PMcJ

REFERENCE MANUAL
Q. E. D.
TIME-SHARING EDITOR

D. C. Angluin
L. P. Deutsch

Document No. R-15
Revised March 26, 1968
Contract SD-185
Office of Secretary of Defense
Advanced Research Projects Agency
Washington, D. C. 20325

TABLE OF CONTENTS

1.0 Introduction 1-1
1.1 A Summary Description 1-1
1.2 Command Mode, Formats 1-1
1.3 Notation and Conventions 1-2
2.0 Addressing Text 2-1
2.1 Legal Addresses 2-1
2.2 Type Address Commands 2-3
2.3 Examples for Section 2.0 2-3
3.0 Printing Text 3-1
4.0 Saving Text on Files 4-1
5.0 Destroying, Creating and Changing Lines 5-1
5.1 Deleting Lines 5-1
5.2 Adding Lines 5-1
5.3 Changing Parts of Lines 5-3
5.4 Control Characters for Text Input 5-4
6.0 Substitute 6-1
7.0 String Buffers 7-1
7.1 Loading, Deleting, and Printing Buffers 7-1
7.2 The Uses of String Buffers 7-3
8.0 Mode and Tab-Setting Commands 8-1
8.1 Quick/Verbose Mode 8-1
8.2 Ignore Characters Mode 8-1
8.3 Tab-Setting 8-2
9.0 Returning from QED and Panic Messages 9-1
9.1 The Normal Return 9-1
9.2 Rubout 9-1
9.3 Panic Messages 9-2

APPENDIX A: Index of Control Characters

APPENDIX B: Index of Commands

1.0 Introduction

1.1 A Summary Description

QED is a rather powerful program for editing symbolic text which runs under the 930 time-sharing system. Its input and output are symbolic files which can also be handled by the executive COPY command. It has extensive facilities for inserting, deleting and changing lines of text, a line edit feature, a powerful symbolic search feature, automatic tabs which may be set by the user, and thirty-six buffers. Text can be read from any file and written onto any file. A replace command permits all occurrences of a specified string of characters to be replaced with another string.

1.2 Command Mode, Formats

To enter QED from the executive type QED. (underlined portions are typed by the user.) QED will type a * and the user may type any of the commands described in this manual. When the operation of a command are terminated and QED is ready for another command, it again types *; at this point QED is waiting for commands and is said to be in "command mode."

The formats of the various commands are of three basic types:

1. A single (non-alphanumeric) character (possibly preceded by line address(es)), for example: / and ↑, which requires no confirming dot, but rather begins its specified function as soon as the character is typed.
2. A command word (possibly preceded by line address(es)), for example: TABS and MODIFY, which require a confirming dot before the specified function is begun. In typing the command word, the user only types the initial letter, then QED recognizes the command and types the rest of it. The user may then type a dot (.) to confirm the command, or any other character, in which case the command will be aborted.

3. Mavericks, namely: READ FROM, WRITE ON, SUBSTITUTE and the buffer commands (LOAD, GET, JAM, BUFFER and KILL) each of which is specified by typing its initial letter (QED types the rest) but otherwise has a unique format (for fuller explanation see individual descriptions).

For a list of all commands recognized in command mode (and the locations of their descriptions) see Appendix B.

1.3 Notation and Conventions

A character with a following superscript C is a control character and is typed by pressing the control key together with the key for the basic character, for example, D^C is control D and is typed by holding down the control key and pressing the key for D. Control characters do not have any printing characters normally associated with them; to emphasize that nothing is printed when they are typed, in the examples they are enclosed in parentheses (which, apart from the function of emphasizing this fact, should be ignored). In certain contexts, QED will print some character when a control character is typed (e.g., # for B^C, ↑ for A^C, \ for W^C). This is noted in the individual descriptions of the control characters, section 5.4. Also, when a control character is part of the text being input from output to the teletype, QED will represent C^C as &C (where C is some printing character), for example, $\uparrow^C$ will be typed as &↑.

'C' is used throughout this manual to indicate a single (arbitrary) character. 'A', 'B' and 'I' are described in under text addressing, section 2.1. Apart from these special symbols, (in the examples and format-specifications) lower case letters, parentheses, and underlining are used for comments, explanation, and variable structures (always specified more fully in their descriptions); and all other characters are literally typed by QED or the user in specifying or executing the operation concerned.

2.0 Addressing Text

The text being edited is held in a single buffer, called the main text buffer. It consists of a series of lines delimited by carriage returns (CR's). The line is the only addressable unit of text.

2.1 Legal Addresses

Lines may be addressed in the following ways:

1. By decimal numbers. The first line is numbered 1.
2. By . (called 'dot'), which refers to the current line. The value of . is changed by many of the editor's commands, as noted below.
3. By $, which always refers to the last line in the main text buffer.
4. By labels. The structure :text: typed in command mode causes a search for the indicated text at the beginning of a line and followed by a character which is not a letter or a digit. When typing in the text, A^c, W^c, Q^c and V^c have their usual functions (see section 5.4). The text may contain any characters, may be a maximum of 30 characters long, and is terminated by a colon (:). The search begins with the line after the current line, and cycles to the beginning of the buffer if it runs off the end. Buffer 0 is loaded with the text searched for. If the search succeeds, . addresses the line where the label was found, and the value of :text: is also this line. (Note that QED does not type any response specifically to indicate that a search succeeded.)

   If there is no line with the specified label in the main text buffer, QED types ? and restarts command input. (In this case . is not changed.)

   The search may be begun at line A where A is any legal address), instead of at .+1, by typing A:text: For example, .:ABC: begins the search at the current line.

5. By arbitrary text. The structure [text] causes a search for the specified text anywhere in the main text buffer. The text may contain any characters (and arbitrarily many of them) and is terminated by]. The search proceeds in the same way as the label search (4), and has the same effect when it succeeds or fails. A[text] starts the search of line A, as in label search.
6. By a legal address followed by +, - or space, followed by another legal address. This construction has the obvious meaning with the following qualifications:
   a) Space is treated as +
   b) A+B, A B, or A-B where A is any legal address and B is a search is equivalent to AB, i.e., the search for B is begun at A and if it is successful, . and A+B (or A B or A-B) address the line where the search succeeded.
   c) After a search has been given in an address, . and $ may not be added to or subtracted from the address being constructed. E.g., :text:+$, [text]-., and :text:+9-. are not legal addresses.
   d) There can only be one occurrence of . or $ in a given address. E.g., $-., .+., $+$-15 are not legal addresses.

