" in Chapter 3.)

To save keystrokes, and for convenience in aligning multi-line insertions, CRT/EDIT allows the user to omit the "I" command character if the string begins with <TAB> or <SPACE>. For instance:

<TAB>PERFORM OPEN-ONE is equivalent to I<TAB>PERFORM OPEN-ONE

ON INSERTING PAGE BREAKS: A page break may inserted simply by inserting a <CTRL-L> wherever the break is wanted. For example, the command:

ISECURITY NONE.<CR><CTRL-L>ENVIRONMENT DIVISION.<CR>

results in a page break between the two source lines entered.

<arg>K — Delete <arg> lines

default value of <arg>: 1

<arg> = 0	deletes all characters preceding the cursor location to the beginning of the line.
<arg> > 0	deletes the rest of the current line and <arg> - 1 complete lines following.
<arg> < 0	deletes all characters preceding the cursor location to the beginning of the line, and <arg> complete lines preceding.

<arg>D — Delete <arg> characters

default value of <arg>: 1

<arg> = 0	no action.
<arg> > 0	deletes <arg> characters from the cursor location forward.
<arg> < 0	deletes <arg> characters preceding the cursor location.

SEARCH AND REPLACEMENT COMMANDS

<arg>Sstring<ESC>	Search for string through remainder of page
<arg>Nstring<ESC>	Search for string through remainder of file

default value of <arg>: 1

Searches forward or backward from the cursor location for the <arg>th occurrence of the specified string. The cursor is positioned just past the end of the string. The value of @S is updated to the search string length: @S is negative for a forward search, positive for a backward search.

If the "S" command does not find ~string~, the error message "MISSING" is given. If the "N" command does not find ~string~, it proceeds to the next logical page and continues the search. An "END OF FILE" error message is displayed if ~string~ is not located through the remainder of the input file. Both commands set the value of ?, the error flag, to -1 after an unsuccessful search. See "Error Suppression" in Chapter 3.

It is frequently much faster to search for a unique substring in the text than to move the cursor by cursor movement keys or the "M" and "L" commands.

CHANGING THE "S" COMMAND SEARCH TERMINATOR. For forward searches with the "S" command, the user may temporarily change the boundary at which CRT/EDIT gives up the search. Normally the "null" character (octal 0) which ends the edit buffer is used as the terminator.

<arg>TS — Set search terminator

default value of <arg>: 0 (null character)

Temporarily changes the forward search terminator to the character whose ASCII code is <arg>. The terminator is reset to null (octal 0) after every successful search/replace operation.

<arg>Cstring1<ESC>string2<ESC>	Change string on current page
<arg>Vstring1<ESC>string2<ESC>	Change string in remainder of file

default value of <arg>: 1

Searches forward, starting at the current cursor location for the <arg>th occurrence of string1. If the string found, CRT/EDIT replaces it with string2.

If the "C" command does not find ~string1~, the error message "MISSING" is given. If the "V" command does not find ~string1~, it proceeds to the next logical page and continues the search. An "END OF FILE" error message is given if ~string1~ is not located through the remainder of the input file. Both commands set the value of ?, the error flag, to -1 after an unsuccessful search. See "Error Suppression" in Chapter 3.

Ustring<ESC> — Replace inserted string

Causes the string just inserted to be replaced by the specified string. This command may be used immediately following an insert, search, replace, or change command.

The "U" command may be executed several times in succession, allowing the user to replace ~string1~ by ~string2~, then replace ~string2~ by ~string3~, and so on. The command "U<ESC><ESC>" replaces the inserted string with nothing -- that is, deletes it.

Examples:

Sabc<ESC>Udef<ESC>	Searches for "abc" and replaces it with "def". Equivalent to Cabc<ESC>def<ESC>.
Cabc<ESC>def<ESC> Ughijk<ESC>	Changes "abc" to "def", and then changes "def" to "ghijk".
Iabcdefg<ESC> U<ESC><ESC>	Inserts "abcdefg" and then deletes the inserted string.

