Tiny PILOT: An Educational Language for the 6502

PILOT is a higher level language used for computer aided instruction. This version includes an editor and an interpreter. It requires fewer than 800 bytes of memory.

Nicholas Vrtis
5863 Pinetree S.E.
Kentwood, MI 49508

```
*********************************************************************
* CHAR    * EDIT FUNCTION
*********************************************************************
*          * START EXECUTION OF THE PILOT PROGRAM
* UPARROW* MOVE EDIT POINTER TO START OF PROGRAM
* /        * DISPLAY NEXT LINE OF THE PROGRAM
* %        * PAD TO END OF LINE WITH DELETE CHARACTERS
* B/S      * BACKSPACE TO CORRECT TYPING ERROR
* C/R      * CARRIAGE RETURN - INDICATE END OF STATEMENT
* ANY      * CHARACTER IS STORED IN PROGRAM (MAX 127 PER LINE)
*********************************************************************
* FORMAT * STATEMENT               * WHAT IT DOES
*********************************************************************
* T:TEXT * TYPE                    * DISPLAY THE TEXT ON THE TERMINAL
*        *                         *
* A:     * ACCEPT                  * INPUT UP TO 40 CHARACTERS INTO
*        *                         * ANSWER FIELD
* ?:     * ACCEPT NAME             * INPUT UP TO 40 CHARACTERS INTO
*        *                         * NAME AND ANSWER FIELD.
* M:TEXT * MATCH                   * COMPARE TEXT TO LAST IMPUT FROM
*        *                         * TERMINAL AND SET MATCH FLAG TO
*        *                         * Y IF EQUAL, N IF NOT EQUAL.
* J:N    * JUMP                    * JUMP TO LABEL N FOR NEXT LINE.
*        *                         * J:A MEANS JUMP TO LAST ACCEPT.
*        *                         * J:* MEANS RESTART FROM BEGINNING.
* U:N    * USE SUBROUTINE N        * SAVE ADDRESS OF START OF NEXT
*        *                         * LINE AND THEN PERFORM AS IN JUMP.
* E:     * EXIT FROM SUBROUTINE    * RETURN TO ADDRESS SAVED BY PRIOR
*        *                         * USE STATEMENT.
* S:     * STOP                    * STOP PROGRAM AND RETURN TO EDITOR
*        *                         *
* C:     * COMPUTE                 * PERFORMS ARITHMETIC ON VARIABLES
*        *                         * NAMED A THROUGH Z. ALLOWED
*        *                         * OPERATIONS ARE =, +, AND -
*        *                         * RANGE IS + OR - 999
*        *                         * C:$= WILL PLACE RESULT IN ANSWER
*        *                         * FIELD INSTEAD OF A VARIABLE
* R:     * REMARKS                 * PROGRAM REMARKS - NOT EXECUTED
*        *                         *
*        * CONDITIONALS            * MAY PRECEED ANY STATEMENT.
* N      *                         * EXECUTE ONLY IF MATCH FLAG IS N
* Y      *                         * EXECUTE ONLY IF MATCH FLAG IS Y
*        *                         *
* *N     * LABEL                   * MAY PRECEED ANY STATEMENT OR
*        *                         * CONDITIONAL.  ACTS AS DESTINATION
*        *                         * FOR A JUMP OR USE STATEMENT
* $X     * VARIABLE ITEM           * AS PART OF TEXT CAUSES CONTENTS
*        *                         * OF VARIABLES TO BE DISPLAYED OR
*        *                         * MATCHED.
*        *                         * $? INDICATES NAME FIELDS.
*********************************************************************
```

Are you envious of the guys on your block who have big BASIC systems? Have you ever tried to teach machine language to someone who thinks HEX is an evil spell? I had the same problem until I discovered PILOT, and implemented a small version on my SYM-1. For those who haven't heard of PILOT yet, it is an educational, high level language intended for computer aided instruction. It is a very simple language, with only ten basic instructions, but it incorporates a number of features that make it easy enough to use as a method for introducing people to computers. I have written some math drill programs for my six- and eight-year olds, and in turn, my eight-year old loves to write programs for her little brother to run.

This implementation of PILOT is not a full "standard" version. After all, what do you expect from an interpeter and editor that run in less than 800 bytes? I also could not resist the temptation to change things a little here and there. It is close enough to give a flavor for what PILOT can do, and it makes a nice language to have fun with, even on a 2K system.

The editor performs only the most elementary functions required to get a program in and running. It accepts characters without checking syntax rules, the only limitation being that each line is a maximum of 127 characters long. I compromised at 127, instead of 80, because the sign of the index register changes at 128, and so I avoided a compare.

The program looks for the ASCII back-space character, hex 08, because my CRT actually backspaces. If your terminal doesn't, you might want to change this to a printable character such as the underscore used by many timesharing systems. A check is also made for the backspace in the code for the ACCEPT statement, so be sure to change it there as well.

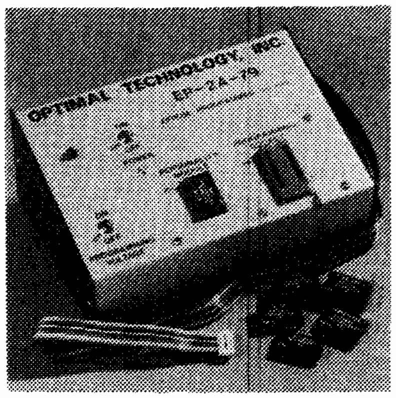
The editor doesn't have a provision for inserting a line between existing lines, but it is possible to change a line, provided you replace it with one of the same length or shorter. The percent key fills from the current position to the next end of line with delete characters, hex FF. Since most terminals ignore these, it works effectively as a delete to the end of the line. The program has to check for these during MATCH and COMPUTE statement processing, since they represent the logical end of line.

The carriage return, entered as the end of line, is converted to a zero by the editor. This simplifies looking for the end of each line, later on, since the zero flag is set as the byte gets loaded. The SYM monitor routine CRLF outputs both the carriage return and the line feed, so one doesn't save anything by keeping the return in the line to output it.

The locations CURAD and CURAD + 1 address the start of each PILOT line. Initially, this is set to $500 by the routine SETBGN. The Y register is incremented o access the next character in the line. At the end of each line, subroutine SCURAD bumps Y one more time to get past the end of line character, and then adds the resulting Y value to the current address and resets Y to zero.

This sets things up for the start of the next line. Performing the line scan in this way saves two bytes each time I need to get to the next character because an INY is used instead of a JSR, and it also makes it easy to check for a line too

EPROM PROGRAMMER
Model EP-2A-79



SOFTWARE AVAILABLE FOR F-8, 8080, 6800, 8085, Z-80, 6502, KIM-1, 1802, 2650.
EPROM type is selected by a personality module which plugs into the front of the programmer. Power requirements are 115 VAC, 50/60 HZ at 15 watts. It is supplied with a 36 inch ribbon cable for,connecting to microcomputer. Requires 1½ I/O ports. Priced at $155 with one set of software. Personality modules are shown below.