Throughout the remainder of this manual, the symbols A and B will represent any legal line addresses. Also, the symbol I will indicate that a single line address, A, may be given; or a pair of line addresses separated by a comma (thus: A,B), where the second address is at least as great as the first. In this latter case, the interval is inclusive, i.e., A,B specifies lines A through B. (A,B where B is less than A is treated as an illegal parameter.) Or, one may type @, which addresses all the lines in the main text buffer. (And is thus equivalent to 1,$.)

Generally, if no line addresses are typed before a command, the interval is taken to be just the current line. For example: *DELETE. is equivalent to *.DELETE. and */ is equivalent to *./ . Exceptions to this are:

*APPEND. , which is equivalent to *$APPEND.
*READ FROM, which is equivalent to *$READ FROM
*WRITE ON, which is equivalent to *1,$WRITE ON

If an address being typed in is deemed to be illegal, QED types ? and restarts command input. When a command is given, negative line addresses are converted to 1, and an address greater than that of the last line is treated as an illegal parameter, i.e., QED types ? and restarts command input.

(Note: examples for text-addressing and type address commands are in section 2.3.)

2.2 Type Address Commands

To facilitate text addressing, there are two commands which convert between absolute line numbers and symbolic addresses.

A= (where A is any legal address) causes QED to type the line number of the line addressed by A.

A← causes QED to type the "symbolic address" of the line addressed by A. (That is, the label of the last line preceding A which has a label, followed by a decimal number indicating its displacement from A. (The label part of the address is enclosed in colons)). For this command, any line which begins with something other than a carriage-return or a blank has a label, and the label typed out will terminate with the first character after the initial one which is not a letter or a digit. For example, the label typed out for the line *AB**C will be :*AB:, and that for **AB*C will be :*:

2.3 Examples for Section 2.0

It is assumed that the reader is familiar with the command / which types out the line or lines addressed.

```
*@/
FIRST LINE
SECOND
     THIRD
FOURTH!!LINE
##FIFTH
*1/
FIRST LINE
*.=1
*.←:FIRST
*$=5
*$/
##FIFTH
*:SECOND:=2
*.=2
*:FOURTH:/
FOURTH!!LINE
*:FOUR:?
*:FOURTH!:/
FOURTH!!LINE
*:FOURTH!!:?
*:FOURTH!:←:FOURTH:
*[#]=5
*[H!!L]=4
*[ST]=1
*.←:FIRST:
*[FT]←:#:
*[INE]=1
*.=1
*[INE]=4
*[INE]=1
*.[INE]=1
*2[INE]=4
*5[INE]=1
*4[INE]=4
*:FIRST:[INE]=1
*1[IRST][IRD][INE]=4
*:FIRST::SECOND:[IRD][FI]←:#:
*.=5
*.1/
?
*1/
FIRST LINE
*.+2/
     THIRD
*.←:SECOND:1
*.+2=5
*. 2←:#:
*$-2=3
*
```

3.0 Printing Text

As text is kept "out of sight" in the main text buffer, the user must explicitly direct QED to type out a line or lines for his inspection.

There are four basic print commands (in each, the value of . is changed to the last line typed out):

I/	types out the lines(s) addressed by the interval I.
line-feed	types out the next line (i.e., .+1).
↑	types out the previous line (i.e., .-1).
IPRINT.	allows formatted printing of the line(s) addressed by I; on the next line, QED types DOUBLE? and expects the user to type Y or N (which will be completed as YES or NO, respectively) in response, to indicate whether he desires double-spacing of the text. QED then types the lines in pages of 54 (single-spaced) or 27 (double-spaced) lines; before each page, QED types a number of line-feeds, a short dashed line, and an equal number of line-feeds. Also, the last page is filled out to the size of the others and marked at the bottom with a dashed line.

Examples:

```
*1,$/
FIRST LINE
SECOND
     THIRD
FOURTH!!LINE
##FIFTH
*↑
FOURTH!!LINE
*↑
     THIRD
*
FOURTH!!LINE
*
##FIFTH
*
?
*1/
```

```
FIRST LINE
*↑?
*2,:FOURTH:PRINT.
DOUBLE? YES


-----


SECOND

     THIRD

FOURTH!!LINE


.
.
.


-----


*
```

4.0 Saving Text On Files

The following two commands enable the user to move text between QED's main text buffer and symbolic files. They accept file names in the format required by the executive.

AREAD FROM (file name). QED reads all the text from the specified file (which should be type 3, symbolic) and appends it after line A.

*READ FROM is the same as *$READ FROM.

After QED has finished reading the text, it types out the number of "words" read, (One "word" is approximately 3 characters.) and returns to command mode.

The main text buffer is not cleared before the READ.

IWRITE ON (file name). QED replaces the contents of the specified file with the lines in the interval I.

*WRITE ON is the same as *1,$WRITE ON

After QED has finished writing the text, it prints out the number of "words" written and returns to command mode.

The main text buffer is unaffected by the WRITE.

Examples:

```
*15,17/
A       BCD
12345678
EFGH
*15,17WRITE ON /TEST/.
9 WORDS.
*$=115
*READ FROM /TEST/.
9 WORDS.
*$=118                          (the lines read were appended, and
*$-2,$/                         the main text buffer was not cleared
A       BCD                     first.)
12345678
EFGH
*
```

5.0 Destroying, Creating and Changing Lines

The following sections describe the heart of QED: the commands by which the user changes the text held in the main text buffer:

5.1 Deleting Lines

IDELETE.	causes the line(s) addressed to be deleted from the main text buffer; . is set to the line before the first one deleted.

Example:

```
*1,4/                    print the first four lines
FIRST LINE
SECOND
     THIRD
LINE FOUR
*$=115                   there are 115 lines in the main text buffer.
*2,3DELETE.              delete the second and third lines.
*.=1                     . is the line before the first one deleted (#2).
*$=113                   there are 2 fewer lines in the main text buffer.
*1,2/
FIRST LINE               the old second and third lines have been deleted.
LINE FOUR
*
```

5.2 Adding Lines

After each of the following three commands -- APPEND, INSERT, and CHANGE -- QED expects the user to type in a series of lines (each terminated by a carriage-return), the whole series followed by a D^c to indicate the end of the series. QED then takes the lines so collected and puts them into the main text buffer in the position specified by the command (see individual descriptions.) (If the user does not follow the last line with a carriage-return before he types the D^c, QED will insert the necessary carriage-return.) In addition to D^c, all of the control characters described in section 5.4 are recognized; the line being edited is taken to be a null line, i.e., one containing no characters. . is changed to the last line collected.

