# Applesoft Search and Replace

*by Sandy Mossberg*
*50 Talcott Rd.*
*Rye Brook, NY 10573*

Applesoft Search and Replace (ASR) is a utility which performs either of two functions on the Applesoft program currently in memory:

**(1) Lists all program lines which contain a given string.**

**(2) Replaces one string with another throughout a program.** A wildcard character is supported, replacement may be selective and a null replacement string is accepted. I think you'll like it!

### COMMAND STRUCTURE AND SYNTAX

From immediate mode, **CTL-S** calls the SEARCH routine and **CTL-R** invokes the REPLACE function. Command syntax is listed below:

**CTL-S d-Sstring-(d-linerange/linenum)**

**CTL-R d-Sstring-d-Rstring-(d-linerange/linenum)**

**CTL-R d-Sstring-d-d**

Text within parentheses is optional. The **slash** means **either/or**. The last command replaces the search string with a null string, effectively deleting the search string.

Valid **delimiters (d)** are listed in LINES 207-208 of the A.L. program. A potential delimiter may be placed in either string if another delimiter is actually used. The most popular delimiters are the **quotation mark** and the **slash**. To search for the quote, use the slash as the valid delimiter, and vice versa. Unless a line number or range is required, the ending delimiter may be omitted.

### RULES AND EXAMPLES

**CTL-C will abort input. Strings may contain no more than 30 characters. Any control character may be included in a string by preceding its entry with CTL-O. If the character following CTL-O is not a control character, it will be rejected.**

Despite the listing format used by Applesoft and ASR, spaces exist only between quotation marks and after REM and DATA statements.

To search for the string **A = 15**, enter **CTL-S/A=15**, not **CTL-S/A = 15**.

To search for **MH E/P following a REM statement**, enter **CTL-S"MH E/P**, not **CTL-S" MHE/P**. **CTL-S/MH E/P** would also be incorrect because the delimiter is used within the string.

```
]LIST

10  REM THIS IS A TEST OF
20  REM SEARCH & REPLACE
30  PRINT "THIS IS A TEST OF:"
40  PRINT "     APPLESOFT"
50  PRINT "SEARCH AND REPLACE"
60  REM THE DELIMITER IS '/'
70  PRINT " TRY IT.. YOU'LL LIKE
    IT"

]/IS/
10  REM THIS IS A TEST OF
30  PRINT "THIS IS A TEST OF"
60  REM THE DELIMITER IS '/'
]
```

Line numbers from 0-65535 are accepted. A line range must be separated by a comma and will produce an error message if the ending range is smaller than or equal to the starting range. Failure to enter a line number or range will result in a complete program search.

### WILDCARD AND CHANGE-CHECKING

**The search string supports a universal match (wildcard) character.** I chose the "at" symbol (@), but you may redefine it by changing LINE 381 of the program listing. If found in the replace string, the wildcard character will be interpreted literally and placed into program memory.

In using the REPLACE feature, you are given an opportunity to check changes. If your answer is **No**, then all replacements will be made automatically.

```
]LIST

10  REM THIS IS A TEST OF
20  REM SEARCH & REPLACE
30  PRINT "THIS IS A TEST OF:"
40  PRINT "     APPLESOFT"
50  PRINT "SEARCH AND REPLACE"
60  REM THE DELIMITER IS '/'
70  PRINT " TRY IT.. YOU'LL LIKE
    IT"

]/IS/WAS/
CHECK CHANGES (Y/N)? N
10  REM THWAS WAS A TEST OF
30  PRINT "THWAS WAS A TEST OF"
60  REM THE DELIMITER WAS '/'
]
```

By accepting the option, each change will be presented for review. Pressing **Y** will direct the change to be made, **N** will suppress the change (the original line will reappear) and **CTL-C** will return you to BASIC. (**Displays 3-5** show a sequence of checks being performed as the Replace proceeds through the program.)

```
]LIST

10  REM  THIS IS A TEST OF
20  REM  FIND AND REPLACE
30  PRINT "THIS IS A TEST OF"
40  PRINT "     APPLESOFT"
50  PRINT "FIND AND REPLACE"
60  REM  THE DELIMITER IS '/'
70  PRINT " TRY IT.. YOU'LL LIKE
    IT"

]/IS/WAS/
CHECK CHANGES (Y/N)? Y
 10  REM  THWAS IS A TEST OF
```

```
]LIST

10  REM  THIS IS A TEST OF
20  REM  FIND AND REPLACE
30  PRINT "THIS IS A TEST OF"
40  PRINT "     APPLESOFT"
50  PRINT "FIND AND REPLACE"
60  REM  THE DELIMITER IS '/'
70  PRINT " TRY IT.. YOU'LL LIKE
    IT"

]/IS/WAS/
CHECK CHANGES (Y/N)? Y
10  REM  THIS IS A TEST OF
10  REM  THIS WAS A TEST OF
```

```
10  REM  THIS IS A TEST OF
20  REM  FIND AND REPLACE
30  PRINT "THIS IS A TEST OF"
40  PRINT "     APPLESOFT"
50  PRINT "FIND AND REPLACE"
60  REM  THE DELIMITER IS '/'
70  PRINT " TRY IT.. YOU'LL LIKE
    IT"

]/IS/WAS/
CHECK CHANGES (Y/N)? Y
10  REM  THIS IS A TEST OF
10  REM  THIS WAS A TEST OF
30  PRINT "THIS IS A TEST OF"
30  PRINT "THIS WAS A TEST OF"
60  REM  THE DELIMITER WAS '/'
```

Any other response will ring a bell and encourage the cursor to continue blinking its eye at you.

### SYNTAX ERRORS

?SYNTAX ERROR messages occur for a variety of reasons. A line number larger than 65535 produces an ?ILLEGAL QUANTITY ERROR.

ASR internally generates a !LINE #XX TOO LONG message if a replacement causes BUF to exceed 249 characters (program aborted) or if a line in program memory detokenizes to more than 249 characters (program continues at next line).

When a replacement moves the end of a program to within $100 bytes of HIMEM, an ?OUT OF MEMORY ERROR halts the program.

I control scrolling by keeping my right forefinger on the RETURN key (start and stop listing) and my left forefinger on the ESCAPE key (abort listing).

### INSTALLING THE PROGRAM

The machine language code may be entered directly beginning at $9000. (The NIBBLE Machine Language Editor is a wonderful tool for this purpose[1]), or the Assembly Language source program may be assembled. MERLIN (Southwestern Data Systems) was used for the appended listing and should present no interpretative difficulty for those who possess other assemblers. Once the code is in memory, **BSAVE ASR, A$9000, L$518**.

**For more instructions on entering Machine Language directly into memory, see the Letters section of this issue.**

Be sure to place ASR on a normal diskette that sets **HIMEM** to **$9600** on booting. On installing the program (BRUN ASR from immediate mode or PRINT CHR$(4)"BRUN ASR" from deferred mode), HIMEM is set to $9000, thus offering reasonable protection. The BASIC command **FP** and the system monitor commands **CTL-B** and **E000G** will reset HIMEM to $9600 getting **MAXFILES3** or less will reset HIMEM to a variable value, exposing but not harming ASR. Immediately pressing RESET or CALLing 36864 will restore HIMEM to $9000.

**A MAXFILES value larger than 3 will cause the functioning DOS buffers to clobber ASR.**

The RESET key will fix ASR hooks (and HIMEM:36864) and return the user to Applesoft. The ampersand is a convenience which disconnects ASR by reestablishing normal I/O hooks. Do not hesitate to use Mr. Ampersand to call another machine language program, since IN#0 and PR#0 will unhook ASR input and output, respectively.

Those of you who read my other ramblings will wonder why I did not place ASR in my favorite location, between DOS and its buffers. The answer is, "I just didn't feel like doing it!" Can you understand that? If so, you're as nutty as I am! If you really want the protection of DOS, refer to the subscripted[2] article for explicit instructions.

Using ASR concurrently with PLE (Synergistic Software) requires a few modifications:

(1) Replace LINES 116-118 with a single line:

**UNHOOK JMP $9700 ;PLE entry**

(2) Assemble the altered program at $8A00 and BSAVE ASR.PLE, A$8A00, L$512.

(3) RUN PLE first.

(4) BRUN ASR.PLE. Now the ampersand will install the PLE hooks and RESET will establish the ASR hooks. Switching between these two vectors (or reversing them) is quite simple.

## DECISIONS, DECISIONS

When faced with a programming task, the first order of business is to turn off your Apple, pick up a pad and pencil and start planning your tactics. Charles Engelsher's recent article in NIBBLE[3] brilliantly illustrates this point. I am not "into" flowcharts mainly because they create in me an irrepressible urge to yawn. Lists are my thing. In the early stages of development, ASR seemed straightforward, but as lists turned into sublists, several nuances became apparent. Decisions had to be made.

