This installment tells you how to add a useful editing function to your computer.

Thoughts on The 68XX Systems

By Garry O. Caudell and Ron Silver

As part of writing this continuing series on 68xx systems, I often receive suggestions on useful topics from avid fans of the 6800 and 6809. This month I'll sit back and combine two contributions from readers into an article under their name.—Pete Stark.

BASIC interpreters usually require that each line be numbered, and automatically insert new lines into line number order, thus providing a certain amount of editing capability.

It is possible to insert, delete or substitute new lines into a program by making use of the line numbers. When a program statement is no longer needed, it can be deleted simply by entering its line number. A line can be changed by retyping a line with the same line number. And finally, a line can be inserted between two existing lines simply by using a line number that is between the values of the line numbers of those two statements.

The above features of BASIC are useful. However, in the process of writing and debugging a BASIC program, you'll often need to make minor modifications to quite a few lines. In most 6800 and 6809 BASICs, each of these modifications requires that you retype the entire line, even when only one or two characters need to be changed.

On disk systems, users often resort to the text editor to make such changes. Yet it is possible to overcome this limitation of the BASIC interpreter by adding to it a more powerful edit function. This modification would make the BASIC system easier to use by letting the user edit a line without retyping the whole line.

Many other computers (such as the TRS-80) have such editing capability in their BASIC, since not much additional code is required.

This article presents the modifications to implement this editing function in SWTP 8K BASIC versions 2.0 and 2.2 (see Listing 1), SWTP Disk BASIC version 3.5 (Listing 2) and Percom Super BASIC (Listing 3).

Although there are differences in the three implementations, the idea is the same. In fact, it is also easily applied to the TSC Text Editor itself (see Listing 4, and further details later). Although the TSC Text Editor provides many ways of editing entire lines, its provisions for manipulating characters within a line are somewhat awkward. This modification fixes that problem.

Control Character	ASCII Code	Function
Control-E	$05	Enters edit mode—may be used during any input.
Control-R	$12	Recalls a line from memory for editing. The line recalled will be the line whose line number appears at the beginning of the line being typed when control-R is entered.
Control-A	$01	Adds a space at cursor position.
Control-D	$04	Delete character at cursor position.
Control-H	$08	Backspace—move cursor left.
Control-I	$09	Tab—move cursor right.
(CR)	$0D	Carriage return—enters the line into memory as if it had just been just typed in, regardless of cursor position.

Table 1. Summary of edit commands.

What It Does

To implement this feature, you must use a CRT terminal. The modification dynamically shows each edit-

Address correspondence to Ron Silver, Kelvin High School, 155 Kingsway Ave., Winnipeg, Man-

ing change as it occurs by rapidly spacing forward or backward to erase and rewrite the changed line. Since most teleprinters do not backspace, this would not work on such a terminal; even with backspacing, it would repeatedly overprint a line and produce a black mess.

The modified BASIC interpreter lets the user insert, delete or change any character in any statement of a program. The edit functions are controlled by six control keys.

BASIC program entry and editing is unaffected until a control-E ($05 code) is typed. This causes a branch to the edit subroutine in the patch. This subroutine then processes all succeeding characters and commands entered from the keyboard until hitting the carriage return returns back to normal BASIC entry.

Editing occurs on the line currently in BASIC's input buffer. If a line is being entered, then that line will be edited. This mode, however, is probably not that useful, since in many cases it is easier to simply backspace to the wrong character and retype from there.

A far more powerful feature of this editor modification is its ability to recall lines previously entered. Typing the number of this line followed by control-E to enter the editor, and then control-R to recall the line, brings a line from memory back into the buffer for editing. The line now appears on the CRT as if it had just been typed, and the cursor is positioned at the end of the line.

At this point, the cursor would normally be moved to the character where a change is to be made.

Backspace (control-H) moves the cursor back to the left without erasing the characters it is stepping over.

Horizontal tab (control-I) moves the cursor to the right, again without erasing the characters it is stepping over.

Once the cursor is positioned where a change is wanted, there are three ways to proceed:

Control-A inserts (adds) a space at the current cursor position and moves the following characters one place to the right. On a slow terminal you can see the rest of the line being rewritten; on a fast terminal (9600 baud, for example) this occurs so fast that it is nearly invisible.

itoba, Canada R3M 0G3; or Garry O. Caudell, 3125 Robin Lynn Drive, Ashland, KY 41101.

Listing 1. Editing modification in SWTP 8K BASIC versions 2.0 and 2.2.

```
00010                          NAM    EDITTOR
00015                          OPT    O
00016                  *  BASIC ENTRY POINTS
00020        0485      PSHX   EQU    $0485     PUSH INDEX
00030        00B0      BUFFER EQU    $B0       BEGINNING OF THE BASIC BUFFER
00035        00F8      BUFEND EQU    $F8       END OF BUFFER
00040        0440      BERROR EQU    $0440     ERROR ROUTINE
00050        0395      INPUTC EQU    $0395     INPUT ROUTINE
00060        0386      OUTC   EQU    $0386     OUTPUT ROUTINE
00070        049A      PULX   EQU    $049A     PULL INDEX
00080        004A      COUNTR EQU    $4A
00085        0024      BUFPNT EQU    $24       POINTER TO BUFFER POSITION
00086        002C      MEMPNT EQU    $2C       POINTER TO MEMORY POSITION
00087        00AC      TEMP   EQU    $AC
00090        0AC7      BERR2  EQU    $0AC7     ERROR ROUTINE
00100        0ACA      NUMC   EQU    $0ACA     NUMERIC CONVERSION
00110        0A87      LINEC  EQU    $0A87     FIND LINE IN MEMORY
00120        0467      PSTR   EQU    $0467     PRINT STRING ROUTINE
00130        0453      PLINEC EQU    $0453     BRANCH POINT BACK TO BASIC
00132                  * MISC CONSTANTS
00133        0008      BACK   EQU    $08       BACK SPACE CHARACTER
00134        0009      TAB    EQU    $09       FORWARD TAB CHARACTER
00140 1EAF                     ORG    $1EAF
00150 1EAF BD 0485 EDIT   JSR    PSHX
00160 1EB2 6F 00   CLEAR  CLR    ,X        CLEAR BUFFER TO END
00170 1EB4 08             INX
00180 1EB5 8C 00F8        CPX    #BUFEND   IS IT THE END OF THE BUFFER?
00190 1EB8 26 F8          BNE    CLEAR
00200 1EBA BD 049A        JSR    PULX
00210 1EBD BD 0395 INPUT  JSR    INPUTC    INPUT LOOP
00220 1EC0 81 0D          CMP A  #$0D      CHECK FOR CARRIAGE CONTROL
00230 1EC2 26 03          BNE    CONT1
00240 1EC4 7E 1FBD        JMP    ENDRTN    IF SO JUMP BACK TO BASIC
00250 1EC7 81 08   CONT1  CMP A  #BACK     CHECK FOR BACKSPACE
00260 1EC9 27 2D          BEQ    BACKSPC
00270 1ECB 81 09          CMP A  #TAB
00280 1ECD 27 15          BEQ    FORWARD
00290 1ECF 81 04          CMP A  #$04      CONTROL D
00300 1ED1 27 34          BEQ    DELETE
00310 1ED3 81 01          CMP A  #$01      CONTROL A
00320 1ED5 27 5D          BEQ    ADD
00330 1ED7 81 12          CMP A  #$12      CONTROL R
00340 1ED9 26 03          BNE    CONT3
00350 1EDB 7E 1F52        JMP    RECALL
00360 1EDE 81 1F   CONT3  CMP A  #$1F      CHECK IF CONTROL CHARACTER
00370 1EE0 2B DB          BMI    INPUT     IF SO IGNORE
00380 1EE2 A7 00   STORCH STA A  ,X        STORE CHARACTER IN BUFFER
00385                  *  MOVE CURSOR FORWARD
00390 1EE4 A6 00   FORWAR LDA A  ,X
00395                  *    CHECK IF AT END OF LINE IN BUFFER
00400 1EE6 26 07          BNE    CONT2     BRANCH IF NOT
00410 1EE8 86 08          LDA A  #BACK
00415                  *  PUT CURSOR BACK WHERE IT BELONGS
00420 1EEA BD 0386        JSR    OUTC

00430 1EED 20 CE          BRA    INPUT
00435                  *    MOVE TO NEXT POSITION IN BUFFER
00440 1EEF 08      CONT2  INX               MOVE TO NEXT POSITION IN BUFF
00450 1EF0 8C 00F8        CPX    #BUFEND   CHECK FOR END OF BUFFER
00460 1EF3 26 C8          BNE    INPUT
00470 1EF5 7E 0440 ERROR  JMP    BERROR
00475                  * BACKUP CURSOR
00480 1EF8 8C 00B0 BACKSP CPX    #BUFFER   CHECK IF AT START OF BUFFER
00490 1EFB 27 03          BEQ    ATBEG     BRA IF SO
00500 1EFD 09             DEX
00510 1EFE 20 BD   LOOPBA BRA    INPUT
00520 1F00 86 09   ATBEG  LDA A  #TAB      MOVE CURSOR FORWARD
00530 1F02 BD 0386        JSR    OUTC
00540 1F05 20 B6          BRA    INPUT



00547                  *  DELETE A CHARACTER FROM BUFFER AND CRT
00550 1F07 BD 0485 DELETE JSR    PSHX
00555                  * ZERO COUNTER IN ORDER TO REPOSITION CURSOR
00560 1F0A 7F 004A        CLR    COUNTR
00565                  * MOVE CURSOR AND BUFFER CONTENTS ONE POSITION LEFT
00570 1F0D A6 01   MOVELF LDA A  1,X
00580 1F0F BD 0386        JSR    OUTC
00590 1F12 A7 00          STA A  X
00600 1F14 08             INX
00610 1F15 7C 004A        INC    COUNTR    COUNT CHARACTERS TO END
00620 1F18 4D             TST A
00630 1F19 26 F2          BNE    MOVELFT
00640 1F1B 86 20   SPACE  LDA A  #$20      PRINT SPACE
00650 1F1D BD 0386        JSR    OUTC
00655                  * PUT CURSOR BACK TO CORRECT POSITION
00660 1F20 7D 004A MOVECS TST    COUNTR    CHECK TO SEE IF FINISHED
00670 1F23 27 0A          BEQ    ENDD
00680 1F25 86 08          LDA A  #BACK
00690 1F27 BD 0386        JSR    OUTC
00700 1F2A 7A 004A        DEC    COUNTR
00710 1F2D 20 F1          BRA    MOVECSR
00720 1F2F BD 049A ENDD   JSR    PULX
00730 1F32 20 CA          BRA    LOOPBACK
```