AAPPEND. — QED expects the user to type in a sequence of lines; when D^c is typed, the APPEND is terminated, i.e., QED takes the lines collected and inserts them after line A in the main text buffer. If the address A is omitted, the collected lines will be added to the end of the main text buffer. This is the usual command for creating a body of text from scratch.

AINSERT. — QED expects the user to type in a sequence of lines; when D^c is typed, the INSERT is terminated and the collected lines are inserted before line A in the main text buffer.

ICHANGE. — QED expects the user to type in a sequence of lines; when D^c is typed the CHANGE is terminated, the line(s) addressed by I are deleted, and the collected lines are put in their place. (The interval I and the collected text need not have the same number of lines.)

Examples:

```
*1,$/                          there is nothing in the main text buffer.
*APPEND.
NEW TEXT(Dc)                   a CR is supplied by QED to terminate the line.
*:NEW:INSERT.
FIRST(CR)                      the CR does not terminate the sequence of lines.
SECOND(Dc)
*1,$/                          the collected lines were inserted before #1.
FIRST
SECOND
NEW TEXT
*2APPEND.
THIRD(Dc)
*1,$/                          the collected line was inserted after #2.
FIRST
SECOND
THIRD
NEW TEXT
*$CHANGE.
FOURTH(CR)
FIFTH(Dc)
*1,$/
FIRST
SECOND
THIRD
FOURTH                         the collected lines were inserted into the main
FIFTH                          text buffer in place of the line NEW TEXT.
*
```

5.3 Changing Parts of Lines

These two commands -- EDIT and MODIFY -- allow the user (via the control characters described in section 5.4) to change just part of a line, and thus usually require less typing than the same change made with CHANGE (which forces the user to type a whole line over to make a small change in it.) . is changed to the last line edited.

AEDIT.	QED types out line A and then expects a new line to be typed in to replace A. All of the control characters described in section 5.4 are recognized; the "old line" or line being edited is line A. In particular, CR, D^C, or F^C will terminate the EDIT, and the new line replaces the old as line A in the main text buffer.
A,BEDIT.	is a convenience permitting repeated single-line edits. Line A is typed out, and when the edit of that line is terminated (with CR, D^C or F^C), the next line (A+1) is typed out for editing. When the edit of the last line (line B) is terminated, and the new lines replace lines A through B in the main text buffer.
AMODIFY.	is exactly equivalent to AEDIT. except that the line being edited is not typed out at the beginning of the edit.
A,BMODIFY.	is exactly equivalent to A,BEDIT. except that the successive lines are not typed out before the user begins to edit them.

Examples:

(The underlined characters are assumed to be those typed by the user in the EDIT.)

```
*:CHCR:EDIT.
CHCR    SKG     =77B
CECR    SKE     =155B(CR)
*./
CECR    SKE     =155B
*.MODIFY.
RLP     CLA(CR)
*./
RLP     CLA
*3,4EDIT.
A       ZRO             old line #3.
A1      BSS     1       new line #3.
B       ZRO             old line #4.
B1      BSS     1       new line #4.
*3,4/
A1      BSS     1
B1      BSS     1
*
```

For more (and more enlightening) examples of these two commands see the examples in section 5.4 on control characters.

5.4 Control Characters for Text Input

The control characters described in this section facilitate text input. All of them are recognized (and have the functions described) in text input in the commands APPEND, INSERT, CHANGE, EDIT, and MODIFY. In addition, those marked with an asterisk (*) are recognized in the JAM INTO command, and in specifying the text in searches and SUBSTITUTE. Finally, the following two control characters (I^c and B^c) are always recognized (except immediately after a V^c):

Table of Contents for

Section 5.4

Tab and Buffer Call	Character	Page
Buffer Call	$B^{c}\underline{C}$	5-5, 7-3
Tab	I^{c}	5-5
Line Terminate		
Carriage Return	CR or M^{c}	5-5
Escape Character		
*take C literally	$V^{c}\underline{C}$	5-5
Backspace		
*one character	A^{c}	5-5
*one word	W^{c}	5-6
*one line	Q^{c}	5-7
one character (restorative)	N^{c}	5-7
Copy		
one character	C^{c}	5-8
to tab stop	U^{c}	5-8
to end of line	H^{c}	5-9
up to C	$O^{c}\underline{C}$	5-9
through C	$Z^{c}\underline{C}$	5-10
rest of line (terminate)	D^{c}	5-10
rest of line (no typing)	F^{c}	5-11
Skip		
one character	S^{c}	5-12
up to C	$P^{c}\underline{C}$	5-12
through C	$X^{c}\underline{C}$	5-13
Retype		
fast	R^{c}	5-14
aligned	T^{c}	5-14
Re-Edit		
concatenate-re-edit	Y^{c}	5-14
Mode Change		
insert/replace	E^{c}	5-15
ignore/usual	K^{c}	5-16
buffer 1/usual	L^{c}	5-17

* Recognized in APPEND, INSERT, CHANGE, EDIT, MODIFY, JAM INTO, SUBSTITUTE, and searches.

Tab and Buffer Call (Always Recognized)

B^C C — (where C is a letter or a digit) is call of string buffer C. B^C is echoed as #. Typing B^CC is equivalent to typing in the whole string of characters in buffer C. (For a full description and examples see section 7.2)

I^C — causes QED to space to the next tab stop (tab stops are set with the command TABS, q.v.). If there are no more tab stops on the line, QED types bell and takes no further action.

The Carriage Return

* M^C — is exactly equivalent to carriage-return (i.e., M^C and the CR key are two ways of typing the same character.) QED automatically supplies a line-feed. Carriage-returns serve to delimit lines of text. In addition, a carriage-return terminates editing of the current line in EDIT or MODIFY. (It does not terminate an APPEND, INSERT or CHANGE; only D^C terminates these latter operations.)

The Escape Character

* V^CC — causes the character C to be appended literally to the text being collected and disables any control function C might otherwise have. C may be any character.

Examples:

```
*:A(V^C):B:/                  V^C prevents : which follows it from terminating the label.
A:B,A RATIO
*$APPEND.
(V^C)(D^C)&D,144B(D^C)        (&D is typed by QED) V^C allows the user to enter a D^C
*$/                                  (which types as &D) without terminating the APPEND.
&D,144B
*
```

Backspace Characters

The following control characters delete one or more characters from the end of the text already typed in; all of them may be iterated. If any of these backspace characters causes the whole line currently being typed in to be deleted, QED gives a carriage return and line-feed; typing may then continue.