Most basic was the method of controlling output and input. Since a search might uncover many matches, lines would fly by unless scrolling could be manipulated. Clearly, this called for control of the **output (CSW) hook**. Input could be handled by Mr. Ampersand or MAMA[4], but the most elegant method would be to employ the **input (KSW) hook**. So far so good.

## APPLESOFT LINE STRUCTURE

A more sticky problem was the manner in which the program was to be searched. By now you should be well-acquainted with Applesoft line structure[5].

To review briefly, each Applesoft program line is stored in the following way:

**(1) Bytes 0-1 (link bytes) contain the address of the beginning of the next line.**

**(2) Bytes 2-3 hold the line number. As usual, LSB precedes MSB.**

**(3) Bytes 4 through n contain the contents of the line in tokenized form.**

**(4) Byte n+1 holds a hex zero which marks the end of the line (EOL). When the link bytes of a given line are both zero, the end of the program (EOP) has been reached.**

The important feature of this description is "tokenization". To conserve space, Applesoft represents keywords[6] with negative ASCII tokens (except when the reserved word is enclosed in quotes or follows a REM or DATA statement). All other line characters are in positive ASCII. Encoding (parsing) and decoding (listing) keywords is accomplished by referring to a table at **$D0D0-$D25F**. Consider the following line and its tokenized equivalent:

## MORE DECISIONS

If one wanted to search for the string "SUB", one would not find it in the CONTENT column because it comprises a portion of the keyword "GOSUB"; yet "SUB" surely exists in LINE 100. Should accuracy then depend upon the search string containing complete keywords? My firm decision was to be maximally flexible by decoding the program line in the **input buffer (BUF)** and allowing an **unrestricted literal search**. In this way one could replace "SUB" with "TO" to form "GOTO", or replace "MDATA" with "STORE" to create "RESTORE". More about this later.

The third major decision was whether to allow the user a chance to **examine a line following replacement and to accept or reject that change**. Less than 5 nanoseconds were required for my affirmative response. Unfortunately, this raised other questions. If the user opted to examine each changed line and if a given line contained more than one match, should the user be able to check each individual change or should the entire changed line be presented for review? Again, I selected the more flexible former option. Finally, if a change was rejected, should the screen display revert to the original form? Of course!

On examining my decisions, it became obvious that I repeatedly was attracted to **user-friendliness** despite the overhead of increased complexity and code. At this stage I felt much like a latter-day saint. One week later, after completing the program, I peered into the mirror and saw Daffy Duck instead of St. Sandy!

## HOW ASR WORKS — INITIALIZATION

Usually, I define local storage locations and flags within program space. Employing page zero would save space (1 byte per instruction), but often not enough unused zero page addresses are available. For many moons I have wondered why the code at **$F1C2-$F1D4** utilizes $00-$05 for jump vectors to the Applesoft main command loop ($D43C) and STROUT ($DB3A). Nowhere can I find a routine that refers to these vectors; in fact, I suspect that they are superfluous (not a unique situation for Applesoft). In ASR, despite heavy usage of Applesoft ROM subroutines, I took the plunge and used these locations. No difficulty was encountered, thus fortifying my hunch. Local pointers occupy 4 patently safe addresses, $18-$19 and $1E-$1F.

The opening code (LINES 84-107) points **the RESET vector** at RESETEV (LINE 111), **the ampersand vector** at UNHOOK (LINE 116), **HIMEM** at the program start (LINE 84), **the input hook** at KBDEV (LINE 141) and the **output hook** at SCREV (LINE 122). The **RESET** key will reconnect program hooks **and the** ampersand will replace program **hooks** with normal hooks (IN#0 and PR#0 **accomplish** the same feat).

## OUTPUT CONTROL

**LINES** 122-137 allow output, e.g. LIST, to **be aborted** by pressing ESCAPE, to be halted **by hitting** any other key and to be restarted by **pressing** yet another non-ESCAPE key. Since **a carriage** return initiates this process, a **pause** always occurs at the end of a printed **statement** or line.

## OBTAINING INPUT

LINES 141-157 poll keyboard input for three parameters: **mode, character and position**. In deferred (program execution) mode, PROMPT contains a positive value (6) rather than the negative value ($DD) of the normal Applesoft prompt character. Only the immediate (command) mode will allow ASR to continue. If command mode is operative, CTL-R (REPLACE) and CTL-S (SEARCH) are the two acceptable characters. On encountering a valid character in immediate mode, the cursor must be in the second column (CH=1) and the prompt must be immediately to the left (CH=0). The latter criterion is not water-tight, but its defeat would be an act of wanton cruelty (if you dunno how, I ain't gonna squeal)! If all 3 criteria are met, the character is placed on the screen (inverse "R" or inverse "S", LINES 161-163) and RPLFLG is set to minus ($FF) if REPLACE is invoked, or to plus (0) if SEARCH is requested (LINES 164-166).

Why did I write my own input routine (LINES 170-211) when GETLN ($FD6A) and NXTCHAR ($FD75) already are built into monitor ROM? My main reason was to provide the user with a **visible control character** in the search or replace string (no ROM routine does this).

A secondary gain was the ability to disable the backspace key when no input has been made. Placing CTL-H (backspace) or CTL-U (forward copy) directly into a string would be unsuccessful, and entering CTL-M (RETURN) would end the input. Thus, an override function was provided so that **any control character could be accepted after pressing CTL-O** (LINES 181-188). Since search and replace strings will later be limited to 30 characters, the maximal number of characters allowed by this segment is 74 (2 strings of 30, 3 delimiters, 10 numerals and 1 comma). RDCHAR was chosen to gather data in preference to RDKEY so that the ESCAPE key would be active.

When input is completed (RETURN pressed), BUF is converted to Applesoft format by GDBUFS (LINE 204) which changes negative to positive ASCII and sets the zero end of buffer (EOB) marker. If the delimiter is valid, it is saved in BUFCNT.

**The search string is stored in SBUF (LINES 215-226) and may not exceed 30 characters.** The carry bit is cleared if no delimiter ends the string and is set if a second delimiter is present. The size of the search string is saved in SRCHSIZ (LINE 226). On testing RPLFLG (LINE 227) a search request routes flow to MAXRNG on carry clear (no ending delimiter) and to FNDRNG on carry set (ending delimiter entered). This program logic is simple — the absence of a second delimiter precludes the possibility of a range entry, whereas the presence of a terminal delimiter indicates that a range may have been given. A replace request is a bit more complex. Absence of a second delimiter (at least two must be present) or absence of input after a second delimiter forces a syntax error (LINES 231-234). LINES 238-250 place the replace string in RBUF. No more than 30 characters are allowed, and a terminal delimiter has the same implications as described for the search string.

# Applesoft Search and Replace (Cont.)

## LINE RANGES

FNDRNG (LINES 255-298) first checks for a given range and branches to MAXRNG if none is found. FRMNUM and GETADR place each range into LINNUM. A single line number is accepted (LINES 276-278) and a comma must separate starting and ending ranges (LINE 279). FNDLIN points LOWTR to the address of the starting range. LINES 262-267 prevent entry of an "E", the only non-numeric character that "fools" FRMNUM, which interprets it as "exponent". If you have been reading DISASSEMBLY LINES, you will be aware of this pitfall. As FNDRNG ends, the ending range is stored in LINNUM. Please note that the expedient of using LINGET, the normal Applesoft routine, in place of FRMNUM/GETADR was eschewed because LINGET rejects any line number larger than 63999, and we certainly desire the ability to search all possible line numbers. Are you listening, Applesoft?

MAXRNG (LINES 299-305) merely points LOWTR at the first program line and sets LINNUM to 65535, the highest possible line number.

The option of checking replacement changes is provided in LINES 311-317. A positive response sets CCFLG to negative, whereas a negative response leaves the previously zeroed CCFLG (LINE 239) positive. A handy print routine is utilized by setting a pointer to the desired ASCII string (which must end with a hex zero) and calling PRINT (LINES 619-626). The subroutine to process the "yes" or "no" response (INPYN, LINES 628-646) provides exit from the program by pressing CTL-C (LINES 629-630, 639-646). A negative answer clears the carry flag; a positive reply set it.

## DECODING AND MATCHING

Regardless of which choices have been made, control now passes to SETBUF (LINES 322-365) which checks for the true EOP (LINES 327-329) or user-defined ending range (LINES 332-341). If neither is found the current line (LOWTR points to the start) is detokenized by an adaptation of the Applesoft ROM LIST routine. STORBUF (LINES 648-668) places each character in BUF.

