SECOND FEATURE

THE STEPPER

Step and trace *through 6502 machine language programs with ease! You can set break points, monitor memory locations, and more, with the Stepper!*

The Stepper is a powerful step-and-trace debugger that executes an assembly language program line by line on the Apple II, II Plus, IIe or IIc. It has features that help assembly language programmers track down even the most elusive bugs. It lets you change the contents of the A, X, or Y Registers and the Stack Pointer. You can set break points and reset flags, or scan through several lines in succession.

USING STEPPER

The comments in the program listing (**Listing 1**) explain how the code works. Therefore, the following comments will be directed primarily to the Stepper's features and operation.

BRUNning Stepper enables the Monitor's Control-Y command. To begin stepping, enter from the Monitor the address at which you want to begin, enter Control-Y and press Return. For example,

```
*4000 Control-Y-Return
```

would start the trace at $4000 with all of the Stepper's flags initialized. If for some reason you exit the Stepper, you may start again where you left off by entering **Control-Y-Return**, without adding an address.

On the screen you'll see an inverse prompt line that shows the available options, the current register contents (including the stack pointer and status byte), and the instruction stored in the current address. When you want to proceed, press the Space bar. The displayed instruction will be executed, and another prompt line is displayed. This single-step mode can be continued until a break (BRK) is encountered or until the last RTS empties the stack. Stepper then returns you to the Monitor.

To begin stepping, enter from the Monitor the address at which you want to begin, enter a Control-Y and press Return.

Options

Your options are as follows:

1. To alter the contents of the A, X or Y Registers or the stack pointer, press the A, X, Y or S key. The current contents will be displayed in inverse below the regular display. You may then enter any 8-bit hex value desired.
2. To alter one of the bits in the status byte, press the N, V, D, I, Z or C key. This will toggle the current value and display the new value just below it. For instance, if the Carry bit is zero, pressing C will change it to a one. Lower-case b and e will toggle the B bit and the unused bit, respectively, in the status register. However, the BREAK bit is always set when Stepper is running and the unused bit is always set by the 6502.
3. Pressing Q will exit the Stepper. You may reenter at the current address with the current values by typing **Control-Y-Return**. If you alter a zero-page value at $55 or below, you must enter with an address parameter to allow the Stepper to pick up the current contents of zero page. (See the Zero Page section below.)
4. Pressing T sets/resets the Stack Display flag. When the flag is set, the contents of up to 11 stack locations just below the current stack pointer are displayed after each step, on top of the prompt line.
5. Pressing P sets/resets the Printer flag. When set, output is directed to a printer. The printer format is somewhat different from the screen display. To increase speed, no prompt line is included, and all printing is done in 80 columns instead of 40. If your printer is in a slot other than slot 1, see Modifications to accommodate another slot.

6. Pressing J sets/resets the JSR flag. When set, Stepper will execute all JSR's without stepping through them. Monitor routines, DOS routines, and routines known to be bug-free may be executed without time-consuming single-steps. Stepper will execute the subroutine and return, displaying the next instruction after the JSR.
7. Pressing R cancels the pause after each step. The Stepper traces continuously through your routine until encountering a BRK, a final JSR, or a break point. (See number 8 below.) Pressing any key returns to single-step mode.
8. K and Control-K (denoted in the listings by [K]) are the break-point controls. K allows you to enter a break point. When you are in run mode (R) and an instruction at an address listed among the break points is encountered, the Stepper halts and returns to single-step mode. The break point is not cleared and may be used again. Control-K erases the entire break-point table. Up to five break points may be in effect at one time.
9. E and Control-E ([E]) are the memory display controls. E allows you to enter an address. After each step, Stepper displays this address and its contents above the prompt line. Control-E clears the memory display table. (M would make more sense, but Control-M (alias, Return) could easily be pressed accidentally.)

THE ZERO PAGE

To allow an operation that is as independent as possible, Stepper saves the portion of the zero page it uses and swaps it in and out as needed. On entry into the Stepper when an address parameter is specified, the values in addresses $00-$55 are saved for use by the source program. Entry without an address parameter uses the last-stored zero page. Be careful if any values from $00 to $55 are changed; reenter the Stepper with an address parameter specified. To allow stepping through I/O routines, the four addresses DOS uses to store its input and print data are also swapped by the Stepper.