* A^C — QED types ↑ and deletes the preceding character. In editing, A^C does not affect the status of the old line.

Examples:

Suppose the user has typed in part of a search:

```
*[ACD
```

and then types two A^{C}'s and continues typing:

```
*[ACD↑↑BCD]/
ZABCD      EF
*
```

in this case, the first A^{C} deleted D, the second, C. Suppose the user has begun an EDIT:

```
*:NXC:EDIT.
NXC    STA      CHAR                  old line: 'TA      CHAR'
NXC    S                              new line: 'NXC     S'
```

and now types A^{C}:

```
*:NXC:EDIT.
NXC    STA      CHAR                  old line is still:  'TA      CHAR'
NXC    S↑                             new line is now:    'NXC     '
```

* W^{C} QED types \ and deletes the preceding "word". That is, all preceding blanks are deleted, and all non-blank characters up to the next preceding blank.

Examples:

Suppose the user has begun to load a buffer:

```
*JAM INTO #3.
A LINE                               ('A LINE ' has been typed)
```

and now types W^{C} and continues typing.

```
*JAM INTO #3.
A LINE \CHARACTER(D^C)
*BUFFER #3.
"A CHARACTER"
*
```

that is, the characters 'LINE ' were deleted. Suppose the user has typed:

```
*APPEND.
GC     CIO      NEWF                 ('GC     CIO      NEWF' has been typed)
```

and now types W^{C} and continues typing:

```
*APPEND.
GC     CIO      NEWF\FILENO(D^C)
*/
GC     CIO      FILENO
*
```

that is, the characters 'NEWF' were deleted.

* Q^C — QED types ←, gives a carriage return and line-feed, and deletes the line currently being typed in, or if there are no characters in the current line, the preceding line is deleted (in line-editing, the old line is restored as it was when the edit began).

Examples:

Suppose the user has begun to type in a label:

```
*:NXCH
```

and then types Q^C and continues typing:

```
*:NXCH←
PCHR:/
PCHR      GCD      MSP
*
```

that is, the characters 'NXCH' were deleted. Suppose the user starts the following insert:

```
*3INSERT.
ABC
EFG
H
```

and now types two Q^C's and continues typing:

```
*3INSERT.
ABC
EFG
H←                          line 'H' is deleted.
←                           line 'EFG' is deleted.
DEF
GHI(D^C)
*3,5/
ABC
DEF
GHI
*
```

In this case, the first Q^C deleted the line 'H', and the second deleted the line 'EFG'.

N^C — QED types ↑ and deletes the preceding character. In addition, in edit mode, QED restores the last character obliterated from the old line, if any.

Example:

If the user has begun the following edit:

```
*:NXC:EDIT.
NXC      STA      CHAR              old line:  'TA      CHAR'
NXC      S                          new line:  'NXC       S'
```

and then types N^C:

```
*:NXC:EDIT.
NXC      STA      CHAR              old line is now:   'STA      CHAR'
NXC      S↑                         new line is now:   'NXC      '
```

that is, N^C restored 'S' to the beginning of the old line.

Copy Characters

The following characters copy one or more characters from the old line onto the end of the new line. (Except in D^C and F^C) if the old line contains no more characters, or if the character to be copied to (in Z^C and O^C) does not appear in the old line, QED rings the bell (perhaps more than once) and takes no other action.

C^C — QED copies the next character of the old line onto the new and types out the character copied.

Example:

```
*:NXC:EDIT.
NXCHR      LDA      CHAR              old line:  'CHR      LDA      CHAR'
PR                                    new line:  'PR'
```

If the user now types a C^C:

```
*:NXC:EDIT.
NXCHR      LDA      CHAR              old line:  'HR       LDA      CHAR'
PRC                                   new line:  'PRC'
```

U^C — QED copies characters from the old line onto the new, up to the next tab stop, and types out the characters copied.

Example:

Suppose the tab stops are the usual ones (8,16,32,40) and the user has begun an edit:

```
*32EDIT.
AB34567890                          old line:  '34567890'
12                                  new line:  '12'
```

If the user now types U^C:

```
*32EDIT.
AB34567890                          old line:  '890'
1234567                             new line:  '1234567'
```

H^C — QED copies the rest of the old line onto the new, typing out characters typed; editing may then continue.

Example:

```
*133EDIT.
      STORE     CHR,CNT,FLG1        old line:  '  STORE    CHR,CNT,FLG1'
INIT                                new line:  'INIT'
```

If the user types H^C:

```
*133EDIT.
      STORE     CHR,CNT,FLG1        old line:  ''  (null)
INIT  STORE     CHR,CNT,FLG1        new line:  'INIT      STORE    CHR,CNT,FLG1'
```

O^C_C_ — QED copies the old line up to, and not including, the next occurrence of the character C after the next character, typing out characters copied. (C is never echoed.)

Examples:

```
*[HALLE]EDIT.
SANG HALLELUJAH!                    old line: 'NG HALLELUJAH!'
SI                                  new line: 'SI'
```

If the user now types O^CH:

```
*[HALLE]EDIT.
SANG HALLELUJAH!                    old line:  'HALLELUJAH!'
SING                                new line:  'SING '
```

If he types $O^{C}H$ again:

```
*[HALLE]EDIT.
SANG HALLELUJAH!                        old line:  'H!'
SING HALLELUJA                          new line:  'SING HALLELUJA'
```

If the user again types $O^{C}H$, QED will ring the bell and take no further action since the line beyond the next character (the next character is H, the line beyond is '!') contains no further occurrences of H, i.e. QED has already copied up to the last H of the old line.

$Z^{C}C$ — QED copies up through the next occurrence of the character C in the old line. C is echoed when it is copied, not when it is typed.

Example:

```
*[HALLE]EDIT.
SANG HALLELUJAH!                        old line:  'NG HALLELUJAH!'
SI                                      new line:  'SI'
```

If the user now types $Z^{C}H$:

```
*[HALLE]EDIT.
SANG HALLELUJAH!                        old line: 'ALLELUJAH!'
SING H                                  new line: 'SING H'
```

If he again types $Z^{C}H$:

```
*[HALLE]EDIT.
SANG HALLELUJAH!                        old line: '!'
SING HALLELUJAH                         new line: 'SING HALLELUJAH'
```

If $Z^{C}H$ is typed again, QED will ring the bell and take no further action, as there are no occurrences of H in the rest of the old line.