If more than 249 characters appear, the END flag (BUFINDX) is cleared, !LINE #XXX TOO LONG is printed and the next line is processed by SETBUF. Decoding, per se, is done by storing all positive ASCII characters directly in BUF and by using the negative ASCII token as an index to the keyword table (TKNSET, LINES 670-679).

FNDLIT (LINES 369-392) scans BUF for occurrences of the search string in SBUF. Recognition of the wildcard figure @ (LINES 381-382) causes an automatic character match. A complete search string match occurs when an appropriate number (SRCHSIZ) of character matches has been made, in which instance MTCHFLG is set, RPLFLG is tested and flow branches to either the SEARCH or the REPLACE routine.

## SEARCH MATCH

When a search has been requested, a single search string match causes printing of the current line in program memory. PRLNNMB (LINES 687-693) places the low and high order line number bytes into (X,A) and prints the decimal line number via LINPRT. A somewhat different adaptation of LIST, PRLN (LINES 695-721) presents the line in Applesoft format but uses 40 instead of 33 columns. Since SETBUF and PRLN are variations on the same theme, why did I not combine them and save bytes? Actually, an earlier version of ASR did just this, but program definition was garbled. Thus, in this instance I opted for a slightly longer but "cleaner" program. This short segment ends by pointing LOWTR to the beginning of the next line (NXLTR, LINES 723-731) and reentering SETBUF.

## REPLACE MATCH

The replace routine is vastly more complex. By subtracting the size of the search string from that of the replace string and storing the result in BUFDIFF, we determine whether BUF must be expanded (BUFDIFF negative) or contracted (BUFDIFF positive) (LINES 411-415). Expansion of the program calls for assurance that HIMEM will not be invaded (LINES 416-421) and that BUF will not exceed 249 characters (LINES 422-430). If expansion brings the program within $100 bytes of HIMEM, an ?OUT OF MEMORY ERROR is generated.

Exceeding maximal BUF size sets the END flag (LINE 429), prints !LINE #XXX TOO LONG and aborts ASR. If these two traps are escaped, the appropriate segment of BUF is moved upward (LINES 431-438) or downward (LINES 439-447) by an amount equal to BUFDIFF. Unless a null replacement string exists (LINES 448-449), LINES 450-457 replace the matched search string with the replace string. BUF is retokenized (LINES 458-464) and PGMDIFF is determined by subtracting the content size of the new line from that of the old one (which was calculated in LINE 345). A positive value for PGMDIFF calls for increased program size and a negative value, decreased size.

Please note that **expansion** of BUF length might result in **contraction** of program length. Changing PR = 1 (4 unparsed BUF characters and 4 parsed program line characters) to PRINT I (6 BUF characters, 2 program characters) is an example of this phenomenon. The reverse also may be true.



```
1             **********************
2             *                    *
3             *     APPLESOFT      *
4             *  SEARCH & REPLACE  *
5             *                    *
6             **********************
7
8                     ORG  $9000
9
10            * Local program equates:
11
12            BUFCNT   =   $00       ;Delimiter, BUF count, Temp
13            BUFINDX  =   $01       ;BUF index, Match point, End flag
14            SRCHSIZ  =   $02       ;Length SEARCH string
15            RPLSIZ   =   $03       ;Length REPLACE string
16            BUFDIFF  =   $04       ;BUF differential
17            PGMDIFF  =   $05       ;Line length, Pgm differential
18            CVSAV    =   $06       ;Save row
19            RPLFLG   =   $07       ;SEARCH/REPLACE flag
20            CCFLG    =   $08       ;CHECK CHANGES flag
21            MTCHFLG  =   $09       ;MATCH flag
22            PTRA     =   $18       ;Pointer
23            PTRB     =   $1E       ;Pointer
24
25            * General equates:
26
27            WNDBTM   =   $23       ;Bottom + 1 of text window
28            CH       =   $24       ;Cursor column
29            CV       =   $25       ;Cursor row
30            BASL     =   $28       ;Left end of current row
31            PROMPT   =   $33       ;Basic prompt char
32            CSWL     =   $36       ;Output hook
33            KSWL     =   $38       ;Input hook
34            LINNUM   =   $50       ;Line #, End user-defined range
35            TXTTAB   =   $67       ;Start of pgm
36            VARTAB   =   $69       ;LOMEM
37            MEMSIZ   =   $73       ;HIMEM
38            FORPNT   =   $85       ;PRINT & TKNTBL index
39            HIGHDS   =   $94       ;BLTU dest of high addr + 1
40            HIGHTR   =   $96       ;BLTU high addr to move + 1
41            LOWTR    =   $9B       ;BLTU low addr, Line start
42            DSCTMP   =   $9D       ;Pointer to keyword table
43            PRGEND   =   $AF       ;End of pgm
44            TXTPTR   =   $B8       ;Text pointer
45            BUF      =   $200      ;Input buffer
46            DOSWRM   =   $3D0      ;DOS warmstart
47            DOSHOOK  =   $3EA      ;Connect I/O hooks to DOS
48            RESET    =   $3F2      ;RESET handler
49            AMPER    =   $3F5      ;Ampersand vector
50            KBD      =   $C000     ;Keyboard data input
51            KBDSTRB  =   $C010     ;Clear keyboard strobe
52            TKNTBL   =   $D0D0     ;Applesoft keyword table
53            BLTU     =   $D393     ;Block transfer upward
54            MEMERR   =   $D410     ;?OUT OF MEMORY ERROR
55            GDBUFS   =   $D539     ;Zero end of BUF & clear hi bits
56            PARSE    =   $D559     ;Tokenize BUF from TXTPTR to EOB
57            FNDLIN   =   $D61A     ;Point LOWTR to line in LINNUM
58            CLEARC   =   $D66C     ;Clear variables and stack
59            CRDO     =   $DAFB     ;Print CR via OUTDO
60            OUTSP    =   $DB57     ;Print space via OUTDO
61            OUTDO    =   $DB5C     ;Set hi bit, print (A) via COUT
62            FRMNUM   =   $DD67     ;Put formula at TXTPTR into FAC
63            CHKCOM   =   $DEBE     ;Check for comma at TXTPTR
64            SYNERR   =   $DEC9     ;?SYNTAX ERROR
65            GETADR   =   $E752     ;Convert FAC to integer in LINNUM
66            LINPRT   =   $ED24     ;Print decimal of (A,X)
67            TABV     =   $FB5B     ;Put cursor at row in (A)
68            SETPWRC  =   $FB6F     ;Validate RESET vector
69            BS       =   $FC10     ;Backspace
70            UP       =   $FC1A     ;Cursor up
71            CLREOP   =   $FC42     ;Clear to end of screen
72            RDKEY    =   $FD0C     ;Call KEYIN via KSWL
73            KEYIN    =   $FD1B     ;Read char into (A)
74            RDCHAR   =   $FD35     ;Call RDKEY, ESC active
75            COUT     =   $FDED     ;Output char in (A) via CSWL
76            COUT1    =   $FDF0     ;Output char in (A) directly
77            SETKBD   =   $FE89     ;Set usual keyboard vector
78            SETVID   =   $FE93     ;Set usual screen vector
79            BELL     =   $FF3A     ;Output bell char via CSWL
80
81            *------------------------------------
82            * SET HOOKS, RESET, HIMEM AND MR. &
83            *------------------------------------
9000: A0 37    84  SETEV    LDY  #<RESETEV  ;RESET
9002: A9 90    85           LDA  #>RESETEV
9004: 8C F2 03 86           STY  RESET
9007: 8D F3 03 87           STA  RESET+1
900A: 20 6F FB 88           JSR  SETPWRC
900D: A2 4C    89           LDX  #$4C       ;Ampersand
900F: A0 3D    90           LDY  #<UNHOOK
9011: A9 90    91           LDA  #>UNHOOK
9013: 8E F5 03 92           STX  AMPER
9016: 8C F6 03 93           STY  AMPER+1
9019: 8D F7 03 94           STA  AMPER+2
901C: A0 00    95  SEV1     LDY  #<SETEV    ;HIMEM
901E: A9 90    96           LDA  #>SETEV
9020: 84 73    97           STY  MEMSIZ
9022: 85 74    98           STA  MEMSIZ+1
9024: A0 6D    99           LDY  #<KBDEV    ;Input hook
9026: A9 90    100          LDA  #>KBDEV
```

## DISPLAY

On testing CCFLG, if changes are not to be checked, flow moves to CHNG1. In this circumstance, when all replacements have been made to a given line, printing of the line is performed in LINES 374-378. Checking changes (LINES 475-492) first requires printing of a prompt, the line number and the parsed contents of BUF. The start of the line is relocated (scrolling may have changed its position) by finding the prompt and INPYN again is called.