Part No.		Programs	Price
PM-0	TMS	2708	$15.00
PM-1		2704, 2708	15.00
PM-2		2732	30.00
PM-3	TMS	2716	15.00
PM-4	TMS	2532	30.00
PM-5	TMS	2516, 2716, 2758	15.00

Optimal Technology, Inc.
Blue Wood 127, Earlysville, VA 22936
Phone (804) 973-5482

```
                 *
                 * PAGE ZERO DATA REFERENCES
                 *
                 LST    *     $0000  ADDRESS OF LAST ACCEPT COMMAND
                 FLG    *     $0002  CURRENT YES/NO FLAG
                 CHRS   *     $0003  ALLOW 40 BYTES OF INPUT
                 NAME   *     $002B  VARIABLE AREAS - 2 BYTES EACH
                 VARIBS *     $0053  VARIABLE AREAS - 2 BYTES EACH
                 IFLAG  *     $0087  SPECIAL INDICATOR FLAG AREA
                 HOLDY  *     $0088  HOLD AREA FOR Y VALUE
                 WORK   *     $0089  TEMP WORK VARIABLE
                 RESULT *     $008B  RESULT HOLD AREA FOR COMPUTATIONS
                 ANSX   *     $008D  HOLD AREA FOR ANSWER INDEX POINTER
                 SIGNIF *     $008E  SIGNIFICANCE INDICATOR
                 OPRATN *     $008F  LAST OPERATION IN COMPUTE STATEMENT
                 NUMDSP *     $0090  DISPLAY VARIABLE BUILD AREA
                 RETURN *     $0095  JUMP RETURN ADDRESS
                 CURAD  *     $0097  ADDRESS OF START OF CURRENT LINE

                 CR     *     $0D    CARRIAGE RETURN CODE
                 *
                 * EXTERNAL ADDRESS REFERENCES
                 *
                 CRLF   *     $834D  OUTPUT A CR AND LF
                 INCHR  *     $8A1B  INPUT ONE CHARACTER
                 OUTCHR *     $8A47  OUTPUT ONE CHARACTER

                        ORG   $0200
                 *
                 * START OF THE EDITOR PORTION
                 *
0200 A9 80       START  LDAIM $80    SET MODE TO EDIT FOR "PRT" ROUTINE
0202 85 87              STA   IFLAG
0204 20 83 04           JSR   SETBGN SET UP STARTING DATA AREA ADDRESS
                 *
                 * HERE IS THE START OF EACH NEW LINE
                 *
0207 A9 3E       ELINE  LDAIM $3E    OUTPUT A ">" PROMPT CHARACTER
0209 20 47 8A           JSR   OUTCHR
                 *
                 * HERE IS WHERE EACH INPUT CHARACTER IS OBTAINED
                 *
020C 20 1B 8A    EGET   JSR   INCHR
020F AA                 TAX          CHECK FOR NULLS AND IGNORE
0210 F0 FA              BEQ   EGET   SO THEY DON'T GET CONFUSED WITH EOL

0212 C9 5E              CMPIM $5E    IS IT AN UPARROW?
0214 F0 EA              BEQ   START  YES - START AT BEGINNING AGAIN

0216 C9 40              CMPIM $40    IS IT "AT" SYMBOL FOR EXECUTE REQUEST?
0218 F0 39              BEQ   EXEC   YES - GO START ON THAT

021A C9 08              CMPIM $08    IS IT A BACKSPACE?
021C D0 06              BNE   TRYDSP NO - GO CHECK FOR DISPLAY REQUEST

021E 88                 DEY          YES - BACK UP ONE CHARACTER
021F 10 EB              BPL   EGET   BUT CHECK FOR PAST START OF LINE
0221 C8                 INY          HE BACKED UP TOO FAR - DISALLOW
0222 10 E8              BPL   EGET   UNCONDITIONAL

0224 C9 2F       TRYDSP CMPIM $2F    IS IT "/" FOR DISPLAY LINE REQUEST?
0226 D0 05              BNE   TRYREP NO - CHECK FOR REPLACEMENT REQUEST
                 *
                 * DISPLAY TO THE NEXT CARRIAGE RETURN
                 *
0228 20 21 04           JSR   PRT    PRINT THE LINE
022B B0 DA              BCS   ELINE  UNCONDITIONAL

022D C9 25       TRYREP CMPIM $25    IS IT "%" REQUEST TO PAD A LINE?
022F D0 0E              BNE   CHAR   NO - MUST BE DATA CHARACTER
                 *
                 * PAD THE LINE FROM CURRENT LOC TO EOL WITH DELETE CHAR
                 *
0231 B1 97       PADLOP LDAIY CURAD  GET CURRENT CHARACTER
0233 F0 18              BEQ   SETNL  IF ZERO, WE ARE DONE
0235 A9 FF              LDAIM $FF    ELSE MAKE IT A DELETE CHAR
0237 91 97              STAIY CURAD
0239 C8                 INY          BUMP TO NEXT CHARACTER
023A 10 F5              BPL   PADLOP LOOP IF HAVEN'T DONE 128
023C 88                 DEY          LINE IS TOO LONG - BACK UP ONE
023D A9 0D              LDAIM CR     FORCE IN AN EOL HERE
                 *
                 * IT WASN'T AN EDIT CHARACTER - MUST BE DATA TO SAVE
                 *
```

```
023F C9 0D     CHAR   CMPIM CR      IS IT CARRIAGE RETURN AS EOL?
0241 D0 02            BNE   CHAR1   SKIP AHEAD IF NOT
0243 A9 00            LDAIM $00     ELSE CONVERT CR TO ZERO AS EOL
0245 91 97     CHAR1  STAIY CURAD   PUT IT AWAY
0247 F0 04            BEQ   SETNL   BRANCH IF YES
0249 C8               INY           ELSE BUMP TO SET UP FOR NEXT ONE
024A 10 C0            BPL   EGET    AND GO GET IT IF STILL ROOM ON LINE
024C 88               DEY           ELSE POINT BACK TO LAST CHAR & FALL THRU

024D 20 57 04  SETNL  JSR   LINEND DO CR/LF AND FIX UP CURAD
0250 B0 B5            BCS   ELINE  GO START A NEW LINE
               *
               * EXECUTION PORTION BEGINS HERE
               *
0252 20 4D 83  EXEC   JSR   CRLF    EXTRA BLANK LINE AFTER EDITOR

0255 20 83 04  RESTRT JSR   SETBGN HERE IF FROM J:*
0258 A2 33            LDXIM $33     ZERO VARIABLE ZREAS
025A A9 00            LDAIM $00
025C 85 96            STA   RETURN +01
025E 95 53     RESTR1 STAX  VARIBS
0260 CA               DEX
0261 10 FB            BPL   RESTR1

0263 B1 97     LSTART LDAIY CURAD  GET CHARACTER FROM THE LINE
0265 C9 2A            CMPIM $2A     CHECK FOR "*" LABEL MARKER
0267 D0 04            BNE   CHKCON IF NOT - GO CHECK FOR CONDITIONAL
0269 C8               INY           OTHERWISE SKIP PAST THE "*"
026A C8        SKPNXT INY           SKIP PAST THE NEXT CHARACTER
026B D0 F6            BNE   LSTART UNCONDITIONAL
               *
               * FLAG DEPENDENT PROCESSING HERE
               *
026D C9 59     CHKCON CMPIM $59     CHECK FOR "Y" REQUEST
026F F0 04            BEQ   TFLAG  BRANCH IF YES
0271 C9 4E            CMPIM $4E     IF NOT - CHECK FOR "N" REQUEST
0273 D0 09            BNE   STRTST BRANCH IF NEITHER
               *
               * SEE IF CONDITIONAL MATCHES FLAG
               *
0275 C5 02     TFLAG  CMP   FLG     SEE IF THEY MATCH
0277 F0 F1            BEQ   SKPNXT SKIP TO NEXT CHAR & EXECUTE LINE
               *
               * NO MATCH - SKIP THIS STATEMENT
               *
0279 20 5A 04  FWD    JSR   FWD1    USE THIS SUBROUTINE
027C B0 E5            BCS   LSTART UNCONDITIONAL

027E 85 87     STRTST STA   IFLAG  THIS WILL CLEAR HIGH BIT FOR EDITOR
0280 C8               INY           POINT TO THE ":" CHAR
0281 C8               INY           AND TO THE FOLLOWING CHARACTER
               *
               * ENTER NAME STATEMENT
               *
0282 C9 3F     XQUEST CMPIM $3F     IS IT "?" FOR ENTER NAME?
0284 D0 05            BNE   XA      BRANCH IF NOT
0286 38               SEC           TURN HIGH ORDER BIT ON TO INDICATE
0287 66 87            ROR   IFLAG   PROCESSING NAME COMMAND
0289 D0 0C            BNE   TAKEIN NOW USE THE ACCEPT LOGIC
               *
               * ACCEPT STATEMENT
               *
028B C9 41     XA     CMPIM $41     SEE IF HAVE ACCEPT STATEMENT
028D D0 34            BNE   XC      BRANCH IF NOT
028F A5 97            LDA   CURAD   SAVE ADDRESS OF THE "A" STATEMENT
0291 85 00            STA   LST     NOTE: WILL INCLUDE CONDITIONALS
0293 A5 98            LDA   CURAD   +01
0295 85 01            STA   LST     +01



0297 A9 3F     TAKEIN LDAIM $3F     DISPLAY "?" PROMPTING CHARACTER
0299 20 47 8A         JSR   OUTCHR

029C A2 27            LDXIM $27     CHRS GETS STORED BACKWARDS
029E 20 1B 8A  ACHR   JSR   INCHR   GET AN INPUT CHARACTER
02A1 C9 08            CMPIM $08     IS IT A BACKSPACE?
02A3 D0 03            BNE   ACHR1   BRANCH IF NOT
02A5 E8               INX           ELSE FORGET ABOUT LAST CHARACTER IN
02A6 D0 F6            BNE   ACHR    UNCONDITIONAL
02A8 C9 0D     ACHR1  CMPIM CR      WAS IT A CARRIAGE RETURN?
02AA D0 02            BNE   ACHR2   NO - SKIP AHEAD
02AC A9 00            LDAIM $00     YES - CONVERT CR TO END OF LINE
02AE 95 03     ACHR2  STAX  CHRS    AND SAVE IT FOR MATCH STATEMENT
02B0 24 87            BIT   IFLAG   SEE IF GETTING NAME FIELD
```