D^{C} — QED copies the rest of the old line onto the new, typing out the characters copied. In addition, D^{C} terminates the edit of the line in EDIT and MODIFY, and terminates text-collection in APPEND, INSERT and CHANGE (also in JAM INTO).

Examples:

```
*210EDIT.
      STORE      CHR,CNT,FLG1          old line:    ' STORE   CHR,CNT,FLG1'
INIT1                                  new line: 'INIT1'
```

If D^c is typed:

```
*210EDIT.
      STORE      CHR,CNT,FLG1
INIT1 STORE      CHR,CNT,FLG1          ← this is the new line #210
*
```

Suppose the user has begun the following edit:

```
*13,14EDIT.
A       BSS      1
$A
```

and now types D^c:

```
*13,14EDIT.
A       BSS      1
$A      BSS      1              the edit of line #13 terminates
B       BSS      1              and line #14 is typed out for editing.
```

(When the user terminates the edit of line #14, the whole edit will terminate and the lines typed in will become the new lines #13 and 14.)

F^c

QED copies the rest of the old line onto the new without typing it. In addition the edit of the line is terminated in EDIT or MODIFY. (F^c does not terminate an APPEND, INSERT, or CHANGE.)

Example:

```
*:NXCHR:EDIT.
NXCHR      LDA      CHAR          old line:  'CHR      LDA      CHAR'
PV                                new line:  'PV'
```

If the user now types F^c:

```
*:NXCHR:EDIT.
NXCHR      LDA      CHAR
PV                                (the characters copied are not typed out).
*./                               Note that F^c terminated the EDIT.
PVCHR      LDA      CHAR
*
```

Skip Characters

The following control characters cause one or more characters from the old line to be skipped; the new line is not affected. QED types % for each character skipped. If there are no more characters in the old line, or if the character to be skipped to (in P^C and X^C) does not occur in the rest of the old line, QED rings the bell and takes no further action. (Editing may then proceed normally.)

S^C QED skips the next character of the old line.

Example:

```
*[CARTO]EDIT.
THE CARTONS OF SHELLS                    old line:  'S OF SHELLS'
THE CARTON                               new line:  'THE CARTON'
```

If the user now types S^C:

```
*[CARTO]EDIT.
THE CARTONS OF SHELLS                    old line:  ' OF SHELLS'
THE CARTON%                              new line:  'THE CARTON'
```

(At this point, the user could type D^C and the new line 'THE CARTON OF SHELLS' would be placed in the main text buffer **and the EDIT would be terminated.**)

$P^C\underline{C}$ QED skips up to (not including) the next occurrence of the character C in the old line after the next character. (P^C is the skip analogue of O^C.) C is never echoed.

Example:

```
*[HALLE]EDIT.
SANG HALLELUJAH!                         old line:  ' HALLELUJAH!'
SING                                     new line:  'SING'
```

If the user now types P^CH:

```
*[HALLE]EDIT.
SANG HALLELUJAH!                         old line is now:  'HALLELUJAH!'
SING%                                    new line is still:  'SING'
```

If the user again types P^CH:

```
*[HALLE]EDIT.
SANG HALLELUJAH!                    old line:  'H!'
SING%%%%%%%%%%%%                   new line is still:  'SING'
```

If the user again types P^CH, QED will ring the bell and take no further action, as it has already skipped up to the last H of the old line.

X^CC — QED skips up through the next occurrence of the character C in the old line. C is never echoed. (X^C is the skip analogue of Z^C.)

Example:

```
*[HALLE]EDIT.
SANG HALLELUJAH!                    old line:  ' HALLELUJAH!'
SING                                new line:  'SING'
```

If the user now types X^CH:

```
*[HALLE]EDIT.
SANG HALLELUJAH!                    old line:  'ALLELUJAH!'
SING%%                              new line is still:  'SING'
```

If the user again types X^CH:

```
*[HALLE]EDIT.
SANG HALLELUJAH!                    old line:  '!'
SING%%%%%%%%%%%%%                  new line is still:  'SING'
```

If X^CH is typed once more, QED will ring the bell and take no further action, as there are no more H's in the old line.

Retype Characters

The following control characters do not affect the state of the edit, but merely retype the old and new lines, to permit the user to recover in case he has become confused about the state of the edit. Editing may then continue normally.

R^c QED types line-feed and then the rest of the old line, and on the next line, the new line so far produced.

Example:

(Assume in this example that ↑ indicates that A^c was typed and % indicates a skipped character):

```
*91EDIT.
THE MANDALA (WHICH FIGURES PROM-          old line:  'URES PROM-'
S↑AE↑%M↑ MD↑ANDALA 8↑↑(↑ (%%%             new line:  'A MANDALA ('
                          URES PROM-      old and new lines are unchanged.
A MANDALA (
```

T^c QED types out the state of the edit as in R^c, except that the rest of the old line is properly aligned with the new.

Example:

Let us take the setup in the R^c example above and assume that T^c is typed instead of R^c:

```
*91EDIT.
THE MANDALA (WHICH FIGURES PROM-
S↑AE↑%M↑MD↑ ANDALA 8↑↑(↑ (%%%
          URES PROM-                    old and new lines are unchanged.
A MANDALA (
```

The Re-Edit Character

Y^c QED copies (without typing) the rest of the old line onto the new and then the result of this concatenation may be re-edited. That is, QED gives a carriage-return and line-feed and editing may continue; the old line is now the result of the concatenation and the new line is null.

Example:

```
*.EDIT.
ZHT      SKG      MAX                 old line:  '  SKG MAX'
ZHTOT                                 new line:  'ZHTOT'
```

If the user now types Y^C:

```
*.EDIT.
ZHT       SKG       MAX
ZHTOT                                   old line:  'ZHTOT      SKG      MAX'
                                        new line:  ''       (null)
```

(Note that the characters copied are not typed out.) Suppose the user now types C and then D^C:

```
*.EDIT.
ZHT       SKG       MAX
ZHTOT
CHTOT     SKG       MAX
*./
CHTOT     SKG       MAX
*
```

Mode Characters

E^C — QED changes the mode from replace to insert, and types <; or from insert to replace, and types >. The mode is replace at the beginning of each line; in replace mode, characters typed by the user replace those of the old line one-for-one. In insert mode, characters typed by the user are appended to the new line, but the old line is unaffected. (Skips and copies proceed as described above in either mode.)