If the change is rejected (carry clear) (LINES 496-509), the match point (BUFINDX) is reset to the character beyond the end of the rejected match, an inverse "N" replaces the flashing cursor and the original line is restored to the screen by printing the unchanged line in program memory. SETBUF is reentered at SB1, thus preserving BUFINDX so that the search may resume at the correct position in BUF.

Accepting the proposed change (LINES 513-570) triggers printing of an inverse "Y" and moving all lines above the current one higher or lower in program memory by an amount equal to PGMDIFF. The upward move employs BLTU, whose parameters are set in LINES 523-534. This Applesoft ROM subroutine requires (A) to equal HIGHDS and (Y) to equal HIGHDS+1. Observe that BLTU transfers the **end** (+1) of the moved code to the destination (+1), whereas MOVE ($FE2C), the monitor ROM routine, transfers the **origin** of the moved code to the destination. The downward move (LINES 541-564) employs the link bytes and PGMDIFF to set move parameters and VARTAB to test the end of the move. MVBUF transfers the tokenized contents of BUF into program space.

Finally, LINES 574-615 reset link bytes, VARTAB, PRGEND and BUFINDX to compensate for altered program size. If the length of the search and replace strings is equal, this segment could be bypassed, but since function is quite rapid, I elected to conserve a few bytes at the expense of unneeded speed. The link resetting sequence (LINES 599-615) parallels the Applesoft ROM code at $D50F-$D52B. It simply places the address of the start of the next line into bytes 0-1 of the current line. On conclusion (LINE 598) control returns to SB1.

There you have it, folks — APPLESOFT SEARCH and REPLACE. I have done extensive debugging; however, to my skeptical mind, an error-free program of this size is one that has not been used long enough by a sufficient number of persons. Should you locate any vermin, I shall be happy to exterminate them.

## REFERENCES


1. **Sprinkle, D., Apple M.L.E. (Machine Language Editor), NIBBLE (Vol.3, No.2, 1982), p. 15 — see also the Letters page of this issue.**
2. **Mossberg, S., LAMP — Part II, NIBBLE (Vol.3, No.3, 1982), p.33**
3. **Engelsher, C., Structured Program Development — Part I, NIBBLE (Vol.3, No.3, 1982), p.53**
4. **Mossberg, S., MAMA (Move Aside Mr. Ampersand), NIBBLE (Vol.3, No.1, 1982), p.105**
5. **Anderson, L.H., Cohen, D., and Searle, R.F., LISZT with Strings, MICRO (May 1982), p.37**
6. **APPLESOFT ][ BASIC PROGRAMMING REFERENCE MANUAL, Apple Computer Inc., 1979, p.121**




```
9028: 84 38    101          STY  KSWL
902A: 85 39    102          STA  KSWL+1
902C: A0 46    103          LDY  #<SCREV     ;Output hook
902E: A9 90    104          LDA  #>SCREV
9030: 84 36    105          STY  CSWL
9032: 85 37    106          STA  CSWL+1
9034: 4C EA 03 107          JMP  DOSHOOK     ;Connect hooks to DOS
               108 *————————
               109 * RESET VECTOR
               110 *————————
9037: 20 1C 90 111 RESETEV  JSR  SEV1        ;RESET program hooks and
903A: 4C C4 91 112          JMP  END         ;  return to BASIC
               113 *————————
               114 * DISCONNECT VECTOR
               115 *————————
903D: 20 89 FE 116 UNHOOK   JSR  SETKBD      ;AMPERSAND disconnects program
9040: 20 93 FE 117          JSR  SETVID      ;  hooks and sets usual hooks
9043: 4C EA 03 118          JMP  DOSHOOK
               119 *————————
               120 * OUTPUT VECTOR - SCROLL CONTROL
               121 *————————
9046: C9 8D    122 SCREV    CMP  #$8D        ;CR?
9048: D0 1A    123          BNE  OUTPUT      ;No. Print char
904A: AD 00 C0 124          LDA  KBD         ;Yes. Keypress?
904D: 10 13    125          BPL  CRPUT       ;No. Print CR
904F: 8D 10 C0 126          STA  KBDSTRB     ;Yes. Clear strobe
9052: C9 9B    127          CMP  #$9B        ;ESC?
9054: F0 11    128          BEQ  ENDPUT      ;Yes. Abort to basic
9056: AD 00 C0 129 KEYPR    LDA  KBD         ;No. Keypress?
9059: 10 FB    130          BPL  KEYPR       ;No. Wait
905B: 8D 10 C0 131          STA  KBDSTRB     ;Yes. Clear strobe
905E: C9 9B    132          CMP  #$9B        ;ESC?
9060: F0 05    133          BEQ  ENDPUT      ;Yes. Abort to BASIC
9062: A9 8D    134 CRPUT    LDA  #$8D        ;No. Put CR in (A)
9064: 4C F0 FD 135 OUTPUT   JMP  COUT1       ;Print char
9067: 20 FB DA 136 ENDPUT   JSR  CRDO        ;Back to BASIC
906A: 4C D0 03 137          JMP  DOSWRM      ;  via DOS
               138 *————————
               139 * INPUT VECTOR - CHECK FOR VALID COMMAND
               140 *————————
906D: 20 1B FD 141 KBDEV    JSR  KEYIN       ;Get char
9070: 24 33    142          BIT  PROMPT      ;Valid mode IMMEDIATE (PROMPT
9072: 10 17    143          BPL  RTS1        ;  contains neg value - $DD)
9074: C9 93    144          CMP  #$93        ;Valid chars CTL-S and CTL-R
9076: F0 04    145          BEQ  CMDPSN
9078: C9 92    146          CMP  #$92
907A: D0 0F    147          BNE  RTS1
907C: 48       148 CMDPSN   PHA              ;Save command char
907D: A4 24    149          LDY  CH          ;Valid position at column 1
907F: C0 01    150          CPY  #1          ;  with prompt (]) at column 0
9081: D0 07    151          BNE  WRGPSN
9083: 88       152          DEY
9084: B1 28    153          LDA  (BASL),Y
9086: C5 33    154          CMP  PROMPT
9088: F0 02    155          BEQ  WHICH
908A: 68       156 WRGPSN   PLA              ;Invalid command
908B: 60       157 RTS1     RTS
               158 *————————
               159 * SEARCH OR REPLACE?
               160 *————————
908C: 68       161 WHICH    PLA              ;Restore command char
908D: 29 7F    162          AND  #$7F        ;Clear hi bit
908F: 20 ED FD 163          JSR  COUT        ;Print command inversely
9092: 38       164          SEC              ;Set S/R flag:
9093: E9 13    165          SBC  #'S'-$40    ;  $FF=REPLACE
9095: 85 07    166          STA  RPLFLG      ;  $00=SEARCH
               167 *————————
               168 * GET INPUT
               169 *————————
9097: 20 35 FD 170 GETINP   JSR  RDCHAR      ;Get char (ESC active)
909A: C9 8D    171          CMP  #$8D        ;CR
909C: F0 3C    172          BEQ  RTN
909E: C9 88    173          CMP  #$88        ;<—
90A0: F0 2F    174          BEQ  BCK
90A2: C9 95    175          CMP  #$95        ;—>
90A4: D0 04    176          BNE  CTLTEST
90A6: B1 28    177          LDA  (BASL),Y    ;If —> pick up current char
90A8: D0 16    178          BNE  ADDINP      ;Always
90AA: C9 A0    179 CTLTEST  CMP  #" "        ;CTL char?
90AC: B0 12    180          BCS  ADDINP      ;No. Print it
90AE: C9 8F    181          CMP  #$8F        ;Yes. CTL-O?
90B0: F0 05    182          BEQ  CTLO        ;Yes
90B2: 20 3A FF 183 CTLNG    JSR  BELL        ;No. Reject it and get
90B5: 10 E0    184          BPL  GETINP      ;  another char
90B7: 20 0C FD 185 CTLO     JSR  RDKEY       ;CTL-O. Get char
90BA: C9 A0    186          CMP  #" "        ;CTL char?
90BC: B0 F4    187          BCS  CTLNG       ;No. Reject it
90BE: 29 7F    188          AND  #$7F        ;Yes. Clear hi bit
90C0: 9D 00 02 189 ADDINP   STA  BUF,X       ;Store char in BUF
90C3: 20 ED FD 190          JSR  COUT        ;Print char
90C6: E8       191          INX              ;Bump BUF index
90C7: E0 4B    192          CPX  #75         ;Less than 75 chars?
90C9: 90 CC    193          BCC  GETINP      ;Yes. Back for more
90CB: 20 1C 90 194 ERR1     JSR  SEV1        ;No. Remind DOS of pgm hooks
90CE: 4C C9 DE 195          JMP  SYNERR      ;?SYNTAX ERROR
90D1: 8A       196 BCK      TXA              ;Disable <— if no input
90D2: F0 C3    197          BEQ  GETINP
90D4: 20 10 FC 198          JSR  BS          ;Backspace
90D7: CA       199          DEX              ;Dec BUF index
90D8: 10 BD    200          BPL  GETINP      ;Always
90DA: 9D 00 02 201 RTN      STA  BUF,X       ;Store CR in BUF
90DD: 20 42 FC 202          JSR  CLREOP
90E0: 20 FB DA 203          JSR  CRDO
90E3: 20 39 D5 204          JSR  GDBUFS      ;Convert to pos ASCII & zero EOB
90E6: AD 00 02 205          LDA  BUF         ;Get delimiter
90E9: F0 E0    206          BEQ  ERR1        ;Valid delimiters are:
90EB: C9 30    207          CMP  #'0'        ;  ! " # $ % & ' (
90ED: B0 DC    208          BCS  ERR1        ;  ) * + , - . /
90EF: C9 21    209          CMP  #'!'
90F1: 90 D8    210          BCC  ERR1
90F3: 85 00    211          STA  BUFCNT      ;Save delimiter
               212 *————————
               213 * SAVE SEARCH STRING
               214 *————————
90F5: E8       215          INX              ;Zero (X) - see GDBUFS
90F6: BD 01 02 216 SRCHSAV  LDA  BUF+1,X     ;Put SEARCH string into SBUF
90F9: 9D D7 94 217          STA  SBUF,X
90FC: F0 0B    218          BEQ  NODELIM     ;String ends with no delimiter
90FE: C5 00    219          CMP  BUFCNT
9100: F0 08    220          BEQ  DELIM       ;String ends with delimiter
9102: E8       221          INX
9103: E0 1F    222          CPX  #31
9105: B0 C4    223          BCS  ERR1        ;Syntax error on > 30 chars
9107: 90 ED    224          BCC  SRCHSAV     ;Always
9109: 18       225 NODELIM  CLC              ;CC=no delimiter
910A: 86 02    226 DELIM    STX  SRCHSIZ     ;CS=delimiter. Save size S string
910C: 24 07    227          BIT  RPLFLG
910E: 30 04    228          BMI  RPL
9110: B0 27    229          BCS  FNDRNG      ;SEARCH. If delimiter find range.
9112: 90 7B    230          BCC  MAXRNG      ;  If no delimiter set max range
```