Stepper cannot step through ProDOS MLI calls, because ProDOS enables alternate banks of memory in the upper 16K, and disconnects RAM in the process. Since Stepper is in RAM, it cannot function properly.

ENTERING THE PROGRAM

If you have an assembler, you may enter the assembly code in **Listing 1** and assemble it to create a working program. If you don't have an assembler, you may enter the hexadecimal code directly from the Monitor. If you are running ProDOS, change HIMEM to protect the code by entering:

```
HIMEM:35584
```

at the Applesoft prompt before you enter the program. Save the program with:

LISTING 1: STEPPER

```
                  ********************************
                  * STEPPER                      *
                  * BY WAYNE EASTWOOD            *
                  * COPYRIGHT (C) 1987           *
                  * BY MICROSPARC, INC.          *
                  * CONCORD, MA  01742           *
                  ********************************
                  *MERLIN PRO ASSEMBLER

                  * My Ø-page use
                  CH       =     $24        ;cursor position locations
                  CV       =     $25
                  LMNEM    =     $2C        ;used in opcode disassembly
                  RMNEM    =     $2D
                  LENGTH   =     $2F        ;instruction length - 1
                  PC       =     $3A        ;PC used by PRADR1
                  A1       =     $3C        ;loc. of 16-bit # on entry (####[Y]

                  * Constant definitions

                  CR       =     $8D        ;C/R
                  SPACE    =     $AØ        ;ASCII space

                  * System and DOS calls

                  STACK    =     $Ø1ØØ      ;65Ø2 stack
                  HOOKUP   =     $Ø3EA      ;DOS hook up
                  YVECTOR  =     $Ø3F8      ;jump to here on ctrl-Y
                  DOSINOUT =     $AA59      ;48K RAM DOS I/O values storage
                  KYBOARD  =     $CØØØ      ;key input
                  STROBE   =     $CØ1Ø      ;key strobe
                  PRODOS   =     $BFØØ
                  VECTIN   =     $BE32
                  VECTOUT  =     $BE3Ø

                  * Monitor calls

                  INSDS2   =     $F88E      ;look up opcode info
                  PRADR1   =     $F91Ø      ;print 16-bit address (mid-routine)
                  PRBLNX   =     $F948      ;print 3 blanks
                  MNEML    =     $F9CØ      ;mnemonic table
                  MNEMH    =     $FAØØ
                  SETTXT   =     $FB39      ;txt mode & full window
                  VTAB     =     $FC22      ;cursor VTAB
                  RDKEY    =     $FDØC      ;read key
                  CROUT    =     $FD8E      ;output C/R
                  PRYX3    =     $FD99      ;print 16-bit at X,Y
                  PRBYTE   =     $FDDA      ;print hex in A
                  COUT     =     $FDED      ;char out
                  SETINV   =     $FE8Ø      ;set screen inverse
                  SETNORM  =     $FE84      ; "      "   normal
                  OUTPORT  =     $FE95      ;reset I/O
                  MON      =     $FF69      ;reenter monitor


                  * Assemble one in high memory and one in low memory
                  *   to allow tracing programs at various locations

                           ORG   $8FØØ


                  * Enters here on BRUN or monitor G
                  *   to set up [Y] vector & initialize
                  *   program parameters

8FØØ: A9 4C       ENTER    LDA   #$4C       ;'JMP' is stored for ctrl-Y
8FØ2: 8D F8 Ø3             STA   YVECTOR
8FØ5: A9 1D                LDA   #<STEP     ;point to STEPPER address
8FØ7: 8D F9 Ø3             STA   YVECTOR+1
8FØA: A9 8F                LDA   #>STEP
8FØC: 8D FA Ø3             STA   YVECTOR+2
8FØF: A2 ØA                LDX   #1Ø        ;clear all information registers
8F11: A9 ØØ       ENTER1Ø  LDA   #Ø
8F13: 9D 28 91    ENTER2Ø  STA   aSAVE,X
8F16: CA                   DEX
8F17: 1Ø FA                BPL   ENTER2Ø
8F19: CE 2B 91             DEC   sSAVE      ;all new stack
8F1C: 6Ø                   RTS


                  * Enter here: ADRS[Y] = step at new PCNT, save current Ø-page
```

```
BSAVE STEPPER,A$8FØØ,L$686
```

If you are using Key Perfect, you should make changes to the object file before you run Key Perfect. BLOAD the file, enter the Monitor with CALL −151 and perform the following instructions:

```
9024:00 00 00
9123:ØØ
9124<9123.9201M
```

Then save the modified program using the BSAVE instruction above.