Example:

```
*112EDIT.
RESTORING THE DAMAGED                   old line:  ' THE DAMAGED'
RESTORATI                               new line:  'RESTORATI'
```

If the user types E^C and continues typing:

```
*112EDIT.
RESTORING THE DAMAGED                   old line is still:  ' THE DAMAGED'
RESTORATI<ON OF                         new line is now:    'RESTORATION OF'
```

And if the user now types H^C (q.v.):

```
*112EDIT.
RESTORING THE DAMAGED                   old line is now:       '' (null)
RESTORATI<ON OF THE DAMAGED             the new line is:  'RESTORATION OF THE
                                                            DAMAGED'
```

K^c QED types " and changes mode from the usual one to one in which no characters are appended to the new line; or from this latter mode back to the usual one. Mode is set to the usual one at the beginning of each line.

Example:

(Underlined characters are those which would normally have been appended to the new line but were not because of the K^c mode.)

```
*19,20EDIT.
CHC      CIO    FILE                  old line:  'CIO   FILE'
CHC                                   new line:  'CHC     '
```

If the user types K^c and then H^c (q.v.):

```
*19,20EDIT.
CHC      CIO    FILE                  old line:  ''   (null)
CHC      "CIO    FILE                 new line:  'CHC      '
```

If the user now types carriage return:

```
*19,20EDIT.
CHC      CIO     FILE
CHC      "CIO     FILE(CR)
         SKG      =77B                old line:  '   SKG     =77B'
                                      new line:  'CHC      '
```

Note that although the CR caused the edit of the next line to begin, it was not appended to the new line, so that the new line is "left over" for the edit of line #20. Also, the K^c mode has been reset to the usual one by the beginning of the line #20 edit. Suppose the user now types K^c and then U^c (q.v.):

```
*19,20EDIT.
CHC      CIO      FILE
CHC      "CIO      FILE(CR)
         SKG      =77B                old line:  'SKG      =77B'
"                                     new line:  'CHC       '
```

Now suppose the user types K^c and then D^c (q.v.):

```
*19,20EDIT.
CHC      CIO      FILE
CHC      "CIO      FILE(CR)
         SKG      =77B
"         "SKG     =77B
*
```

Note that the single line 'CHC SKG =77B' replaced the old lines #19 and 20.

L^C QED changes mode from the usual one to one in which characters which are (or normally would be) appended to the new line are also collected in a special (non-addressable) internal buffer, and types [; or from this latter mode back to the usual one, and types]. Buffer 1 is cleared. Whenever L^C is echoed as], the text in this internal buffer is loaded into buffer 1. This action is also taken when the user terminates a line in this special mode. The mode is reset to the usual one at the beginning of each line.

Example:

(Assume that [and] indicate that L^C was typed):

```
*BUFFER #1.
"ABC"
*:INIT2:APPEND.
     STORE   [FLG1,FLG2,FLG3(CR)          (note that the mode is reset by CR)
     STORE   [FLG4,FLG5],FLG6(D^C)
*.-1,./
     STORE   FLG1,FLG2,FLG3               (L^C did not affect the append
     STORE   FLG4,FLG5,FLG6                itself)
*BUFFER #1.
"FLG1,FLG2,FLG3
FLG4,FLG5"
*
```

6.0 Substitute

The SUBSTITUTE command allows the user to substitute one string of characters for another in all or some of its occurrences in the main text buffer. There are options giving the user a variable amount of control over the individual substitutions and allowing him to see each substitution before and/or after it is made. The format of the command is:

In VERBOSE mode:

ISUBSTITUTE (options)/text^n/FOR/text^o/

In QUICK mode:

IS(options)/text^n/text^o/

Where the underlined portions of the command are typed by the user. In place of /, the user may employ any character except : or blank to delimit the two strings of characters, text^n and text^o. To allow the user to make this command more readable, blanks are ignored except in the two strings.

Text^n and text^o are strings of characters; neither may contain carriage-returns, and text^o should not be the null string. The control characters V^c, A^c, Q^c and W^c (described in section 5.4) may be used while typing in the strings.

When the character terminating text^o is typed, the SUBSTITUTE begins and proceeds generally as follows: QED begins on the first line of the specified interval (I) and searches for occurrences of text^o. The search continues through the last line of the specified interval (at which point the search, and the SUBSTITUTE, terminate) or until an occurrence is found. In this case QED may or may not make the substitution of text^n for text^o (according to the "options" specified). In either case, the search continues immediately after that occurrence of the text^o (or the text^n that replaced it) and proceeds as above, until the end of the interval is encountered. At that point, QED types out an integer which is the number of substitutions actually made.

The options possible are:

:G — which causes QED to make all substitutions without typing. (It will terminate after N substitutions if :N (q.v.) has been typed.)

:W — Each time an occurrence of the string text° is found, QED types out the line containing it (with the occurrence in question enclosed in double quotes) and expects the user to type:

- S — which causes QED to make the substitution and continue, or
- option — which causes QED to change to that option and wait for the user to type S, another option, or some other character.
- any other non-blank character — which causes QED to continue without making the substitution.

:L — after all the substitutions in a given line are made, the line is typed out. (There must be at least one substitution made in the line for this to happen.)

:V — is the combination of :W and :L.

:N — where N is a string of digits (terminated by the first non-digit following it) causes QED to make at most N substitutions. That is, QED will terminate the SUBSTITUTE normally when it has made N substitutions. (If this option is omitted, the SUBSTITUTE will terminate only when the end of the interval I is reached.)

Options may be concatenated (e.g., SUBSTITUTE :L:19/A/ FOR /B/ which will make at most 19 substitutions and list each one made) and are interpreted thusly:

:C — (C is G, W, L, or V) overrides all previous :C.

:N — (N is an integer) overrides all previous :N.

The final C and N are merged.

One may choose to give no options, which causes QED to make all substitutions in the interval, without typing.

As examples of the SUBSTITUTE command, consider: (underlined portions typed by the user.)

```
*.SUBSTITUTE  :V.HAS GIVEN.  FOR.GAVE.
HE  "GAVE" SEVERAL RECITALS, OF                    note use of . instead of /
S
HE  HAS GIVEN  SEVERAL RECITALS, OF
1
*:ADDR1:,.+3
ADDR1 ADD   ALPHA
      BRX   *-1
ADDR2 ADD   BETA
      ADD   GAMMA
*.-3,.SUBSTITUTE  :L/ SUB/  FOR  / ADD/
ADDR1 SUB   ALPHA                                  (note use of blanks to keep
ADDR2 SUB   BETA                                   the labels ADDR1 and ADDR2 from
      SUB   GAMMA                                  being changed)
3
*:ADDR31:,.+2
ADDR31 ADD  DELTA
       BRX  *-1
       ADD  EPS
*.-2,.SUBSTITUTE  :W/SUB/  FOR  /ADD/
"ADD"R31 ADD  DELTA
:G K
2
*.-2,./
ADDR31 SUB  DELTA                                  (note the use of  :G to change
       BRX  *-1                                    options, and K to prevent a
       SUB  EPS                                    substitute)
*
```

7.0 String Buffers

Thirty-six string buffers are available to the user, named by the digits (0-9) and letters (A-Z). Their contents may be any string of characters. The contents of buffers 0 and 1 are affected by searches, SUBSTITUTE, and L^C, as noted in the description of these features. In general, string buffers 0-9 are reserved to joint use by QED and the user, i.e., both may affect their contents, and this should be borne in mind when buffers 0-9 are used. It is advisable to use the lettered buffers (A-Z) when one wants the contents of a buffer to be changed only when he explicitly changes them (with LOAD, GET, KILL or JAM).

