CROMEMCO CDOS TEXT EDITOR

CROMEMCO INCORPORATED

2400 Charleston Road, Mountain View, California

```
************
INTRODUCTION
************
```

The Cromemco CDOS Text Editor enables one to create, edit, and save
ASCII text or program files. The CDOS Text Editor is very versatile in
that it can be used to manipulate and edit text on a line, word, or
character basis. Characters and words can be inserted in, deleted from,
or changed within a line of text. The point of change can be chosen
to be the beginning or end of the text, the beginning or end of any line,
or the beginning or end of any word. Insertions and deletions can be
made that cover more than one line of text. The Text Editor is not
encumbered by line numbers or other extraneous information, and operates
using only the text itself as a guideline to changes.

```
==============================
General Procedure of Editing
==============================
```

The usual procedure for creating a new text file is to call the Text
Editor (hereafter called the Editor), enter text from the system console
in Insert Mode, edit or change the text using String Mode, and then
output the text file to a diskette for storage. The procedure for
editing an existing file is exactly similar; however, prior to
editing a usual step is to input the old text from a diskette file.
These procedures are described in detail in the following pages ;
calling the Editor is described under GETTING STARTED.

```
==================================================
Definitions of Important Terms and Symbols
==================================================
```

Memory Requirements -

The CDOS Text Editor occupies approximately 8600 bytes of RAM memory
within the resident routines of CDOS, which require about 5500 bytes.
The remaining RAM memory is available as work space for the Editor.
This amounts to the following number of bytes (ie, number of characters)
for these systems: 2300 for 16K system, 18700 for 32K system, 35100 for
48K system, and 51500 for 64K system. Although not required, a 32K
system is probably the minimum acceptable for most editing.

Text Buffer -

The text buffer is the work space as well as the storage space of the
Editor. All entered text is kept here during an editing session, and all
existing text to be edited must first be transferred here from the disk
before changes can be made. The size of the text in the buffer varies,
increasing as text is entered, and decreasing as text is deleted (see V
command for an explanation of how to determine the remaining space in the
buffer). When the buffer is empty, the beginning and end of it coincide.

Character-Pointer -

The character-pointer is used to locate the position in the text
buffer where editing is to occur. The character-pointer (or pointer)
is in most cases positioned BETWEEN characters (ie, after one
character and before the next). However, the pointer can also be
positioned immediately before the first character in the text buffer (the
"top" of the buffer) or immediately after the last character in the buffer
(the "bottom" of the text buffer). In Insert Mode (or under the S or G
commands) text is placed into the buffer at a point immediately
preceding the character-pointer.

The character-pointer can be moved to any position within the text
buffer. Any command attempting to move the pointer past the boundaries
of the text buffer is terminated when the pointer reaches the boundary,
even though the command specified may not have been comleted. An
exception to this are the -nP command, which will continue to re-type
the top page of the buffer the number of times specified (unless a key
is pressed to abort the command) and the Macros, <>, for which execution
is carried out the specified number of times even after the buffer
boundary is reached (although no changes are made in the text after the
boundary is reached). This can also be aborted by pressing any key.

The character-pointer can be moved by characters, words, or lines,
and accordingly, the Editor can be used for "character editing," "word
editing," or "line editing." A line of text in the buffer is a string
of characters having a line feed as its last character. A word of text is
a string of characters having a space, tab, carriage return, or line
feed as its last character. The next character in the text buffer
immediately following the line feed, <LF>, is in the next line and will
be so displayed on the system console. If no <LF> characters are used,
the entire text is considered to be one line. When one enters text, the
Editor automatically generates a <LF> following a carriage return, <CR>,
whenever the <CR> key is depressed. Thus, if you want to delete a
carriage return, you must delete two characters to eliminate the line
feed also.

Current Character
Current Word
Current Line -

The current character, word, or line is that which currently con-
tains the character-pointer. Note that if the pointer is following the
<LF> of one line and immediately preceding the first character of the
next line, that next line will be the one considered the current line.

Character Editing
Word Editing
Line Editing -

The term character editing applies when you are using the C and D
commands for deletion and pointer positioning. The term word editing
applies when you are using the J and X commands. The term line
editing applies when you are using the L and K commands. The S and
I commands may be used in all three cases for text insertion.

Default Value -

This is used through-out the following command descriptions to
indicate the assumption that is made if a given quantity is omitted.
For example, the statement, "The DEFAULT VALUE of n is 1," means
that the value of n is assumed by the Editor to be 1 if it is
omitted from the command.

Forward
Bottom
Down -

These terms are synonymous and mean to move through the text buffer
towards its end. This can be a "forward" movement of the pointer, "down"
through the buffer, or towards the "bottom."

Backward
Top
Up -

These terms are synonymous and mean to move through the text buffer
towards its beginning (the point at which you would begin entering text
for a NEW FILE). This can be a "backward" movement of the pointer, "up"
through the buffer, or towards the "top."

<>

These brackets are used in the examples in the following material to
set off ASCII characters or other quantities which are user-typed
but cannot be easily represented otherwise. For example, the symbol
<CR> stands for a user-typed carriage return. Note that you would
NOT type the four individual characters: <, C, R, and >. Note that these
symbols are also used for Macros; this use will always be so labeled.

+/-

This is the printed representation in what follows of the familiar
symbol, "plus-or-minus."

n

This prefix is part of the mnemonic for a large number of the Editor
commands. However, it would never be typed in actual use; it is instead
a symbol for the integer values:

        n = 0, 1, 2, 3, ... 65535

Note that n is ALWAYS positive. If the particular command allows
negative values of n, the command mnemonic will be written: +/-n?.

↑

This symbol is echoed back to the console by the Editor and stands
for the word "Control-." That is, this symbol together with the letter
immediately following it stand for one of the Control Characters. For
example, the symbol "↑Z" stands for "Control-Z." Note that the symbol
for the <ESC> or escape key is "↑[."

#

This is a USER-ENTERED character and stands for the maximum number
of times for carrying out any function on the Editor:

        # = 65535

It may be used with any of the commands of the form: n? or +/-n? in place
of the "n," and used in this way will cause the command to execute until:
(a)it is executed 65535 times, (b)the top or bottom of the buffer is
reached, (c)the command can no longer be carried out successfully.   An
example is:

        *B#T<CR>

which will first move the character-pointer to the top of the buffer and
then type out all the lines of the present file.