*continued on next page*

# Applesoft Search and Replace

```
9114: 90 B5     RPL     BCC  ERR1         ;REPLACE. Abort if no 2nd
9116: E8                INX               ;  delimiter or no output
9117: BD 01 02          LDA  BUF+1,X      ;  after 2nd delimiter
911A: F0 AF             BEQ  ERR1
                *-------------------
                * SAVE REPLACE STRING
                *-------------------
911C: A0 00             LDY  #0
911E: 84 08             STY  CCFLG        ;Zero flags
9120: 84 03             STY  RPLSIZ
9122: BD 01 02  RPLSAV  LDA  BUF+1,X      ;Put REPLACE string into RBUF
9125: 99 F7 94          STA  RBUF,Y
9128: F0 65             BEQ  MAXRNG       ;String ends with no delimiter
912A: C5 00             CMP  BUFCNT
912C: F0 0B             BEQ  FNDRNG       ;String ends with delimiter
912E: E8                INX
912F: C8                INY
9130: 84 03             STY  RPLSIZ
9132: C0 1F             CPY  #31
9134: 90 EC             BCC  RPLSAV       ;Syntax error on > 30 chars
9136: 4C CB 90  ERR2    JMP  ERR1
                *-------------------
                * SET RANGE PARAMETERS
                *-------------------
9139: E8        FNDRNG  INX
913A: E8                INX
913B: BD 00 02          LDA  BUF,X        ;Get char after final delimiter
913E: F0 4F             BEQ  MAXRNG       ;No range given. Set max range
9140: 86 B8             STX  TXTPTR       ;Range given. Point TXTPTR
9142: A9 02             LDA  #2           ;  at starting range
9144: 85 B9             STA  TXTPTR+1
9146: BD 00 02  FR1     LDA  BUF,X        ;Don't allow that troublesome
9149: F0 07             BEQ  FR2          ;  "E" to sneak in
914B: C9 45             CMP  #'E'
914D: F0 E7             BEQ  ERR2
914F: E8                INX
9150: D0 F4             BNE  FR1
9152: 20 1C 90  FR2     JSR  SEV1         ;Remind DOS of pgm hooks
9155: 20 67 DD          JSR  FRMNUM       ;Put starting range
9158: 20 52 E7          JSR  GETADR       ;  into LINNUM
915B: A5 51             LDA  LINNUM+1
915D: 48                PHA               ;Save start hi
915E: A5 50             LDA  LINNUM
9160: 48                PHA               ;Save start lo
9161: 20 1A D6          JSR  FNDLIN       ;Set LOWTR to start
9164: A0 00             LDY  #0
9166: B1 B8             LDA  (TXTPTR),Y
9168: F0 33             BEQ  CHKCHNG      ;No ending range given
916A: 20 BE DE          JSR  CHKCOM       ;Comma must separate ranges
916D: A5 9B             LDA  LOWTR        ;Save LOWTR
916F: 48                PHA
9170: A5 9C             LDA  LOWTR+1
9172: 48                PHA
9173: 20 67 DD          JSR  FRMNUM       ;Put ending range into FAC
9176: A0 00             LDY  #0
9178: B1 B8             LDA  (TXTPTR),Y   ;Disallow non-numeric char
917A: D0 BA             BNE  ERR2         ;  following ending range
917C: 20 52 E7          JSR  GETADR       ;Put ending range into LINNUM
917F: 68                PLA               ;Restore LOWTR
9180: 85 9C             STA  LOWTR+1
9182: 68                PLA
9183: 85 9B             STA  LOWTR
9185: 68                PLA               ;Restore starting range and
9186: C5 50             CMP  LINNUM       ;  compare to ending range.
9188: 68                PLA
9189: E5 51             SBC  LINNUM+1
918B: B0 A9             BCS  ERR2         ;Syntax error if start >= end
918D: 90 0E             BCC  CHKCHNG      ;Always
918F: A5 67     MAXRNG  LDA  TXTTAB       ;Set max range by placing zero
9191: 85 9B             STA  LOWTR        ;  into LOWTR and $FFFF (65535)
9193: A5 68             LDA  TXTTAB+1     ;  into LINNUM
9195: 85 9C             STA  LOWTR+1
9197: A9 FF             LDA  #$FF
9199: 85 50             STA  LINNUM
919B: 85 51             STA  LINNUM+1
                *-------------------
                * CHECK CHANGES?
                *-------------------
919D: 24 07     CHKCHNG BIT  RPLFLG       ;REPLACE routine?
919F: 10 13             BPL  SETBUF       ;No
91A1: A0 AC             LDY  #<TXCC       ;Yes
91A3: A9 94             LDA  #>TXCC
91A5: 20 D5 93          JSR  PRINT        ;Print CHECK CHANGES (Y/N)?
91A8: 20 E5 93          JSR  INPYN        ;Get user response
91AB: B0 02             BCS  CC1          ;YES
91AD: C6 08             DEC  CCFLG        ;NO. Set flag
91AF: 91 28     CC1     STA  (BASL),Y     ;Print Y or N
91B1: 20 FB DA          JSR  CRDO
                *--------------------------------
                * PUT DETOKENIZED LINE INTO INPUT BUFFER
                *--------------------------------
91B4: A0 FF     SETBUF  LDY  #$FF
91B6: 84 01             STY  BUFINDX
91B8: 46 09             LSR  MTCHFLG      ;Clear flag
91BA: A0 FF     SB1     LDY  #$FF
91BC: 84 05             STY  PGMDIFF
91BE: A0 01             LDY  #1           ;If link byte hi (2nd byte)
91C0: B1 9B             LDA  (LOWTR),Y    ;  is zero, EOP reached
91C2: D0 06             BNE  CHKEND
91C4: 20 6C D6  END     JSR  CLEARC       ;End pgm
91C7: 4C D0 03          JMP  DOSWRM
91CA: A0 03     CHKEND  LDY  #3           ;End if user-defined EOP (LINNUM)
91CC: B1 9B             LDA  (LOWTR),Y    ;  exceeds line # (bytes 2,3)
91CE: C5 51             CMP  LINNUM+1
91D0: D0 08             BNE  CKE1
91D2: 88                DEY
91D3: B1 9B             LDA  (LOWTR),Y
91D5: C8                INY
91D6: C5 50             CMP  LINNUM
91D8: F0 02             BEQ  ZX
91DA: B0 E8     CKE1    BCS  END
91DC: A2 00     ZX      LDX  #0           ;(X)=BUF index
91DE: 86 00             STX  BUFCNT       ;Zero BUF char counter
91E0: C8        NXCHR   INY               ;(Y)=pgm index
91E1: E6 05             INC  PGMDIFF      ;Pgm line char counter
91E3: B1 9B             LDA  (LOWTR),Y    ;Get pgm line char
91E5: F0 27             BEQ  EOB          ;EOL
91E7: 30 05             BMI  PGMTKN       ;Token
91E9: 20 0D 94          JSR  STORBUF      ;Nontoken. Store it
91EC: 90 F2             BCC  NXCHR        ;Always