Special Characters In Searches

The control characters <CTRL-@>, <CTRL-P>, and <CTRL-N> have special meanings when they appear in a search argument string:

<CTRL-@>	Matches any single character in the edit buffer.
<CTRL-P>	Matches any string in the edit buffer, including the null string.
<CTRL-N>	Acts as a "not" prefix to the following character. The two-character sequence <CTRL-N>~char~ matches any character except ~char~.

Examples:	A<CTRL-@>B	matches	AXB, AyB, A1B, etc.
	a<CTRL-P>b	matches	axxb, ab, a1234567b, etc.
	1<CTRL-N>23	matches	1Q3, 153, but not 123.

PARAGRAPH FORMATTING COMMANDS

PJ — Paragraph justify

Redraws lines, using the current line justification width, from the beginning of the current line to the end of the paragraph. CRT/EDIT recognizes an end-of-paragraph when it encounters the end of the text buffer or one of these two-character sequences:

<CR><CR> <CR><SPACE> <CR><TAB>

By converting <CR>s to <SPACES> and vice-versa, CRT/EDIT formats the text into the longest possible lines that don't exceed the current line justification width. This width may be adjusted with the "TW" command (see below).

<arg>TW — Set paragraph justification width

<arg> may not be defaulted

Set the line justification width. The PJ command uses this value to determine the line width of the justified paragraph. Initially, the line justification width is set to 79.

COMMAND EDITING KEYS

The following keys may be used to edit the command string:

KEY	ACTION
<DEL>	Erase last character from command string.
<f9>	Blank the command string and redraw the text window centered on cursor location.
<f10>	Begin/end command string editing.

<f10> (function key #10) is a very powerful tool. When struck the first time, it causes the edit buffer to be saved and the current command string to be placed in a new edit buffer. At this point, the command string may be edited using any CRT/EDIT commands. Finally, striking <f10> a second time restores the saved text buffer and places the edited command string back into the command window, ready for execution.

This procedure has several uses:

1. TO VIEW THE ENTIRE COMMAND STRING. Since CRT/EDIT displays only the last four lines of the command string on the screen, it is otherwise impossible to view the beginning of a multi-line command string without deleting the end of the command string.

2. TO CORRECT AN ILLEGAL COMMAND. A common error is to omit the "I" from the beginning of an insertion command. CRT/EDIT, attempting to interpret the string to be inserted as a command, frequently detects an illegal condition. The operator can avoid having to retype the entire string by clearing the error condition, pressing <f10>, inserting an "I" at the beginning of the command, and pressing <f10> again. Similar recovery from other errors is also possible.

3. MODIFYING OR CREATING A COMMAND. <f10> may be used to create a slightly modified version of the last command or to build a new command from scratch. To insert <ESC> into the command string, use the command "<ESC>I.

THE HELP KEY

<f11> (function key #11) may be pressed at any time to display a CRT/EDIT command summary. Pressing <CR> or <NEWLINE> resumes the edit session.

CHAPTER 3 ADVANCED COMMAND DESCRIPTIONS

This chapter describes advanced CRT/EDIT commands in detail. These commands are categorized as follows:

* Input and Output File Commands
* Auxiliary File Handling Commands
* Cursor Movement Command
* Insertion and Deletion Commands
* Conditional Replacement Commands
* Cut and Paste Commands (Text Registers)
* Numeric Arguments to Commands
  - The Numeric Argument Radices
  - Numeric Argument Variables
* Branching and Labels
* Error Suppression
* Repetition Brackets
* Command File Execution
* Miscellaneous Commands

INPUT AND OUTPUT FILE COMMANDS

GRfilename<ESC> — Set the input file to filename

There must not be any currently open input file.

GCI — Close the input file

Another input file may be opened with the "GR" command.

GCO — Close the output file

Any information already written to the output file with the "R" command is preserved.

<arg>A — Append <arg> lines from the input file

default value of <arg>: 0

If <arg> is absent or 0 an entire logical page is appended to the edit buffer. The current input page number (@I) is incremented. The cursor is positioned to the beginning of the appended text.

Any character which generates a parity error on input is replaced by the character "\". This command updates the value of the error flag (?). See "Error Handling" below.

