Program Writer™

WRITES PROGRAMS IN A FRACTION OF THE TIME IT TAKES YOU NOW!

BY **ALAN BIRD**

Reference Manual

Software Touch

9625 Black Mountain Road, Suite 204, San Diego, California 92126, (619) 549-3091

PROGRAM WRITER COMMAND CHART

Editor Commands

- * A-A: Auto line numbering
- A-B: Beginning of line
- * A-C: Copy (use with Paste)
- * A-D: Delete lines
- A-E: Change cursor mode
- * A-F: Find text
- * A-G: Get macros
- A-I: Insert line
- A-J: Jump to line number
- * A-K: Calculator
- A-L: Convert to lower case
- * A-M: Edit macros
- A-N: End of line
- A-O: Override
- * A-P: Paste (use with Copy)
- A-Q: Quit editor
- * A-R: Replace text
- * A-S: Save macros
- * A-T: Split into two lines
- A-U: Convert to upper case
- * A-V: List program variables
- A-X: 40/80 columns
- A-Y: Clear to end of line
- A-Z: Remove editor

A-1 thru A-9: Position in program

- * A-#: Renumber
- A-.: Increase margin
- A-,: Decrease margin

ESC: Restore line, stop macro, quit macro editor, redisplay screen
TAB: Move to next tab stop
A-TAB: Move to previous tab stop
RETURN: Next line, accept input
DELETE: Delete to left of cursor
A-DELETE: Delete at cursor

A-left arrow: Previous word
A-right arrow: Next word
A-up arrow: Previous screen
A-down arrow: Next screen

*Not available in EDITOR.SMALL

Macro Commands

ctrl-A: Execute editor command
ctrl-Z: Execute macro

Find Control Commands

ctrl-A: Alphabetic character
ctrl-B: Blank
ctrl-C: Control character
ctrl-D: Digit (0-9)
ctrl-E: End of line
ctrl-G: Garbage
ctrl-L: Literal mode
ctrl-N: Not
ctrl-R: Repeat
ctrl-S: To end of statement
ctrl-V: Alphabetic or digit
ctrl-X: Any single character
ctrl-\: Or

Apple II or II+ Replacements

IIe/IIc	II/II+
Open-apple	ctrl-A
Solid-apple	ctrl-Z
Up arrow	ctrl-K
Down arrow	ctrl-J
TAB	ctrl-I
DELETE	ctrl-D
ctrl-\	ctrl-O

PROGRAM WRITER™

BASIC Program Screen Editor
by Alan Bird

Published by *THE SOFTWARE TOUCH™*
9625 Black Mountain Rd., Suite 204
San Diego, CA 92126
(619) 549-3091

TABLE OF CONTENTS

COPYRIGHT

LIMITATION ON WARRANTIES AND LIABILITY

Even though *THE SOFTWARE TOUCH* has tested the software described in this manual and reviewed its contents, neither *THE SOFTWARE TOUCH* nor its software suppliers make any warranty or representation, either express or implied, with respect to this manual or to the software described in this manual, their quality, performance, merchantability, or fitness for any particular purpose. As a result, this software and manual are sold "as is," and you the purchaser are assuming the entire risk as to their quality and performance. In no event will *THE SOFTWARE TOUCH* or its software suppliers be liable for direct, indirect, incidental, or consequential damages resulting from any defect in the software or manual, even if they have been advised of the possibility of such damages. In particular, they shall have no liability for any programs or data used with *THE SOFTWARE TOUCH* products, including the costs of recovering or reproducing these programs or data. Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you.

ProDOS and DOS 3.3

This product includes software, ProDOS™ and DOS 3.3, licensed from Apple Computer, Inc. Apple Computer, Inc. makes no warranties, either express or implied, regarding the enclosed computer software package, its merchantability or its fitness for any particular purpose. Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you.

SITE LICENSING AND NETWORK COPIES

Schools and businesses call (619) 549-3091 for information on site licensing and network copies.

INTRODUCTION

PROGRAM WRITER is a utility program that allows you to quickly and effortlessly make changes or add new lines to a BASIC program. Changes are made to the program much like editing a text file with a word processor. With *PROGRAM WRITER* you can quickly scroll to the location in your program you want to edit and easily change, insert, or delete. You may optionally use a mouse for faster cursor control.

With *PROGRAM WRITER* you can insert, delete, find and replace, renumber, copy and paste, delete lines, add new lines with automatic line numbering, insert control characters, split a line in two, list all the variables in the program, convert to upper/lower case or use the built-in BASIC calculator.

The editor is a memory–resident program. It stays in memory with and is hidden from your BASIC programs. It is always available for you to use once it has been installed. Because the program relocates itself, it should be compatible with most other machine language utilities you may be using already.

INSTALLING THE EDITOR

IMPORTANT: Before using the editor, make a backup copy of the disk. Since it is not copy–protected, you may use one of the standard copy programs such as COPYA, FILER, or SYSTEM UTILITIES.