More

Listing 1 continued.

```
00740 1F34 BD 0485 ADD    JSR    PSHX
00746              * ADD SPACE TO BUFFER AT CURSOR POSITION
00750 1F37 7F 004A        CLR    COUNTR   ZERO COUNTER
00760 1F3A C6 20          LDA B  #$20     PUT SPACE IN BUFFER
00765              * MOVE CURSOR AND BUFFER RIGHT ONE PLACE
00770 1F3C 17      MOVERT TBA
00780 1F3D E6 00          LDA B  X
00790 1F3F A7 00          STA A  X
00800 1F41 BD 0386        JSR    OUTC
00810 1F44 08             INX
00820 1F45 8C 00F8        CPX    #BUFEND  CHECK FOR FILLED BUFFER
00830 1F48 27 AB          BEQ    ERROR
00840 1F4A 7C 004A        INC    COUNTR   COUNT CHARACTERS FILLED
00850 1F4D 4D             TST A
00860 1F4E 26 EC          BNE    MOVERT
00870 1F50 20 C9          BRA    SPACE



00877              * RECALL LINE FROM MEMORY AND PLACE IN BUFFER
00880 1F52 8D 5B   RECALL BSR    ZERCSR   MOVE CURSOR TO START OF LINE
00890 1F54 CE 00B0        LDX    #BUFFER  FIND LINE NUMBER IN BUFFER
00900 1F57 BD 0ACA        JSR    NUMC     CONVERT TO BCD FOR LINEC
00910 1F5A 24 03          BCC    CONT4    INVALID LINE NUMBER?
00920 1F5C 7E 0AC7 ERR2   JMP    BERR2
00925              *   STORE POSITION OF END OF LINE NUMBER
00930 1F5F DF 24   CONT4  STX    BUFPNT
00940 1F61 BD 0A87        JSR    LINEC    FIND LINE IN MEMORY
00950 1F64 25 F6          BCS    ERR2     BRANCH IF NOT FOUND
00955              *  SKIP PAST LINE NUMBER STORED IN MEMORY
00960 1F66 08             INX
00970 1F67 08             INX
00975              *   STORE POSITION OF LINE IN MEMORY
00980 1F68 DF 2C          STX    MEMPNT
00990 1F6A 8D 1F          BSR    STORKY   STORE KEYWORD IN BUFFER
01000 1F6C 08             INX
01005              * TRANSFER CHARACTERS FROM MEMORY TO BUFFER
01010 1F6D A6 00   SLOOP  LDA A  X
01020 1F6F 4D             TST A
01025              *  BRANCH IF LAST CHARACTER IN LINE
01030 1F70 27 0A          BEQ    ENDR
01040 1F72 81 19          CMP A  #$19     ANOTHER KEYWORD?
01050 1F74 2C 04          BGE    STRCHR   BRANCH IF NOT
01060 1F76 8D 13          BSR    STORKY   STORE KEYWORD IN BUFFER
01070 1F78 20 F3          BRA    SLOOP
01080 1F7A 8D 1C   STRCHR BSR    TRNSFR+2 STORE CHARACTER IN BUFFER
01090 1F7C DE 24   ENDR   LDX    BUFPNT   STORE $00 IN BUFFER
01100 1F7E A7 00          STA A  X
01110 1F80 CE 00B0        LDX    #BUFFER  PRINT CONTENTS OF THE BUFFER
01120 1F83 BD 0467        JSR    PSTR
01125              *   SET INDEX TO LAST CHARACTER IN BUFFER
01130 1F86 DE 24          LDX    BUFPNT
01140 1F88 7E 1EBD        JMP    INPUT
01150 1F8B EE 00   STORKY LDX    X        STORE KEYWORD IN BUFFER
01160 1F8D 09             DEX
01170 1F8E 09      BACKUP DEX              LOCATE KEYWORD IN TABLE
01180 1F8F A6 00          LDA A  X
01190 1F91 26 FB          BNE    BACKUP
01195              *  SKIP PAST JUMP ADDRESS IN TABLE
01200 1F93 08             INX
01210 1F94 08             INX
01220 1F95 08             INX
01230 1F96 A6 00   TRNSFR LDA A  X        TRANSFER CHARACTERS TO BUFFER
01240 1F98 81 20          CMP A  #32      LAST CHARACTER? OR KEY WORD?
01250 1F9A 2D 0E          BLT    ENDSTR   BRANCH IF SO
01260 1F9C 08             INX
01270 1F9D DF AC          STX    TEMP     STORE MEMORY POINTER
01280 1F9F DE 24          LDX    BUFPNT   PICK UP BUFFER INDEX
01290 1FA1 A7 00          STA A  X
01300 1FA3 08             INX
01310 1FA4 DF 24          STX    BUFPNT
01320 1FA6 DE AC          LDX    TEMP
01330 1FA8 20 EC          BRA    TRNSFR
01335              *  LOAD POSITION OF BUFFER AFTER LINE#
01340 1FAA DE 2C   ENDSTR LDX    MEMPNT
01345              *   LOAD POSITION OF BUFFER AFTER LINE#
01350 1FAC 08             INX
01360 1FAD 08             INX
01370 1FAE 39             RTS
01380 1FAF 8C 00B0 ZERCSR CPX    #BUFFER  AT BEGINNING OF BUFFER?
01390 1FB2 27 08          BEQ    ENDZ     BRANCH IF YES
01400 1FB4 86 08          LDA A  #BACK
01410 1FB6 BD 0386        JSR    OUTC
01420 1FB9 09             DEX
01430 1FBA 20 F3          BRA    ZERCSR
01440 1FBC 39      ENDZ   RTS



01450 1FBD CE 00AF ENDRTN LDX    #BUFFER-1  FIND END OF LINE IN BUFFER
01460 1FC0 08      LOOP   INX
01470 1FC1 A6 00          LDA A  X
01480 1FC3 4D             TST A
01490 1FC4 26 FA          BNE    LOOP
01500 1FC6 7E 0453        JMP    PLINEC
01510      1FC9    PRGEND EQU    *
```

More

Control-D deletes the character at the current cursor position and moves all following characters one place to the left.

The preceding edit commands are summarized in Table 1.

Alternatively, typing any other character substitutes the new character instead of the old one, and moves the cursor right one place.

This recall-and-edit feature can be used to change any part of the line, even the line number. Changing the line number makes it into a new line, to be inserted into the program in line number order. The old line stays in the program, and the new line is added. Thus it is possible to produce identical (or slightly modified) copies of existing lines.

How It Works

Let's confine ourselves to describing the version 2.0 BASIC patch in Listing 1; the other versions work similarly.

The edit patch works by manipulating characters in the BASIC interpreter buffer that starts at location 00B0.

Lines 20–130 of the program define entry points to the BASIC interpreter. PSHX and PULX are two subroutines used to push and pull the index register into an index stack kept by BASIC. INPUTC and OUTC are the BASIC character input and output routines. NUMC is a BASIC routine which converts a number in ASCII in memory to a binary-coded decimal (BCD) number used by LINEC to find that line in memory.

PSTR is a routine which prints a string of characters. PLINEC is the entry point when a carriage return has been entered.

Lines 150–540 represent the edit input routine, which branches to the appropriate routines when the proper control characters are entered. It starts at 1EAF for BASIC version 2.0, 1EE2 for BASIC version 2.2 or 2698 for BASIC version 3.5.

The delete routine, called with a control-D, consists of lines 550–730. MOVELF shifts all characters in the buffer which are past the cursor position one place to the left. As it does this, it prints those characters on the CRT and counts them. MOVECS then puts the cursor back to where it started by printing that number of backspaces.