long. If Y is minus after it has been incremented, more than 128 characters have gone by since the start of the line.

The editor inserts an end of line at this point and continues on. If this occurs during line print or scan for end of line, it probably means that the PILOT program has gone off the end, so these routines branch to SETBGN to start at the beginning again. This does not prevent the PILOT program from looping while looking for an undefined label, but it does prevent printing some garbage.

The first character on a line is not necessarily useful for executing a PILOT statement. There might be a line feed or some other control character present there. The asterisk and the label are not used except as a destination for a USE or JUMP statement. If we do find one of these, we not only need to skip it, but we must also skip the next character, since that is the label. The routine SKPJNK takes care of skipping over everything but the asterisk, since the same routine is used by both normal command start and by the label search routine.

Once the program has searched out the first probable command character on the line, the next thing it has to do is look for a conditional flag. This will determine whether it must examine the rest of the line. A "Y" or an "N" is a conditional, and if the character of one of these lines, it is checked against the current value in FLG. If they do match, the program simply increments Y to point to the following character, and also starts again, but this time Y is pointing to the operation code following the conditional.

Most of the other operations execute in a similar manner. They look at the current character in A, do their processing if it is their turn, or branch to the next routine if it isn't theirs. There are some exceptions to this (naturally). The TEXT command is last because, if the character isn't a valid statement, the whole line must be printed anyway. One of the other exceptions is the processing for ENTER NAME (?:) and ACCEPT statements, which share much of the same code. Another is the code for JUMP and USER statements, which also share common code.

Logically, the only difference between the "?:" statement and the "A:" statement is that the "?:" inputs characters into both CHRS and into NAME, while the "A:" saves the starting address of the line for use in "J:A" (jump to last accept) processing. In fact, the processing of the ENTER NAME statement merely involves setting the high order bit of IFLAG on and skipping the save of the line address that the ACCEPT statement performs. The high order bit of IFLAG is normally turned off by storing the ASCII command character in it. The code for the ACCEPT statement checks the high order bit of IFLAG and stores the input character in NAME if the bit is on.

One thing to note is that data saved in NAME and CHRS are stored backwards, with the first input character in CHRS + 39,the second in CHRS + 38, etc. Since I have to initialize the X register anyway, I could initialize it with zero and count up, or with 39 and count down. If I am counting up, though, I need to do a compare to see if I have reached the maximum value. If I am counting down, the minus flag will automatically set when I reach the end.

The COMPUTE statement uses decimal arithmetic. Each variable is two bytes long, with the high order first. The high order decimal digit (bits 0-3 of the first byte) are used to indicate the sign. A value of 8 or 9 indicates a negative number, while anything else is considered positive. It works out to be tens' complement arithmetic. To illustrate, assume I want to calculate 1 minus 2, which everybody knows is −1. The actual result from the decimal subtract is $9999, much as it would be $FFFF in binary.

In order to display this as −1, we have to subtract $9999 from zero to get $0001. Using decimal arithmetic does have some disadvantages, particularly the fact that the range of numbers is −2000 to +7999 ($8000 to $7999) for two bytes instead of −32768 to +32767 for binary. Another disadvantage is that INC is not a decimal instruction.

The primary advantage of using decimal mode is the ease of translating from ASCII to internal and back. The ASCII characters zero through nine are $30 through $39 in hex. Multiplying by 10 in order to accept the next digit into a number is also very easy, since it only requires a four bit shift left. Converting to display merely means shifting each digit to the low order four bits. ANDing off the high order part, and ORing in $30.

The MATCH statement is the most complicated statement apart from COMPUTE. In theory, all that has to be done is compare the characters in CHRS against those in the MATCH statement line, and then set FLG to Y if they match, and to N if they don't. This works fine if they match. The problems come when they are different. Before the flag gets set to N, we have to determine why they did not match.