There are 3 versions of the editor on the disk. The one called *EDITOR* is for normal use. Another called *EDITOR.SMALL* is much smaller and is to be used when little memory is available. Many of the more powerful editing features are not available with *EDITOR.SMALL* (see the Command Chart).

The third version, *EDITOR.LC*, loads into the "language card" or bank–switched memory. Use this version, if possible, when you have very large programs or have several utilities in main memory at the same time. *EDITOR.LC* requires only a few bytes of main memory space.

Under ProDOS, *EDITOR.LC* requires a IIc or a IIe with 128K of memory. It loads into the auxiliary bank–switched memory. Under DOS 3.3, it may load into main memory (for a 64K machine) or optionally into auxiliary memory (for a 128K machine). Use the *CONFIGURE* program to select which bank of memory the DOS 3.3 version will load into (see page 20). The default version loads into main memory. Be careful not to load *EDITOR.LC* into main memory if you are using a DOS 3.3 mover that relocates DOS up into the "language card".

Each version of the editor runs under DOS 3.3 or ProDOS. Both versions are on the same side of the *PROGRAM WRITER* disk, which is a hybrid disk. When you boot up the disk, you will automatically be in the ProDOS version.

IMPORTANT: Be sure to save your program before loading the editor. If during installation of the editor you receive a message notifying you to reload your BASIC program, it means that the editor has loaded in over part of your program and your program has been destroyed. This only occurs with very long

BASIC programs. The editor initially loads into memory at 16384 ($4000) and then moves itself as high as it can go in memory.

ProDOS
To use the ProDOS version of the editor, simply boot the disk and select the appropriate editor from the *STARTUP* menu. If you have already booted with another ProDOS disk, you may insert the *PROGRAM WRITER* disk and enter the following from the "]" prompt:

```
-EDITOR
 or
-EDITOR.SMALL
 or
-EDITOR.LC
```

DOS 3.3
To use the DOS 3.3 version of the editor, boot the disk and select *SWITCH TO DOS 3.3* from the *STARTUP* menu. When the DOS 3.3 menu appears, select the editor. If you have already booted with another DOS 3.3 disk, you may insert the *PROGRAM WRITER* disk and enter the following from the "]" prompt:

```
BRUN EDITOR
 or
BRUN EDITOR.SMALL
 or
BRUN EDITOR.LC
```

HOW TO BOOT A DISK
Insert the disk to be booted into drive 1 and turn the power on, or if you have a IIe or IIc, you may hold down the OPEN–APPLE key and the CONTROL key and while holding those 2 keys down, press the RESET key.

USING THE EDITOR

IMPORTANT: Periodically quit the editor and save your program in case you make any changes that you later may regret. It is important to always have a backup version of the program you can return to.

To enter the editor, type 2 ampersands (&&) followed by RETURN. After entering the editor, most of the screen will contain program statements from the beginning of the BASIC program that is currently in memory. If there is no program in memory, this editing area will be blank.

THE CURSOR
Most program editing can be done by placing the cursor on the screen where you want to make the change (see MOVING THE CURSOR, page 7), verifying that the cursor is in the correct mode (insert or replace), and by typing in the corrections to the line.

In replace mode, the cursor is a flashing white box. In this mode, anything you type will replace the character the cursor is currently on. In insert mode, the cursor is a flashing underscore (_). In insert mode, anything you type will be inserted in front of the character currently under the cursor. See EDIT CURSOR on page 9 for information on switching between insert and replace modes.

The first change that is made to a BASIC line will cause the cursor to switch to a flashing plus sign (+). It will be normal or inverse depending on which cursor mode is in use. The flashing plus indicates that a change has been made to the line and that the line may be restored to its original state by entering ESC. Moving the cursor off of the line or entering certain editor commands will cause the BASIC line to be entered into the program and the cursor will change back to a flashing box or underscore. When this happens, the line cannot be restored using the ESC key.

SCREEN WIDTH
The editor works in 40 columns or in 80 columns on a IIe or IIc. It does not work with II or II+ 80–column cards. The editor starts in the screen width that is active when it is entered. You may switch between 40 and 80 columns by using the A–X command (see MISC COMMANDS, page 18).

LINES
The word "line" has two meanings when using the editor. BASIC lines are lines in a program that start with a line number and end with the end of the program line. Screen lines are any of the 24 lines of characters that you see on your video screen.

PROMPT LINE
The bottom line of the screen is the prompt area. Most of the time it simply indicates the number of free bytes you have remaining in memory for your program. You need not be concerned with this number unless you have an unusually large program and are in danger of running out of memory or if you are doing some program optimizing and are concerned about how much memory you are saving. The dashed line above the prompt line is for separating the prompt area from the main editing area.

Some commands will prompt you for information the editor needs, such as what to look for when doing a FIND or what line numbers to provide when doing ADD. This information is entered in the prompt area.