****
GETTING STARTED
****

The CDOS Text Editor is contained in a file named EDIT.COM on the CDOS
system diskette.   It is called by entering the filename as a CDOS
command.   The syntax of the EDIT command is:

        EDIT <filename.ext>

where "filename" is either the 8-letter name of a new file to be created
and edited or the name of an existing file to be edited, and "ext" is
its 3-letter extension if there is one.   When a new file is created, the
Editor prints the message:

        NEW FILE

at the start of the editing session.

There is another way of calling EDIT which is useful for simultaneously
editing and transfering a file to another diskette. The format of this is:

        EDIT X:FILENAME Y:

where X and Y each stand for one of the disk drives, A, B, C, or D.   Note
that the file "FILENAME" is initially on drive X, and is to be transferred
to drive Y following the edit.   Also NOTE that X is the drive containing
the sourcefile and Y is the drive which will contain the output file, and
that this is exactly opposite the convention used in CDOS for XFER.
The file "FILENAME" will be found (after editing) on drive Y with the same
name it had on drive X.

Command Syntax

The Editor signals its readiness to accept commands by prompting with
an asterisk (*).   Commands can be entered in upper or lower case; for
example, the Type-out command can be "T" or "t."

Commands can be entered singly or in a command string. The commands
entered in a command string are executed in the order they are entered.
A command string is terminated and executed with a carriage return;
some commands may be terminated but not executed by an ESC or ↑Z (S and
F commands for example). They will be executed when the <CR> is pressed.

****
Control Characters
****

Following are descriptions of the Control Characters used by the CDOS
Editor. Each character will be represented by the symbol, ↑?, where
the ? stands for the character name. This symbol, ↑?, stands for
"Control-?" or, on some consoles, "CTRL-?". Under each Control
Character are two headings: String and Insert, explaining the control's
function in each of these two modes. Note that Control Characters
are called "controls" in the following to distinguish them from
the "commands," which are those responsible for text re-arrangement
and insertion. Also, the differences between "normal" and "alternate"
Insert Modes are explained under the I command.

Control-A

String - The ↑A has no effect in this mode; it is simply echoed to
the console.

Insert - The ↑A is a very convenient feature of "normal" Insert Mode.
When pressed at any time, it serves to type out the previous 23 lines
or "page" of text above the character-pointer. The pointer's position
is unchanged by this, and further text may be inserted following the
issue of the ↑A. However, this control does not work in "alternate"
Insert Mode, that where the inserted text is all on one line (see
explanation under Insert Mode).

Control-C

String - The ↑C restarts CDOS, which means that all text in the
text buffer is lost. This generally would not be used to get out
of EDIT unless a system problem had occurred (insufficient RAM, etc.).
The Control-C has no effect if issued during a type-out of text. It
may or may not abort other Editor commands while they are in progress, but
it is not advisable to use it for this purpose because of the danger of
losing the contents of the buffer.

Insert - The ↑C has no effect in "normal" Insert Mode other than to be
entered as a character of text. It does, however, have the restart CDOS
effect in "alternate" Insert Mode. Be careful of this difference.

Control-E

String - The ↑E is a physical carriage return--line feed. That is, it
has the effect of returning the console cursor to the left side of the
screen WITHOUT having any effect on entered text.

Insert - The ↑E has no effect in Insert Mode other than to be entered
as a character of text. Note, however, that some console terminals
will provide an automatic ↑E at the end of a full line while text
is being entered; this is NOT entered as a <CR><LF> in the text itself.
Thus, it's possible to have the entire contents of the buffer entered
on "one line," ie, as a single character string without carriage returns.

Control-H

String & Insert - The ↑H has the same effect in both modes: a backspace
without character-echo. On some terminals a BACK key is actually present
and serves to implement the ↑H. Other terminals have a RUB or DEL key;
the Editor will also accept these as backspace even though the ↑H is not
used for this. Note that the DEL or RUB key will work in upper- or lower-
case, and consequently, the <underline> key cannot be used with the
Editor. Hence, any of the following may be used to backspace, depending
on the console terminal used: ↑H, BACK, RUB, DEL, <underline>.

Control-I

String - If ↑I is given in a string of commands, it will produce an
error message. If it is given WITHIN a command, however, such as an
F or S command, it will be treated as a <tab> character.

Insert - The ↑I in this mode is a <tab> character. Some terminals
may have an actual TAB key which will implement the ↑I also. A
↑I is stored in the text buffer as a single character. The system
accepts this character to generate a sufficient number of spaces
on the console to position the next character at the next tab location.
Tab stops are located every eight character positions across a line
on the console. Note that the <tab> will NOT generate a single
character if the Z command has been used to change the spacings.
See Z command for a further explanation of this.

Control-L

String - The ↑L is a logical carriage return--line feed. That is, if
it is used in a string as part of the F, N, or S commands, it will be
treated as the two characters, <CR> and <LF>. This is necessary
because the actual carriage return key has the effect of terminating
the F, N, or S commands. For example, suppose one wanted to find every
time that two <CR><LF>'s had been entered consecutively, and sub-
stitute one <CR><LF>. The user would type:

        *B#S↑L↑L↑[↑L<CR>

Insert - The ↑L is also a logical carriage return--line feed in
"alternate" Insert Mode and is used as described above (part of

a string after the I command). In "normal" Insert Mode, however,
the ↑L is inserted into the text as the character itself and
NOT as a <CR> and <LF>. This is a very useful feature, because
the ↑L character is accepted by most printers as a form-feed.
Thus, you can insert the ↑L into the text every 56 lines, for
example, which will advance the paper at the bottom of each printed
page. Note that ↑L, ↑E, and ↑C are the only of the Control Character
commands which can actually be stored in the text as those characters.
Any random control characters which are not commands may also be stored
in the text.

Control-P

String & Insert - The ↑P is used in both these modes to turn the printer
on and off. When the printer has been turned on, everything typed
out on the console screen will also be printed. If it is later no
longer desired to have printout, another ↑P should be typed. The ↑P works
by a "toggle" action, ie, if the printer is on when ↑P is pressed, it will
go off, and if it was off, ↑P will turn it on. Thus, typing an even
number of Control-P's (say two) will leave the printer as it is.

The ↑P should NOT be used with systems not having a printer as it may
cause the system to hang up; if one ↑P is typed accidentally, be sure to
type another one immediately BEFORE any other characters are typed.
Should another character be typed, the Editor will go into a "wait" for
the printer to be ready. Since no printer is connected to the system, it
can never be ready; hence, the system gets hung up and the contents of
the buffer will be lost in restarting CDOS.