For one thing, it might be the end of the MATCH statement line. Since all the characters up to that point have matched, the program treats this condition as a complete match. PILOT uses the comma as a seperator in the match statement to indicate alternate possible matches, so if the mismatch character is a comma, it is treated as the end of line, and FLG is set to Y.

```
02B2 10 02           BPL    ACHR3  BRANCH IF NOT
02B4 95 2B           STAX   NAME   ELSE SAVE IN NAME FIELD ALSO
02B6 C9 00    ACHR3  CMPIM  $00    IS IT DONE YET?
02B8 F0 C3           BEQ    ADONE  BRANCH IF HE HAS SIGNALLED END
02BA CA              DEX           ELSE BUMP FOR NEXT INPUT
02BB 10 E1           BPL    ACHR   AND GO GET IT IF ROOM STILL LEFT
02BD 20 4D 83 ADONE  JSR    CRLF   DO CR/LF TO LET GUY KNOW
02C0 4C 79 02        JMP    F[illegible]D
              *
              * COMPUTE STATEMENT
              *
02C3 C9 43    XC     CMPIM  $43    IS IT A "C" FOR COMPUTE?
02C5 F0 03           BEQ    XC1    BRANCH IF IT IS
02C7 4C 56 03        JMP    XM     ELSE LONG JUMP TO TEST FOR M
02CA 20 94 04 XC1    JSR    GETIDX GET INDEX POINTER TO RESULT
02CD 86 8D           STX    A[illegible]X   SAVE IT FOR NOW
02CF A9 00           LDAIM  $00    CLEAR RESULT
02D1 85 8B           STA    RESULT
02D3 85 8C           STA    RESULT +01
02D5 C8              INY           POINT TO "="
02D6 A2 2B           LDXIM  $2B    SET 1ST OPERATION TO "+" FOR ADD
02D8 D0 4A           BNE    OPWRAP GO SAVE & SET UP WORK AREA
              *
              * LOOP FOR EACH NEW CHARACTER IN COMPUTE PROCESSING
              *
02DA C8       CMPLOP INY           BUMP TO NEXT CHARACTER
02DB B1 97           LDAIY  C[illegible]AD  GET A CHARACTER
02DD 30 20           BMI    ISOPR  MINUS IS DELETE/ALSO LAST "OPERATOR"
02DF C9 2F           CMPIM  $2F    IS IT "/" FOR AN OPERATION SPECIFIED?
02E1 90 1C           BCC    ISOPR  BRANCH IF YES
02E3 C9 3A           CMPIM  $3A    IF NOT - IS IT ":" FOR A NUMBER?
02E5 B0 12           BCS    NOTNMB BRANCH IF NOT - MUST BE A VARIABLE

02E7 29 0F           ANDIM  $0F    CONVERT NUMBER TO BINARY
02E9 6A              RORA          SPIN TO HIGH ORDER PART OF A
02EA 6A              RORA
02EB 6A              RORA
02EC 6A              RORA          LEAVE BIT 3 IN CARRY
02ED A2 04           LDXIM  $04    4 BITS TO ROLL INTO WORK
02EF 26 8A    BITROL ROL    WORK   +01 RIPPLE CARRY INTO WORK
02F1 26 89           ROL    WORK   FOR 16 BITS
02F3 0A              ASLA          PUT NEXT BIT INTO CARRY
02F4 CA              DEX           COUNT ONE JUST DONE
02F5 D0 F8           BNE    BITROL CONTINUE IF MORE TO GO
02F7 F0 E1           BEQ    CMPLOP ELSE GET NEXT CHARACTER (DIGITS)

02F9 20 9C 04 NOTNMB JSR    VTRANS TRANSFER VARIABLE TO WORK AREA
02FC 4C DA 02        JMP    CMPLOP GO GET NEXT CHARACTER (OPERATION?)
              *
              * GOT AN OPERATION - FIRST PERFORM PREVIOUS REQUEST
              *
02FF F8       ISOPR  SED           SET TO DECIMAL MODE
0300 AA              TAX           SAVE NEW OPERATION IN X FOR NOW
0301 A5 8F           LDA    OPRATN GET PREVIOUS OPERATION
0303 C9 2D           CMPIM  $2D    WAS IT A "-" FOR SUBTRACT?
0305 F0 10           BEQ    OPMNUS BRANCH IF YES
0307 18              CLC           ALL OTHERS ASSUME IT IS ADD
0308 A5 8A           LDA    WORK   +01
030A 65 8C           ADC    RESULT +01
030C 85 8C           STA    RESULT +01
030E A5 89           LDA    WORK
0310 65 8B           ADC    RESULT
0312 85 8B           STA    RESULT
0314 4C 24 03        JMP    OPWRAP GO WRAP UP THE OPERATION

0317 38       OPMNUS SEC           SUBTRACTION
0318 A5 8C           LDA    RESULT +01
031A E5 8A           SBC    WORK   +01
031C 85 8C           STA    RESULT +01
031E A5 8B           LDA    RESULT
0320 E5 89           SBC    WORK
0322 85 8B           STA    RESULT

0324 D8       OPWRAP CLD           GET OUT OF DECIMAL MODE
0325 86 8F           STX    OPRATN SAVE NEW OPERATION
0327 8A              TXA           DO TRANSFER TO CHECK FOR "00"/"FF"
0328 F0 0A           BEQ    CMPDON DONE IF IT WAS ZERO (EOL)
032A 30 08           BMI    CMPDON OR DELETE CHARACTERS (FROM FILLING)

032C A9 00           LDAIM  $00    ELSE CLEAR WORK AREA FOR NEXT ONE
032E 85 89           STA    WORK
0330 85 8A           STA    WORK   +01
0332 F0 A6           BEQ    CMPLOP AND GO DO NEXT CHARACTER
```

```
0334 A6 8D    CMPDON LDX   ANSX   GET INDEX TO RESULT
0336 10 13           BPL   TOVRIB PLUS IS NORMAL INDEX TO A VARIABLE

0338 A2 38           LDXIM $38    ELSE FUDGE INDEX FOR "FROM" RESULT
                                    USING "RESULT - VARIBS"
033A 20 9F 04        JSR   VTRANS +03 MOVE RESULT TO WORK AREA
033D 20 AB 04        JSR   CNVDSP +03 CONVERT IT TO DISPLAY FORM
0340 A2 04           LDXIM $04    TRANSFER DISPLAY TO ANSWER AREA
0342 B5 90    TALOOP LDAX  NUMDSP
0344 95 26           STAX  CHRS   +23 NOTE OFFSET TO PUT IT AT THE END
0346 CA              DEX
0347 10 F9           BPL   TALOOP
0349 30 08           BMI   XFWD   UNCONDITIONAL
034B A5 8C    TOVRIB LDA   RESULT +01 DESIRED VARIABLE
034D 95 54           STAX  VARIBS +01
034F A5 8B           LDA   RESULT
0351 95 53           STAX  VARIBS
0353 4C 79 02 XFWD   JMP   FWD    AND GO DO NEXT ONE
              *
              * PROCESS MATCH STATEMENT
              *
0356 C9 4D    XM     CMPIM $4D    IS IT "M" FOR MATCH?
0358 D0 4F           BNE   XU     BRANCH IF NOT
035A 88              DEY          BACK UP ONE FOR WHAT FOLLOWS
035B C8       MCHKX  INY          POINT TO MATCH CHARACTER
035C A2 27           LDXIM $27    START AT FIRST ACCEPTED CHARACTER
035E B1 97    MCHK   LDAIY CURAD  GET THE MATCH CHARACTER
0360 F0 08           BEQ   MXY    THEY HAVE MATCHED TO END OF "M:" STMT
0362 D5 03           CMPX  CHRS   CHECK FOR MATCH
0364 D0 08           BNE   MXNMCH BRANCH IF MATCH FAILED
0366 C8              INY          ELSE BUMP TO NEXT PAIR OF CHARACTERS
0367 CA              DEX
0368 10 F4           BPL   MCHK   AND GO CHECK IF STILL DATA LEFT
036A A2 59    MXY    LDXIM $59    BOTH EQUAL - SET FLAG TO "Y"
036C D0 37           BNE   MX     UNCONDITIONAL
036E C9 24    MXNMCH CMPIM $24    IS IT "$" FOR VARIABLE REQUEST?
0370 F0 13           BEQ   MNUMB  YES - MATCH TO NUMERIC VARIABLE
0372 C9 2C           CMPIM $2C    IS IT A COMMA GROUP SEPARATOR?
0374 F0 F4           BEQ   MXY    YES - MATCHED SO FAR - SET IT AS YES

0376 C8       MCOMMA INY          NO - SO NEED TO SKIP AHEAD TO COMMA
0377 B1 97           LDAIY CURAD
0379 F0 28           BEQ   MXSETN IF TO EOL, THERE IS NO MORE TO CHECK
037B C9 2C           CMPIM $2C    CHECK FOR A COMMA CHARACTER
037D F0 DC           BEQ   MCHKX  RESTART COMPARE AT NEXT MATCH CHARACTER
037F D0 F5           BNE   MCOMMA LOOP IN SEARCH OF A COMMA
0381 A4 88    MCOMX  LDY   HOLDY  RESET Y TO CURRENT LINE POINTER
0383 D0 F1           BNE   MCOMMA AND GO LOOK FOR NEXT COMMA

0385 C8       MNUMB  INY          VARIABLE - BUMP TO VARIABLE ID
0386 86 8D           STX   ANSX   SAVE CURRENT X FOR NOW
0388 20 A8 04        JSR   CNVDSP CONVERT VARIABLE TO DISPLAY FORM
038B A6 8D           LDX   ANSX   GET POINTER TO INPUT BACK
038D 84 88           STY   HOLDY  SAVE CURRENT "Y" POINTER
038F A0 04           LDYIM $04    HAVE TO SEARCH UP TO 5 BYTES

0391 B9 90 00 MXNOLP LDAY  NUMDSP GET ONE NUMERIC CHARACTER
0394 F0 08           BEQ   MXDIFF BRANCH IF END - MIGHT BE MATCH
0396 D5 03           CMPX  CHRS   ELSE CHECK AGAINST INPUT
0398 D0 E7           BNE   MCOMX  BRANCH IF NO MATCH
039A CA              DEX          ELSE CONTINUE MATCHING
039B 88              DEY
039C 10 F3           BPL   MXNOLP UNCONDITIONAL

039E A4 88    MXDIFF LDY   HOLDY  RESET Y TO CURRENT LINE POINTER
03A0 C8              INY          BUMP TO CHARACTER AFTER VARIABLE
03A1 D0 BB           BNE   MCHK   UNCONDITIONAL CONTINUE CHECKING

03A3 A2 4E    MXSETN LDXIM $4E    GET "N" - MATCH WAS UNSUCCESSFUL
03A5 86 02    MX     STX   FLG    STORE IT
03A7 D0 AA           BNE   XFWD   UNCONDITIONAL FOWRARD TO NEXT LINE
              *
              * PROCESS USE SUBROUTINE STATEMENT
              *
03A9 C9 55    XU     CMPIM $55    IS IT A "U" FOR USE SUBROUTINE?
03AB D0 11           BNE   XJ     BRANCH IF NOT
03AD B1 97           LDAIY CURAD  GET DESTINATION
03AF 48              PHA          SAVE THE LABEL CHARACTER
03B0 20 5A 04        JSR   FWD1   MOVE TO START OF NEXT LINE
03B3 A5 97           LDA   CURAD
03B5 85 95           STA   RETURN SAVE FOR RETURN ADDRESS
03B7 A5 98           LDA   CURAD  +01
03B9 85 96           STA   RETURN +01
03BB 68              PLA          GET DESTINATION BACK
```