AUXILIARY FILE HANDLING COMMANDS

These three commands use disk files to move sections of text in and out of the edit buffer. Unlike the text registers (described later in this chapter), files created and modified by the "FO" and "FA" commands survive after the edit session is terminated.

<arg>FOfilename<ESC>	Output <arg> lines to new file
<arg>FAfilename<ESC>	Append <arg> lines to existing file

default value of <arg> for both commands: entire edit buffer

CRT/EDIT either creates a new file ("FO" command) or appends to the existing file ("FA" command) with the specified name. Device/directory specifiers may be included.

Output begins at the current cursor location. If <arg> is absent or 0, the entire edit buffer is output. A negative value of <arg> is illegal.

These commands may be used to output lines to the system printer. The command <arg>FO$LPT<ESC> precedes the output lines with a form feed. The command <arg>FA$LPT<ESC> does not.

<arg>FIfilename<ESC> — Insert <arg> lines from disk file

default value of <arg>: entire file

Inserts <arg> lines of the named file at the current cursor location. If <arg> is missing or 0, the entire file is inserted. Any page breaks in the file are represented by the character ^L (octal 14). A negative value of <arg> is illegal.

FDtemplate<ESC> — Delete all files matching a template

An error results from attempting to delete either the input or output file. The specified template may contain the special characters "*" and/or "-".

* * matches any single character

* - matches any string (including the null string) that doesn't include "."

FRfilename1<ESC>filename2<ESC> Rename file

Renames the file currently identified as ~filename1~ to ~filename2~.

<arg>FLtemplate<ESC> List files

default value of <arg>: 1

Inserts a listing of all filenames in the working directory matching the specified template at the current cursor location. If <arg> is negative, all files including permanent and link entries are listed. The template may contain "*" and "-" as above.

@S is set to the total number of characters in the listing and the cursor is left at the beginning of the listing. The inserted information may conveniently be deleted with the command "U<ESC><ESC>".

The first line inserted lists the total number of blocks occupied by the listed file(s) and the total numbers of blocks left and used on the default directory device.

CURSOR MOVEMENT COMMAND

Y Move to opposite end of inserted string

This command may be used to position the cursor to the beginning of text just inserted from the keyboard, a text register, or a file. Similarly, it may be used to position the cursor to the end of text just located in a backwards search.

The "Y" command is equivalent to "@SM". It changes the sign of @S so that immediate reexecutions of the command toggle the cursor between the beginning and the end of the inserted string.

Example: Imultiple-line-string<ESC>YPJ

After an insertion of text which spans several lines, the "Y" command positions the cursor back to the beginning of the inserted text in preparation for the "PJ" (paragraph justify) command.

INSERTION AND DELETION COMMANDS

The insertion commands described in this section perform character insertion according to their numeric arguments. The insertion takes place at the current cursor location. The value of @S is updated to the negative of the length of the inserted text.

Since two of these commands require the user to input an ASCII code, CRT/EDIT provides an "ASCII look up" feature.

> The two-character sequence "x (x = any keyboard character) may be used as a numeric argument. The value of the argument is the ASCII code for the character x.

<arg>I	Insert the single character whose ASCII code is <arg>
<arg>TC	Replace with the character whose ASCII code is <arg>

<arg> may not be defaulted

These commands are used to place a single character in the edit buffer. The "I" command inserts a character at the cursor location; the "TC" command deletes the character at the cursor location before performing the insertion.

These commands may be used to enter control characters, such as <ESC>, which may not be included in a string argument. For example,

"<ESC>I inserts <ESC>

A null (ASCII code 0) may not be inserted with this command, since CRT/EDIT uses it to mark the end of the edit buffer.

<arg><CTRL-U> Insert the number <arg>

default value of <arg>: 1

Inserts the value <arg> at the current cursor location.

<arg>Qstring<ESC> Delete everything until ~string~

default value of <arg>: 1

Deletes all characters from the current cursor location up to, but not including, the <arg>th next occurrence of ~string~ in the edit buffer. If the search is successful, all characters between the cursor location and ~string~ (but not including ~string~) are deleted.