The add routine, called by a control-A and shown in lines 740–879, operates in a similar manner except

Listing 1 continued.

```
01520 040F                    ORG    $040F
01530 040F 81 05              CMP A  #5
01540 0411 26 03              BNE    CONT6
01550 0413 7E 1EAF            JMP    EDIT
01560 0416 B1 0153 CONT6      CMP A  $153      CHECK FOR CONTROL X
01570 0419 27 DE              BEQ    $3F9
01580 041B B1 0154            CMP A  $154
01590 041E 27 08              BEQ    CONT7
01600 0420 81 0D              CMP A  #$0D      CHECK FOR CAR RETURN
01610 0422 27 2F              BEQ    PLINEC
01620 0424 2D E6              BLT    $40C
01630 0426 20 0E              BRA    $436
01640 0428 B6 0155 CONT7      LDA A  $0155
01650 014E                    ORG    $14E
01660 014E 1FC9               FDB    PRGEND
01665 0154                    ORG    $154
01670 0154 08                 FCB    BACK,0
      0155 00
01680 A048                    ORG    $A048
01690 A048 0100               FDB    $100
```

Listing 2. Editing modification in SWTP Disk BASIC version 3.5.

```
00010                         NAM    EDITOR
00020                 *
00030                 * EDITOR FOR BASIC V3.5 AS SUPPLIED FOR DMAF1 DISK
00040                 *
00050                 * WRITTEN BY R.SILVER
00060                 *
00070                 * MODIFIED FOR THE DMAF1 DISK BASIC BY K.J.KROEKER
00080                 *
00090                 *
00100       0547      PSHX   EQU    $0547
00110       04F0      BERROR EQU    $04F0
00120       0428      INPUTC EQU    $428
00130       03F1      OUTC   EQU    $3F1
00140       055C      PULX   EQU    $055C
00150                         OPT    O,S
00160       0BA1      BERR2  EQU    $0BA1
00170       0BA4      NUMC   EQU    $0BA4
00180       0B61      LINEC  EQU    $0B61
00190       0520      PSTR   EQU    $0520
00200       050C      PLINEC EQU    $050C
00210 2698                    ORG    $2698
00220 2698 BD 0547 EDIT       JSR    PSHX
00230 269B 6F 00   CLEAR      CLR    ,X
00240 269D 08                 INX
00250 269E 8C 00F8            CPX    #$00F8
00260 26A1 26 F8              BNE    CLEAR
00270 26A3 BD 055C            JSR    PULX
00280 26A6 BD 0428 INPUT      JSR    INPUTC
00290 26A9 81 0D              CMP A  #$0D
00300 26AB 26 03              BNE    CONT1
00310 26AD 7E 27A6            JMP    ENDRTN
00320 26B0 81 08   CONT1      CMP A  #$08
00330 26B2 27 2D              BEQ    BACKSPC
00340 26B4 81 09              CMP A  #$09
00350 26B6 27 15              BEQ    FORWARD
00360 26B8 81 04              CMP A  #$04
00370 26BA 27 34              BEQ    DELETE
00380 26BC 81 01              CMP A  #$01
00390 26BE 27 5D              BEQ    ADD
00400 26C0 81 12              CMP A  #$12
00410 26C2 26 03              BNE    CONT3
00420 26C4 7E 273B            JMP    RECALL
00430 26C7 81 1F   CONT3      CMP A  #$1F
00440 26C9 2B DB              BMI    INPUT
00450 26CB A7 00   STORCH     STA A  ,X
00460 26CD A6 00   FORWAR     LDA A  ,X
00470 26CF 26 07              BNE    CONT2
00480 26D1 86 08              LDA A  #$08
00490 26D3 BD 03F1            JSR    OUTC
00500 26D6 20 CE              BRA    INPUT
00510 26D8 08      CONT2      INX
00520 26D9 8C 00F8            CPX    #$00F8
00530 26DC 26 C8              BNE    INPUT
00540 26DE 7E 04F0 ERROR      JMP    BERROR
00550 26E1 8C 00B0 BACKSP     CPX    #$00B0
00560 26E4 27 03              BEQ    ATBEG
00570 26E6 09                 DEX
00580 26E7 20 BD   LOOPBA     BRA    INPUT
00590 26E9 86 09   ATBEG      LDA A  #$09
00600 26EB BD 03F1            JSR    OUTC
00610 26EE 20 B6              BRA    INPUT
00620 26F0 BD 0547 DELETE     JSR    PSHX
00630 26F3 7F 004A            CLR    $004A
00640 26F6 A6 01   MOVELF     LDA A  1,X
00650 26F8 BD 03F1            JSR    OUTC
00660 26FB A7 00              STA A  X
```

More

that it inserts a space at the cursor position and then shifts all characters from the cursor position one character to the right.

The recall routine is called by a control-R and uses lines 880–1490 of the program. To recall the line from memory into the BASIC buffer, lines 890–980 locate the requested line in memory and store its position in MEMPNT. Lines 1010–1370 then transfer the characters from memory into the buffer.

In SWTP BASICs, each line is stored in memory as a two-byte BCD line number, a two-byte address which represents the encoded keyword (such as PRINT, INPUT or IF), followed by the remaining characters of the line. Before the line can be physically transferred from memory back into the buffer, this two-byte code must be translated back into the keyword; this is done by subroutine STORKY, which locates the keyword in the BASIC jump table and stores it in the buffer.

When editing is completed, lines 1450 through 1500 position the index at the end of the buffer and jump to the BASIC reentry point to process the carriage return and place the line back into memory in its correct location.

The patch to the BASIC input routine that calls the editor when a control-E is first entered is shown at lines 1520–1640. This patch is the same for both BASIC versions 2.0 and 2.2. BASIC version 3.5, however, uses a different input routine as well as different entry points. Listing 2 shows how Ken Kroeker of Dakota Collegiate Institute, Winnipeg, Canada, modified the patch to work with this BASIC.

Finally, lines 1650 through 1670 change BASIC's end-of-program pointer to leave room for the editor patch, and also change the backspace and echo characters so that the standard backspace (control-H or ASCII $08) is used instead of the standard control-O and underline echo.

TSC Text Editor Patch

The TSC Text Editor has a large variety of commands and operational modes, but it is line- rather than character-oriented. Moreover, it was originally intended for hard-copy as well as CRT terminal use. Thus, it is rather awkward to change characters in the middle of a line. The text editor patch described here can easily be implemented in the TSC editor to greatly

enhance its operation.

The original TSC Text Editor was cassette-based. It was later adapted, with relatively minor patches but without reassembly, to work on the SWTP disk system under MiniFlex, and on the Percom disk system under MINIDOS. Since all three versions resulted from the same assembly, Listing 4 applies to all of them except for the end-of-program pointers.

This affects the two ORG statements labelled as ***SEE TEXT*** in Listing 4. The addresses shown ($1D7D and $16DB) apply to Percom's "Touchup" version of the Editor.

In the original cassette version, the first ORG should read ORD $1492. The last ORG, as well as the LDX #PRGEND following it, should be omitted.

In the MiniFlex version, the first ORG should read $19DB, and the last ORG and the following LDX should also be omitted.

A/BASIC Modifications

Of all the 6800 higher-level language compilers, A/BASIC is the only one available in a cassette version. Since some recent statistics seem to indicate that there are more tape users than was thought, it's worth some space and time.

Though A/BASIC is relatively inexpensive ($65 in the cassette version), it isn't nearly as popular as one would expect. There are several reasons.

First, Microware's ads state that the cassette version requires the RT/68 operating system. Though RT/68 is quite impressive as a multitasking, real-time execution program, it is not nearly as versatile as some other, more plain monitors. Nor is it compatible with any of them. (In fact, even if a program is compiled with unmodified A/BASIC on an RT/68 system, it will not run on computers with other monitors.) This, combined with its price of $55, has not made it popular for the nonprofessional. (Microware, 5835 Grand Ave., Des Moines, IA 50304) has sold many more RT/68 manuals —to colleges and universities for text use—than ROMs.)

Second, the cassette version of A/BASIC is awkward to use. In fact, any compiler is awkward to use, but in a cassette form it is downright painful.

To run A/BASIC from cassette normally requires the following steps:

Listing 2 continued.

```
00670 26FD 08              INX
00680 26FE 7C 004A         INC     $004A
00690 2701 4D              TST A
00700 2702 26 F2           BNE     MOVELFT
00710 2704 86 20   SPACE   LDA A   #$20
00720 2706 BD 03F1         JSR     OUTC
00730 2709 7D 004A MOVECS  TST     $4A
00740 270C 27 0A           BEQ     ENDD
00750 270E 86 08           LDA A   #$08
00760 2710 BD 03F1         JSR     OUTC
00770 2713 7A 004A         DEC     $4A
00780 2716 20 F1           BRA     MOVECSR
00790 2718 BD 055C ENDD    JSR     PULX
00800 271B 20 CA           BRA     LOOPBACK
00810 271D BD 0547 ADD     JSR     PSHX
00820 2720 7F 004A         CLR     $4A
00830 2723 C6 20           LDA B   #$20
00840 2725 17      MOVERT  TBA
00850 2726 E6 00           LDA B   X
00860 2728 A7 00           STA A   X
00870 272A BD 03F1         JSR     OUTC
00880 272D 8C 00FB         CPX     #$FB
00890 2730 08              INX
00900 2731 27 AB           BEQ     ERROR
00910 2733 7C 004A         INC     $4A
00920 2736 4D              TST A
00930 2737 26 EC           BNE     MOVERT
00940 2739 20 C9           BRA     SPACE
00950 273B 8D 5B   RECALL  BSR     ZERCSR
00960 273D CE 00B0         LDX     #$00B0
00970 2740 BD 0BA4         JSR     NUMC
00980 2743 24 03           BCC     CONT4
00990 2745 7E 0BA1 ERR2    JMP     BERR2
01000 2748 DF 24   CONT4   STX     $24
01010 274A BD 0B61         JSR     LINEC
01020 274D 25 F6           BCS     ERR2
01030 274F 08              INX
01040 2750 08              INX
01050 2751 DF 2C           STX     $2C
01060 2753 8D 1F           BSR     STORKYWD
01070 2755 08              INX
01080 2756 A6 00   SLOOP   LDA A   X
01090 2758 4D              TST A
01100 2759 27 0A           BEQ     ENDR
01110 275B 81 19           CMP A   #$19
01120 275D 2C 04           BGE     STRCHR
01130 275F 8D 13           BSR     STORKYWD
01140 2761 20 F3           BRA     SLOOP
01150 2763 8D 1C   STRCHR  BSR     TRNSFR+2
01160 2765 DE 24   ENDR    LDX     $24
01170 2767 A7 00           STA A   X
01180 2769 CE 00B0         LDX     #$B0
01190 276C BD 0520         JSR     PSTR
01200 276F DE 24           LDX     $24
01210 2771 7E 26A6         JMP     INPUT
01220 2774 EE 00   STORKY  LDX     X
01230 2776 09              DEX
01240 2777 09      BACKUP  DEX
01250 2778 A6 00           LDA A   X
01260 277A 26 FB           BNE     BACKUP
01270 277C 08              INX
01280 277D 08              INX
01290 277E 08              INX
01300 277F A6 00   TRNSFR  LDA A   X
01310 2781 81 20           CMP A   #32
01320 2783 2D 0E           BLT     ENDSTR
01330 2785 08              INX
01340 2786 DF AC           STX     $AC
01350 2788 DE 24           LDX     $24
01360 278A A7 00           STA A   X
01370 278C 08              INX
01380 278D DF 24           STX     $24
01390 278F DE AC           LDX     $AC
01400 2791 20 EC           BRA     TRNSFR
01410 2793 DE 2C   ENDSTR  LDX     $2C
01420 2795 08              INX
01430 2796 08              INX
01440 2797 39              RTS
01450 2798 8C 00B0 ZERCSR  CPX     #$B0
01460 279B 27 08           BEQ     ENDZ
01470 279D 86 08           LDA A   #$08
01480 279F BD 03F1         JSR     OUTC
01490 27A2 09              DEX
01500 27A3 20 F3           BRA     ZERCSR
01510 27A5 39      ENDZ    RTS
01520 27A6 CE 00AF ENDRTN  LDX     #$AF
01530 27A9 08      LOOP    INX
01540 27AA A6 00           LDA A   X
01550 27AC 4D              TST A
01560 27AD 26 FA           BNE     LOOP
01570 27AF 7E 050C         JMP     PLINEC
01580 27B2 81 05   PATCH3  CMP A   #5
01590 27B4 26 03           BNE     GOBACK
01600 27B6 7E 2698         JMP     EDIT
01610 27B9 B1 AC00 GOBACK  CMP A   $AC00
01620 27BC 7E 04CA         JMP     $4CA
```