91EE: 20 3B 94  PGMTKN  JSR  TKNSET       ;Set pointer to keyword table
91F1: CA        NXTKN   DEX               ;Dec index to keyword location
91F2: F0 07             BEQ  PUTKYW       ;When (X)=0, keyword found
91F4: 20 4C 94  NT1     JSR  KYW          ;Get char in keyword table
91F7: 10 FB             BPL  NT1          ;If pos ASCII get another
91F9: 30 F6             BMI  NXTKN        ;If neg ASCII dec location index
91FB: 20 4C 94  PUTKYW  JSR  KYW          ;KEYWORD FOUND. Get char in table
91FE: 30 05             BMI  PK1          ;It's the final char
9200: 20 0D 94          JSR  STORBUF      ;Store non-final char
9203: 90 F6             BCC  PUTKYW       ;Always
9205: 29 7F     PK1     AND  #$7F         ;Convert final char to pos ASCII
9207: 20 0D 94          JSR  STORBUF      ;Store final char
920A: A4 85             LDY  FORPNT       ;Restore pgm index
920C: 90 D2             BCC  NXCHR        ;Always
920E: 9D 00 02  EOB     STA  BUF,X        ;Zero marks EOB
                *-------------------
                * CHECK FOR A MATCH
                *-------------------
9211: A2 00     FNDLIT  LDX  #0           ;(X)=RBUF index
9213: A4 01             LDY  BUFINDX      ;Restore BUF index
9215: C8                INY               ;(Y)=BUF index
9216: C4 00             CPY  BUFCNT       ;At EOB?
9218: 90 0A             BCC  FL1          ;No
921A: 24 08             BIT  CCFLG        ;Yes. Print line if both CCFLG
921C: 10 33             BPL  NXLN         ;  and MTCHFLG set. Used if
921E: 24 09             BIT  MTCHFLG      ;  changes not checked and
9220: 30 1E             BMI  SEARCH       ;  one or more changes made
9222: 10 2D             BPL  NXLN
9224: 84 01     FL1     STY  BUFINDX      ;Save BUF index
9226: BD D7 94  FL2     LDA  SBUF,X       ;Get S string char
9229: C9 40             CMP  #'@'         ;If wildcard char, match
922B: F0 05             BEQ  FL3          ;  is automatic
922D: D9 00 02          CMP  BUF,Y        ;Do S string & BUF chars match?
9230: D0 DF             BNE  FNDLIT       ;No. Restart at next BUF char
9232: C8        FL3     INY               ;Yes. Bump BUF index
9233: E8                INX               ;  and RBUF index
9234: E4 02             CPX  SRCHSIZ      ;String (complete) match?
9236: 90 EE             BCC  FL2          ;No. Keep going
9238: A9 FF             LDA  #$FF         ;Yes. Match point=BUFINDX
923A: 85 09             STA  MTCHFLG      ;Set flag
923C: 24 07             BIT  RPLFLG       ;Go to appropriate routine
923E: 30 17             BMI  REPLACE
                *-------------------
                * SEARCH MATCH
                *-------------------
9240: A5 9B     SEARCH  LDA  LOWTR        ;Point PTRB at start
9242: 85 1E             STA  PTRB         ;  of current line
9244: A5 9C             LDA  LOWTR+1
9246: 85 1F             STA  PTRB+1
9248: 20 FB DA          JSR  CRDO
924B: 20 54 94          JSR  PRLNNMB      ;Print line #
924E: 20 61 94          JSR  PRLN         ;Print line contents
9251: 20 9E 94  NXLN    JSR  NXLTR        ;Set LOWTR at start of next line
9254: 4C B4 91          JMP  SETBUF       ;Put next line into BUF
                *-------------------
                * REPLACE MATCH
                *-------------------

                * Change input buffer

9257: 38        REPLACE SEC               ;BUFDIFF=SRCHSIZ-RPLSIZ
9258: 8A                TXA               ;  pos=SEARCH > REPLACE
9259: E5 03             SBC  RPLSIZ       ;  neg=REPLACE > SEARCH
925B: 85 04             STA  BUFDIFF
925D: 10 2F             BPL  BUFDWN       ;BUF gets smaller. Don't worry
925F: A5 B0             LDA  PRGEND+1     ;  about HIMEM & BUF size tests
9261: 69 01             ADC  #1           ;Is program end within
9263: C5 74             CMP  MEMSIZ+1     ;  256 bytes of HIMEM?
9265: 90 06             BCC  BUFSIZ       ;No. Continue
9267: 20 1C 90          JSR  SEV1         ;Yes. Remind DOS of pgm hooks
926A: 4C 10 D4          JMP  MEMERR       ;?OUT OF MEMORY ERROR
926D: 38        BUFSIZ  SEC
926E: A5 00             LDA  BUFCNT       ;# chars in BUF minus BUFDIFF
9270: E5 04             SBC  BUFDIFF      ;  =new # char in BUF
9272: B0 05             BCS  BFS1         ;> 255 char. !TOO LONG message
9274: A8                TAY               ;(Y)=destination of upward move
9275: C0 FA             CPY  #250         ;> 250 char?
9277: 90 05             BCC  BUFUP        ;No. Move BUF up
9279: 46 01     BFS1    LSR  BUFINDX      ;Yes. Set END FLAG (plus)
927B: 4C 1D 94          JMP  TOOLONG      ;!LINE #XX TOO LONG
927E: A6 00     BUFUP   LDX  BUFCNT       ;(X)=origin of upward move=EOB
9280: BD 00 02  BFU1    LDA  BUF,X        ;Move BUF up
9283: 99 00 02          STA  BUF,Y
9286: E4 01             CPX  BUFINDX
9288: F0 14             BEQ  MVRBUF       ;End of move=match point
928A: CA                DEX
928B: 88                DEY
928C: D0 F2             BNE  BFU1
928E: 98        BUFDWN  TYA               ;(Y)=origin of downward move
928F: E5 04             SBC  BUFDIFF      ;Subtract differential
9291: AA                TAX               ;(X)=destination of downward move
9292: B9 00 02  BFD1    LDA  BUF,Y        ;Move BUF down
9295: 9D 00 02          STA  BUF,X
9298: F0 04             BEQ  MVRBUF       ;End of move=EOB
929A: C8                INY
929B: E8                INX
929C: D0 F4             BNE  BFD1
929E: A5 03     MVRBUF  LDA  RPLSIZ       ;If null R string, no
92A0: F0 10             BEQ  ZPTRS        ;  move is necessary
92A2: A2 00             LDX  #0
92A4: A4 01             LDY  BUFINDX
92A6: BD F7 94  MVR1    LDA  RBUF,X       ;Move R string into BUF
92A9: 99 00 02          STA  BUF,Y
92AC: C8                INY
92AD: E8                INX
92AE: E4 03             CPX  RPLSIZ
92B0: 90 F4             BCC  MVR1
92B2: A2 00     ZPTRS   LDX  #0           ;Point TXTPTR and PTRB
```