There is also the possibility it might be caused by a request to match against the current value of a variable. To perform variable matching, the program calls CNVDSP which converts the variable to display format with leading zeros suppressed. It then matches the display format against the characters in CHRS. If the variable value matches, the program continues checking the rest of the MATCH statement.

If, even after all this, we still have a no-match condition, all is not lost yet. We have to scan forward in the MATCH statement, to look for a comma or the end of line. If we find the end of line, then FLG gets set to N. If we find a comma, the program starts the whole match process over again, from the character after the comma in the MATCH statement and from the beginning of CHRS. All this sounds confusing but, for example, the statement "M:YE,OK,SUR" will provide a Y indication for most affirmative responses such as YES or YES SIR or YEP or SURE WILL or OK.

As I mentioned earlier, the USE subroutine statement shares much of its code with the JUMP statement. The main difference is that the USE statement must save the address of the start of the *next* statement, while the JUMP statement doesn't need to. Note that the USE statement does not nest levels (sorry about that).

There are two reserved labels in PILOT. The first is the asterisk, which is used to completely restart the PILOT program (including zeroing the variables). The second reserved label is "A". This label indicates a JUMP (or USE) to the last ACCEPT statement. If the label in the statement is not one of the reserved labels, the program sets CURAD back to the start of the PILOT program via a call to SETBGN + 3 and starts the search for that label.

The STOP statement is trivial. It merely requires a jump back to the start of the editor.

Processing of the EXIT from subroutine statement is slightly more complex. It involves a check of the high order byte of the address contained in RETURN. If it is zero, then there was no USE statement executed to get there, and the program merely advances to the next line. The high order byte can never be zero, since all the lines are stored above $500. After restoring the return address to CURAD, the program resets the high order byte to zero. This means that the PILOT program can either "fall through" a subroutine, or use it in a normal fashion.

The REMARKS processing rivals that of the STOP statement for complexity. It merely involves advancing to the next statement. One final PILOT statement is the TYPE statement. It is also the default statement if none of the above sections processed it. If the statement is not a true TYPE statement, Y is backed up twice, so the whole line will be printed. Otherwise, the line is printed following the "T:".

The remainder of the program consists of subroutines used by various PILOT statements. The routine PRT prints the current line to the end. It uses the high order bit of IFLAG to see if the program is in editor mode. If it is, then all characters are printed, instead of being checked for a "$" to indicate a variable. After the line has been printed, a carriage return and line feed are output. It then falls through to FWD1.

The purpose of this routine is to advance to the end of the current line, and set up CURAD for the next line. Since it checks for end of line first, before incrementing Y, the fall through from PRT will immediately exit this routine, thus saving a branch in PRT.

FWD1, in turn, exits to a routine called SCURAD. This adds one to Y, and adds the result to CURAD as the start of the next line. Finally, this routine falls through to SKPJNK, which skips over any unwanted junk at the start of the line and executes the return.