Many of the editing functions such as insert, delete, clear to end of line, etc., can be used in the prompt area. Note that unlike typing information from the Applesoft prompt ( ]), everything after the cursor is accepted as part of the input. If, for example, you type in *PRINTING* as something for the editor to find, and you want to change it to *PRINT*, you must delete the *ING*—you cannot simply backspace over the last 3 characters.

Some commands prompt for a simple yes or no answer. You may enter *Y* or RETURN for yes; or *N*, SPACE, or ESC for no.

<u>CONTROL CHARACTERS</u>
If there are any control characters embedded in the program in REM statements, DATA statements, or between quotes, these will be listed in inverse on the screen. If you enter control characters in find–strings, macros, etc., these will also appear in inverse.

<u>LONG LINES</u>
A program may contain lines that are too long to edit. Normally lines this long can only be obtained by using a program that compacts a BASIC program to its smallest size by concatenating lines. A line that is too long to edit from the keyboard is shown by the editor as a single asterisk (*) following the line number. Most programmers will never see this.

Some very long lines are listed by the editor without any spaces between keywords to keep the lines short enough that they can still be entered into the program.

FULL SCREEN EDITOR COMMANDS

The following are all of the available editor commands. AppleWorks owners will be pleased that many of the commands are the same as those used by the AppleWorks word processor and are very easy to learn.

Some keys are not available on the II and II+. Their replacement keys are listed in parenthesis.

<u>DEFINITIONS</u>
ctrl– Hold down the CONTROL key then press the key following *ctrl-*.

A– If you are using the editor with a IIe or IIc, hold down the OPEN–APPLE key then press the key following *A-*.

If you are using the editor with a II or II+, press instead the ctrl–A, release it, then press the key following *A-*. For example, A–J means enter ctrl–A then J. After pressing ctrl–A, the cursor will change to a flashing inverse ">" to signal that the following character will be an editor command. If you accidentally press ctrl–A, you may press ESC to cancel.

MOVING THE CURSOR

<u>Left arrow:</u>
Moves the cursor to the left one space. If the cursor is at the left edge of the screen, it will go up 1 line and be moved to the far right edge of the screen.

<u>Right arrow:</u>
Moves the cursor to the right one space. If the cursor is at the right edge of the screen, it will go down 1 line and be moved to the far left edge of the screen.

<u>Up arrow:</u> (CONTROL–K for II or II+)
Moves the cursor up one line. If the cursor is already at the top of the screen, the editor will scroll to the previous line in the BASIC program (if one exists).

<u>Down arrow:</u> (CONTROL–J for II or II+)
Moves the cursor down one line. If the cursor is already at the bottom of the screen, the editor will scroll to the next line in the BASIC program (if one exists).

<u>MOUSE:</u>
If your Apple has a mouse, you may use it to quickly position the cursor on the screen. If you attempt to move the cursor off the top or bottom of the screen, it will scroll.

If you need to scroll many lines, push and hold down the mouse button. If the cursor was in the top half of the screen when the button was pushed, the editor will continuously scroll up. If the cursor was in the bottom half, the editor will scroll down. Press the OPEN–APPLE key to scroll a page at a time.

<u>A–left arrow:</u>
Moves the cursor to the previous word on the screen. A word is one or more non–space characters preceded by a space.

A–right arrow:
Moves the cursor to the next word.

A–up arrow: (CONTROL–A CONTROL–K for II or II+)
Moves the cursor to the top of the screen or if the cursor is already at the top, the editor will scroll to the previous page in the BASIC program (if not already at the beginning).

A–down arrow: (CONTROL–A CONTROL–J for the II or II+)
Moves the cursor to the bottom of the screen or if the cursor is already at the bottom, the editor will scroll to the next page in the BASIC program (if not already at the end).

RETURN:
Hitting the RETURN key anywhere on a line will move the cursor to the beginning of the next BASIC line. RETURN is also used to indicate that you are finished entering something on the prompt line.

TAB: (CONTROL–I for II or II+)
Advances the cursor to the next tab stop. Tab stops are not adjustable and are preset to every 8th position on the screen.

A–TAB: (CONTROL–A CONTROL–I for II or II+)
Moves the cursor to the previous tab stop.

A–B: {BEGINNING}
Moves the cursor to the beginning of the BASIC line the cursor is on.

A–N: {END}
Moves the cursor to the end of the BASIC line the cursor is on. This is the fastest way to get to the end of the line to add more statements.

A–1 thru A–9:
Scrolls to relative positions within the program. A–1 goes directly to the first page in the program, A–4 is about halfway through the program, A–7 is very near the end, and A–9 lists the last full page of the program.

A–J: {JUMP}
Jumps to the line number you specify. That line number will appear at the top of the screen. If the line number you specified is not in the program, the first line previous to the non–existent line number will be listed.

The editor remembers the last line you jumped to. It becomes the default line number. If you are doing a lot of work in a specific area of a long program, do a jump to that area of the program. When you quit the editor to test your program and then return to the editor, you can quickly jump to that area of the program since it is the default jump line number.