More

Listing 2 continued.

```
01630       27BF    PRGEND EQU    *
01640 04C7                 ORG    $4C7
01650 04C7 7E 27B2         JMP    PATCH3
01660 014E                 ORG    $14E
01670 014E 27BF            FDB    PRGEND
01680 A048                 ORG    $A048
01690 A048 0100            FDB    $100
01700                      END
PSHX   0547
BERROR 04F0
INPUTC 0428
OUTC   03F1
PULX   055C
BERR2  0BA1
NUMC   0BA4
LINEC  0B61
PSTR   0520
PLINEC 050C
EDIT   2698
CLEAR  269B
INPUT  26A6
CONT1  26B0
CONT3  26C7
STORCH 26CB
FORWAR 26CD
CONT2  26D8
ERROR  26DE
BACKSP 26E1
LOOPBA 26E7
ATBEG  26E9
DELETE 26F0
MOVELF 26F6
SPACE  2704
MOVECS 2709
ENDD   2718
ADD    271D
MOVERT 2725
RECALL 273B
ERR2   2745
CONT4  2748
SLOOP  2756
STRCHR 2763
ENDR   2765
STORKY 2774
BACKUP 2777
TRNSFR 277F
ENDSTR 2793
ZERCSR 2798
ENDZ   27A5
ENDRTN 27A6
LOOP   27A9
PATCH3 27B2
GOBACK 27B9
PRGEND 27BF

TOTAL ERRORS 00000
```

Listing 3. Editing modification for Percom Super BASIC.

```
                NAM   EDITOR    FOR SUPER BASIC
               *           R.SILVER MAY 13,1979
               * BASIC ENTRY POINTS
(08B5)         PSHX  EQU   $08B5     PUSH INDEX
(0081)         BUFFER EQU  $81       BEGINNING OF THE BASIC BUFFER
(00C9)         BUFEND EQU  $C9       END OF BUFFER
(10C7)         BERROR EQU  $10C7     ERROR ROUTINE
(0570)         INPUTC EQU  $0570     INPUT ROUTINE
(0561)         OUTC   EQU  $0561     OUTPUT ROUTINE
(08CA)         PULX   EQU  $08CA     PULL INDEX
(002A)         COUNTR EQU  $2A
(004A)         BUFPNT EQU  $4A       POINTER TO BUFFER POSITION
(002E)         MEMPNT EQU  $2E       POINTER TO MEMORY POSITION
(0047)         TEMP   EQU  $47
(10C7)         BERR2  EQU  $10C7     ERROR ROUTINE
(1414)         NUMC   EQU  $1414     NUMERIC CONVERSION
(0F22)         LINEC  EQU  $0F22     FIND LINE IN MEMORY
(08A4)         PSTR   EQU  $08A4     PRINT STRING ROUTINE
(0885)         PLINEC EQU  $0885     BRANCH POINT BACK TO BASIC
               * MISC CONSTANTS
(0008)         BACK   EQU  $08       BACK SPACE CHARACTER
(0009)         TAB    EQU  $09       FORWARD TAB CHARACTER
(3000)               ORG   $3000
```

More

1. Load Microware's RTEDIT editor (supplied on the A/BASIC cassette) into the computer.

2. Run RTEDIT to enter and edit the source code.

3. When done, save the source code to cassette.

4. Load the A/BASIC compiler into memory.

5. Set up two cassette recorders—one to read the source code, the other to record the machine-language code. The recorder used to read the source code must have motor control so that A/BASIC can start and stop it as needed.

6. Start A/BASIC and start reading the source code.

7. A/BASIC is a two-pass compiler which must read the source code twice. After the first read is finished, rewind the tape.

8. Start both tape recorders and continue A/BASIC.

9. When the compiler is done, rewind both tapes and use the monitor's L function to read the machine language into the computer.

10. Now execute the program.

11. Since programs never run the first time, figure out what went wrong and go back to step 1 to start all over again, as many times as are needed to make it work.

Sound tedious? Right! Add to this the necessity to get RT/68, two tape recorders and motor control, and the process becomes expensive as well.

But there is a way out. With a little patching it is possible to use A/BASIC without spending as much time and without all the extra hardware. The patches do two things—allow RTEDIT and A/BASIC to work with MIKBUG or SWTBUG, and keep the source code in memory so it does not have to be read twice from tape by the compiler.

Listing 5 shows the patches required to RTEDIT. The first portion (down to location 01A6) substitutes the MIKBUG addresses for INEEE and OUTEEE instead of the addresses used by RT/68.

Immediately following is a short subroutine, at A04A, for printing a carriage return and line feed. RT/68 contains such a CR/LF subroutine, and Microware uses it both in the editor as well as in A/BASIC and even compiled programs. Thus, we need a short subroutine in RAM.

This first portion of the patch is all you need if you are using the SWTP AC-30 cassette interface with motor control and two recorders. However,

Listing 3 continued.

```
3000 BD 08B5  EDIT   JSR   PSHX
3003 6F 00    CLEAR  CLR   ,X          CLEAR BUFFER TO END
3005 08              INX
3006 8C 00C9         CPX   #BUFEND     IS IT THE END OF THE BUFFER?
3009 26 F8           BNE   CLEAR
300B BD 08CA         JSR   PULX
300E BD 0570  INPUT  JSR   INPUTC      INPUT LOOP
3011 81 0D           CMP A #$0D        CHECK FOR CARRIAGE CONTROL
3013 26 03           BNE   CONT1
3015 7E 310E         JMP   ENDRTN      IF SO JUMP BACK TO BASIC
3018 81 08    CONT1  CMP A #BACK       CHECK FOR BACKSPACE
301A 27 2D           BEQ   BACKSP
301C 81 09           CMP A #TAB
301E 27 15           BEQ   FORWRD
3020 81 04           CMP A #$04        CONTROL D
3022 27 34           BEQ   DELETE
3024 81 01           CMP A #$01        CONTROL A
3026 27 5D           BEQ   ADD
3028 81 12           CMP A #$12        CONTROL R
302A 26 03           BNE   CONT3
302C 7E 30A3         JMP   RECALL
302F 81 1F    CONT3  CMP A #$1F        CHECK IF CONTROL CHARACTER
3031 2B DB           BMI   INPUT       IF SO IGNORE
3033 A7 00    STORCH STA A ,X          STORE CHARACTER IN BUFFER
              * MOVE CURSOR FORWARD
3035 A6 00    FORWRD LDA A ,X
              * CHECK IF AT END OF LINE IN BUFFER
3037 26 07           BNE   CONT2       BRANCH IF NOT
3039 86 08           LDA A #BACK
              * PUT CURSOR BACK WHERE IT BELONGS
303B BD 0561         JSR   OUTC
303E 20 CE           BRA   INPUT
              * MOVE TO NEXT POSITION IN BUFFER
3040 08       CONT2  INX               MOVE TO NEXT POSITION IN BUFFER
3041 8C 00C9         CPX   #BUFEND     CHECK FOR END OF BUFFER
3044 26 C8           BNE   INPUT
3046 7E 10C7  ERROR  JMP   BERROR
              * BACKUP CURSOR
3049 8C 0081  BACKSP CPX   #BUFFER     CHECK IF AT START OF BUFFER
304C 27 03           BEQ   ATBEG       BRA IF SO
304E 09              DEX
304F 20 BD    LOOPBA BRA   INPUT
3051 86 09    ATBEG  LDA A #TAB        MOVE CURSOR FORWARD
3053 BD 0561         JSR   OUTC
3056 20 B6           BRA   INPUT



              * DELETE A CHARACTER FROM BUFFER AND CRT
3058 BD 08B5  DELETE JSR   PSHX
              * ZERO COUNTER IN ORDER TO REPOSITION CURSOR
305B 7F 002A         CLR   COUNTR
              * MOVE CURSOR AND BUFFER CONTENTS ONE POSITION LEFT
305E A6 01    MOVLFT LDA A 1,X
3060 BD 0561         JSR   OUTC
3063 A7 00           STA A X
3065 08              INX
3066 7C 002A         INC   COUNTR      COUNT CHARACTERS TO END
3069 4D              TST A
306A 26 F2           BNE   MOVLFT
306C 86 20    SPACE  LDA A #$20        PRINT SPACE
306E BD 0561         JSR   OUTC
              * PUT CURSOR BACK TO CORRECT POSITION
3071 7D 002A  MOVECS TST   COUNTR      CHECK TO SEE IF FINISHED
3074 27 0A           BEQ   ENDD
3076 86 08           LDA A #BACK
3078 BD 0561         JSR   OUTC
307B 7A 002A         DEC   COUNTR
307E 20 F1           BRA   MOVECS
3080 BD 08CA  ENDD   JSR   PULX
3083 20 CA           BRA   LOOPBA



3085 BD 08B5  ADD    JSR   PSHX
              * ADD SPACE TO BUFFER AT CURSOR POSITION
3088 7F 002A         CLR   COUNTR      ZERO COUNTER
```