CONDITIONAL REPLACEMENT COMMANDS

```
{  <arg><CTRL-B>string1<ESC>string2<ESC>            Conditional change }
{                                                 through remainder of page }
{  ------------------------------------------------------------------------ }
{                                                                           }
{  <arg><CTRL-V>string1<ESC>string2<ESC>            Conditional change }
{                                                 through remainder of file }
{  ------------------------------------------------------------------------ }
```

default value of <arg>: 1

These commands allow the user to selectively change one string to another through the remainder of the page (CTRL-B command) or through the remainder of the file (CTRL-V command).

With both commands, CRT/EDIT locates the <arg>th next occurrence of ~string1~, displays it in the text window, and prompts the user with the message "TYPE A KEY" in the command window. The user may respond as follows:

<ESC>	REPLACES ~string1~ with ~string2~ and searches for the <arg>th next occurrence of ~string1~.
<CR> or <NEWLINE>	BYPASSES: CRT/EDIT does not perform the string replacement, but searches for the <arg>th next occurrence of ~string1~.
Any other key	TERMINATES the command.

CUT AND PASTE COMMANDS (TEXT REGISTERS)

CRT/EDIT supports thirty-six text registers that may be used to store portions of the edit buffer. The registers are labeled as follows:

```
#0  #1  #2  ...  #9     and     #A  #B  #C  ...  #Z
```

The buffers are dynamically allocated and may be of any size, subject to the limitations of available memory space. Text may be transferred between the edit buffer and text registers with several commands.

```
{  <arg> X register-label switch(es) C                          Copy text  }
{  ------------------------------------------------------------------------  }
{                                                                            }
{  <arg> X register-label switch(es) M                          Move text  }
{  ------------------------------------------------------------------------  }
{              default value of <arg>: 1        switches:  A and/or B  }
```

Places <arg> lines/characters from the cursor location in the labeled text register. If <arg> is negative, the <arg> lines preceding the cursor location are processed.

The "XC" command copies text only -- the edit buffer remains unchanged. The "XM" command moves the text -- that is, copies it then deletes it from the edit buffer.

The switches alter these two commands as follows:

```
switch A included        append to present contents of register
switch A not included    clear register before copying/moving text
switch B included        <arg> is number of characters
switch B not included    <arg> is number of lines
```

Examples:

```
4X#5C       clear text register 5, then copy in next 4 lines
4X#5AC      append next 4 lines to text register 5
4X#5ABM     append next 4 characters to text register 5 and delete
              them from the edit buffer
```

X register-label D — Display text

Displays the contents of the labeled text register. Pressing <CR> or <NEWLINE> resumes the edit session.

<arg> X register-label G — Get text

default value of <arg>:1

Inserts <arg> copies of the contents of the labeled text register at the cursor locations.

X register-label K — Clear register

Deletes the contents of the labeled text register.

X register-label I string<ESC> — Insert into register

Inserts the typed string into the labeled text register. @S is ~not~ updated.

NUMERIC ARGUMENTS TO COMMANDS

Many CRT/EDIT commands may begin with a numeric argument. These arguments may function as counters or as logical flags:

COUNTER ARGUMENTS are used to specify a repetition factor, a number of characters, lines, pages, etc.

LOGICAL ARGUMENTS ("true" or "false") are used to control conditional execution of commands, most notably those commands that branch to a label.

Any numeric argument may consist of any algebraic expression (add, subtract, multiply, divide) involving numbers and the argument variables described in this section. In expressions, all operators have equal precedence and operations are performed from left to right. Parentheses may be used to override the left-to-right order of evaluation.

A numeric argument consisting of only a minus sign evaluates to -1. If no numeric argument is specified, a default value of 1 is normally assumed. Default values are noted under the individual command descriptions.

The two-character sequence consisting of the double quote (") followed by any keyboard character may be used as a numeric argument. The value of the argument is the ASCII code for the character.

LOGICAL ARGUMENTS. CRT/EDIT represents all numeric arguments as integer numbers. In general, a non-zero number may be used as the logical argument "true". A zero argument is a logical "false". CRT/EDIT can also compute logical expressions using the relations "LESS THAN", "GREATER THAN", and "EQUAL", and the operators "OR", "AND", "NOT" and parentheses to group. The result of a logical expression is either "true" (represented by -1) or "false" (represented by 0).