EDITING COMMANDS

ESC:
Restores the BASIC line you are currently editing to its original form. After you make any changes to a line, the cursor changes to a normal or inverse plus (+) depending on whether the editor is in replace or insert mode. Hitting ESC while the "+" cursor is active will cause the BASIC line to be restored to its original state before you made any changes. This is useful if you have made errors and would like to start all over on the line.

If the cursor is no longer a plus, the BASIC line has been entered into the program and you cannot automatically restore it with the ESC key.

ESC also breaks you out of many commands that require input from the keyboard. If for example you are entering a string to FIND and you suddenly decide you do not want to FIND anything, enter ESC.

A–E: {EDIT CURSOR}
Switches between insert mode and replace mode. A flashing white box cursor indicates replace mode and a flashing underscore (_) indicates insert mode.

DELETE: (CONTROL–D for II or II+)
The DELETE key deletes the character to the left of the cursor.

A–DELETE: (CONTROL–A CONTROL–D for II or II+)
Deletes the character under the cursor.

Note that DELETE by itself is used for deleting text to the left of the cursor and A–DELETE is used for deleting text to the right of the cursor (including the character under the cursor).

A–Y: {CLEAR}
Deletes everything from the cursor to the end of the BASIC line.

A–O: {OVERRIDE}
Overrides any function the following key you enter has and inserts it directly into the text. This is necessary for inserting certain control characters such as CONTROL–M (RETURN), CONTROL–H (BACKSPACE), etc. After entering A–O, the cursor will change to an inverse question mark (?) so that you may know that the following character will be inserted as–is.

ESC will not break out of override mode as it will many other commands since it may be necessary to insert an ESC character into the program text.

The following is a list of control characters you may want to insert in REM statements or between quotes for PRINT statements. Control characters are signified by a preceding caret (^).

ctrl–D:	Use this between quotes instead of CHR$(4) for DOS commands: PRINT "^DCATALOG"
ctrl–G:	Causes a beep
ctrl–H:	Backspace
ctrl–J:	Line feed or down arrow

ctrl–K:	Reverse line feed or up arrow
ctrl–L:	Form feed or HOME in 80 columns
ctrl–M:	RETURN
ctrl–U:	Forward arrow
ESC:	Used extensively in printer control strings

A–L: {LOWER CASE}
Converts the character at the cursor to lower case if possible. Since the cursor is also advanced one character, you may repeatedly enter the A–L command to convert several characters.

A–U: {UPPER CASE}
Converts the character at the cursor to upper case if possible. Since the cursor is also advanced one character, you may repeatedly enter the A–U command to convert several characters.

A–D: {DELETE}
Deletes one or more BASIC lines from the program. Place the cursor anywhere on the first (or last) line you want to delete, then enter A–D. Move the cursor to the other end of the range of lines you want to delete. If you are only deleting one line, you may leave the cursor where it is. If you are deleting a large number of lines, you may use the OPEN–APPLE number keys or OPEN–APPLE arrows to quickly position the cursor. Enter RETURN and the editor will display the starting and ending line numbers and ask you if you are sure you want to delete those lines. Enter *Y* or *N* if you do or do not want the lines deleted.

IMPORTANT: Always think twice before deleting lines. It may be wise to save the program first in case you later decide you need the lines restored.

A–T: {TWO}
Splits a BASIC line into two lines. Position the cursor where the split is to be made. Everything to the left of the cursor will remain in the same line. Everything to the right will be moved to the new line. The character under the cursor will be deleted. Normally this will be the colon (:) that separates two statements.

After entering A–T, you will be prompted for the line number of the new line. The default number is the one immediately following the line that is being split. You may change this if you desire. Press RETURN after you have specified the line number.

ADDING NEW LINES

A–I: {INSERT}
Allows you to insert a new line into the program at any location. If the cursor is on a BASIC line when you enter A–I, a blank line is created before the line so you may insert a new one. If the cursor is already on a blank line or a partial line, the cursor is moved to the first available space for adding a line. Don't forget to type in a line number.

A–A: {ADD}
Allows you to easily add new lines to the program by providing automatic line numbering. After entering A–A, you will be prompted to enter the starting line number for auto numbering. You may enter RETURN if you want the default number provided for you. You then must enter the increment between line numbers. Once again you may use the default value.

If, for example, you would like the following line numbers to be provided for you: 200, 210, 220, etc.; use 200 as the starting value with an increment of 10.

Once you have entered the starting and incremental values, the first line number will be provided for you at the bottom of the current page on the screen. It may be useful before entering A–A to scroll to the point in the program where the text will be added so that you can examine the program statements that lead up to what you are adding.

After typing in the line, press RETURN and the editor will provide a new line number. If you are finished with auto numbering, press ESC. You may also exit from auto numbering mode by moving the cursor away from the line you are adding.