7.1 Loading, Deleting and Printing Buffers

Each of the following five commands is specified by its first letter; QED then completes the command up through the number sign (#). The user then types a letter or a digit (C) to specify a buffer and then gives a confirming dot (.). (In QUICK mode, QED only types out # to complete the command; the user then proceeds as above. For example, in QUICK mode the command to print buffer E will look like: *B#E. where the underlined portions are typed by the user.)

Any string buffer may be loaded with one of the three following commands. A buffer is always cleared before it is loaded.

ILOAD #C. where C is a letter or a digit causes QED to load string buffer C with the lines specified by I. . is changed to the last line loaded.

IGET #C. causes QED to load buffer C with the specified lines, which are then deleted from the main text buffer. . is changed to the line before the first one loaded.

JAM INTO #C. causes QED to go into text input mode. The user may type in text (V^C, A^C, Q^C, W^C are recognized and have the functions described in section 5.4), terminated by a D^C. QED then loads buffer C with the collected text.

The contents of a string buffer may be printed with:

BUFFER #C. which types the contents of buffer C, enclosed in double quotes.

To delete the contents of a buffer, type

KILL #C. which clears buffer C. (Note: to delete the contents of the main text buffer type 1,$DELETE.)

Examples: (suppose buffer E contains 'ABC')

```
*BUFFER #E.
"ABC"                          (The double quotes are supplied by
*JAM INTO #E.                  the BUFFER command.)
NEW CONTENTS(D^C)              (Note that no CR is supplied before
*BUFFER #E.                    the D^C; the contents of buffers need not
"NEW CONTENTS"                 be lines.)
*1/
FIRST LINE
*1LOAD #E.
*BUFFER #E.
"FIRST LINE
"
*1,4/
FIRST LINE
SECOND
THIRD
FOURTH!!LINE
*2,3GET #5.
*1,2/
FIRST LINE                     (The old second and third lines were deleted.)
FOURTH!!LINE
*BUFFER #5.
"SECOND
THIRD
"
*KILL #5.
*BUFFER #5.
""                             (Buffer 5 contains no text.)
*
```

7.2 The Uses of String Buffers

B^CN is recognized at all times, and is equivalent to the user's typing the string of characters in buffer N (with the exception of command errors, noted below). The B^C is echoed or a letter, as #, and if B^CC is typed, where C is not a digit QED types ? and ignores both B^C and the character C.

Thus, string buffers can be used to minimize typing in text input; for example, suppose buffer W contains ' BSS 1' and the user does the following INSERT. Assume in this example that # indicates that B^C was typed:

```
*11INSERT.
A1#W(CR)
A2#W(CR)
B1#W(CR)
B2#W(CR)
*11,14/
A1      BSS     1
A2      BSS     1
B1      BSS     1
B2      BSS     1
*
```

Another use of string buffers is that of moving text. For example (assume that # indicates that B^C was typed):

```
*:TEMP1:,.+2/
TEMP1   BSS     1
TEMP2   BSS     1
TEMP3   BSS     1
*.-2,.GET #X.          (GET deletes the lines after loading the buffer)
*115INSERT.
#X
*115,117/
TEMP1   BSS     1
TEMP2   BSS     1
TEMP3   BSS     1
*                      (Note:  buffer W still contains the three lines)
```

This sequence of commands was used to take the three lines in question from their old position and insert them before line #115.

Also, buffers may be used as a source of commands. (An error in a command taken from a buffer causes control to return directly to the user, i.e., the whole hierarchy of buffers is aborted; the contents of the buffers are not changed, of course. Rubout has the same effect.) When commands are taken from buffers only characters explicitly printed with the commands /, PRINT, line-feed, ↑, =, ←, ", or BUFFER are typed out. As an example of commands from buffers consider:

```
*JAM INTO #J.
:ABC:E.XYZ(V^C)(D^C)&D(D^C)      &D is typed by QED; note the use of V^C
                                      to enter a D^C.
*BUFFER #J.
":ABC"E.XYZ&D"                   QED types out D^C as &D.
*(B^C)#J                         # is typed by QED.
*
```

At this point, all labels 'ABC' in the main text buffer have been changed to 'XYZ'. The error of the search :ABC: when there are no more labels of that form causes control to return to the user.

Buffers used in this way may call other buffers; that is, if $B^C\underline{N}$ is inserted in a buffer (with V^C) then when those characters are accessed by QED in reading from the buffer, they will cause a transfer to buffer N until this latter is exhausted, at which time control returns to the characters following the $B^C\underline{N}$ in the original buffer. Buffer N may call other buffers, in the same manner. (However, if the contents of the calling buffers are altered by the operations of the called buffer, peculiar things may result.) For example, if we have:

```
*BUFFER #F.
"1E.&H!
&BN"
*BUFFER #N.
".+1E.&H!
&BN"
*
```

Then if(B^C)F is typed, each line of the main text buffer will be edited in turn (nothing will be typed out), and at the end, i.e., when QED finally types *, each line will have ! at the end. The error of calling for .+1 (in buffer N) when . is the last line returns control to the user.

8.0 Mode and Tab-Setting Commands

8.1 Quick/Verbose Mode

QUICK. causes command completion (in command mode) to be suppressed except in the cases of READ FROM and WRITE ON.

VERBOSE. restores command completion disabled by QUICK. This is the usual mode.