With the exception of CNVDSP, the remaining routines are short and pretty much to the point. The VTRANS routine must transfer the high order byte of the variable last, so it sets the sign flag for CNVDSP. The format of the NUMDSP array is set up in the same "backward" manner used for CHRS and NAME, and it is the output of CNVDSP. If the variable is negative, a " – " is inserted as the first character.

The high order bit of SIGNIF is used to keep track of whether a non-zero digit has been encountered in the number being converted. If the bit is off and the current digit is zero, the index is not decremented, but the zero is stored anyway. If the bit is on, the digit gets stored regardless of its value. Any non-zero digit turns on the high order bit, just to make sure. An end of line zero is inserted after the last digit.

There are three SYM monitor routines used in this program. If you plan to bring Tiny PILOT up on another system you will have to change the addresses for these routines. They are all fairly standard, so most systems should have equivalents. INCHR gets one ASCII character from the terminal into the A register, without parity; OUTCHR outputs one ASCII character from A; and CRLF outputs a carriage return then a line feed. Tiny PILOT assumes that all registers are preserved by these routines. μ

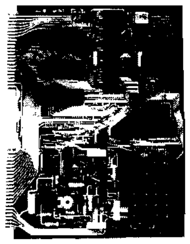
```
03BC D0 06           BNE   JDO    NO GO HANDLE AS JUMP STATEMENT
               *
               * PROCESS JUMP STATEMENT
               *
03BE C9 4A     XJ    CMPIM $4A    IS IT "J" FOR JUMP STATEMENT?
03C0 D0 2E           BNE   XS     BRANCH IF NOT
03C2 B1 97           LDAIY CURAD  GET DESTINATION

03C4 85 87     JDO   STA   IFLAG  SAVE LABEL CHARACTER
03C6 C9 2A           CMPIM $2A    HAVE "*" TO REQUEST RETURN TO BEGINNING?
03C8 F0 23           BEQ   IREST  BRANCH IF SO
03CA C9 41           CMPIM $41    SEE IF A LABELLED JUMP
03CC D0 0A           BNE   JF     IF NOT "A", IT'S A NORMAL JUMP


03CE A5 00           LDA   LST    ELSE SET TO START OF LAST ACCEPT
03D0 85 97           STA   CURAD
03D2 A5 01           LDA   LST    +01
03D4 85 98           STA   CURAD  +01
03D6 D0 43           BNE   ILNEXT UNCONDITIONAL

03D8 20 86 04  JF    JSR   SETBGN +03 AND GET BACK TO START OF PROGRAM


03DB B1 97     FNDMRK LDAIY CURAD GET FIRST CHARACTER
03DD C9 2A           CMPIM $2A    IS IT "*" FOR A MARKER?
03DF D0 07           BNE   FMNEXT NOPE - GO AHEAD TO NEXT LINE
03E1 C8              INY          ELSE BUMP TO MARKER CHARACTER
03E2 B1 97           LDAIY CURAD  GET LABEL
03E4 C5 87           CMP   IFLAG  SEE IF ITS THE ONE WE WANT
03E6 F0 33           BEQ   ILNEXT YES - GO EXECUTE IT
03E8 20 5A 04  FMNEXT JSR  FWD1   ELSE GO TO NEXT LINE
03EB B0 EE           BCS   FNDMRK AND CONTINUE LOOKING
03ED 4C 55 02  IREST JMP   RESTRT INDIRECT TO RESTRT
               *
               * STOP STATEMENT
               *
03F0 C9 53     XS    CMPIM $53    IS IT AN "S" FOR STOP STATEMENT?
03F2 D0 03           BNE   XE     BRANCH IF NOT
03F4 4C 00 02        JMP   START  ELSE RETURN TO EDITOR START
               *
               * EXIT FROM SUBROUTINE
               *
03F7 C9 45     XE    CMPIM $45    IS IT AN "E"
03F9 D0 10           BNE   XR     BRANCH IF NOT
03FB A5 96           LDA   RETURN +01 MOVE RETURN ADDRESS TO CURAD
03FD F0 10           BEQ   XXFWD  SKIP LINE IF NOT SET
03FF 85 98           STA   CURAD  +01

0401 A5 95           LDA   RETURN
0403 85 97           STA   CURAD
0405 A9 00           LDAIM $00    NOW SET TO NOT-USED AGAIN
0407 85 96           STA   RETURN +01
0409 F0 10           BEQ   ILNEXT UNCONDITIONAL
               *
               * REMARK STATEMENT
               *
040B C9 52     XR    CMPIM $52    IS IT AN "R"
040D D0 03           BNE   XT     BRANCH IF NOT - ELSE SKIP THE LINE
040F 4C 79 02  XXFWD JMP   FWD    CAN'T REACH THAT FAR ALONE
               *
               * TYPE STATEMENTS AND SYNTAX ERRORS
               *
0412 C9 54     XT    CMPIM $54    IS IT A VALID "T" STATEMENT
0414 F0 02           BEQ   TE     BRANCH IF SO
0416 88              DEY          ELSE BACK UP TO ORIGINAL START
0417 88              DEY
0418 20 21 04  TE    JSR   PRT    NOW PRINT THE LINE
041B 20 6E 04  ILNEXT JSR  SKPJNK CURAD IS SET - SKIP OVER LEADING JUNK
041E 4C 63 02        JMP   LSTART AND GO START ON THE LINE
               *
               * PRINT A LINE FROM CURRENT LOCATION TO
               * NEXT EOL AND THEN SET UP FOR NEXT LINE
               *
0421 B1 97     PRT   LDAIY CURAD  GET THE CURRENT CHARACTER
0423 F0 32           BEQ   LINEND BRANCH IF TO END OF LINE
0425 24 87           BIT   IFLAG  SEE IF IN EDITOR
0427 30 26           BMI   CHROUT IF SO, DON'T LOOK FOR "$"

0429 C9 24           CMPIM $24    IS IT A SPECIAL ONE ("$")
042B D0 22           BNE   CHROUT BRANCH IF NOT
042D C8              INY          ELSE BUMP TO NEXT ONE
042E B1 97           LDAIY CURAD  GET VARIABLE
0430 C9 3F           CMPIM $3F    IS IT REQUEST FOR NAME ("$")?
0432 F0 0F           BEQ   NAMEO  BRANCH IF YES
```

```
0434 20 A8 04         JSR    CNVDSP CONVERT VARIABLE TO DISPLAY
0437 A2 04            LDXIM $04    GOT 5 BYTES POSSIBLE

0439 B5 90     VBDISP LDAX   NUMDSP GET A CHARACTER
043B F0 15            BEQ    CHROUT +03 BRANCH IF TO END OF VARIABLE
043D 20 47 8A         JSR    OUTCHR ELSE OUTPUT IT
0440 CA               DEX           AND COUNT IT
0441 10 F6            BPL    VBDISP UNCONDITIONAL LOOP

0443 A2 27     NAMEO  LDXIM $27    REMEMBER - IT CAME IN BACKWARDS
0445 B5 2B            LDAX   NAME
0447 F0 09            BEQ    CHROUT +03 BRANCH IF TO END OF NAME
0449 20 47 8A         JSR    OUTCHR
044C CA               DEX
044D 10 F6            BPL    NAMEO  +02 UNCONDITIONAL

044F 20 47 8A  CHROUT JSR    OUTCHR
0452 C8               INY
0453 10 CC            BPL    PRT    LOOP IF NOT TOO MANY
0455 30 2C            BMI    SETBGN RESET TO BEGINNING IF PAST THE END
0457 20 4D 83  LINEND JSR    CRLF   OUTPUT A CR AND THE LINE FEED
               *
               * ENTER HERE TO SKIP A LINE WITHOUT PRINT
               * AND INITIALIZE FOR THE NEXT LINE
               *
045A B1 97     FWD1   LDAIY CURAD  GET A CHARACTER
045C F0 05            BEQ    SCURAD BRANCH IF END OF LINE
045E C8               INY           ELSE BUMP TO NEXT ONE
045F 10 F9            BPL    FWD1   LOOP IF NOT TOO MANY
0461 30 20            BMI    SETBGN RESET TO BEGINNING IF PAST THE END
               *
               * HERE FIXES UP CURAD TO POINT TO BEGINNING OF A LINE
               * CURAD SHOULD INDEX END OF LINE (WITH Y) ON ENTRY
               *
0463 C8        SCURAD INY           BUMP PAST THE CR
0464 98               TYA           MOVE COUNT TO A
0465 18               CLC           CLEAR CARRY FOR ADD
0466 65 97            ADC    CURAD  ADD TO LOW ORDER FIRST
0468 85 97            STA    CURAD  AND SAVE RESULT
046A 90 02            BCC    SKPJNK SKIP IF NO CARRY FORWARD
046C E6 98            INC    CURAD  +01 ELSE BUMP HIGH ORDER
               *
               * HERE TO SKIP PAST LEADING JUNK ON A LINE
               *
046E A0 FF     SKPJNK LDYIM $FF    SET UP Y THIS WAY
0470 C8        SJLOOP INY           INCREMENT TO NEXT CHARACTER
0471 24 87            BIT    IFLAG  SEE IF IN EDIT MODE
0473 30 0C            BMI    SJRTS  DON'T TRY SKIPPING JUNK IF SO
0475 B1 97            LDAIY CURAD  GET CHARACTER TO LOOK AT
0477 30 F7            BMI    SJLOOP IGNORE DELETE CHARACTER ALSO
0479 C9 2A            CMPIM $2A    LOOK FOR "*" LABEL MARKER
047B F0 04            BEQ    SJRTS  RETURN IF FOUND
047D C9 3F            CMPIM $3F    LOOK FOR POSSIBLE OPERATION CHARACTER
047F 90 EF            BCC    SJLOOP CONTINUE SKIPPING IF TOO LOW
0481 38        SJRTS  SEC           SET CARRY FOR BRANCHES AFTER RETURN
0482 60               RTS           BEFORE RETURN
               *
               * SET UP BEGINNING ADDRESS OF USER AREA
               *
0483 20 4D 83  SETBGN JSR    CRLF   START ON A NEW LINE
0486 A0 00            LDYIM $00    EVEN PAGE BOUNDARY
0488 84 97            STY    CURAD
048A 84 00            STY    LST    ALSO SET UP THIS GUY AS DEFAULT
048C A9 05            LDAIM $05
048E 85 98            STA    CURAD  +01
0490 85 01            STA    LST    +01
0492 D0 DA            BNE    SKPJNK UNCONDITIONAL
               *
               * COMPUTE INDEX FOR A VARIABLE
               *
0494 B1 97     GETIDX LDAIY CURAD  GET VARIABLE LETTER
0496 38               SEC
0497 E9 41            SBCIM $41    SUBTRACT "A" TO MAKE RELATIVE TO ZERO
0499 0A               ASLA          TIMES TWO BYTES PER VARIABLE
049A AA               TAX           MOVE TO INDEX REGISTER
049B 60               RTS           AND RETURN
               *
               * TRANSFER A VARIABLE'S DATA TO WORK AREA
               *
049C 20 94 04  VTRANS JSR    GETIDX GET INDEX POINTER FIRST
049F B5 54            LDAX   VARIBS +01 NOW MOVE TO WORK AREA
04A1 85 8A            STA    WORK   +01
04A3 B5 53            LDAX   VARIBS
04A5 85 89            STA    WORK
04A7 60               RTS
```