More

the rest of the patch will allow the source code to stay in memory while the compiler is loaded. (You'd still want to make a cassette tape of the edited source code now and then to guard against program loss, but this tape would be strictly for backup and not needed by the compiler.)

Listing 6 shows the patches required to the A/BASIC compiler itself. As before, the first part of Listing 6 must always be done to use A/BASIC with non-RT/68 systems. It patches A/BASIC to work with MIKBUG/SWTBUG-compatible monitors. Again, we see a CR/LF routine at A04A.

If you are not using two recorders and motor control, then the second half of the patch is required as well. This part allows A/BASIC to use the source code which was left in memory by RTEDIT. Note that A/BASIC needs to read the source code twice, but since the code is already in memory, that is easy to arrange without any extra work.

A/BASIC still needs a cassette recorder to store the machine-language object code, but this one need not have motor control. A/BASIC outputs the machine code fairly fast, but still slowly enough that the monitor will have no trouble reading it later. (It comes out in the standard S1....MIKBUG format.)

Once the machine-language code is on cassette, there is one more thing to do. A/BASIC does not insert CR/LF codes into the object code, but normally calls the appropriate routine in RT/68. The code output by the modified compiler will now call the routine at A04A instead. This means that this routine must be inserted into memory any time the object program is run.

If the program is to be run often, or if you intend to give a copy to someone else, it is much more convenient to append a short CR/LF routine to the end of the compiled program (the A/BASIC compiler tells you where the end is) and change all the references from A04A to the new address. Use a search program to find all occurrences of A04A in the object code and replace them with the address of the added subroutine.

A/BASIC Disk Operation

A/BASIC is, of course, most convenient to use in a disk system. Microware sells a disk version of A/BASIC for SWTP and SSB disk systems, but it costs $150 rather than

$65 for the cassette version.

If you don't need the disk files which the disk version allows, then you can patch the cassette version to work with a disk. A patch for the Percom disk system is part of the Percom Users' Library, and the patch for miniFlex is shown in Listing 7.

This program patches the cassette compiler to work with MIKBUG/SWTBUG systems, and to allow the use of miniFlex text files. The source code can be generated with either the BUILD command or, better yet, with the Text Editor.

Rather than write the compiled object code back on disk, this patch leaves it in memory starting at location $2000 and returns to the monitor when done.

A/BASIC source code specifies by means of an ORG statement the location where the object code should go. Regardless of what origin is specified, this patch will place the object code starting at $2000. If the code was origined at $2000, then it is in the right place and can be executed immediately after compilation, or saved with the disk operating system's SAVE command for later use.

If the program was origined for a different location, then it must be moved there before it can be run. A memory move utility is part of the TSC Flex Utilities package, or else the MOVE routine (described in the September 1980 installment) can also be used.

Both the A/BASIC compiler as well as the compiled object program use Flex's PCRLF routine at $711E for carriage returns and line feeds. If the compiled program is to be run on a non-miniFlex system, then it will again be necessary to insert a CR/LF routine somewhere, search out all the jumps to $711E and substitute jumps to the new routine.

Listing 7 is set up for a 32K or larger system. If it is run on a smaller system, the instruction at 16BA will have to be changed to reflect the lower memory limit. It is presently set at $6FFF so that the compiled program has 20K of room from $2000 to $6FFF, but does not erase miniFlex at $7000.

One last thought—although it is tempting to try disassembling a compiler such as A/BASIC, and then reassembling on a 6809 system to produce a 6809 compiler, don't bother. Even if it runs, it will still generate 6800 code! Changing it enough to produce 6809 output is a tough job. ■

Listing 3 continued.

```
308B C6 20          LDA B #$20      PUT SPACE IN BUFFER
                  * MOVE CURSOR AND BUFFER RIGHT ONE PLACE
308D 17           MOVERT TBA
308E E6 00          LDA B X
3090 A7 00          STA A X
3092 BD 0561        JSR   OUTC
3095 08             INX
3096 8C 00C9        CPX   #BUFEND   CHECK FOR FILLED BUFFER
3099 27 AB          BEQ   ERROR
309B 7C 002A        INC   COUNTR    COUNT CHARACTERS FILLED
309E 4D             TST A
309F 26 EC          BNE   MOVERT
30A1 20 C9          BRA   SPACE
                  * RECALL LINE FROM MEMORY AND PLACE IN BUFFER
30A3 8D 5B        RECALL BSR  ZERCSR    MOVE CURSOR TO START OF LINE
30A5 CE 0081        LDX   #BUFFER   FIND LINE NUMBER IN BUFFER
30A8 BD 1414        JSR   NUMC      CONVERT TO BCD FOR LINEC
30AB 24 03          BCC   CONT4     INVALID LINE NUMBER?
30AD 7E 10C7      ERR2  JMP   BERR2
                  * STORE POSITION OF END OF LINE NUMBER
30B0 DF 4A        CONT4 STX   BUFPNT
30B2 BD 0F22        JSR   LINEC     FIND LINE IN MEMORY
30B5 25 F6          BCS   ERR2      BRANCH IF NOT FOUND
                  * SKIP PAST LINE NUMBER STORED IN MEMORY
30B7 08             INX
30B8 08             INX
                  * STORE POSITION OF LINE IN MEMORY
30B9 DF 2E          STX   MEMPNT
30BB 8D 1F          BSR   STORKY    STORE KEYWORD IN BUFFER
30BD 08             INX
                  * TRANSFER CHARACTERS FROM MEMORY TO BUFFER
30BE A6 00        SLOOP LDA A X
30C0 4D             TST A
                  * BRANCH IF LAST CHARACTER IN LINE
30C1 27 0A          BEQ   ENDR
30C3 81 19          CMP A #$19      ANOTHER KEYWORD?
30C5 2C 04          BGE   STRCHR    BRANCH IF NOT
30C7 8D 13          BSR   STORKY    STORE KEYWORD IN BUFFER
30C9 20 F3          BRA   SLOOP
30CB 8D 1C        STRCHR BSR  TRNSFR+2  STORE CHARACTER IN BUFFER
30CD DE 4A        ENDR  LDX   BUFPNT    STORE $00 IN BUFFER
30CF A7 00          STA A X
30D1 CE 0081        LDX   #BUFFER   PRINT CONTENTS OF THE BUFFER
30D4 BD 08A4        JSR   PSTR
                  * SET INDEX TO LAST CHARACTER IN BUFFER
30D7 DE 4A          LDX   BUFPNT
30D9 7E 300E        JMP   INPUT
30DC EE 00        STORKY LDX  0,X       STORE KEYWORD IN BUFFER
30DE 09             DEX
30DF 09           BACKUP DEX            LOCATE KEYWORD IN TABLE
30E0 A6 00          LDA A X
30E2 26 FB          BNE   BACKUP
                  * SKIP PAST JUMP ADDRESS IN TABLE
30E4 08             INX
30E5 08             INX
30E6 08             INX
30E7 A6 00        TRNSFR LDA A 0,X      TRANSFER CHARACTERS TO BUFFER
30E9 81 20          CMP A #32       LAST CHARACTER? OR KEY WORD?
30EB 2D 0E          BLT   ENDSTR    BRANCH IF SO
30ED 08             INX
30EE DF 47          STX   TEMP      STORE MEMORY POINTER
30F0 DE 4A          LDX   BUFPNT    PICK UP BUFFER INDEX
30F2 A7 00          STA A X
30F4 08             INX
30F5 DF 4A          STX   BUFPNT
30F7 DE 47          LDX   TEMP
30F9 20 EC          BRA   TRNSFR
                  * LOAD POSITION OF BUFFER AFTER LINE#
30FB DE 2E        ENDSTR LDX  MEMPNT
                  * LOAD POSITION OF BUFFER AFTER LINE#
30FD 08             INX
30FE 08             INX
30FF 39             RTS
3100 8C 0081      ZERCSR CPX  #BUFFER   AT BEGINNING OF BUFFER?
3103 27 08          BEQ   ENDZ      BRANCH IF YES
3105 86 08          LDA A #BACK
3107 BD 0561        JSR   OUTC
310A 09             DEX
```