continued on page 7

# Applesoft Search and Replace (Cont.)

```
92B4: 86 B8        STX  TXTPTR     ;  at start of BUF
92B6: 86 1E        STX  PTRB
92B8: A2 02        LDX  #2
92BA: 86 B9        STX  TXTPTR+1
92BC: 86 1F        STX  PTRB+1
92BE: 20 59 D5     JSR  PARSE      ;Tokenize BUF
92C1: 38           SEC
92C2: 98           TYA             ;(Y)=# chars + 5 in parsed BUF
92C3: E9 05        SBC  #5         ;Subtract 5
92C5: E5 05        SBC  PGMDIFF    ;Subtract # chars in old line
92C7: 85 05        STA  PGMDIFF    ;PGMDIFF=NEW-OLD
92C9: 24 08        BIT  CGFLG      ;Check changes?
92CB: 30 53        BMI  CHNG1      ;No. Skip Y/N choice

      * Check each change as it is made

92CD: 20 FB DA     JSR  CROUT      ;Yes
92D0: A9 60        LDA  #$60
92D2: 20 ED FD     JSR  COUT       ;Print flashing space (prompt)
92D5: 20 54 94     JSR  PRLNMB     ;Print line number
92D8: A9 FF        LDA  #$FF
92DA: 85 85        STA  FORPNT     ;Set index to line contents
92DC: 20 61 94     JSR  PRLN       ;Print contents of changed line
92DF: A6 25        LDX  CV
92E1: 86 06        STX  CVSAV      ;Save row position
92E3: A0 00        LDY  #0
92E5: 84 24        STY  CH         ;Column at left margin
92E7: B1 28  FNDCUR LDA (BASL),Y   ;Locate prompt (start of line)
92E9: C9 60        CMP  #$60
92EB: F0 06        BEQ  YN
92ED: 20 1A FC     JSR  UP
92F0: 4C E7 92     JMP  FNDCUR
92F3: 20 E5 93 YN  JSR  INPYN      ;Accept or reject change?
92F6: B0 1E        BCS  CHANGE     ;Accept

      * Change rejected

92F8: A5 9B        LDA  LOWTR      ;Reject.  Set print pointer at
92FA: 85 1E        STA  PTRB       ;  original (unchanged) pgm line
92FC: A5 9C        LDA  LOWTR+1
92FE: 85 1F        STA  PTRB+1
9300: A5 02        LDA  SRCHSIZ    ;Reset match point (BUFINDX)
9302: 65 01        ADC  BUFINDX    ;  to char beyond end of
9304: AA           TAX             ;  rejected match
9305: CA           DEX
9306: 86 01        STX  BUFINDX
9308: A9 0E        LDA  #'N'-$40
930A: 20 ED FD     JSR  COUT       ;Print inverse N
930D: 20 54 94     JSR  PRLNMB     ;Restore original pgm line
9310: 20 61 94     JSR  PRLN
9313: 4C BA 91     JMP  SB1        ;Reset BUF

      * Change accepted

9316: A9 19  CHANGE LDA #'Y'-$40
9318: 20 ED FD     JSR  COUT       ;Print inverse Y
931B: A5 06        LDA  CVSAV      ;Restore row position
931D: 20 5B FB     JSR  TABV
9320: A6 9B  CHNG1 LDX  LOWTR      ;PTRB=LOWTR
9322: 86 1E        STX  PTRB
9324: A6 9C        LDX  LOWTR+1
9326: 86 1F        STX  PTRB+1
9328: 24 05        BIT  PGMDIFF    ;Move pgm which way?
932A: 30 25        BMI  PGMDWN     ;Down
932C: 20 9E 94     JSR  NXLTR      ;UP. BLTU start hi and lo
932F: A5 6A        LDA  VARTAB+1
9331: 85 97        STA  HIGHTR+1   ;BLTU end hi
9333: 18           CLC
9334: A5 69        LDA  VARTAB
9336: 85 96        STA  HIGHTR     ;BLTU end lo
9338: 65 05        ADC  PGMDIFF    ;Add increase in pgm size
933A: 90 02        BCC  PGMUP
933C: E6 6A        INC  VARTAB+1
933E: 85 94  PGMUP STA  HIGHDS     ;BLTU destination lo
9340: A4 6A        LDY  VARTAB+1
9342: 84 95        STY  HIGHDS+1   ;BLTU destination hi
9344: 20 93 D3     JSR  BLTU       ;Move pgm segment upward
9347: A5 1E        LDA  PTRB       ;Restore LOWTR to point at
9349: 85 9B        STA  LOWTR      ;  start of present line
934B: A5 1F        LDA  PTRB+1
934D: 85 9C        STA  LOWTR+1
934F: D0 2C        BNE  MVBUF      ;Always
9351: 18     PGMDWN CLC            ;MOVE PGM DOWN
9352: A0 01        LDY  #1
9354: B1 9B        LDA  (LOWTR),Y  ;Get link byte hi
9356: 85 1F        STA  PTRB+1     ;Origin hi
9358: 85 19        STA  PTRA+1     ;Destination hi
935A: 88           DEY
935B: B1 9B        LDA  (LOWTR),Y  ;Get link byte lo
935D: 85 1E        STA  PTRB       ;Origin lo
935F: 65 05        ADC  PGMDIFF    ;Add increase in pgm size
9361: 85 18        STA  PTRA       ;Destination lo
9363: B0 02        BCS  PGD1
9365: C6 19        DEC  PTRA+1
9367: B1 1E  PGD1  LDA  (PTRB),Y   ;Move pgm segment down
9369: 91 18        STA  (PTRA),Y
936B: 98           TYA
936C: C5 69        CMP  VARTAB     ;Compare origin and LOMEM
936E: A5 1F        LDA  PTRB+1
9370: E5 6A        SBC  VARTAB+1
9372: B0 09        BCS  MVBUF      ;End when origin=LOMEM
9374: C8           INY
9375: D0 F0        BNE  PGD1
9377: E6 1F        INC  PTRB+1
9379: E6 19        INC  PTRA+1
937B: D0 EA        BNE  PGD1
937D: A0 04  MVBUF LDY  #4         ;Point to contents of pgm line
937F: B9 FC 01 MVB2 LDA BUF-4,Y    ;Move changed BUF contents
9382: 91 9B        STA  (LOWTR),Y  ;  into pgm storage area
9384: F0 03        BEQ  FNDEOP
9386: C8           INY
9387: D0 F6        BNE  MVB2
      * Reset link bytes, LOMEM, EOP pointer & match point

9389: A5 67  FNDEOP LDA TXTTAB     ;Point PTRA at start of pgm
938B: 85 18        STA  PTRA
938D: A5 68        LDA  TXTTAB+1
938F: 85 19        STA  PTRA+1
9391: 18           CLC
9392: A0 01  FEOP1 LDY  #1
9394: B1 18        LDA  (PTRA),Y   ;EOP marker?
9396: D0 20        BNE  SETLINK    ;No. Set link bytes
9398: A5 18        LDA  PTRA       ;Yes. Reset LOMEM and EOF pointer
939A: 69 02        ADC  #2
939C: 85 69        STA  VARTAB
939E: 85 AF        STA  PRGEND
93A0: 90 02        BCC  FEOP2
93A2: E6 19        INC  PTRA+1
93A4: A5 19  FEOP2 LDA  PTRA+1
93A6: 85 6A        STA  VARTAB+1
93A8: 85 B0        STA  PRGEND+1
93AA: A5 02        LDA  SRCHSIZ    ;Reset match point position
93AC: 65 01        ADC  BUFINDX
93AE: 38           SEC
93AF: E5 04        SBC  BUFDIFF
93B1: AA           TAX
93B2: CA           DEX
93B3: 86 01        STX  BUFINDX
93B5: 4C BA 91     JMP  SB1        ;Search for more changes
93B8: A0 04  SETLINK LDY #4        ;Point to 1st line content byte
93BA: C8     SLK1  INY
93BB: B1 18        LDA  (PTRA),Y   ;EOL?
93BD: D0 FB        BNE  SLK1       ;No. Recycle
93BF: C8           INY             ;Yes. (Y) at start of next line
93C0: 98           TYA
93C1: 65 18        ADC  PTRA       ;Add start of current line
93C3: AA           TAX
93C4: A0 00        LDY  #0
93C6: 91 18        STA  (PTRA),Y   ;Set link byte lo
93C8: A5 19        LDA  PTRA+1
93CA: 69 00        ADC  #0         ;Pick up carry
93CC: C8           INY
93CD: 91 18        STA  (PTRA),Y   ;Set link byte hi
93CF: 86 18        STX  PTRA       ;Reset pointer
93D1: 85 19        STA  PTRA+1
93D3: 90 BD        BCC  FEOP1      ;Always
      *-----------
      * SUBROUTINES
      *-----------
93D5: 84 1E  PRINT STY  PTRB       ;PRINT ASCII STRING
93D7: 85 1F        STA  PTRB+1     ;Point to ASCII string
93D9: A0 00        LDY  #0
93DB: B1 1E  PR1   LDA  (PTRB),Y   ;Get char
93DD: F0 1B        BEQ  RTS2       ;End on hex zero
93DF: 20 ED FD     JSR  COUT       ;Output char
93E2: C8           INY
93E3: D0 F6        BNE  PR1        ;Always

93E5: 20 0C FD INPYN JSR RDKEY     ;Get response
93E8: C9 83        CMP  #$83       ;CTL-C
93EA: F0 0F        BEQ  CTLC
93EC: C9 D9        CMP  #"Y"       ;YES
93EE: F0 0A        BEQ  RTS2
93F0: C9 CE        CMP  #"N"       ;NO
93F2: F0 05        BEQ  INY1
93F4: 20 3A FF     JSR  BELL       ;Trap incorrect keypress
93F7: 10 EC        BPL  INPYN      ;Always
93F9: 18     INY1  CLC             ;NO=carry clear
93FA: 60     RTS2  RTS             ;YES=carry set
93FB: 68     CTLC  PLA             ;Abort REPLACE
93FC: 68           PLA
93FD: 20 57 DB     JSR  OUTSP
9400: 20 FB DA     JSR  CRDO
9403: A5 23        LDA  WNDBTM     ;Cursor at lowest row
9405: E9 02        SBC  #2         ;  in text window
9407: 20 5B FB     JSR  TABV
940A: 4C C4 91     JMP  END        ;End it all

940D: A6 00  STORBUF LDX BUFCNT    ;STORE CHAR IN BUF
940F: 9D 00 02     STA  BUF,X      ;Store char
9412: E8           INX
9413: 86 00        STX  BUFCNT     ;Save position index
9415: E0 FA        CPX  #250       ;> 249 chars
9417: 90 E1        BCC  RTS2       ;No. Return
9419: 86 01        STX  BUFINDX    ;Yes. Clear END FLAG (minus)
941B: 68           PLA
941C: 68           PLA
941D: A0 C3  TOOLONG LDY #<TXTOO1  ;Print !LINE #XX TOO LONG
941F: A9 94        LDA  #>TXTOO1
9421: 20 D5 93     JSR  PRINT
9424: 20 54 94     JSR  PRLNMB
9427: A0 CC        LDY  #<TXTOO2
9429: A9 94        LDA  #>TXTOO2
942B: 20 D5 93     JSR  PRINT
942E: 20 3A FF     JSR  BELL
9431: 24 01        BIT  BUFINDX    ;END pgm?
9433: 30 03        BMI  TOO1       ;No. Go to next line
9435: 4C C4 91     JMP  END        ;Yes. Back to BASIC
9438: 4C 51 92 TOO1 JMP NXLN

943B: 38     TKNSET SEC            ;SET POINTER TO KEYWORD TABLE
943C: E9 7F        SBC  #$7F       ;Token minus $7F equals index
```