COPYING AND PASTING

A–C: {COPY}
This command is used to specify the text that will be copied or pasted into other areas of the program. After entering A–C, use the arrow keys to highlight the text you want to copy, then press RETURN. Copied text may not extend across more than one BASIC line.

A–P: {PASTE}
After using the COPY command to specify the text to be copied, use the PASTE command to copy the text into the program at the current cursor location one or more times. You may paste using either insert or replace cursor modes.

FINDING AND REPLACING TEXT

A-F: {FIND}
After entering A-F, you will be prompted to enter something to search for called a *find-string*. The string may be as long as the prompt area. In 80 columns, the *find-string* may be about twice as long as in 40 columns. If you have previously entered a *find-string*, it will be the default *find-string* and you may use it again or type in a new one. Be sure to use A-Y to delete the default *find-string* if you are typing in a new one.

Once you enter RETURN, the editor will search for and highlight any text in the program which matches your *find-string*. The editor starts searching from the beginning of the line that the cursor was on when the A-F was pressed. To start searching from the beginning of the program, do a A-1 first.

The editor searches within each line separately. Nothing is found that starts in one BASIC line and continues into another.

If no text is found that matches the *find-string*, the editor will so indicate in the prompt area and a warning bell will be sounded. If a match is found, the text will be highlighted (printed in inverse) and the editor will ask if you want to search again. Answering *Y* (or RETURN) for YES will cause the editor to search again. Answering *N* (or SPACE) will cause the editor to quit searching and will place the cursor at the beginning of the text that was found.

All spaces in the program text are ignored, but any space you enter in the *find-string* is required. For example, entering *GOTO* would find "GO TO", "G OTO", or "G O T O" since all the spaces in the program are ignored. If you enter *GO TO*, then "GO TO" will be found but "GOTO" will not since a space is required between the first "O" and the "T".

No distinction is made between upper and lower case. For example, searching for *Mark* will find "MARK", "mark", or "MaRk".

A-R: {REPLACE}
Allows you to search for and replace text in the program.

First, enter a *find-string* as with the FIND command. Next, enter the string you would like to replace the *find-string* with. This will be the *replace-string*. Finally, the editor will ask if you want to replace some or all of the occurrences of the *find-string*. If you answer *Y* , the editor will automatically search for and replace each occurrence in the program that matches the *find-string*. If you answer *N* , the editor will stop when it finds an occurrence of the *find-string* and ask you if you want to replace it. Answer *Y* to replace or *N* to not replace the text. To stop replacing, press ESC.

FIND CONTROL COMMANDS

Certain control characters take on special meaning when entered into a *find-string*. Use of control commands adds a lot of flexibility to the FIND and REPLACE commands.

Pressing the CONTROL key is indicated by a caret (^) in the examples in the following descriptions. For example, CONTROL-A is indicated by ^A.

ctrl-A: {ALPHA}
Will match any lower- or upper-case alphabetic character. Users of a II or II+ will need to use the A-O override command to enter a CONTROL-A.

S^A$	will find *ST$* or *Sx$* but not *S1$* or *S"$*

ctrl-B: {BLANK}
Will match a blank or space. Use this at the end of a *find-string* if you are specifying that there must be a space at the end.

LEN^B	will find *LEN (A)* but not *LENGTH*

ctrl-C: {CONTROL CHARACTER}
Will match any single control character.

REM^C	will find a REM followed by any control character

ctrl-D: {DIGIT}
Will match any single digit (0 thru 9). Users of a II or II+ will need to use the A-O override command to enter a CONTROL-D.

A^D%	will find *A1%* or *A7%* but not *AB%* or *Ax%*

ctrl-E: {END OF LINE}
Will match only if at the end of the BASIC line.

PRINT^E	will find a PRINT statement only if it is the last statement on the BASIC line

ctrl-G: {GARBAGE}
Will match any "garbage" until a match is found for the rest of the string. It is used as a multi-character wildcard.

T^GN	will find *TN, TAN, TIGHTEN* or *TOO OFTEN*
"^G"	will find and highlight any number of characters between and including quotes
A(^G)=	finds every assignment to the array *A*

ctrl-L: {LITERAL}
Everything after the ctrl-L is searched for literally. Control characters may be searched for and the search is case sensitive, meaning that upper and lower case characters must match exactly.

^L^A	will find any *REM, DATA*, or string literal that contains a CONTROL–A. Without the CONTROL–L, any alphabetic character would be found.
P^Lr	will find *Pr* or *pr* but not *PR*

ctrl–N; {NOT}
Matches anything provided it is NOT the following character.

S^NT	will find *S* followed by anything but *T*
NUM^N^D	will find *NUM* followed by anything but a digit
12^N^B	will find *12* followed by anything but a space
PRINT^N^A	will find a *PRINT* followed by anything but an alphabetic character (a variable name)

ctrl–O; {OR}
If the character preceding doesn't match, try the next one.