OPERATORS FOR NUMERIC ARGUMENT EXPRESSIONS

Algebraic operators: evaluate to a signed integer	
+	signed integer addition
-	signed integer subtraction or negative number indicator
*	signed integer multiplication
/	signed integer division
Logical Operators: evaluate to -1 (true) or 0 (false)	
&	logical AND
!	logical inclusive OR
%	logical exclusive OR
'	logical NOT (complementation)
<	signed LESS THAN
>	signed GREATER THAN
=	EQUAL TO

THE NUMERIC ARGUMENT RADICES

By default, CRT/EDIT interprets numbers typed at the keyboard as decimal. That is, the "input radix" is 10. Similarly, CRT/EDIT's default radix for displaying values is decimal ("output radix" = 10).

Both the input and output radices may be changed to any value between 2 and 36. Changing the radix does not alter values already stored by CRT/EDIT -- it only alters the way in which the editor interprets numeric arguments and displays numeric values.

<arg>TI	Set input radix to <arg>
<arg>TO	Set output radix to <arg>
default value of <arg>: 8 (that is, octal)	

WARNING: The value of <arg> is interpreted in the currently-defined input radix. If both the input radix and output radix are to be changed, change the output radix first to avoid confusion and unexpected results.

NUMERIC ARGUMENT VARIABLES

Numeric argument variables may be used as individual numeric arguments or in algebraic or logical argument expressions. There are several types of numeric argument variable:

* CRT/EDIT MAINTAINED VARIABLES: Certain values pertaining to the current editing situation are maintained automatically be CRT/EDIT:

@C	The current character number. (First character in edit buffer is numbered 0)
@L	The current line number. (First line is numbered 0)
@T	The ASCII code of the character at the current cursor location.
@I	The number of pages read so far from the input file. (Also displayed on editor status line)
@O	The number of pages written so far to the output file. (Also displayed on editor status line)
@S	LENGTH OF INSERTED/SEARCH STRING: After a search command, @S is updated with the relative character position of the other end of the string with respect to the cursor -- for a forward search, @S gets a negative value; for a backward search, @S gets a positive value. After an insert or change command, @S always gets a negative value.
?	THE ERROR FLAG: set to -1 ("true") if an error has occurred set to 0 ("false") if an error has not occurred This flag may be used in an argument expression immediately after one of the following commands is executed.

Command	Error Condition
S, N	search string was not found within search range
C, V	change string was not found within search range
M, L	cursor was ordered to move past beginning or end of edit buffer
D, K	deletion was ordered past beginning or end of edit buffer
R	end of input file OR no input file open

* THE NUMERIC ARGUMENT STACK: A series of values may be "pushed" and "popped" during execution of a single command string or command file. The stack is cleared before the next command string is executed.

@P	The value on top of the numeric stack. This value is popped off the stack, making the next value available.
@Q	The value on top of the numeric stack. This value remains on the stack.

Values are "pushed" onto the numeric argument stack with the following command:

<arg>TP — Push <arg> onto numeric argument stack

default value of <arg>: 1

Pushes <arg> onto the last-in-first-out (LIFO) stack.

* USER/COMMAND-STRING INTERFACE: A command string or command file can pause to have the operator press a single key, or to enter a number. This number may then be used as an argument or in an argument expression.

@K	The ASCII code of the key struck by the user. When CRT/EDIT encounters this argument, it prompts the operator with "TYPE A KEY", and waits for a key to be pressed.
@N	The number entered by the user. When CRT/EDIT encounters this argument, it prompts the operator with "TYPE A NUMBER", and waits for a number to be entered. The number may be one or more digits long, terminated with <CR> or <NEWLINE>. Pressing <CR> alone enters the number zero.