For help with entering *Nibble* programs, see "A Welcome to New *Nibble* Readers" at the beginning of this issue.

ASSEMBLY

Assemble the Stepper wherever you like. I keep two versions, one at $8F00 and one at $805, so that one will be ready wherever my source program happens to be. I've written the code as compactly as possible while still providing user-friendly operation.

The use of $2C warrants some explanation. In an assembly source listing it looks like the following:

```
          LDA   #Ø
          HEX   2C
TOGGLE    LDA   #1
```

etc.

but it assembles as:

```
          LDA   #Ø
          BIT   $Ø1A9
```

etc.

Entering from the top loads the Accumulator with zero. The program then falls through the BIT operation with the Accumulator unscathed. Yet the program may enter at TOGGLE instead to load the Accumulator with a one, saving a byte of code.

At times I've used the ADC operation with the Carry bit set to avoid a CLC, thus saving a byte. In such cases, the result is greater by one than might be expected.

MODIFICATIONS

If your printer interface is not in slot 1, change the AND #1 instructions in **lines 482, 565, 625, 692 and 696** to AND *#n*, where *n* is the slot number of your printer interface. Reassemble the file.

If you are making the changes from the Monitor, BLOAD the object file, enter the Monitor with CALL −151 and make the following changes:

```
92EØ:n
938E:n
9418:n
94A9:n
94B4:n
```

BSAVE the resulting file according to the instructions in the Entering the Program section.