Control-R

String & Insert - The ↑R is used in both modes to retype the current line
being displayed on the console screen. In Insert Mode this is a line of
entered text; in String Mode it is the present command string. No char-
acters are sent to the text buffer and Control-R may be typed as many
times in succession as is desired. It is most useful if several
Control-X's have been given in the line and it has become hard to read.

Control-S

String & Insert - The ↑S is used in both modes to start and stop the type-
out on the console screen. This would occur after issuing a ↑A in Insert
Mode, or after issuing either a P or T command in String Mode. This is
a most useful control; for example, you might give the command string
"B#T" in String Mode which would type out all the current contents of
the text buffer. The Control-S would be used as often as needed to stop
the type-out so that parts of it could be read. Like the ↑P, the ↑S works
by a "toggle" action; note that it does not abort the type-out, it merely
causes it to pause. A type-out may be aborted by tapping any key (other
than ↑S) during its progress.

Control-U

String - The ↑U is used to delete the current line on the console. In
String Mode this means deleting the current command string. This is a
quicker way of eliminating a long command string than ↑H would be.

Insert - The ↑U is also used in Insert Mode to delete the current line on
the console. It differs, however, from String Mode in that it is used to
delete the current line of inserted text, ie, the line back to but not
including the last previous carriage return and line feed. If the <CR> and
<LF> are then deleted by using ↑H, the next previous line may be deleted
using Control-U, and so on.

Control-X

String - The ↑X is used to delete commands in a command string WITH echo.
One character will be deleted for each time ↑X is pressed and it will
proceed in a last-typed, first-deleted manner. The first time ↑X is
pressed it displays 3 "\" (backslashes) to demark the beginning of the
deletions, and the first character you type following the release of the
Control-X will display 3 "/" (forward-slashes) to demark the beginning
of correctly entered text.

Insert - The ↑X is used in an exactly similar way in this mode except
that it is inserted text which is deleted rather than part of a command
string. The 3 "\" followed by the 3 "/" are also used here to mark the
beginning and end of the deletions. Notice that the difference between
↑X and ↑H (or RUB) is that the Control-X will echo the deleted characters
to the terminal so that you can see how far back you have deleted.

Control-Z or ESC Key (↑[)

String - The ↑Z and ESC keys (Escape Key echoed to the terminal as "↑[")
are exactly equivalent in their actions for the Editor. The ↑Z is used in
String Mode as a delimiter for the F, N, and S commands. These commands
are terminated by a carriage return; however, sometimes it is desired to
terminate them (or a part of them) in another way. Consider the following
example using Macros to find the first three occurrences of the word
"Editor" and type the line containing each one:

        *B3<FEditor↑ZOTT>

where the ↑Z is used as a delimiter for the string which is to be found.
Note that the following string using ESC is exactly equivalent:

        *B3<FEditor↑[OTT>

Insert - The ↑Z is used in a slightly different way in Insert Mode as
a delimiter. In "normal" Insert Mode the ↑Z (or ESC) is the key pressed

to return to String Mode.   The character-pointer will remain where it is:
at the end of the just inserted text.   In "alternate" Insert Mode the ↑Z
(or ESC) is used as a delimiter just as it was for the F, N, or S
commands.   Suppose in the previous example it was desired to pluralize
the first three occurrences of "Editor":

        *B3<FEditor↑ZIs↑ZOTT>

where the ↑Z was used as a delimiter for both the F and I commands.   Note
that no carriage return was needed to terminate the I command since the
↑Z was used as a delimiter.

RUB or DEL Key

(See Control-H for an explanation of this function.)

****
COMMAND DESCRIPTIONS
****

++++++++++++
Insert Mode
++++++++++++

I - Insert Text Before Character-Pointer

The I command is the only one which can be used to insert text into
the text buffer using Insert Mode.   (See R, G, and S commands for
means of inserting text using String Mode.)   The usual format for the
I command has been called "normal" Insert Mode in this manual.   This
format is:

        I<CR>

after which the user may begin entering text.   Note that as you enter
text, carriage returns may be used as may any other character.   Then
when all the desired text is entered, you leave Insert Mode by pressing
either Control-Z (↑Z) or Escape (<ESC> key which is echoed as ↑[).
If the I command is preceded by any prefix other than part of a valid
command string, it is either ignored or produces an error message.   There
is a special form of the I command (called "alternate" Insert Mode in
this manual) which is used to insert text on one line only:

        I<this is inserted text><CR>

This form will not require a ↑Z or <ESC> to return to String Mode.
(There are other differences between "normal" and "alternate"
Insert Modes; see Control Characters for a description of these.)
Instead the <CR> both returns the user to String Mode and also inserts
the <CR> and <LF> at the end of the inserted line.   You would not use

this form to insert one or several words at the beginning of an existing
line, however, as it would insert an undesirable carriage return. If
the character-pointer is in the middle of a line when the I command
is given, the Editor types the portion of the current line before
the pointer, and then enters Insert Mode.   The user may then use the
<DEL> or <RUB> key as though he/she had just typed all the characters
displayed.   Thus, the I command may be used to delete previous lines
(without returning to String Mode) by "backing up" over lines that are
already there.   It is important when using the I command to position the
pointer BEFORE going into Insert Mode so that the inserted text
will be positioned where desired.   The text will be inserted before the
character-pointer position in the text buffer; the following example
will illustrate this further.   Suppose we have the text:

         LINE 1
         LINE 3

and wish to insert the words, "LINE 2. " The character-pointer must first
be positioned before the "L" in "LINE 3":

         *BFLINE 3↑[OLT<CR>
         LINE 3

where the F command is used to find the line desired, and the OL command
is used to locate the character-pointer at the beginning of that line.
The user now makes the following insertion:

         *ILINE 2<CR>
         *-2TT<CR>

and the typed-out text appears as:

         LINE 1
         LINE 2
         LINE 3

The user might also have typed out the first two lines of this text by
issuing a Control-A, <↑A>, while still in Insert Mode (see Control
Characters for a further explanation of this).   Most of the control
characters will work in Insert Mode as well as String Mode; the
differences are explained in the Control Characters section.

Note that if you are in Insert Mode and completely fill the Text
Buffer, the Editor automatically returns control to String Mode
and further commands will NOT have any effect unless they are
one of the write commands: W, E, Q, H, or O.   Following writing
out some of the buffer, you may return to Insert Mode.