* THE NUMERIC REGISTERS: CRT/EDIT supports thirty-six number registers. Each register may store a single integer between -32768 and +32767. The registers are labeled as follows:

#0 #1 #2 ... #9 and #A #B #C ... #Z

The contents of a number register may be used in a numeric argument expression by typing the two-character label. Two commands manipulate the numeric registers:

<arg>Tregister-label — Store value in numeric register

default value of <arg>: 1

Sets the value of the labeled register to <arg>. The pound sign (#) may be omitted from the label of the registers with number labels. Thus, both of the following commands store the value 23 in numeric register #5:

(20+3)T#5 46/2T5

<arg>TD — Display values of all numeric registers

default value of <arg>: 1

The text window is cleared and the values of all 36 numeric registers displayed. The value <arg> is displayed at the top of the screen. Pressing <CR> or <NEWLINE> resumes the edit session.

BRANCHING AND LABELS

Command execution may be altered from the normal left-to-right sequence using conditional branch commands and labels. A label is a two-character pair of the form "\x", where x is any character except <ESC>. Labels should only be placed between commands, not within commands. If CRT/EDIT encounters a label without having been commanded to branch there, it ignores the label. If more than one label with the same character exists, only the first one is recognized.

<arg>;x — Branch to the label " \x "

default value of <arg>: -1 (true)
x may be any character except <ESC>

If <arg> is missing or non-zero (that is, "true"), control branches to the location labeled "\x". If <arg> = 0, execution continues at the next command. If no label "\x" exists, the error message "MISSING LABEL" is displayed.

Examples:

1. #9;aTB\a Sounds the bell if number register 9 contains 0.

2. The following command string computes the factorial of a number the user specifies, and inserts the number in the edit buffer. The command string is broken onto several lines for clarity. CRT/EDIT permits this, since it ignores <CR>s between commands.

Command String	Explanation
@NT#C	Have user imput number; put it in register C.
1T#P	Set register P to 1.
\a#C=1;b	Define label "a", if value in register C =1, go to b.
#C*#PT#P	Set register P to (register C * register P).
#C-1T#C	Decrement register C.
;a	Go to label "a".
\b#P^U	Define label "b", insert register P value in edit buffer.

ERROR SUPPRESSION

Normally, when an error occurs in CRT/EDIT operation, an error message is displayed and command execution terminates. For command string and command file execution, however, error termination may be suppressed in many cases. In addition, the error condition can be detected and used to alter the flow of command execution.

E — Suppress error termination

Suppresses error termination for the command immediately following. The error flag (see below) is updated, and may be used as a numeric argument.

THE ERROR FLAG. CRT/EDIT maintains an error flag, a numeric argument variable named "?", which stores the current error condition:

* If an error has occurred, then ? = -1 (true)
* If an error has not occurred, then ? = 0 (false)

In particular, the error flag may be tested just after a command for which error termination has been suppressed. In this way, the user may change the flow of command execution if an error has occurred. See "Branching and Labels" above. The error flag is cleared (set to 0) at the beginning of every command.

Example:

Sto the victor<ESC>Cboils<ESC>spoils<ESC>

Command execution terminates before changing "boils" to "spoils" if "to the victor" is not found.

ESto the victor<ESC>Cboils<ESC>spoils<ESC>

With the search error condition suppressed, the change command is executed whether or not the search was successful.

REPETITION BRACKETS

Repetition brackets are used to repeatedly execute a sequence of commands. Repetition brackets are used as follows:

```
<arg>[ ...commands to be iterated...  ]
```

This causes the command or sequence of commands enclosed in brackets to be executed <arg> times. If <arg> is omitted, a repetition count of 65536 is assumed. If an error occurs, command execution terminates unless error termination has been suppressed with the "E" command. Repetition brackets may be nested.

CONDITIONAL REPETITION. A right repetition bracket may be preceded by a logical argument that allows/disallows continuance of the repetition. This facility can be used to override the repetition count preceding the bracketed command string.

This argument is designed to detect an error condition when error termination has been suppressed, though any numeric argument may be used.

* If <arg> = 0 (false -- no error has occurred), repetition continues with control returning to the left backet.

* If <arg> is missing or non-zero (true -- an error has occurred), execution falls through to the command following the right bracket.

The status of error suppression is saved on entering iteration brackets and restored before each iteration. Thus an "E" command immediately preceding an iteration loop will suppress errors on each iteration.