LISTING 1: STEPPER *(continued)*

```
                  82   *                        [Y] = step at current PCNT, use prev Ø-page
                  83
8F1D: 8A          84   STEP      TXA                  ;if X=Ø, then no ADRS
8F1E: FØ 12       85             BEQ   STEPLOOP       ;go directly to main routine
8F2Ø: 2Ø 6B 95    86             JSR   ZSAVEØ         ;new ADRS, so save current Ø-page
8F23: A2 Ø5       87             LDX   #5             ;clear registers & run flag
8F25: 2Ø 11 8F    88             JSR   ENTER1Ø
8F28: A5 3C       89             LDA   A1             ;put new address into PCNT
8F2A: 8D 83 95    9Ø             STA   PCNT           ;A1 set from monitor routine
8F2D: A5 3D       91             LDA   A1+1           ;  when reading [Y]
8F2F: 8D 84 95    92             STA   PCNT+1         ;fall into main tracing routine
                  93
                  94
                  95   * Main tracing routine:
                  96   *     1) Display memory and stack per user direction
                  97   *     2) Display current registers & next instruction
                  98   *     3) Seek user input for step/run, register change, etc.
                  99   *     4) Perform next instruction
                  1ØØ  *     5) Return to 1) above
                  1Ø1
                  1Ø2  *     1) Display memory and stack per user direction
                  1Ø3
8F32: 2Ø 39 FB    1Ø4  STEPLOOP  JSR   SETTXT         ;always start with new screen
8F35: 2Ø 8E FD    1Ø5            JSR   CROUT          ;insure new line
8F38: 2Ø FØ FD    1Ø6            JSR   COUT+3         ;2 for screen for readability
8F3B: AE 31 91    1Ø7            LDX   mCOUNT         ;display any memory?
8F3E: FØ Ø3       1Ø8            BEQ   SL1Ø           ;no
8F4Ø: 2Ø 2C 94    1Ø9            JSR   DSPMEM         ;display memory
8F43: AE 3Ø 91    11Ø  SL1Ø      LDX   tFLAG          ;display stack?
8F46: FØ 24       111            BEQ   SL4Ø           ;no
8F48: AE 2B 91    112            LDX   sSAVE          ;get current stack location
8F4B: E8          113            INX                  ;at bottom?
8F4C: FØ 1E       114            BEQ   SL4Ø           ;yes, none to see
8F4E: AØ ØA       115            LDY   #1Ø            ;11 max
8F5Ø: BD ØØ Ø1    116  SL2Ø      LDA   STACK,X        ;display values separated w/spaces
8F53: 2Ø DA FD    117            JSR   PRBYTE
8F56: A9 AØ       118            LDA   #SPACE
8F58: 2Ø ED FD    119            JSR   COUT
8F5B: E8          12Ø            INX                  ;more stack?
8F5C: FØ Ø3       121            BEQ   SL3Ø           ;no
8F5E: 88          122            DEY                  ;more list?
8F5F: 1Ø EF       123            BPL   SL2Ø           ;yes
8F61: 2Ø 48 F9    124  SL3Ø      JSR   PRBLNX         ;flush with 3 blanks to tab printer
8F64: 88          125            DEY                  ;  (not noticed on screen display)
8F65: 1Ø FA       126            BPL   SL3Ø
8F67: A9 8D       127            LDA   #CR            ;new line for screen but not printer
8F69: 2Ø FØ FD    128            JSR   COUT+3
                  129
                  13Ø  *     2) Display current registers & next instruction
                  131
8F6C: 2Ø Ø3 92    132  SL4Ø      JSR   DSPREG         ;display current registers
8F6F: AE 83 95    133            LDX   PCNT           ;print PCNT as hex followed by '-'
8F72: AC 84 95    134            LDY   PCNT+1
8F75: 86 3A       135            STX   PC             ;save for disassembly by PRADR1
8F77: 84 3B       136            STY   PC+1
8F79: 2Ø 99 FD    137            JSR   PRYX3          ;print
8F7C: 2Ø 24 95    138            JSR   ZRESTORE       ;restore source Ø-page
8F7F: A2 Ø2       139            LDX   #2             ;get next instruction (3-bytes max)
8F81: 2Ø 82 95    14Ø  SL5Ø      JSR   PRGGET         ;  & store for program reference
8F84: 9D 25 91    141            STA   NXTINST,X      ;  at NXTINST, NXTINST+1, NXTINST+2
8F87: CA          142            DEX
8F88: 1Ø F7       143            BPL   SL5Ø
8F8A: 2Ø 51 95    144            JSR   ZSAVE          ;return my own Ø-page
8F8D: A9 EA       145            LDA   #$EA           ;clear execution area with
8F8F: 8D 25 9Ø    146            STA   EXECUTE+1      ;  'NOP' ($EA) instructions
8F92: 8D 26 9Ø    147            STA   EXECUTE+2
8F95: AD 25 91    148            LDA   NXTINST        ;load opcode of instr. & use monitor
8F98: 2Ø 8E F8    149            JSR   INSDS2         ;  to find format, len, & index
8F9B: A2 Ø3       15Ø            LDX   #3             ;three chars in mnemonic
8F9D: A8          151            TAY                  ;mnemonic table index
8F9E: B9 CØ F9    152            LDA   MNEML,Y        ;fetch 3-char mnemonic
8FA1: 85 2C       153            STA   LMNEM          ;  (packed in 2 bytes)
8FA3: B9 ØØ FA    154            LDA   MNEMH,Y        ;(this is an imitation of the Mon.
8FA6: 85 2D       155            STA   RMNEM          ;  rout. except we disass. the