++++++++++
String Mode
++++++++++

The following commands may all be used in String Mode. They are
sub-divided into logical sub-headings. For example, the B, C, J, and L
commands are all used to reposition the character-pointer and are thus
grouped under one sub-heading, Character-Pointer Positioning Commands.

==================
Type-out Commands
==================

T - Type a Specified Number of Lines on the Console

The T command is used to type back the entered text so one can make
sure it was entered correctly. The format of the command is

        +/-nT

The command nT causes text to be typed from the current position of the
character-pointer forward to the nth carriage return; the command -nT
causes text to be typed from n lines before the current line (containing
the pointer) up to the position of the pointer. The default value of
n is 1, which causes typing from the pointer to the end of the current
line (if the pointer is located at the beginning of a line, this causes
the entire line to be typed). The command OT causes typing from the
beginning of a line up to the pointer. It may be seen from this that
the command string "OTT" causes the current line to be typed regardless
of the position of the character-pointer within it and without moving
the position of the pointer. Note that if the pointer is in the middle
of a line and the command T is issued, the results will be printed on
the console beginning in column 1, even though that is not the way
it is stored in the buffer (ie, an entire line is still stored). Pressing
any key during a type-out will abort the command and return control
to the Editor with a prompt (see also Control-S).

P - Move and Print a Specified Number of Pages

The P command is used to type back the printed text in blocks of 23
lines each called pages. The format of the command is

        +/-nP

The command nP causes the character-pointer to move down the text buffer
(n x 23) lines and begin typing there, typing out 23 lines. The command
-nP causes the character-pointer to move up the buffer (n x 23) lines and
type out 23 lines. The command OP does not cause the pointer to move;
hence, this command is used to type out the current page (ie, the 23 lines
beginning with and immediately following the pointer). The default value
of n is 1; giving successive P commands will cause the text to be typed
out page by page.

An example of the use of the P command command is:

        *BOP#P<CR>

which causes the entire contents of the buffer to be typed out by pages
beginning with the first line. After each page is displayed there is
about a 1 second pause before the next page is typed. During this time
the type out can be stopped using a Control-S (see Control Characters).
The type-out can be aborted (as opposed to stopped) at any time by hitting
any character on the keyboard. If the command -P is given just upon
coming out of Insert Mode, the last 23 lines of inserted text will be
typed. 23 lines was chosen because this is the standard size of a
page which can be displayed at one time on most CRT's; persons having
a 12-line CRT will have to move by half-pages. For example, the command
string:

        *B12tOP#<-12LP>

is suitable for typing out an entire body of text by 12-line pages (see
Macros). Also, type-out for the P command is aborted upon reaching the
lower boundary of the text buffer; if it is not immediately aborted at
the upper boundary, you may stop the type-out by hitting any key.

+/-n - Move Character-Pointer and Type a Line (+/-nLT)

The +/-n command is a convenient combination of two other commands:
the L and T commands. Its format is simply: +/-n, which is exactly
equivalent to the command string "+/-nLT;" however, unlike this string
the +/-n command cannot be preceded or succeeded by any other command
characters. In other words the +/-n must be the ONLY command in
the string; if it is not, it will either be ignored or will produce an
error message. The default value of n is 1; thus, the command "-<CR>"
will type the line immediately preceding the line containing the
character-pointer; the command "<CR>" will type the line immediately
following that containing the pointer. The command "O<CR>" will type
the current line. (There may be slight variations in this if the pointer
is not located at the beginning of a line.) From the above it can
be seen that it is possible to step through the text buffer line by
line, examining each one in turn, simply by hitting successive
carriage returns. It is also possible to skip through the buffer
with little typing by successively issuing a command such as:
"20<CR>" (see also L and T commands).

```
================================================
```
Character-Pointer Positioning Commands
```
================================================
```

B - Move Character-Pointer to Beginning or Bottom of Text Buffer
----

The format for this command is

        +/-B

The (+)B command moves the character-pointer to immediately before the
first character in the buffer.   This might be useful for the following:
(1)defining a starting point for typing out the whole text buffer con-
tents, (2)setting a reference point for counting lines of text or search-
ing for a word or phrase, (3)inserting text before text already in the
buffer.   The -B command moves the character-pointer to immediately after
the last character in the buffer, and is useful for adding text at the end
of the present text in the buffer.   The -B command may also be used with
the W or N commands if it is desired to write out the present buffer
and replace it with more of the input file (see W and N commands).

C - Move Pointer Over One or More Characters

The C command will locate the pointer within a line and is useful for
character editing.   The format of the command is

        +/-nC

The command nC causes the character-pointer to move forward n
characters from its present position; the command -nC causes
the pointer to move back n characters.   The default value for n is 1.
n=0 is undefined for this command (ie, the character-pointer does
not move).   Remember that the pointer always resides BETWEEN
characters and that its traditional location while doing line editing
is before the first character of the current line.   The C command treats
the <CR><LF> at the end of every line of edited text the same as any other
two characters and thus the pointer may pass from line to line if
n is large.

J - Jump the Character-Pointer Forward or Backward Over Words
----

The J command jumps the character-pointer over a specified number of
words.   The format of the command is

        +/-nJ

The command nJ moves the pointer to just before the first character of
the nth word after the current word.   The command -nJ moves the pointer
to before the first character of the nth word preceding the word now con-
taining the pointer.   The command 0J moves the pointer to the beginning

of the current word (ie, the one containing the pointer). The default
value for n is 1. The delimiters defining the ends of words are: <space>,
<tab>, <LF>, and <CR>. When the pointer jumps to a new word, it includes
the delimiter as the last character of the word being jumped. Thus, the
command string "JT" would cause the word following the current word to be
typed, beginning in the first column of the screen. Also, it's important
to note that any two delimiters following each other also constitute a
word. Thus, the <CR> and <LF> at the end of every line of edited text
will constitute a word. The J command (along with the associated X
command) is most useful for word editing. As with the C and L commands
the J command is aborted if it reaches either boundary of the text buffer.

L - Move Character-Pointer Forward or Backward Over a Number of Lines

---

The L command moves the character-pointer a specified number of lines
forward or backward. The format of the L command is

        +/-nL

The command nL advances the character-pointer to the beginning of the nth
line following the current line; the command -nL moves the pointer to the
beginning of the nth line preceding the current line. The default value
of n is 1, moving the pointer to the beginning of the line immediately
following the current line; a -L moves it to the beginning of the
immediately preceding line. The command OL moves the pointer to the
beginning of the current line. Note that the L command always
places the pointer at the BEGINNING of a specific line, thus making
it most useful for line editing. Also, if the value specified
for n is too large, the character-pointer will stop at the top or
bottom of the text buffer.