TES^OXT	will find *TEST* or *TEXT*
ZA^O^D^B	will find *Z* followed by an *A* or a single digit and ending with a space
A^OB^OC	will match *A, B*, or *C*

ctrl–R; {REPEAT}
Searches for repeated instances of the following character.

^R.	will match . or ..... or 100 periods
^R^D	will match *7, 255,1234567890*, or any string of digits

ctrl–S; {STATEMENT}
Matches everything up to the end of the current statement. This is very useful for finding and highlighting complete statements of a certain type such as REM's.

REM^S	will highlight entire REM statements
PRINT^S	will highlight entire PRINT statements

ctrl–V; {VARIABLE CHARACTER}
Matches either a digit or an alphabetic character (a variable character).

X^V	will find *X1* or *XA* but not *X$*

ctrl–X; {ANY CHARACTER}
Will match any single character, regardless of what it is. It is used as a single–character wildcard, unlike CONTROL–G which is a multi–character wildcard.

L^XST	will find *LAST, LIST, LOST,* or *LUST* but not *LEAST*

RENUMBERING

A–#; {RENUMBER} (# is shift–3)
There are at least 4 reasons for using the renumber command on a program:

1. Renumbering by an even number such as 10 gives your final product a nice, organized look.

2. If you need to insert some new lines between two consecutive numbers (for example, 20 and 21) you will need to renumber.

3. Renumbering using the smallest numbers available will help make your program as small as possible to reduce memory usage (start at zero using an increment of one).

4. Renumbering a portion of a program allows you to move lines (for example, subroutines) to another location in the program.

To use renumber, place the cursor on the first line that you wish to renumber. When renumbering the entire program, place the cursor on the first line of the program (A–1 will do this for you). Press A–# and then select the last line number you wish to renumber by placing the cursor somewhere on that line and pressing RETURN. When renumbering the entire program you should put the cursor on the last line of the program (use A-9). After selecting the ending line number to renumber, you will be prompted to verify that those line numbers are correct. If they are not correct, type *N* to start again.

Next, you will be prompted to specify the starting line number. This is the line number that you want the first line of the renumbered portion of your program to become. For example, if you are renumbering the entire program and you want the program to start with the number 100, then specify 100. If you want a subroutine that starts with line 300 to move to line 2500, then specify 2500 as the starting line number. When specifying the starting line number, you will be prompted with a default value that you may change. The default value is the last value that you specified for a starting line number.

Finally, you will be asked to specify the increment. If, for example, you want to renumber the entire program starting with line 100 followed by 110, 120, 130, etc., then use the default increment of 10. After specifying the increment value and hitting RETURN the program will be renumbered.

MOVING LINES
You can use the RENUMBER command to move one or more lines to another location in the program. In the following example, line 10 is the first line to renumber, line 20 is the last line to renumber, the starting line number is 50 and the increment is 10:

```
BEFORE                  AFTER

10  TEXT                30  PRINT  "HELLO"
20  HOME                40  END
30  PRINT  "HELLO"      50  TEXT
40  END                 60  HOME
```

Make sure when moving portions of the program by renumbering that the location it will be moved to will not overlap part of the program. If this occurs, you will receive a "LINE OVERLAP" error message.

MACROS

A macro is a sequence of characters that can be generated automatically by the editor by entering a single macro command.

A-M: (MACRO EDITOR)
The first step in using macros is to create them. Entering A-M gives you a screen with a column of macro command characters down the left side of the screen. Move the cursor to the macro you want to add or modify and type in its definition. For example, if you want the macro *F* to generate *FOR I = 1 TO 100,* just type those characters following "F: " followed by RETURN.

The maximum length of a macro is 37 characters.

Macros may be nested. That means that a macro may call another macro. To call a macro from within a macro, enter CONTROL-Z followed by the name of the macro. If you need to enter a macro that is longer than 37 characters, you may enter part in one macro then have the macro call another macro containing the remainder of the characters.

Macros may do editor commands. A command is indicated by putting CONTROL-A followed by the command character into the macro. For example, a macro that would cause a search for the word *PRINT* would contain the following:

```
^AF^AYPRINT^M
```

Which generates:

FIND command (A-F)
Clear to the end of line command (A-Y)
"PRINT"
RETURN (CONTROL-M)

You will need to use the override command (A-O) to insert control characters such as RETURN (CONTROL-M) into macros.

When you are finished defining macros, press ESC to return to the normal editing mode. If you have just finished editing a macro, hit RETURN before entering ESC, otherwise ESC will cause the macro to be restored back to what it was before you started editing it.

To call a macro, hold down the SOLID-APPLE key (to the right of the space bar) and press the key for the name of the macro (A thru X). For example, to invoke macro *F*, hold down the SOLID-APPLE key and type *F*. If you are using a II or II+, enter CONTROL-Z followed by the macro name. The cursor changes to an inverse *M* after entering CONTROL-Z to let you know that a macro name is expected.