More →

Listing 3 continued.

```
310B 20 F3          BRA   ZERCSR
310D 39       ENDZ  RTS



310E CE 0080  ENDRTN LDX  #BUFFER-1 FIND END OF LINE IN BUFFER
3111 08       LOOP  INX
3112 A6 00          LDA A X
3114 4D             TST A
3115 26 FA          BNE   LOOP
3117 7E 0885        JMP   PLINEC
   (311A)     PRGEND EQU  *



   (0847)           ORG   $0847
0847 81 05          CMP A #5
0849 26 03          BNE   CONT6
084B 7E 3000        JMP   EDIT
084E B1 0153  CONT6 CMP A $153      CHECK FOR CONTROL X
0851 27 D4          BEQ   $827
0853 B1 0154        CMP A $154
0856 27 09          BEQ   CONT7
0858 81 0D          CMP A #$0D      CHECK FOR CAR RETURN
085A 27 29          BEQ   PLINEC
085C 2D E3          BLT   $841
085E 20 0F          BRA   $86F
0860 01             NOP
0861 B6 0155  CONT7 LDA A $0155
   (014E)           ORG   $14E
014E 31 1A          FDB   PRGEND
```

Listing 4. Editing modification for the TSC Text Editor.

```
                    NAM   TSCEDIT   CURSOR EDIT FOR TSC EDITOR
                  * R. SILVER APR 17,1979
   (04BB)     BUFLIM EQU   $4BB
   (0040)     TEMP   EQU   $40
   (008E)     CHRCNT EQU   $8E
   (0099)     FILEND EQU   $99
   (0097)     FILBEG EQU   $97
   (0091)     NUMBER EQU   $91
   (0206)     INPUTC EQU   $206
   (073B)     CLASS  EQU   $73B
   (0209)     OUTC   EQU   $209
   (038A)     TEDIT  EQU   $38A
   (00BB)     BUFFER EQU   $BB
   (045F)     BERR2  EQU   $45F
   (0755)     NUMC   EQU   $755
   (07AB)     LINEC  EQU   $7AB
   (0485)     PSTR   EQU   $485
   (03CC)     PLINEC EQU   $3CC

   (1D7D)           ORG   $1D7D      *** SEE TEXT ***
1D7D BD 0206  PATCH JSR   INPUTC
1D80 81 05          CMP A #5         IS IT CONTROL E?
1D82 27 03          BEQ   EDIT
1D84 7E 049C        JMP   INCHR1
1D87 DF 40    EDIT  STX   TEMP
1D89 6F 00    CLEAR CLR   ,X
1D8B 08             INX
1D8C 8C 0143        CPX   #BUFFER+136
1D8F 26 F8          BNE   CLEAR
1D91 DE 40          LDX   TEMP
1D93 BD 0206  INPUT JSR   INPUTC
1D96 81 0D          CMP A #$0D
1D98 26 03          BNE   CONT1
1D9A 7E 1E77        JMP   ENDRTN
1D9D 81 08    CONT1 CMP A #$08
1D9F 27 29          BEQ   BACKSP
1DA1 81 09          CMP A #$09
1DA3 27 15          BEQ   FORWRD
```

More

Listing 4 continued.

```
1DA5 81 04          CMP A #$04
1DA7 27 30          BEQ   DELETE
1DA9 81 01          CMP A #$01
1DAB 27 59          BEQ   ADD
1DAD 81 12          CMP A #$12
1DAF 26 03          BNE   CONT3
1DB1 7E 1E23        JMP   RECALL
1DB4 81 1F    CONT3 CMP A #$1F
1DB6 2B DB          BMI   INPUT
1DB8 A7 00    STORCH STA A ,X
1DBA A6 00    FORWRD LDA A ,X
1DBC 26 07          BNE   CONT2
1DBE 86 08          LDA A #$08
1DC0 BD 0209        JSR   OUTC
1DC3 20 CE          BRA   INPUT
1DC5 BD 04BB  CONT2 JSR   BUFLIM
1DC8 20 C9          BRA   INPUT
1DCA 8C 00BB  BACKSP CPX  #BUFFER
1DCD 27 03          BEQ   ATBEG
1DCF 09             DEX
1DD0 20 C1    LOOPBA BRA  INPUT
1DD2 86 09    ATBEG LDA A #$09
1DD4 BD 0209        JSR   OUTC
1DD7 20 BA          BRA   INPUT
1DD9 DF 40    DELETE STX  TEMP
1DDB 7F 0091        CLR   NUMBER
1DDE A6 01    MOVLFT LDA A 1,X
1DE0 16             TAB
1DE1 BD 0209        JSR   OUTC
1DE4 17             TBA
1DE5 A7 00          STA A X
1DE7 08             INX
1DE8 7C 0091        INC   NUMBER
1DEB 4D             TST A
1DEC 26 F0          BNE   MOVLFT
1DEE 86 20    SPACE LDA A #$20
1DF0 BD 0209        JSR   OUTC
1DF3 7D 0091  MOVECS TST  NUMBER
1DF6 27 0A          BEQ   ENDD
1DF8 86 08          LDA A #$08
1DFA BD 0209        JSR   OUTC
1DFD 7A 0091        DEC   NUMBER
1E00 20 F1          BRA   MOVECS
1E02 DE 40    ENDD  LDX   TEMP
1E04 20 CA          BRA   LOOPBA
1E06 DF 40    ADD   STX   TEMP
1E08 7F 0091        CLR   NUMBER
1E0B C6 20          LDA B #$20
1E0D 17       MOVERT TBA
1E0E E6 00          LDA B X
1E10 A7 00          STA A X
1E12 BD 0209        JSR   OUTC
1E15 BD 04BB        JSR   BUFLIM
1E18 7C 0091        INC   NUMBER
1E1B 17             TBA
1E1C 26 F0          BNE   MOVERT+1
1E1E 20 D3          BRA   MOVECS
1E20 7E 045F  ERROR JMP   BERR2
1E23 8D 44    RECALL BSR  ZERCSR
1E25 CE 00BB        LDX   #BUFFER
1E28 BD 073B        JSR   CLASS
1E2B C1 01          CMP B #1
1E2D 26 F1          BNE   ERROR
1E2F BD 0755        JSR   NUMC
1E32 86 3D          LDA A #'=
1E34 A7 00          STA A X
1E36 08             INX
1E37 DF 40          STX   TEMP
1E39 BD 07AB        JSR   LINEC
1E3C 5D             TST B
1E3D 26 E1          BNE   ERROR
1E3F 08             INX
1E40 08             INX
1E41 08             INX
1E42 A6 00    SLOOP LDA A X
1E44 81 0D          CMP A #$0D
1E46 27 0E          BEQ   ENDR
```

More →

Listing 4 continued.

```
1E48 08              INX
1E49 DF 91           STX   NUMBER
1E4B DE 40           LDX   TEMP
1E4D A7 00           STA A X
1E4F 08              INX
1E50 DF 40           STX   TEMP
1E52 DE 91           LDX   NUMBER
1E54 20 EC           BRA   SLOOP
1E56 DE 40    ENDR   LDX   TEMP
1E58 86 04           LDA A #4
1E5A A7 00           STA A X
1E5C CE 00BB         LDX   #BUFFER
1E5F BD 0485         JSR   PSTR
1E62 DE 40           LDX   TEMP
1E64 6F 00           CLR   X
1E66 7E 1D93  LOOP2  JMP   INPUT
1E69 8C 00BB  ZERCSR CPX   #BUFFER
1E6C 27 08           BEQ   ENDZ
1E6E 86 08           LDA A #8
1E70 BD 0209         JSR   OUTC
1E73 09              DEX
1E74 20 F3           BRA   ZERCSR
1E76 39       ENDZ   RTS
1E77 CE 00BA  ENDRTN LDX   #BUFFER-1
1E7A 5F              CLR B
1E7B 08       LOOP   INX
1E7C 5C              INC B
1E7D A6 00           LDA A X
1E7F 4D              TST A
1E80 26 F9           BNE   LOOP
1E82 D7 8E           STA B CHRCNT
1E84 86 0D           LDA A #$D
1E86 A7 00           STA A X
1E88 7E 03CC         JMP   PLINEC
1E8B 0D              FCB   $0D SET END
   (1E8C)    PRGEND EQU   *

   (0499)           ORG   $499
0499 7E 1D7D INCHAR JMP   PATCH
049C 81 08   INCHR1 CMP A #8

   (0358)           ORG   $358
0358 CE 1E8C        LDX   #PRGEND

   (16DB)           ORG   $16DB     *** SEE TEXT ***
16DB CE 1E8C        LDX   #PRGEND
                    END
```

Listing 5. Patches required for RTEDIT to work with MIKBUG or SWTBUG.

```
                * THIS PROGRAM PATCHES THE MICRO-WARE EDITOR
                * TO WORK WITH MIKBUG/SWATBUG SYSTEMS
                *
                * MIKBUG EQUATES
E1AC            INEEE   EQU    $E1AC
E1D1            OUTEEE  EQU    $E1D1
E0D0            MON     EQU    $E0D0
                *
0D17                    ORG    $0D17
0D17 E1 D1              FDB    OUTEEE
0D31                    ORG    $0D31
0D31 E1 D1              FDB    OUTEEE
0DCF                    ORG    $0DCF
0DCF E1 D1              FDB    OUTEEE
0D0E                    ORG    $0D0E
0D0E E1 D1              FDB    OUTEEE
                *
0D5D                    ORG    $0D5D
0D5D A0 4A              FDB    CRLF
0DD6                    ORG    $0DD6
0DD6 A0 4A              FDB    CRLF
                *
0D23                    ORG    $0D23
```