+/-n - Move and Type a Line
P - Move and Print Pages

These two commands will also reposition the character-pointer. They are
described in detail under the Type-out Commands section.

===========================
Deletion of Text Commands
===========================

D - Delete Characters from Text Buffer

The D command deletes a specified number of characters from the text.
The format ot the D command is

        +/-nD

The command nD causes deletion of n characters following the character

pointer.   The command —nD causes deletion of n characters preceding
the character-pointer.   The default value for n is 1.   n=0 is undefined
for this command.

K - Kill Lines of Text

The K command deletes lines of text.   The format of the K command is

        +/-nK

The command nK causes text to be deleted from the current position of
the character-pointer forward to and including the nth carriage return
and line feed.   The command -nK causes deletion to start from n lines
before the beginning of the current line (ie, the line containing the
character-pointer) and continues until the character-pointer is
reached.   The command OK causes text deletion from the beginning of
the current line up to the character-pointer.   The default value for
n is 1, which causes deletion from the character-pointer of the rest
of the current line.   Note that if one wishes to delete the last portion
of a line, a <CR> (and <LF>) must be re-inserted as the K command will
remove the one already there.

X - Delete Words of Text

The X command deletes a specified number of words from the text.   The
format of the command is

        +/-nX

The command nX deletes the portion of the current word which lies after
the character-pointer plus the first n-1 words after that.   The command
-nX deletes the n words before the pointer plus the portion of the
current word which lies before the pointer.   The command OX deletes
only the portion of the present word which lies before the pointer.
The default value of n is 1.   Thus, a word of text might be deleted by
positioning the pointer before the first letter of the word and giving
the command "X," or by positioning the pointer after the space
immediately following the word and issuing the command "OX."   The
delimiter for a word is a <space>, <tab>, <CR>, or a <LF>; thus, dele-
tion of a word always includes the space immediately following the word.
Also, one delimiter immediately followed by another constitutes a word by
itself.   See J command for more information on the definition of a word.

=================================
Search and Substitute Commands
=================================

F - Find a Character String in the Text Buffer

The f command is used to find a string of characters in the text buffer. The format of the command is

        +/-nF<textstring>↑[(or <CR>)

The command nF will find the first n occurrences of the "textstring" AFTER the present position of the character-pointer, and will reposition the pointer immediately following the nth occurrence. The command -nF will search in the other direction, ie, BEFORE the present pointer position, and will reposition the pointer immediately following the nth occurrence, counting UPWARD, of the "textstring". n=0 is undefined for this command, and may produce an error. The default value for n is 1. If the search can be completed only m times, where m is a number >= 0, the character-pointer is left at the end of the mth occurrence of the "textstring" and the Editor prints the message:

        CANNOT FIND "<textstring>"

If m is 0, ie, if the search is unsuccessful in all cases, the pointer is left at its position prior to the issue of the command.

It is wise to verify that the search has been successful by typing out the line prior to making any modifications on text near the just-found string. The "textstring" can often appear in unusual places; for example, suppose you have the line of text:

        This is, however, the first day we
                                        will be able to go.

and you desire to remove the incorrect <LF> by using the command string:

        *Fwe↑[DOLT

The resultant line printed out would be:

        This is, howeer, the first day we
                                        will be able to go.

The F command string is limited to 128 characters including the "textstring", and if more than this are used, the command is automatically terminated without completion. Remember to use a ↑L instead of <CR><LF> within the "textstring".

N - Find Next Occurrence in File

The N command may be used to search through an entire file for a parti-
cular character string. However, it may also be used to search for the
null string, which causes half the text buffer space to be filled with
text from the input file. This is really an I/O feature; the N command
is therefore described in detail under the I/O Commands section.

S - Substitute One Character String for Another

The S command is used to substitute one character string for another
in the text buffer. The format of the S command is

        +/-nS<oldtext>↑[<newtext>↑[(or <CR>)

The command nS looks AFTER the present location of the character-pointer
for the nth occurrence of the character string called "oldtext", removes
the n strings, and replaces each of them with the string called "newtext".
The substitution is made only if the search is successful. The command
-nS searches BEFORE the present location of the pointer for the nth prev-
ious occurrence of the string "oldtext" and substitutes the string
"newtext" n times. Note that this means that the substitution is made
one time in each of n places.   n=0 is undefined for this command,
and may produce an error. The default value for n is 1. The
character-pointer following a successful substitution is left at
the end of the string "newtext"; if the search is not successful, the
pointer is left where it was just prior to the issue of the command.
If the substitution is made some arbitrary number of times, say m (where
m may be 0), and the string "oldtext" can no longer be found, then the
last n-m substitutions are ignored, and the Editor prints the message:

        CANNOT FIND "<oldtext>"

Note that the appearance of this message does not necessarily mean that
no substitutions occurred, but only that the (m+1)th substitution did
not occur. It is usually a good idea to print back the line on which
the substitution was made to be sure it was done in the correct place.
For example, the command string issued might be:

        *SOLD↑[NEW↑[OTT<CR>

The total length of the S command including "oldtext" and "newtext"
should not exceed 128 characters; more than this will cause the command
to terminate without completion. If "newtext" is omitted, the "oldtext"
string is found and deleted; thus the S command may be used for word
deletions. Keep in mind that the string searched for by the S command
must exactly match the string specified in the command, including
upper and lower case, punctuation, etc. Remember to use a ↑L to repre-
sent a <CR><LF> in a string of characters within the S command.

=================================
I/O Commands from/to Input File
=================================

A - Append Lines to End of Buffer

The A command is one of two commands (see also N command) used to
bring text from the input file into the text buffer. The format of the
A command is

        nA

With this command lines of text are appended to any text already present
in the text buffer. (The A command will be aborted, however, once the
buffer becomes filled.) Each iteration of this command causes one line
of text to be read into the buffer. The character-pointer position is
NOT changed by A commands. n=O is undefined for this command.

E - End EDIT, Close Files, and Return to CDOS

The E command writes the contents of the text buffer to the output
file and exits to CDOS. Any prefix to E is ignored or produces an
error message (unless E is preceded by other commands in a command
string); the format is simply: E. Also note that even if the entire
input file was not loaded into the text buffer, E will write it
through the text buffer in its entirety onto the output file. The
E command is the one normally used to end an editing session. Note
that E creates a BAK file only after the second time a file is edited;
a new file will be returned to the disk without a backup file.