Examples:

```
*llEDIT.              (verbose mode:  'DIT' is typed by QED)
      CLA
      CLB
*QUICK.
*.E.                  (quick mode:  'DIT' not typed by QED)
      CLB
      CLX
*READ FROM /T/.       ('READ FROM' is unaffected)
901 WORDS.
*V.                   ('ERBOSE' is not typed by QED)
*llEDIT.              (mode is verbose again)
      CLX
      CLAB
*
```

8.2 Ignore Characters Mode

" causes characters typed by the user to be ignored (carriage return is still supplied with line-feed; I^C (tab) and B^C (buffer call) are still recognized) until the next D^C. Rubout also restores the mode to the usual one. In addition, if this command is read from a buffer, characters up to the next D^C are typed out. This is useful for printing messages from buffers, as usually nothing except explicit print commands causes printing from buffers.

Examples:

```
*"THIS IS NOT(CR)
RECOGNIZED(CR)
BY QED(D^C)
*llOAD.
"(CR)
MESSAGE(V^C)(D^C)&D(D^C)    &D is typed by QED; note the use of V^C
*llBUFFER.                     to enter a D^C without terminating the LOAD.
""
MESSAGE&D"                  Only the first and last " are supplied by QED.
*(B^C)#1
MESSAGE
*
```

8.3 Tab-Setting

TABS. QED gives a carriage return and line-feed and then expects a string of at most twelve decimal numbers separated by commas (,) and terminated by a dot (.); none of the numbers is to exceed 80. Also, the numbers should be in ascending order of magnitude to avoid peculiar results. If the input is deemed illegal, QED types ? and then the user may continue typing. QED sets the tab stops to the specified positions. The tab character is I^c. (Full description in section 5.4.) The tab stops are initialized upon entry into QED to 8, 16, 32, 40.

Example:

```
*TABS.
5,10.
*
```

sets the tab stops to 5 and 10.

9.0 Returning from QED and Panic Messages

9.1 The Normal Return

When the user is in command mode, the command

FINISHED. may be used to return to the exec. If the last command previous to the FINISHED command was a WRITE command, or if there is no text in the main text buffer, QED simply returns the user to the exec. If there is text in the main text buffer and the last command was not WRITE ON, QED types WRITE OUT! (to remind the user to save his text on a file before he leaves QED) and returns to the exec.

If the user has returned to the exec from QED and called no other subsystem, nor done anything to cause a RESET, he may continue QED by typing

@CONTINUE QED.

This preserves the state of QED as it was before he returned to the exec (in particular, the main text buffer is unchanged). However, typing

@QED

will get the user a "fresh" copy of QED, and in particular, one with nothing in the main text buffer.

For example, after the sequence:

```
*FINISHED.
WRITE OUT!

@DRUM BLOCKS LEFT = 10 OUT OF 110
@CONTINUE QED.
*
```

the user may continue using QED just as though this sequence had not been typed.

9.2 Rubout

The rubout button may be pressed at any time. If QED is inputting text, rubout will cause QED to ring the bell. If a second rubout is typed (with no intervening typing) the command will be aborted and the text being input will be lost.

In all other cases, typing a single rubout during the execution of a command will cause the current operation to be aborted, and QED will return to command mode. If QED is in the middle of printing or writing a large number of lines, . will be set to the last line printed or written. The value of . may be unpredictably affected by aborting commands in this way.

In command mode, two rubouts with no intervening typing will return the user to the exec. This is not the normal return (see section 9.1 for the normal method of returning), but the user may continue QED with the executive command

@CONTINUE QED.

as described in section 9.1.

9.3 Panic Messages

In certain contexts, QED will type out a message to warn the user of a condition he might not otherwise be aware of:

NO ROOM.	indicates that the operation being executed caused a memory trap. (Check machine size if UNUSED MEMORY is >0.)
WON'T FIT.	indicates that the attempt to load a string buffer will overflow the area allocated for string buffers. The buffer concerned will have been cleared but not loaded.
#1 FULL.	indicates that the operations of L^c (q.v.) have filled the special internal buffer allocated for them, and no more characters will be collected using L^c. Text input may continue normally.
I-O ERROR.	indicates that a READ or WRITE was terminated on some abnormal condition, possibly an unexpected end of record.
NEARLY FULL	indicates that text input has caused the internal text-collection buffer to fill nearly to capacity. The user may continue inputting text as usual, but if he does not terminate text input before the internal buffer overflows, he will get the message:
EDIT TERMINATED	indicating that QED simulated a D^c and terminated text input. QED then returns to command mode.

APPENDIX A: INDEX OF CONTROL CHARACTERS

		Page
A^c	backspace one character (↑)	5-5
$B^c\underline{N}$	call of buffer N (#)	5-5, 7-3
C^c	copy (typing) one character	5-8
D^c	copy (typing) rest of line and terminate	5-10
E^c	change insert/replace mode (< , >)	5-15
F^c	copy (no typing) rest of line and terminate	5-11
G^c	(bell) no function	
H^c	copy (typing) rest of line	5-9
I^c	tab	5-5
J^c	cannot be typed in	
K^c	change ignore/usual mode (")	5-16
L^c	change buffer 1/usual mode ([,])	5-17
M^c	(carriage return) terminate line (and edit)	5-5
N^c	backspace one character, restorative (↑)	5-7
$O^c\underline{C}$	copy (typing) up to C	5-9
$P^c\underline{C}$	skip up to C (%%...%)	5-12
Q^c	backspace one line (←)	5-7
R^c	retype, fast	5-14
S^c	skip one character (%)	5-12
T^c	retype, aligned	5-14
U^c	copy (typing) to next tab stop	5-8
$V^c\underline{C}$	take C literally	5-5
W^c	backspace word ( )	5-6
$X^c\underline{C}$	skip through C (%%...%)	5-13
Y^c	concatenate and re-edit	5-14
$Z^c\underline{C}$	copy (typing) through C	5-10
$\leftarrow^c$	no function	
$@^c$	no function	
$[^c$	no function	
$\backslash^c$	no function	
$\uparrow^c$	no function	
$]^c$	no function	

(See also the table of contents for section 5.4)

APPENDIX B: INDEX OF COMMANDS

Command	Page
"	8-1
/	3-1
=	2-3
↑	3-1
←	2-3
(line-feed)	3-1
APPEND	5-2
BUFFER	7-2
CHANGE	5-2
DELETE	5-1
EDIT	5-3
FINISHED	9-1
GET	7-1
INSERT	5-2
JAM INTO	7-1
KILL	7-2
LOAD	7-1
MODIFY	5-3
PRINT	3-1
QUICK	8-1
READ FROM	4-1
SUBSTITUTE	6-1
TABS	8-2
VERBOSE	8-1
WRITE ON	4-1

Note: A description of searches is in #4 and 5 under section 2.1, pages 2-1 and 2-2.