Examples:

1. 10[CPlease enter name<ESC>Type a name, please<ESC>]

2. 10[ECPlease enter name<ESC>Type a name, please<ESC>?]

Both of these command strings change the next ten occurrences of "Please enter name" in the edit buffer to "Type a name, please". With the first command string, CRT/EDIT displays a "MISSING" error message when it cannot find any more occurrences of "Please enter name". With the second command, string the "E" command suppresses this error message, and the "?" argument causes the repetition to end if an error occurs (that is, there are fewer than ten occurrences of "Please enter name").

COMMAND FILE EXECUTION

FXfilename<ESC> — Execute the contents of a file as a command string

This command allows the user to store CRT/EDIT commands in disk files and execute them as if they had been entered at the keyboard. The user may pass information from the keyboard to an executing command file by including the @K or @N arguments in the command file.

To return control back to the keyboard, the executed file must contain one or more occurrences of the command:

<arg>FX<ESC><ESC>

If <arg> is missing or non-zero, command file execution terminates and control returns to the keyboard. If <arg> = 0, command file execution continues at the next command.

MISCELLANEOUS COMMANDS

TN — Complement contol character suppression flag

Suppresses interpretation of all cursor movement keys and all other control keys except for <CR>, <NEWLINE>, <DEL>, and <ESC>, which are essential to command entry. This command allows the control characters to be included in string arguments. To restore interpretation of the control characters, the "TN" command is issued a second time.

TB — Ring Bell

Rings the bell on the terminal. This command is useful for notifying the operator after CRT/EDIT has executed a long command string.

Text Formatting Commands

<arg>TF — Set the text window size

default value of <arg>: 8

The text window will consist of <arg> lines on either side of the line containing the cursor. See the description of the text window in Chapter 1.

TT — Complement line-truncation mode

In line-truncation mode, only the first 79 characters of a line are displayed. Without line trunctation, long lines are displayed in their entirety, a dash (-) in column 80 acting as a continuation character. See the description of the text window in Chapter 1.

COMMAND SUMMARY

```
<arg>A                          Append <arg> lines from the input file

     B                          Go to beginning

<arg><CTRL-B>string1<ESC>string2<ESC>  Conditional change on current page

<arg>Cstring1<ESC>string2<ESC>         Change string on current page

<arg>D                          Delete <arg> characters

     E                          Suppress error termination

<arg>FAfilename<ESC>            Output <arg> lines to existing file

     FDtemplate<ESC>            Delete all files matching a template

<arg>FIfilename<ESC>            Insert <arg> lines from disk file

<arg>FLtemplate<ESC>            List all files matching template

<arg>FOfilename<ESC>            Output <arg> lines to new file

     FRfilename1<ESC>filename2<ESC>         Rename file

     FXfilename<ESC>            Execute file as a command string

     GCI                        Close the input file

     GCO                        Close the output file

     GRfilename<ESC>            Set the input file to filename

     GWfilename<ESC>            Define an output file

     GX                         Abort an edit pass

     H                          Terminate CRT/EDIT

     Istring<ESC>               Insert a String

<arg>I                          Insert the character with ASCII code <arg>

     J<ESC>                     Start a new edit pass

     Jnewfilename<ESC>          Rename and start a new edit pass
```

<arg>K	Delete <arg> lines
<arg>L	Move <arg> lines
<arg>M	Move <arg> characters
<arg>Nstring1<ESC>	Serach for string in remainder of file
Ofilename<ESC>	Open a file
PJ	Paragraph justify
<arg>Qstring<ESC>	Delete everything until string
<arg>R	Read next page
<arg>Sstring1<ESC>	Search for string on current page
TB	Ring Bell
<arg>TC	Replace with character with ASCII code <arg>
<arg>TD	Display values of all numeric registers
<arg>TF	Set the text window size
<arg>TI	Set input radix to <arg>
TN	Complement contol character suppression flag
<arg>TO	Set output radix to <arg>
<arg>TP	Push <arg> on numeric argument stack
<arg>TS	Set search terminator
TT	Complement line-truncation mode
<arg>TW	Set paragraph justification width
<arg>Tregister-label	Store value in numeric register
Ustring<ESC>	Replace inserted string
<arg>Vstring1<ESC>string2<ESC>	Change string through remainder of file
<arg><CTRL-V>string1<ESC>string2<ESC>	Conditional change string through remainder of file
W<ESC>	Close a file
Wfilename<ESC>	Close a file and rename

```
     X register-label D                 Display text
<arg>X register-label G                 Get text
     X register-label I string<ESC>     Insert into register
     X register-label K                 Clear register
<arg>X register-label switch(es) C      Copy text
<arg>X register-label switch(es) M      Move text
     Y                    Move to opposite end of inserted string
     Z                    Go to end
<arg>;x                   Branch to the label " \x "
<arg><CTRL-U>             Insert the number <arg>
```