H - End EDIT and Reopen at Head
O - Obliterate Buffer and Reopen Input File

These two commands are useful for saving edited work on the disk (H) or
restarting EDIT (O). However, because of their many valuable features,
they are described separately in the EDIT Restart Commands section.

N - Find Next Occurrence of a Character String in Input File

The N command is similar to the F command; however, it automatically
appends and writes lines as the search proceeds. The format of the
command is

        nN<textstring>↑[(or <CR>)

The command nN causes the Editor to search for the nth occurrence of
"textstring" and to reposition the character-pointer immediately after

the last character of that string. It will search the entire source-file if necessary. The command works in the following manner: The text buffer is half-filled with text from the input file, and this text is then searched for the "textstring". If it is not found, the entire contents of the buffer are then written out to the disk, and the Editor again proceeds to half-fill the buffer with text and search that. This continues until all the text has been searched and the entire input file has been written into the buffer and then written to the output file. The search can proceed only in a forward direction because of the read and write operations; therefore, a negative or zero prefix to the N command will produce an error message.

When half-filling the buffer, the command will ALWAYS print the full line up to and including the <CR><LF> immediately following the event of the buffer becoming half-filled. It will thus find the next occurrence of any "textstring" which is all on one line. It is not advised, however, to search for strings which lie on two separate lines because of the possibility that one of them will be read in one read operation, and the other will be read in the subsequent read operation.

There is a special form of the N command which is most useful for reading a file into the text buffer. This form is

        *N<CR>

This causes the Editor to search for the null string. Initially, before any text is present in the buffer, this means that half the buffer space will be filled with text and then control is returned to the user. This is because any character will satisfy the null string and thus that character is found when anything is present in the buffer. The advantage of this form of the command over the A command is that you can always be sure of filling the buffer at least half-full, but also be sure of not over-filling it. The command #A is useful for very short files, but there is the danger of completely filling the text buffer with large files.

To read more of the input text into the buffer after having edited some of the first portion, the command is issued: "-BN<CR>" which moves the pointer to the end of the buffer before again searching for the null string. If this was not done, the N command alone would stop upon reading the first character it encountered.

Q - Quit EDIT With No File Changes and Return to CDOS

The Q command can be used to terminate an editing session in progress without doing any output. Any prefix to the command is either ignored or produces an error message (unless Q is preceded by other valid commands in a command string); therefore, the format is simply: Q. The input file is unchanged following a Q command; note, however, that NO output file is created even if lines of text had been written out of the text buffer prior to the Q command. The Q command is therefore very powerful in its effect and is usually invoked only just after entering EDIT if one decides

not to continue. Also note that the Q command will leave no BAK
file on the disk regardless of whether or not one was present just
prior to entering the Editor; it leaves only the original sourcefile.
If Q is issued after having entered EDIT with a new file, the new
filename and sourcefile will not be left on the disk.

R - Read File from Disk Into Text Buffer

The R command is used to read files from the disk into the text buffer
without leaving EDIT. This prevents having to exit from the Editor to
concatenate or transfer files. The format of the R command is:

        R<filename.ext>

where filename is the 8-letter disk filename, and ext is the 3-letter
extension (if any) to that filename. Note that there can be no space
between the "R" and the filename. Any prefix to the R command is ignored
or produces an error message (unless part of a valid command string).
The text which is read in is always placed immediately AFTER the char-
acter-pointer; thus, the pointer must be positioned prior to issuing an
R command. If only part of a particular file is desired, the entire
file must be first read in and then the undesired portions deleted
out. Be careful not to exceed the text buffer space; if in doubt,
first write out some portion of the buffer, and then read in the file.
Any file may be read including BAK files EXCEPT the BAK file for the
file which is presently being edited. Also note that to follow an R
by other commands in a command string, a ↑Z or ESC (↑[) must be typed
following the filename as a delimiter.

There is a special form of the R command for reading files from any of
the other disk drives. This form is

        #RX:<filename.ext>

where X stands for any of the disk drives A, B, C, or D. This feature
is very time-saving as files can be read from other drives without
leaving EDIT.

W - Write Lines Before Pointer to Output File

The W command is used to write out a number of lines from the text
buffer to the output file. The format of the command is simply: W;
any prefix to the W command is ignored or causes an error indication
(unless W is preceded by a valid command string). Text is written
to the output file from the beginning of the text buffer through the
character immediately preceding the character-pointer, even if the
pointer is in the middle of a line. If only part of any given line
is written out, the remainder will be written out correctly by
any other write operation such as E or H. Note that one must
take care to position the pointer so that the text one wishes

to be written out actually does get written.   After a write the
remaining lines (if any) are moved to the top of the buffer freeing
more space at the bottom for insertions.   If one wishes to use the
W command to write out a specific number of lines, one should first
use the +/-n command to locate the character-pointer at the line
number, and then issue a W command.   If one writes out more lines
than intended, it is possible to return to the beginning of the file
by issuing an H command followed by locating the pointer correctly
and re-issuing the W command (see H and +/-n commands).


```
========================
EDIT Restart Commands
========================
```

```
--------------------------------------------------------
H - End Edit, Close and Reopen Files at Head
--------------------------------------------------------
```

Any prefix to H is ignored or produces an error message (unless H
is preceded by other commands in a command string); therefore the format
is simply: H.   The H command is similar to the E command; however, you
never leave the Editor.   Its action is similar to issuing an E command,
followed by recalling EDIT using the same filename.   However, the H
command preserves both the tab settings and the contents of the Save
Buffer.   This is useful for file re-arrangement.   The original
sourcefile (before invoking H) is then retained as sourcefile.BAK,
and the edited version of it becomes the new sourcefile.   Often
during a long editing session, one may want to either view text
which has already been written onto the disk or save large sections
of text material on the disk followed by a restart at the beginning
of the sourcefile; this is the advantage of the H command: a
convenience and a safety feature for long passages of edited text.