KIM-1

by Commodore

The Original 6502 System



20 mA Current Loop TTY Interface

Audio Cassette Interface

15 User I/O lines

2 Interval Timers

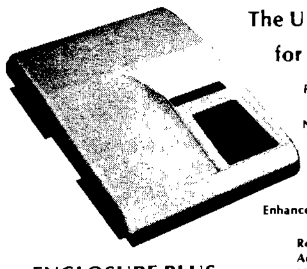
1K+ RAM

2K KIM Monitor ROM

Hex Keypad/LED Display

KIM-1: $180^00

ENCLOSURE PLUS™



The Ultimate Enclosure for the KIM-1

Protects Your KIM-1

Neat, Attractive, Professional

Full Access to the Expansion and Application Connectors

Enhances the LED Display with a Red Lense

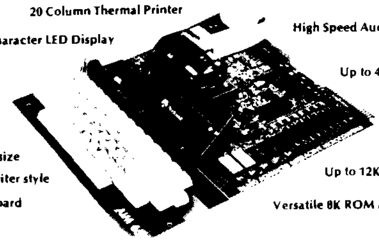
Room for the KIM-1 and One Additional Board such as MEMORY PLUS or VIDEO PLUS.

ENCLOSURE PLUS for KIM: $30^00

AIM 65

by Rockwell International

The Complete 6502 System



20 Column Thermal Printer

20 Character LED Display

High Speed Audio Cassette

Up to 4K RAM on board

Full size Typewriter style Keyboard

Up to 12K additional ROM

Versatile 8K ROM Monitor

AIM 65: $375^00 1K RAM - $420^00 4K RAM

AIM PLUS™

ENCLOSURE
WITH BUILT IN
POWER SUPPLY

SPECIFICATIONS:
INPUT: 110/220 VAC 50/60 Hz
OUTPUT: +5V @ 5A
+24V @ 1A
GROUNDED THREE-WIRE LINE CORD
ON/OFF SWITCH WITH PILOT LIGHT
Enclosure has room for the AIM and one additional board: MEMORY PLUS or VIDEO PLUS

AIM PLUS: $100^00   AIM and AIM PLUS: $475^00

617/256-3649

THE COMPUTERIST INC

PO Box 3
S Chelmsford, MA 01824

MACRO ASSEMBLER and TEXT EDITOR: for PET, APPLE II, SYM, KIM, other. Macros, conditional assembly, 27 commands, 22 pseudo-ops. Cassette and manual for $49.95 ($1.00 for info).
C.W. Moser
3239 Linda Drive
Winston-Salem, NC 27106

ZIPTAPE loads 8K BASIC in 15 seconds! Slower than a speeding disk? Sure, but it only costs $22.50 plus $1.00 S&H. $3.00 extra for software on KIM cassette. Described in MICRO #6. SASE for info.
Order from: Lew Edwards
1451 Hamilton Ave.
Trenton, NJ 08629

GRAFAX, the full screen graphics editor for the OSI 2P, 540 video graphics ROM, polled keyboard. Single keystroke commands make drawing a breeze. $10 + $1.00 postage for BASIC/assembler cassette and documentation:
Mark Bass
269 Jamison Drive
Frankfort, IL 60423