More →

Listing 5 continued.

```
0D23 E1 AC              FDB     INEEE
0CF1                    ORG     $0CF1
0CF1 E1 AC              FDB     INEEE
0CF1                    ORG     $0CF1
0CF1 E1 AC              FDB     INEEE
                *
01A6                    ORG     $01A6
01A6 E0 D0              FDB     MON
                *
A04A                    ORG     $A04A
A04A 86 0A      CRLF    LDA A   #$0A
A04C 8D 02              BSR     JOUT
A04E 86 0D              LDA A   #$0D
A050 7E E1 D1   JOUT    JMP     OUTEEE
                *
                * THE ABOVE IS ALL THAT IS NECESSARY IF YOU
                * WANT TO USE THE AC-30.
                * THE FOLLOWING WILL ALLOW THE SOURCE TO STAY IN
                * MEMORY WHILE THE COMPILER IS BEING LOADED.
                * THE COMPILER OUTPUT WILL STILL BE TO CASSETTE.
                * NOTE SOME OF THE ABOVE PATCHES WILL NOT BE
                * NECESSARY IF YOU DO THE FOLLOWING. ($0D0E, $0CF1,
                * $0CF9) PATCH COMPILER TO USE MEMORY
                *
                *
                *SECTION TO REWIND MEMORY
                *
0100                    ORG     $0100
0100 7E 20 23           JMP     REWIND
0103            RESTART EQU     $0103
                * SECTION TO STORE SOURCE IN MEMORY
                *
0DAB                    ORG     $0DAB
0DAB 86 20              LDA A   #$20
0DAD B7 0D CF           STA A   JMEM+1
0DB0 86 10              LDA A   #$10
0DB2 B7 0D D0           STA A   JMEM+2
0DB5 01                 NOP
0DB6 01                 NOP
0DB7 01                 NOP
0DB8 01                 NOP
0DB9 86 02              LDA A   #$02        LOAD DATA START FLAG
0DBB 8D 11              BSR     JMEM
0DBD 8D D8              BSR     PBUFF
0DBF 86 03              LDA A   #$03        LOAD DATA END FLAG
0DC1 8D 0B              BSR     JMEM
0DC3 86 E1              LDA A   #$E1
0DC5 B7 0D CF           STA A   JMEM+1
0DC8 86 D1              LDA A   #$D1
0DCA B7 0D D0           STA A   JMEM+2
0DCD 39                 RTS
0DCE 7E E1 D1   JMEM    JMP     OUTEEE
0D97            PBUFF   EQU     $0D97
                *
                *
                *
                * SECTION TO LOAD NEMORY TO EDITOR
                *
0CEA                    ORG     $0CEA
0CEA 5F                 CLR B
0CEB 8D 15      IN1     BSR     JIN
0CED 81 02              CMP A   #$02        START OF DATA
0CEF 26 FA              BNE     IN1
0CF1 8D 0F      IN2     BSR     JIN
0CF3 81 03              CMP A   #$03        END?
0CF5 27 08              BEQ     JEND
0CF7 A7 00              STA A   0,X
0CF9 08                 INX
0CFA 5C                 INC B
0CFB C1 80              CMP B   #$80        128 BYTES YET?
0CFD 25 F2              BCS     IN2
0CFF 6F 00      JEND    CLR     0,X
0D01 39                 RTS
0D02 7E 20 00   JIN     JMP     DATAIN
                *
                *
                *
2000                    ORG     $2000
```

More

Listing 5 continued.

```
2000 FF 20 0D  DATAIN  STX    XSAV+1
2003 CE 20 3B  LDATA1  LDX    #DATA
2006 A6 00             LDA A  0,X
2008 08                INX
2009 FF 20 04          STX    LDATA1+1
200C CE 00 00  XSAV    LDX    #0000
200F 39                RTS
                *
                *
                *
2010 FF 20 20  DATAOUT STX    XSAV2+1
2013 CE 20 3B  LDATA2  LDX    #DATA
2016 A7 00             STA A  0,X
2018 08                INX
2019 FF 20 14          STX    LDATA2+1
201C BD E1 D1          JSR    OUTEEE
201F CE 00 00  XSAV2   LDX    #0000
2022 39                RTS
                *
                *
                *
2023 4F        REWIND  CLR A
2024 C6 32             LDA B  #$32
2026 CE 20 3B          LDX    #DATA
2029 FF 20 04          STX    LDATA1+1
202C FF 20 14          STX    LDATA2+1
202F 7E 01 03          JMP    RESTART
                *
                *
2032                   RMB    $9        ALTERNATE CRLF ROUTINE AREA
                *
203B           DATA    EQU    *
                       END
```

Listing 6. Patches required for A/BASIC to work with MIKBUG or SWTBUG.

```
                * THIS PROGRAM PATCHES THE MICRO-WARE COMPILER
                * TO WORK WITH MIKBUG/SWATBUG SYSTEMS
                *
                * MIKBUG EQUATES
E1AC           INEEE   EQU    $E1AC
E1D1           OUTEEE  EQU    $E1D1
E0D0           MON     EQU    $E0D0
E0BF           OUT2H   EQU    $E0BF
                *
16E5                   ORG    $16E5
16E5 E1 AC             FDB    INEEE
18E5                   ORG    $18E5
18E5 E1 AC             FDB    INEEE
18ED                   ORG    $18ED
18ED E1 AC             FDB    INEEE
1917                   ORG    $1917
1917 E1 AC             FDB    INEEE
                *
16EE                   ORG    $16EE
16EE A0 4A             FDB    CRLF
17D0                   ORG    $17D0
17D0 A0 4A             FDB    CRLF
1951                   ORG    $1951
1951 A0 4A             FDB    CRLF
                *
16E2                   ORG    $16E2
16E2 E1 D1             FDB    OUTEEE
17C9                   ORG    $17C9
17C9 E1 D1             FDB    OUTEEE
1902                   ORG    $1902
1902 E1 D1             FDB    OUTEEE
190B                   ORG    $190B
190B E1 D1             FDB    OUTEEE
1925                   ORG    $1925
1925 E1 D1             FDB    OUTEEE
                *
0827                   ORG    $0827
```

More

Listing 6 continued.

```
0827 E0 D0              FDB     MON
                        *
A04A                    ORG     $A04A
A04A 86 0A      CRLF    LDA A   #$0A
A04C 8D 02              BSR     JOUT
A04E 86 0D              LDA A   #$0D
A050 7E E1 D1   JOUT    JMP     OUTEEE
                        *
                        * THIS WAS A TOUGH ONE
                        *
16D1                    ORG     $16D1
16D1 BD A0 4A           JSR     CRLF
16D4 96 42              LDA A   $42
16D6 26 2C              BNE     CONT
16D8 39                 RTS
16D9 EB 00      PATCH   ADD B   0,X
16DB 7E E0 BF           JMP     OUT2H
16DE 7E 16 D9           JMP     PATCH
1704            CONT    EQU     $1704
                        *
                        * THIS WAS EVEN TOUGHER
                        * FIXES OPT S
                        *
0F11                    ORG     $0F11
0F11 01                 NOP
0F12 01                 NOP
                        *
                        *
                        * THE ABOVE IS ALL THAT IS NECESSARY IF YOU
                        * WANT TO USE THE AC-30
                        * THE FOLLOWING WILL ALLOW THE SOURCE TO STAY IN
                        * MEMORY WHILE THE COMPILER IS BEING LOADED
                        * THE COMPILER OUTPUT WILL STILL BE TO CASSETTE.
                        * NOTE SOME OF THE ABOVE PATCHES WILL NOT BE
                        * NECESSARY IF YOU DO THE FOLLOWING. ($18E5, $18ED)
                        * THE EDITOR WILL HAVE TO BE PATCHED TO MATCH
                        *
                        *SECTION TO REWIND MEMORY
                        *
00FB                    ORG     $00FB
00FB BD 20 0D   BACK    JSR     REWIND
00FE 20 10              BRA     FWD
0100 20 F9              BRA     BACK
0110            FWD     EQU     $0110
                        *
                        *
                        *
                        * SECTION TO LOAD NEMORY TO COMPILER
                        *
18DE                    ORG     $18DE
18DE 5F                 CLR B
18DF 8D 15      IN1     BSR     JIN
18E1 81 02              CMP A   #$02        START OF DATA
18E3 26 FA              BNE     IN1
18E5 8D 0F      IN2     BSR     JIN
18E7 81 03              CMP A   #$03        END?
18E9 27 08              BEQ     JEND
18EB A7 00              STA A   0,X
18ED 08                 INX
18EE 5C                 INC B
18EF C1 80              CMP B   #$80        128 BYTES YET?
18F1 25 F2              BCS     IN2
18F3 6F 00      JEND    CLR     0,X
18F5 39                 RTS
18F6 7E 20 00   JIN     JMP     DATAIN
                        *
                        *
                        *
2000                    ORG     $2000
2000 FF 20 16   DATAIN  STX     XSAV+1
2003 CE 20 3B   LOAD    LDX     #DATA
2006 A6 00              LDA A   0,X
2008 08                 INX
2009 81 1A              CMP A   #$1A        END?
200B 26 05              BNE     STORE
200D CE 20 3B   REWIND  LDX     #DATA
2010 86 03              LDA A   #$03
2012 FF 20 04   STORE   STX     LOAD+1
```