Note that the ORIGINAL sourcefile.BAK is not retained.   Also,
following an H command no text will remain in the text buffer; thus,
either the A or N commands must be invoked to reinsert text into
the buffer.   Note that these can be put in the same command string
as the H command.   Keep in mind the difference between the H and O
commands: the H command reopens the file which has just been
edited as a NEW sourcefile; the O command reopens the ORIGINAL
sourcefile as though no editing had been done (they BOTH preserve
the Save Buffer and tab settings; see O command for an example
of using this for file re-arrangement).


```
--------------------------------------------------------
O - Obliterate Contents of Text Buffer and Reopen Input File
--------------------------------------------------------
```

Any prefix to an O command is ignored or produces an error message
(unless O is preceded by other valid commands in a command string);
therefore, the format is simply: O.   The O command is quite similar to
the H command; control never leaves the Editor.   Its action resembles
issuing a Q command followed by recalling EDIT using the same filename.
However, there are three things saved by the O command which would not

be saved by the previous actions: (1)any text already written out prior
to issuing the O is saved on the output file, (2)the contents of the Save
Buffer are preserved, (3)the tabs remain set as they were prior to in-
voking the O command.   These features give one a great capacity for file
re-arrangement.   For example, it can be used to move a block of text
from the end to the beginning of a sourcefile too large to fit entirely
into the text buffer.   First, the user would read in the sourcefile,
killing all lines above the block of text which is to be moved.   When the
desired block is in the buffer, it may either be put into the Save Buffer
or be written directly to the output file using the W command (if it
should go at the beginning of the new file).   Then the O command is
issued, the sourcefile is read in from the beginning and written to the
output file in back of the block already there, taking care to kill the
block of text which still remains at the end of the text.   Variations
on this scheme may be used for other types of file re-arrangement.

Also, note in the above that following an O command no text will remain
in the text buffer; thus, either the A or N commands must be invoked
to reinsert text into the buffer.   Note that it is most convenient to
put such a command into the same command string as the O command.   For
example, the command string "ON<CR>" will first obliterate the buffer and
rewind the input file, and will then refill half the text buffer with
text from the input file.   Keep in mind the difference between the
H command and the O command: The H command reopens the file which
has just been edited; the O command reopens the original sourcefile.
They both kill all portions of the file in the buffer which are not in
the Save Buffer; however, only the O command preserves the text which
has been written to the output file.   Due to the hidden power of the
O command to destroy one's editing work, it is never executed without
asking the user if it was intended by printing the message:

          OBLITERATE TEXT BUFFER AND "REWIND" INPUT FILE (Y/N)?

If one responds with an "N", control continues as though the O command had
never been given (ie, the prompt is issued); if one responds with a "Y",
control continues with the O command.

=====================
Save Buffer Commands
=====================

G - Get the Contents of the Save Buffer

The G command is used to get the contents of the Save Buffer and insert
them into the text buffer just preceding the character-pointer.   The for-
mat of the command is simply: G; any prefix to G is either ignored or pro-
duces an error message (unless part of a valid command string).   For exam-
ple the incorrect command "2G" has the same effect as the command "G"; the
"2" is ignored.   However, the command "GG" would have the effect of dupli-
cating the Save Buffer twice in succession in the text buffer.   The Save
Buffer contents are not destroyed by a G command, and thus, the Save Buf-
fer may be accessed as many times as desired.   This feature may be used
if there are a large number of identical or almost identical lines to
be inserted into the text.

Following a G command, the character-pointer position is not changed
and it will be at the end of the inserted text.   Note that one cannot
insert a portion of the Save Buffer; a single G command inserts the entire
contents.   Upon first starting an editing session in EDIT, the Save Buffer
is cleared.

Y - Put Lines of Text into the Save Buffer

The Y command is used to save text in the Save Buffer.   The format is:

        nY

A negative or zero prefix to Y produces an error message.   The nY
command is used to save the portion of the current line following the
character-pointer plus the following n-1 lines.   The default value
for n is 1.   Note that the lines to be saved must always follow the
pointer; thus, the pointer will have to be positioned beforehand.   Each
time a Y command is given, the contents of the Save Buffer are cleared
as the new lines are written.   Thus, it is not possible to save lines suc-
cessively (ie, using two Y commands) without first writing out the con-
tents of the Save Buffer using the G command (see G command).   Also, note
that the Save Buffer contents are preserved by the O command but not by
the H command.   See O command for an explanation of how this can be used
for file re-arrangement.

The Save Buffer is limited in size to 2048 characters.   An attempt
to write more characters than this will cause the Editor to issue
the message:

        SAVE BUFFER FULL

and abort the Y command at the boundary of the Save Buffer.   Thus, there
may be only a portion of the last line written into it.   A G command
can be issued (after properly positioning the character-pointer), and
the remainder of lines to be transferred can then be written out to
the Save Buffer.

=========================
Miscellaneous Commands
=========================

U - Upper Case Translate

The U command is used to guarantee that all text entered will be in
ASCII upper-case; only alphabetic characters are affected, however.   The
format of the command is

        +/-U

Any prefix to this is ignored (unless an algebraic sign or part of a valid
command string preceding U), and the sign of U is taken as the last
preceding sign in the string.   For example, the command string "-7U"
would be interpreted as the command -U.   The command "OU" is also inter-
preted as -U, but all positive integers are interpreted as (+)U.

Upon entering EDIT the U command is always de-selected (ie, it is as
though -U has just been issued); this means that lower-case ASCII is
entered as lower-case and upper-case ASCII is entered as upper-case. Now
if the U command is given, all text will be stored as upper-case in the
text buffer regardless of how it is entered. The Editor does, however,
echo the characters just as they are typed, so it is not until a type-out
that the upper case is seen. For example, if one enters the following:

```
*U<CR>
*Ithis is an example<CR>
*-T<CR>
```

the Editor then responds by typing:

         THIS IS AN EXAMPLE

The -U command is then used to return to no-translation mode. BE
SURE to issue the -U if you no longer want all text to be entered
in upper-case. Note that the U command will have no effect when used
on terminals which are not capable of producing lower-case; they
are effectively locked into upper-case mode at all times.

V - Verify Remaining Text Buffer Space

The V command computes and displays on the console the amount of work
space remaining in the text buffer for use by the Editor. The format
is simply: V; any prefix to V is ignored (unless part of a valid
command string). The Editor responds with the message:

         ROOM FOR XXXXX MORE CHARACTERS IN BUFFER

where XXXXX is a number between 0 and 65535 (each byte of memory holds
one character). If you are editing a large file and are running out
of work space, you can write the first part of the buffer to the output
file using the W command.