You may halt a macro at any time by entering ESC. Most macros execute so quickly that you do not have time to stop them. However, you may define a macro that calls an editor command or calls other macros that may take a long time to execute. If you find that the macro is not doing what you expected or you would like to stop it for some reason, press the ESC key.

A-S: (SAVE MACROS)
Use this command to save the current macros to disk under a file name you specify. Make sure to type in a valid DOS 3.3 or ProDOS file name. You may optionally enter slot and drive parameters.

A-G: (GET MACROS)
Use this command to "get" or load a macro file from disk.

By saving and loading your own macro files, you can greatly increase the number of macros available and you can easily customize your own macros without having to retype them every time you load in the editor. Use the *CONFIGURE* program to add a macro file to the editor (see page 20).

NOTE: If you have moved DOS 3.3 onto the language card, you may not use A-S or A-G. You may, however, load a macro file into the editor as the default macros (see CONFIGURING THE EDITOR, page 20).

SAMPLE MACRO FILE
There is a sample macro file on the *PROGRAM WRITER* disk called *MACROS*. Use the A-G command to load it into your editor (or use the *CONFIGURE* program to make it a part of your editor). You may customize it for your own use if you desire. The following describes each macro in the *MACROS* file:

A:	Appends the line the cursor is on to the end of the previous line. If you receive a "LINE TOO BIG" error message, you may use the A-P (Paste) function to restore the original second line.
B:	Moves to the beginning of the text in the next line
C:	Generates *CHR$( )*
D:	Generates *DATA*
E:	Moves to the beginning of the last line in the program
F:	Finds the next line with a *PRINT* statement
G:	Generates *GET*
H:	Generates *HOME*
I:	Generates *IF = THEN*
J:	Inserts a RETURN (ctrl-M) into the text
K:	Generates *POKE*
L:	Continuously converts to lower case. Press ESC when finished.
M:	Generates *MID$(, ,)*
N:	Renumbers the entire program starting with 10 with an increment of 10
O:	Generates *ONERR GOTO*
P:	Generates *PRINT*
Q:	Deletes from the previous word to the end of the line. Keep pressing macro *Q* to delete more words
R:	Puts a *REM* statement at the end of the line
S:	Prints the size, in bytes, of the program you are editing
T:	Used by macro *S*
U:	Continuously converts to upper case. Press ESC when finished. Use this macro to convert all text to upper case for a II+ version of your program
V:	Generates *VTAB*
W:	Generates *PEEK( ) + 256*PEEK( )*
X:	Deletes the BASIC line the cursor is on. If you have the courage, you can add a *Y* to the end of this macro for uninterrupted BASIC line deletion

MISCELLANEOUS COMMANDS

A–Q: {QUIT}
This command will return you to BASIC so you may run your program, save your program, load another program, etc.

&&:
Use the double ampersand followed by RETURN to re–enter the editor.

A–Z: {ZAP}
Removes the editor from memory. This may be necessary if your program requires the memory used by the editor to run. This will also recover all memory used by any other memory–resident programs installed after the editor was installed. You will have to re–install the editor if you want to use it again.

A–K: {CALCULATOR}
Gives you a calculator in the prompt area. Enter any arithmetic expression in standard BASIC format. After entering RETURN, the result will be printed in the prompt area.

For example, entering *2 * (4 + SQR (25))* gives 18. Entering *ASC ("A")* returns the ascii value for *A* which is 65.

A–V: {VARIABLES}
Lists alphabetically all of the variables in the program you are editing. Use this command before creating a new variable to determine which variable names have already been used.

Only the first 2 characters of the variable names are listed. String variables are followed by a *$*, integer variables are followed by a %, and arrays are followed by parenthesis.

Remember that in Applesoft BASIC, only the first 2 characters of a variable name are significant. Don't create a new variable *DOG* if the variables command indicates that the variable *DO* is already being used.

It is all right to reuse variable names of different types. For example, the variable *A$* is not the same as *A%, A, A$(X), A(X), or A%(X).*

A–X: {SWITCH SCREEN WIDTH}
Switches between 40 and 80 columns. This is only available on a IIc or a IIe with an 80–column card.

A–,: {SHRINK MARGIN}
The A–comma command reduces the size of the margin when a long BASIC line needs to wrap around. Use this command if you would like to get more text on the screen or to reduce the margin width to zero doing screen layouts in 40 or 80 columns. The "<" above the comma indicates the direction the margin will move.

A–.: {EXPAND MARGIN}
The A–period command increases the size of the margin when a long BASIC line needs to wrap around. Use this command if you would like the editor to indent further. The ">" above the period indicates the direction the margin will move.

ERRORS

The following are explanations for the error messages the editor might give:

INVALID NUMBER
You have entered a BASIC line that does not start with a line number, the line number is greater than 63999, or you have specified an invalid number for the starting number or increment number for auto-numbering or renumbering. Also, if renumbering will create a line number greater than 63999, you will get this error.