More →

Listing 6 continued.

```
2015 CE 00 00  XSAV    LDX    #0000
2018 39                RTS
203B           DATA    EQU    $203B
               *
               * IT WILL BE NECESSARY TO KEEP THE CRLF IN
               * SA04A ANYTIME YOU ARE RUNNING PROGRAMS THAT
               * HAVE BEEN COMPILED BY THE A/BASIC COMPILER
               * A BETTER METHOD IS TO INSERT THE CRLF ROUTINE
               * AT THE END OF THE COMPILED CODE.  THE COMPILER
               * TELLS YOU WHERE THE END IS.  THEN SEARCH OUT
               * THE JUMPS TO $A04A AND PATCH TO THE NEW CRLF
               * ROUTINE.  (FOR AN EXCELLENT SEARCH ROUTINE SEE
               * MAR 1978 73 MAGAZINE)
                       END
```

Listing 7. Program to use miniFlex text files.

```
               *
               * THIS PROGRAM PATCHES THE MICRO-WARE COMPILER
               * TO WORK WITH MIKBUG/SWATBUG SYSTEMS
               * AND TO ALLOW THE USE OF FLEX TEXT FILES.
               * THE TSC EDITOR CAN GENERATE THE FILES.
               * OUTPUT WILL BE TO MEMORY STARTING AT $2000.
               * PROGRAMS WITH ORG=$2000 CAN BE RUN THERE, ELSE
               * RELOCATE THEM TO ACTUAL ORG ADDRESS.
               * MAKE THE FOLLOWING CHANGES TO THE
               * PROGRAM TO THE ABASIC ORIGINAL (CASETTE VERSION).
               * MAKE ABASIC A COMMAND THEN USE ABASIC,<TEXT FILE>
               * THE CRLF IS ROUTED TO FLEX.  TO ALLOW THE
               * COMPILED PROGRAMS TO RUN INDEPENDANT OF FLEX
               * IT WILL BE NECESSARY TO SEARCH OUT THE JUMPS
               * TO PCRLF AND PUT A CRLF ROUTINE AT THE END OF
               * THE COMPILED PROGRAM.
               *
               *
               *
               * MIKBUG EQUATES
E0D0           MON     EQU    $E0D0
E1AC           INEEE   EQU    $E1AC
E1D1           OUTEEE  EQU    $E1D1
E07E           PDATA1  EQU    $E07E
               *
               * FIX MON JUMP FOR COMPILED PROGRAMS
               *
0A1A                   ORG    $0A1A
0A1A 86 E0             LDA A  #$E0
0A1C C6 D0             LDA B  #$D0

               *
               * FLEX EQUATES
               *
7112           PUTCHR  EQU    $7112
710F           GETCHR  EQU    $710F
7118           PSTRNG  EQU    $7118
7139           OUTHEX  EQU    $7139
711E           PCRLF   EQU    $711E
7103           WARMS   EQU    $7103
7127           GETFIL  EQU    $7127
712D           SETEXT  EQU    $712D
713C           RPTERR  EQU    $713C
7806           FMS     EQU    $7806
7803           FMSCLS  EQU    $7803
7740           FCB     EQU    $7740
               *
0100           ABASIC  EQU    $0100
               *


1917                   ORG    $1917
1917 E1 AC             FDB    INEEE
               *
0103                   ORG    $0103
0103 76 1F             FDB    REWIND
```

More

Listing 7 continued.

```
                    *
17D0                        ORG     $17D0
17D0 71 1E                  FDB     PCRLF

1951                        ORG     $1951
1951 71 1E                  FDB     PCRLF
                    *
16E1                        ORG     $16E1
16E1 7E 71 12               JMP     PUTCHR
16E4 7E 71 0F               JMP     GETCHR
16E7 7E 17 0F               JMP     OUT4HS
16EA 7E 17 13               JMP     OUT2HS
16ED 7E 71 1E               JMP     PCRLF
16F0 7E 71 18               JMP     PSTRNG
                    *
170F                        ORG     $170F
170F BD 71 39  OUT4HS       JSR     OUTHEX
1712 08                     INX
1713 BD 71 39  OUT2HS       JSR     OUTHEX
1716 20 42                  BRA     OUTSP
175A                        ORG     $175A
175A B6 20     OUTSP        LDA A   #$20
175C 08                     INX
175D 7E 71 12               JMP     PUTCHR

                    *
17C9                        ORG     $17C9
17C9 E1 D1                  FDB     OUTEEE
1902                        ORG     $1902
1902 E1 D1                  FDB     OUTEEE
190B                        ORG     $190B
190B E1 D1                  FDB     OUTEEE
1925                        ORG     $1925
1925 E1 D1                  FDB     OUTEEE
                    *
0827                        ORG     $0827
0827 E0 D0                  FDB     MON
                    *
                    * GO CLOSE ALL FLEX FILES
                    *
1689                        ORG     $1689
1689 BD 76 2C               JSR     CLOSE
168C 01                     NOP
168D 01                     NOP
168E 01                     NOP

                    *
                    * CARRIAGE RETURN LINE FEED ROUTINE
                    *
                    * LOAD OUTPUT TO MEMORY STARTING AT $2000
                    *
16A7                        ORG     $16A7
16A7 CE 00 90               LDX     #$0090      GET BUFFER ADDRESS
16AA E6 00                  LDA B   0,X         GET LENGTH
16AC 08                     INX                 SKIP TO DATA
16AD 08                     INX
16AE 08                     INX
16AF A6 00     OUT1         LDA A   0,X         GET DATA BYTE
16B1 FF 16 C3               STX     OUT3+1      STORE POINTER
16B4 CE 20 00  OUT2         LDX     #$2000      GET MEMORY ADDRESS
16B7 A7 00                  STA A   0,X         STORE DATA
16B9 08                     INX
16BA 8C 6F FF               CPX     #$6FFF      ANY MEMORY LEFT?
16BD 27 0B                  BEQ     OUT4        NO-REPORT MEMORY FULL
16BF FF 16 B5               STX     OUT2+1      YES-STORE MEMORY POINTER
16C2 CE 00 00  OUT3         LDX     #$000       RESTORE BUFFER POINTER
16C5 08                     INX
16C6 5A                     DEC B
16C7 26 E6                  BNE     OUT1        ALL DONE, NO-GOBACK
16C9 39                     RTS
16CA CE 16 D6  OUT4         LDX     #MSG1
16CD BD E0 7E               JSR     PDATA1
16D0 BD 76 2C               JSR     CLOSE
16D3 7E 71 03               JMP     WARMS
16D6 54        MSG1         FCC     /TOO BIG/
16DD 04                     FCB     4
```

More

Listing 7 continued.

```
                    *
                    * THIS WAS TOUGH
                    * FIXES OPT S
                    *
0F11                        ORG     $0F11
0F11 01                     NOP

0F12 01                     NOP
                    *
                    *
                    * SECTION TO LOAD FLEX TEXT FILE TO COMPILER
                    *
18DE                        ORG     $18DE
18DE 5F                     CLR B               CLEAR B FOR BYTE COUNT
18DF FF 18 F6       IN2     STX     XSAV+1      SAVE BUFFER PONTER
18E2 CE 77 40               LDX     #FCB        POINT TO FCB
18E5 BD 78 06               JSR     FMS         GET DATA
18E8 27 0B                  BEQ     XSAV        NO ERROR
18EA A6 01                  LDA A   1,X         GET ERROR
18EC 81 08                  CMP A   #$08        END OF FILE?
18EE 27 03                  BEQ     JRET2
18F0 7E 76 39               JMP     ERROR

18F3 86 03          JRET2   LDA A   #$3
18F5 CE 00 00       XSAV    LDX     #0000       RESTORE X
18F8 A7 00                  STA A   0,X
18FA 81 03                  CMP A   #$03        END?
18FC 27 06                  BEQ     JEND
18FE 08                     INX
18FF 5C                     INC B
1900 C1 80                  CMP B   #$80        128 BYTES YET?
1902 25 DB                  BCS     IN2
1904 6F 00          JEND    CLR     0,X
1906 39                     RTS
                    *
                    * OPEN TEXT FILE FOR ABASIC

                    *
                    * REWIND FILE
                    *
761F 86 05          REWIND  LDA A   #$5         REWIND CODE
7621 CE 77 40               LDX     #FCB
7624 A7 00                  STA A   0,X
7626 BD 78 06               JSR     FMS
7629 26 0E                  BNE     ERROR
762B 39                     RTS
                    *
                    * ERROR SECTION
                    *

7600                        ORG     $7600
7600 CE 77 40       START   LDX     #FCB        POINT TO FCB
7603 BD 71 27               JSR     GETFIL      GET FILE SPEC
7606 25 31                  BCS     ERROR       ANY ERRORS
7608 CE 77 40               LDX     #FCB
760B 86 01                  LDA A   #1          SET DEFAULT EXT.
760D BD 71 2D               JSR     SETEXT
7610 CE 77 40               LDX     #FCB
7613 86 01                  LDA A   #1          OPEN FOR READ
7615 A7 00                  STA A   0,X
7617 BD 78 06               JSR     FMS
761A 26 1D                  BNE     ERROR
761C 7E 01 00               JMP     ABASIC

762C 86 04          CLOSE   LDA A   #4          CLOSE FILE CODE
762E CE 77 40               LDX     #FCB
7631 A7 00                  STA A   0,X
7633 BD 78 06               JSR     FMS
7636 26 01                  BNE     ERROR
7638 39                     RTS
                    *
                    *
7639 BD 71 3C       ERROR   JSR     RPTERR
763C BD 78 03               JSR     FMSCLS      CLOSE FILES
763F 7E 71 03               JMP     WARMS       GOTO FLEX
                            END     START
```