Z - Set Tabs

The Z command is used to pre-arrange the insertion of a number of
spaces, which will be called by the Control-I command, known as a "tab."
This is similar to the tab setting function of a typewriter. Upon
entering the Editor, the tabs are set at multiples of 8 spaces (0, 8,
16, 24, etc.), which are the standard spacings for entering assembly
language code. Note: these spacings are stored internally as a
SINGLE ASCII tab character and NOT as 8 spaces. However, resetting
the tabs by means of the Z command inserts an equivalent number of
spaces. This may be undesirable for large programs, as a large part
of the text buffer could then be occupied by spaces. The format of the
Z command is simply: Z; any prefix is ignored (unless part of a valid
command string preceding Z). The command is then self-explanatory,
typing out instructions and numbering the columns of the console screen.
Note that even though the columns are numbered only to 63, the actual

line length may be as long as 128 characters, and control returns to
the Editor if one spaces past this length. If one desires to return
again to the standard single-character spacings, the Z command
should be re-issued. In response to the "?" the user should type:

        ?S<CR>

which will restore the standard tabs (note that the upper-case S ONLY
will carry-out this function). Any ASCII character is sufficient to set
the tabs including the <space>. (See also Control-I command.)

****
MACROS OR COMMAND ITERATIONS
****

The Macro Command symbols, <>, allow the user to group EDIT commands
together for repeated evaluation. The form of the command is

        n<command string>

where "command string" is any valid sequence of Editor commands; this
string will then be executed n times, where n may also be the "#" symbol
to stand for 65535 times. A "O" preceding the Macro is defined to be the
same as the "#"; a negative value for n is considered positive. The
default value for n is 1, which is the same as issuing the command string
without a Macro. Unlike the S or F commands, the Macro will not abort
when it can no longer be executed; thus, if the "#" symbol is used, it
will attempt to execute the command string 65535 times. To abort it,
simply press any key. An example of the use of Macros is the following
in which you can make changes in 5 labels used in an assembly language
program. If the original labels are:

            SUBT1:   DW   OH
            SUBT2:   DW   8H
            SUBT3:   DW   10H
            SUBT4:   DW   18H
            SUBT5:   DW   20H

and you issue the Macro:

        *B5<SSUBT↑[ADD↑[>

where the B is used to move the pointer to the beginning of the buffer
prior to executing the Command Iteration. The resultant substitutions
would leave the text:

            ADD1:   DW   OH
            ADD2:   DW   8H
            ADD3:   DW   10H
            ADD4:   DW   18H
            ADD5:   DW   20H

Command Iterations can be nested up to 25 deep. Any attempt to nest
Macros more than 25 deep is trapped and the error message is printed
on the console:

        MACROS NESTED TOO DEEP

An example of the use of nested Macros is to organize columns of data
into more readable fashion. Suppose you have the following columns
of data:

            105.298349817
             98.209370128
            107.242475023
             87.584394879
             94.729398289
            101.239797957
            102.885097399

You might issue the following Macro command:

        *B7<F.↑[2<3CI ↑[>>

which would result in the following columns of data:

            105.198 349 817
             98.209 370 128
            107.242 475 023
             87.584 394 879
             94.729 398 289
            101.239 797 957
            102.885 097 399

Macros are also very useful in conjunction with the Conditional commands
which are described in the following section.

****
CONDITIONAL COMMANDS
****

The Conditional Commands allow the user to conditionally execute commands
in the command string. The format of the commands is

        =<text string>↑[
        /<text string>↑[

The symbols used for conditional commands are '=' and '/'. The '=' stands
for equal and '/' stands for not equal. Any prefix number to '=' or '/'
is ignored, the command is executed only once. The entered text string
is compared to the text following the character-pointer. If the two
strings are identical, the character-pointer is moved to after the string,
otherwise the character-pointer remains unchanged. If the condition is
met, the next instruction in the command string is executed. If the
condition is not met, all commands up to the end of the Macro or the

end of the line are skipped over. An example of using conditionals
follows. Suppose you wish to change all '*' which begin a line to semi-
colons. The command string would be

        B#<<=*↑[-DI;↑[>L>

When this string is executed, it will start at the beginnig of the text
buffer. If the first character is an '*', then it is deleted and a semi-
colon is inserted. Because of the nested macros, the character-pointer
is moved to the next line, whether or not the condition was met.

CDOS Text Editor Command Summary

The following is an alphabetical list of the CDOS Editor commands. The mnemonic source of the command letter is underlined in the description.

Command	Description	Page
nA	Append lines from input file to end of text buffer...	18
±B	Move character-pointer beginning or bottom of text buffer.	13
±nC	Move pointer over one or more characters.	13
±nD	Delete characters from text buffer.	14
E	End edit, close files, and return to CDOS	18
±nF	Find a character string in the text buffer	16
G	Get the contents of the Save Buffer and insert them into the text.	22
H	End edit, close and reopen files at head	21
I	Insert text before character-pointer.	9
±nJ	Jump the character-pointer forward or backward over words.	13
±nK	Kill lines of text.	15
±nL	Move the character-pointer forward or backward over a number of lines	14
nN	Find next occurrence of a character string in input file.	18
O	Obliterate contents of text buffer and reopen input file at head.	21
±nP	Move and print a specified number of pages.	11
Q	Quit edit with no file changes and return to CDOS.	19
R	Read file from disk into text buffer	20
±nS	Substitute one character string for another.	17
±nT	Type a specified number of lines on the console.	11

Control Character Summary

The following is an alphabetical listing of the CDOS Editor control characters. Under "Modes", 'I' means Insert Mode and 'S' means string Mode.

Control	Modes	Description	Page
CTRL-A	I	Type page preceding character-pointer..	5
CTRL-C	I,S	Restart CDOS and lose edited text......	5
CTRL-E	S	Physical carriage return-line feed; not sent to text buffer.	5
CTRL-H or DEL key	I,S	Backspace without echoing character. .	6
CTRL-I	I,S	Generate a "tab" character or equivalent number of spaces.	6
CTRL-L	I,S	Logical carriage return-line feed; these are generated by EDIT and inserted into text.	6
CTRL-P	I,S	Turn printer on or off; a toggle action.	7
CTRL-R	I,S	Retype current line.	7
CTRL-S	I,S	Stop or start type out on console; a toggle action.	7
CTRL-U	I,S	Delete current line.	8
CTRL-X	I,S	Rubout characters with echo.	8
CTRL-Z or ESC Key	I,S	Used as delimiter for F,N,R, and S commands in a command string.	8