LINE OVERLAP
This will occur during renumbering if you specify only a portion of the program to renumber and some or all of it will overlap with part of the program that is not being renumbered. Duplicate or interleaving lines would be the result.

LINE TOO BIG
If you are entering a line that contains many statements, there is a possibility that the line can become too long. When you attempt to move the cursor from the BASIC line that is too long, the editor will tell you that the line is too long and will leave the cursor at the first character in the line that will not fit so you can tell how much of the line is over the limit.

OUT OF MEMORY
Your program is too large to continue editing or you tried to renumber your program and there is not enough free memory to build the line number tables. You may have to remove another machine language utility you have in memory or even remove the editor to make more room for your program. If you are not using *EDITOR.SMALL* or *EDITOR.LC*, you may wish to switch to one of these editors.

PARTIAL LINE
You have attempted to make changes to a blank line or a line that is only seen partially on the screen. You will need to scroll up or down a few screen lines so the entire BASIC line is on the screen.

CONFIGURING THE EDITOR

The program *CONFIGURE* is a BASIC program that lets you change the editor's command keys, load a default macro file into the editor, specify the default cursor type, or specify which bank of memory to load the DOS 3.3 version of *EDITOR.LC*.

Most IIe or IIc owners will not want to change any of the command keys unless they are used to a different editor that has different command keys. Owners of a II or II+ may want to change some or all of the OPEN–APPLE command keys to control keys or other special keys. This will reduce the number of keystrokes needed to enter a command. Be careful not to change a command key to a character you would normally want to type into a program unless you want to use the A–O (override) command to enter it into the program. Don't use control characters such as CONTROL–M or CONTROL–J since CONTROL–M is RETURN and CONTROL–J moves the cursor down.

Run the *CONFIGURE* program and enter the name of the file you want to configure. Normally this will be *EDITOR, EDITOR.SMALL,* or *EDITOR.LC.*

When changing command keys, select the command you want to change by moving the highlighted cursor. There are 2 pages of commands. To go to the next page, move the cursor off the top or bottom edge of the commands on the current page. After selecting the command with RETURN, enter the control character, special character, or OPEN–APPLE character you want to invoke the command. You can specify an OPEN–APPLE character by holding down the OPEN–APPLE key followed by the character or by entering *A* followed by the character.

To load a macro file into the editor, just type in the name of the macro file.

To select the default cursor type or the bank of memory for DOS 3.3 *EDITOR.LC,* highlight the correct choice, then press RETURN.

After configuring the editor, select "QUIT" or press ESC from the main menu. The *CONFIGURE* program will ask you whether or not you want to save the editor with its changes. Specify *Y* for YES if you want the changes saved. Next you will be prompted for the file name to save it to. The default file name is the one that was loaded in. You may press RETURN to use the default name or enter a new name.

INDEX

PROGRAM WRITER COMMAND CHART

Editor Commands

*A-A: Auto line numbering
A-B: Beginning of line
*A-C: Copy (use with Paste)
*A-D: Delete lines
A-E: Change cursor mode
*A-F: Find text
*A-G: Get macros
A-I: Insert line
A-J: Jump to line number
*A-K: Calculator
A-L: Convert to lower case
*A-M: Edit macros
A-N: End of line
A-O: Override
*A-P: Paste (use with Copy)
A-Q: Quit editor
*A-R: Replace text
*A-S: Save macros
*A-T: Split into two lines
A-U: Convert to upper case
*A-V: List program variables
A-X: 40/80 columns
A-Y: Clear to end of line
A-Z: Remove editor

A-1 thru A-9: Position in program

*A-#: Renumber
A-.: Increase margin
A-,: Decrease margin

ESC: Restore line, stop macro, quit macro editor, redisplay screen
TAB: Move to next tab stop
A-TAB: Move to previous tab stop
RETURN: Next line, accept input
DELETE: Delete to left of cursor
A-DELETE: Delete at cursor

A-left arrow: Previous word
A-right arrow: Next word
A-up arrow: Previous screen
A-down arrow: Next screen

*Not available in EDITOR.SMALL

Macro Commands

ctrl-A: Execute editor command
ctrl-Z: Execute macro

Find Control Commands

ctrl-A: Alphabetic character
ctrl-B: Blank
ctrl-C: Control character
ctrl-D: Digit (0-9)
ctrl-E: End of line
ctrl-G: Garbage
ctrl-L: Literal mode
ctrl-N: Not
ctrl-R: Repeat
ctrl-S: To end of statement
ctrl-V: Alphabetic or digit
ctrl-X: Any single character
ctrl-\: Or

Apple II or II+ Replacements

IIe/IIc	II/II+
Open-apple	ctrl-A
Solid-apple	ctrl-Z
Up arrow	ctrl-K
Down arrow	ctrl-J
TAB	ctrl-I
DELETE	ctrl-D
ctrl-\	ctrl-O_