# Applesoft Search and Replace (Cont.)

```
943E: AA                  TAX                 ;  to keyword in table
943F: 84 85               STY   FORPNT        ;Save pgm index
9441: A0 D0               LDY   #<TKNTBL      ;Set keyword table
9443: 84 9D               STY   DSCTMP        ;  pointer at $CFD0
9445: A0 CF               LDY   #>TKNTBL-$100
9447: 84 9E               STY   DSCTMP+1
9449: A0 FF               LDY   #$FF          ;Point at $D0CF
944B: 60                  RTS

944C: C8           KYW    INY                 ;FETCH NEXT CHAR IN KEYWORD TABL
944D: D0 02               BNE   KY1
944F: E6 9E               INC   DSCTMP+1
9451: B1 9D        KY1    LDA   (DSCTMP),Y
9453: 60                  RTS

9454: A0 02        PRLNNMB LDY  #2            ;PRINT LINE NUMBER
9456: B1 9B               LDA   (LOWTR),Y     ;Line # byte lo
9458: AA                  TAX
9459: C8                  INY
945A: 84 85               STY   FORPNT        ;Set index to line contents
945C: B1 9B               LDA   (LOWTR),Y     ;Line # byte hi
945E: 4C 24 ED            JMP   LINPRT        ;Print decimal line #

9461: A9 20        PRLN   LDA   #' '          ;PRINT LINE CONTENTS
9463: A4 85        PRL1   LDY   FORPNT        ;Restore index to line contents
9465: 09 80        PRL2   ORA   #$80          ;Set hi bit
9467: C9 A0               CMP   #" "          ;CTL char?
9469: B0 02               BCS   PRL3          ;No
946B: 29 7F               AND   #$7F          ;Yes. Clear hi bit
946D: 20 ED FD     PRL3   JSR   COUT          ;Output char
9470: C8                  INY
9471: B1 1E               LDA   (PTRB),Y      ;Get next char
9473: D0 06               BNE   PRL4          ;Not EOL. Continue
9475: 20 42 FC            JSR   CLREOP        ;EOL. Return
9478: 4C FB DA            JMP   CRDO
947B: 10 E8        PRL4   BPL   PRL2          ;Nontoken
947D: 20 3B 94            JSR   TKNSET        ;Set pointer to keyword table
9480: CA           PRL5   DEX                 ;Dec index to keyword location
9481: F0 07               BEQ   PRL7          ;When (X)=0, keyword found
9483: 20 4C 94     PRL6   JSR   KYW           ;Get char in keyword table
9486: 10 FB               BPL   PRL6          ;If pos ASCII get another
9488: 30 F6               BMI   PRL5          ;If neg ASCII dec location inde:
948A: 20 57 DB     PRL7   JSR   OUTSP         ;Space
948D: 20 4C 94     PRL8   JSR   KYW           ;KEYWORD FOUND. Get char in tabl
9490: 30 05               BMI   PRL9          ;It's the final char
9492: 20 5C DB            JSR   OUTDO         ;Print non-final char
9495: D0 F5               BNE   PRL8          ;Always
9497: 20 5C DB     PRL9   JSR   OUTDO         ;Print final char
949A: A9 20               LDA   #' '          ;Space
949C: D0 C5               BNE   PRL1          ;Always

949E: A0 01        NXLTR  LDY   #1            ;SET LOWTR TO START OF NEXT LIN
94A0: B1 9B               LDA   (LOWTR),Y     ;Link byte hi
94A2: 48                  PHA
94A3: 88                  DEY
94A4: B1 9B               LDA   (LOWTR),Y     ;Link byte lo
94A6: 85 9B               STA   LOWTR
94A8: 68                  PLA
94A9: 85 9C               STA   LOWTR+1
94AB: 60                  RTS
```

uffer
buffer

```
                   *-----
                   * TEXT
                   *-----
94AC: 8D           TXCC       HEX   8D
94AD: C3 C8 C5
94B0: C3 CB A0
94B3: C3 C8 C1
94B6: CE C7 C5
94B9: D3 A0 A8
94BC: D9 AF CE
94BF: A9 BF A0                ASC   "CHECK CHANGES (Y/N)? "
94C2: 00                      HEX   00

94C3: 8D           TXTOO1     HEX   8D
94C4: A1 CC C9
94C7: CE C5 A0
94CA: A3                      ASC   "!LINE #"
94CB: 00                      HEX   00
94CC: A0 D4 CF
94CF: CF A0 CC
94D2: CF CE C7     TXTOO2     ASC   " TOO LONG"
94D5: 8D 00                   HEX   8D00

                   *-----------------
                   * INTERNAL BUFFERS
                   *-----------------
                   SBUF       DS    $20        ;SEARCH buffer
                   RBUF       DS    $20        ;REPLACE buffer

9517: 00           ENDLIST    HEX   00
```

—End assembly—

1304 bytes

Errors: 0

```
KEY PERFECT 4.0
RUN ON
SEARCH/REPLACE
========================================
CODE          ADDR# - ADDR#
-----         -------------
26AB          5E00 - 5E4F
271B          5E50 - 5E9F
271B          5EA0 - 5EEF
2469          5EF0 - 5F3F
24BD          5F40 - 5F8F
255D          5F90 - 5FDF
2ABB          5FE0 - 602F
2796          6030 - 607F
2C47          6080 - 60CF
2AA5          60D0 - 611F
28C5          6120 - 616F
249B          6170 - 61BF
2B12          61C0 - 620F
28FB          6210 - 625F
28CF          6260 - 62AF
2ACC          62B0 - 62FF
01FE          6300 - 6317
TOTAL PROGRAM CHECK IS : 0518
```

```
APPLE CHECKER

ON: SEARCH/REPLACE
TYPE: B

LENGTH: 0518
CHECKSUM: BA
```

**100 GOSUB 10 : REM DATA**

| ADDR | REL.BYTE | CONTENT | DEFINITION |
|---|---|---|---|
| $923 | 0 | $31 | Link (LSB) |
| 924 | 1 | 09 | Link (MSB) |
| 925 | 2 | 64 | Line number (LSB) |
| 926 | 3 | 00 | Line number (MSB) |
| 927 | 4 | B0 | GOSUB |
| 928 | 5 | 31 | 1 |
| 929 | 6 | 30 | 0 |
| 92A | 7 | 3A | : |
| 92B | 8 | B2 | REM |
| 92C | 9 | 44 | D |
| 92D | A | 41 | A |
| 92E | B | 54 | T |
| 92F | C | 41 | A |
| 930 | D | 00 | EOL |
| 931 | — | — | Start of next line |