Software for the APPLE:
$25 buys SCROLLING WONDER
+ GIANT LETTER
+ HI-RES ALPHANUMERICS
on cassette, 16K. $25 buys:
MULTI-MESSAGE +
INTERLEAVED KALEIDOSCOPE +
MULTI-MESSAGE w/ ABSTRACT ART
on cassette, 32K. Send check or MO to: Connecticut Information Systems
218 Huntington Rd.
Bridgeport, CT 06608

MAILING LIST PROGRAM for APPLE: Maintain complete mailing list! Requires 1 drive and Applesoft II Data base can be added, changed, deleted, reformatted, searched (5 and sorted (5). Send $34.95 to:
SOFTWARE TECHNOLOGY for COMPUTERS
P.O. Box 428
Belmont, Ma. 02178

For only $33 for 12 issues you get CURSOR, the original cassette magazine for the Commodore PET. Each issue has a graphic "Front Cover" program, plus five excellent programs that are ready to Load and Run. The CURSOR NOTES newsletter with each issue provides useful information about the programs and help you with using your PET.
CURSOR
Box 550
Goleta, Ca. 93017

Southeastern Software NEWSLETTER for Apple II Owners. $10.00 for 10 issues per year. Back issues available at $1.00 each. Order from:
SOUTHEASTERN SOFTWARE
7270 CULPEPPER DRIVE
DEPT. MI
NEW ORLEANS, LA. 70126

```
                  *
                  * CONVERT A VARIABLE TO DISPLAY FORM
                  *
04A8 20 9C 04   CNVDSP JSR   VTRANS MOVE TO WORK AREA
04AB 10 17             BPL   ISPLUS BRANCH IF POSATIVE
04AD A9 2D             LDAIM $2D    ELSE PUT IN MINUS SIGN
04AF 85 94             STA   NUMDSP +04
04B1 F8                SED          SET DECIMAL MODE INDICATOR
04B2 38                SEC
04B3 A9 00             LDAIM $00    SUBTRACT FROM ZERO TO COMPLEMENT
04B5 E5 8A             SBC   WORK   +01
04B7 85 8A             STA   WORK   +01
04B9 A9 00             LDAIM $00
04BB E5 89             SBC   WORK
04BD 85 89             STA   WORK
04BF D8                CLD          CLEAR DECIMAL MODE
04C0 A2 03             LDXIM $03    ONLY 4 POSITIONS LEFT
04C2 D0 02             BNE   ISPL1  SKIP INDEX SET

04C4 A2 04      ISPLUS LDXIM $04    PLUS HAS FIVE POSITIONS AVAILABLE
04C6 18         ISPL1  CLC          TURN OFF SIGNIFICANCE INDICATOR
04C7 66 8E             ROR   SIGNIF
04C9 A5 89             LDA   WORK   GET FIRST DIGIT
04CB 20 E6 04          JSR   TOOUT  PUT TO OUTPUT AREA
04CE A5 8A             LDA   WORK   +01 SECOND DIGIT IS HIGH ORDER OF THIS
04D0 4A                LSRA         MOVE TO LOW ORDER
04D1 4A                LSRA
04D2 4A                LSRA
04D3 4A                LSRA
04D4 20 E6 04          JSR   TOOUT
04D7 A5 8A             LDA   WORK   +01 LOW ORDER IS THIRD DIGIT
04D9 20 E6 04          JSR   TOOUT
04DC 24 8E             BIT   SIGNIF SEE IF HAD ANY SIGNIFICANT CHARS
04DE 30 01             BMI   ISPL2  SKIP NEXT IF YES
04E0 CA                DEX          ELSE KEEP THE LAST ZERO THERE
04E1 A9 00      ISPL2  LDAIM $00    INSERT END OF LINE MARKER
04E3 95 90             STAX  NUMDSP
04E5 60                RTS          AND RETURN
                  *
                  * CONVERT CURRENT VALUE TO ASCII AND PUT TO OUTPUT AREA
                  *
04E6 29 0F      TOOUT  ANDIM $0F    KEEP ONLY LOW ORDER
04E8 09 30             ORAIM $30    MAKE IT ASCII
04EA 95 90             STAX  NUMDSP SAVE REGARDLESS
04EC 24 8E             BIT   SIGNIF SEE IF SIGNIFICANCE STARTED
04EE 30 05             BMI   SETSIG YES - ALL ARE IMPORTANT NOW
04F0 C9 30             CMPIM $30    ELSE SEE IF SHOULD START NOW
04F2 D0 01             BNE   SETSIG IMPORTANT IF NOT ZERO
04F4 60                RTS          ELSE RETURN

04F5 38         SETSIG SEC          SET SIGNIFICANCE BIT ON
04F6 66 8E             ROR   SIGNIF ALWAYS
04F8 CA                DEX          AND POINT TO NEXT AVAILABLE POSITION
04F9 60         PGMEND RTS          AND THEN RETURN
```

```
SYMBOL TABLE 2000 226A
ACHR   029E   ACHRQ  02A8   ACHRR  02AE   ACHRS  02B6
ADONE  02BD   ANSX   008D   BITROL 02EF   CHAR   023F
CHARQ  0245   CHKCOM 026D   CHROUT 044F   CHRS   0003
CMPDON 0334   CMPLOP 02DA   CNVDSP 04A8   CR     0D0D
CRLF   834D   CURAD  0097   EGET   020C   ELINE  0207
EXEC   0252   FLG    0002   FMNEXT 03E8   FNDMRK 03DB
FWD    0279   FWDQ   045A   GETIDX 0494   HOLDY  0088
IFLAG  0087   ILNEXT 041B   INCHR  8A1B   IREST  03ED
ISOPR  02FF   ISPLQ  04C6   ISPLR  04E1   ISPLUS 04C4
JDO    03C4   JF     03D8   LINEND 0457   LSTART 0263
LST    0000   MCHK   035E   MCHKX  035B   MCOMMA 0376
MCOMX  0381   MNUMB  0385   MX     03A5   MXDIFF 039E
MXNMCH 036E   MXNOLP 0391   MXSETN 03A3   MXY    036A
NAME   002B   NAMEO  0443   NOTNMB 02F9   NUMDSP 0090
OPMNUS 0317   OPRATN 008F   OPWRAP 0324   OUTCHR 8A47
PADLOP 0231   PGMEND 04F9   PRT    0421   RESTRQ 025E
RESTRT 0255   RESULT 008B   RETURN 0095   SCURAD 0463
SETBGN 0483   SETNL  024D   SETSIG 04F5   SIGNIF 008E
SJLOOP 0470   SJRTS  0481   SKPJNK 046E   SKPNXT 026A
START  0200   STRTS1 027E   TAKEIN 0297   TALOOP 0342
TE     0418   TFLAG  0275   TOOUT  04E6   TOVRIB 034B
TRYDSP 0224   TRYREF 022D   VARIBS 0053   VBDISP 0439
VTRANS 049C   WORK   0089   XA     028B   XC     02C3
XCQ    02CA   XE     03F7   XFWD   0353   XJ     03BE
XM     0356   XQUES1 0282   XR     040B   XS     03F0
XT     0412   XU     03A9   XXFWD  040F
```