NUMERIC ARGUMENT VARIABLES

@C	The current character number. (First character in edit buffer is numbered 0.)
@I	The number of pages read so far from the input file. (Also displayed on editor status line)
@K	The ASCII code of the key struck by the user. When CRT/EDIT encounters this argument, it prompts the operator with "TYPE A KEY", and waits for a key to be pressed.
@L	The current line number. (First line is numbered 0)
@N	The number entered by the user. When CRT/EDIT encounters this argument, it prompts the operator with "TYPE A NUMBER", and waits for a number to be entered. The number may be one or more digits long, terminated with <CR> or <NEWLINE>. Pressing <CR> alone enters the number zero.
@O	The number of pages written so far to the output file. (Also displayed on editor status line)
@P	The value on top of the numeric stack. This value is popped off the stack, making the next value available.
@Q	The value on top of the numeric stack. This value remains on the stack.
@S	LENGTH OF INSERTED/SEARCH STRING: After a search command, @S is updated with the relative character position of the other end of the string with respect to the cursor -- for a forward search, @S gets a negative value; for a backward search, @S gets a positive value. After an insert or change command, @S always gets a negative value.
@T	The ASCII code of the character at the current cursor location.
?	THE ERROR FLAG: set to -1 ("true") if an error has occurred set to 0 ("false") if an error has not occurred This flag may be used in an argument expression immediately after one of the following commands is executed.

Command	Error Condition
S, N	search string was not found within search range
C, V	change string was not found within search range
M, L	cursor was ordered to move past beginning or end of edit buffer
D, K	deletion was ordered past beginning or end of edit buffer
R	end of input file OR no input file open

"x (x = any character)	The ASCII code for the character x.

OPERATORS FOR NUMERIC ARGUMENT EXPRESSIONS

Algebraic operators: evaluate to a signed integer	
+	signed integer addition
-	signed integer subtraction or negative number indicator
*	signed integer multiplication
/	signed integer division
Logical Operators: evaluate to -1 (true) or 0 (false)	
&	logical AND
!	logical inclusive OR
%	logical exclusive OR
'	logical NOT (complementation)
<	signed LESS THAN
>	signed GREATER THAN
=	EQUAL TO

CURSOR MOVEMENT KEYS

KEY	CURSOR MOVEMENT	EQUIVALENT COMMAND
<f1>	Back 4 lines	-4L
<f2>	Back 1 line	-1L
<f3>	Ahead 4 lines	1L
<f4>	Ahead 1 line	4L
<f5>	Back 4 characters	-4M
<f6>	Back 1 character	-1M
<f7>	Ahead 4 characters	1M
<f8>	Ahead 1 character	4M
<f15> or <BLANK KEY>	Beginning of edit buffer	B
<ERASE EOL>	End of edit buffer	Z
<HOME>	Other end of current line	1L-1M or 0L
<LEFT ARROW>	Back 1 character	-1M
<RIGHT ARROW>	Ahead 1 character	1M
<UP ARROW>	Back 1 line	-1L
<DOWN ARROW>	Ahead 1 line	1L

SPECIAL SEARCH CHARACTERS

<CTRL-@>	Matches any single character in the edit buffer.
<CTRL-P>	Matches any string in the edit buffer, including the null string.
<CTRL-N>	Acts as a "not" prefix to the following character. The two-character sequence <CTRL-N>~char~ matches any character except ~char~.