8FA8: A9 ØØ       156  SL6Ø      LDA   #Ø             ;  opcode only w/o displaying hex
8FAA: AØ Ø5       157            LDY   #5             ;  code as well and mod. the spacing)
8FAC: Ø6 2D       158  SL7Ø      ASL   RMNEM          ;shift 5 bits of chr into A
8FAE: 26 2C       159            ROL   LMNEM
8FBØ: 2A          16Ø            ROL
8FB1: 88          161            DEY
8FB2: DØ F8       162            BNE   SL7Ø
8FB4: 69 BF       163            ADC   #"?            ;carry clear, add '?' offset
8FB6: 2Ø ED FD    164            JSR   COUT           ;print it
8FB9: C9 BF       165            CMP   #"?            ;exit on unknown opcode '???'
8FBB: DØ ØF       166            BNE   SL8Ø
8FBD: 2Ø 8Ø FE    167            JSR   SETINV         ;display code
8FCØ: AD 25 91    168            LDA   NXTINST
8FC3: 2Ø DA FD    169            JSR   PRBYTE
8FC6: 2Ø 84 FE    17Ø            JSR   SETNORM
8FC9: 4C 4Ø 9Ø    171            JMP   INTBRK1Ø       ;exit
8FCC: CA          172  SL8Ø      DEX
8FCD: DØ D9       173            BNE   SL6Ø
8FCF: A9 AØ       174            LDA   #SPACE         ;one blank
8FD1: 2Ø ED FD    175            JSR   COUT
8FD4: A4 2F       176            LDY   LENGTH         ;prepare to enter norm. Mon. stream
8FD6: A2 Ø6       177            LDX   #6             ;  to print the address field
```

```
8FD8: 20 10 F9            JSR  PRADR1     ;enter

                *   3) Seek user input for step/run, register change, etc.

8FDB: 20 62 92            JSR  RESPOND    ;read user input

                *   4) Perform next instruction

8FDE: AE 2B 91            LDX  sSAVE      ;ready to go, put on source stack
8FE1: 9A                  TXS
8FE2: AD 25 91            LDA  NXTINST    ;get opcode

                * Normally, instr. fall through to PERFORM.  However, we
                *   must interpret BRK, RTI, RTS, JSR, JMP, and JMP (XXXX)
                *   to maintain control of program.  Relative braching is
                *   controlled at PERFORM.

8FE5: F0 56               BEQ  INTBRK     ;'BRK'
8FE7: C9 40               CMP  #$40       ;'RTI'?
8FE9: F0 58               BEQ  INTRTI
8FEB: C9 60               CMP  #$60       ;'RTS'?
8FED: F0 5D               BEQ  INTRTS
8FEF: C9 20               CMP  #$20       ;'JSR'?
8FF1: D0 05               BNE  SL90       ;no
8FF3: AE 2E 91            LDX  jFLAG      ;yes, run entire routine or step?
8FF6: F0 68               BEQ  INTJSR     ;continue single step
8FF8: C9 4C     SL90      CMP  #$4C       ;'JMP'?
8FFA: F0 76               BEQ  INTJMP
8FFC: C9 6C               CMP  #$6C       ;indirect JMP?
8FFE: F0 7D               BEQ  INTJMPI
9000: A4 2F     PERFORM   LDY  LENGTH     ;get length of instruction
9002: 29 1F               AND  #%00011111 ;is it a relative branch opcode?
9004: 49 14               EOR  #%00010100
9006: C9 04               CMP  #%00000100 ;if equal, branch--have made
9008: F0 03               BEQ  SL110      ;  the offset branch into our prgm
900A: B9 25 91  SL100     LDA  NXTINST,Y  ;move all necess., others 'NOP'
900D: 99 24 90  SL110     STA  EXECUTE,Y
9010: 88                  DEY
9011: 10 F7               BPL  SL100
9013: 20 24 95            JSR  ZRESTORE   ;put on source 0-page
9016: AD 2C 91            LDA  pSAVE      ;restore all registers
9019: 48                  PHA
901A: AD 28 91            LDA  aSAVE
901D: AE 29 91            LDX  xSAVE
9020: AC 2A 91            LDY  ySAVE
9023: 28                  PLP

9024: 00 00 00  EXECUTE   DS   3          ;execute the instruction

9027: 4C B1 90            JMP  NOBRNCH    ;if no branch, continue


                * Branch if taken falls here always.  PCNT is adjusted
                *   then we return to STEPLOOP for next instruction

902A: 20 51 95            JSR  ZSAVE      ;restore my 0-page
902D: D8                  CLD             ;restore correct conditions
902E: 18                  CLC
902F: AD 26 91            LDA  NXTINST+1  ;forward or backward branch?
9032: 10 03               BPL  XQT10      ;forward
9034: CE 84 95            DEC  PCNT+1     ;backward, adjust prgm counter
9037: 20 A5 90  XQT10     JSR  ADDPCNT    ;adjust value of prgm counter
903A: 4C C7 90            JMP  UPDATE     ;get new prgm counter & step again


                * Opcode interpretation routines

903D: 20 03 92  INTBRK    JSR  DSPREG     ;display registers
9040: 4C F9 92  INTBRK10  JMP  EXIT       ;exit program

9043: BA        INTRTI    TSX             ;simulate 'RTI'
9044: E0 FD               CPX  #$FD       ;enough on stack?
9046: B0 F8               BCS  INTBRK10   ;no
9048: 68                  PLA             ;pull processor status
9049: 8D 2C 91            STA  pSAVE      ;  & continue
904C: BA        INTRTS    TSX             ;simulate 'RTS'
904D: E0 FE               CPX  #$FE       ;enough on stack?
904F: B0 EF               BCS  INTBRK10   ;no
9051: 68                  PLA             ;pull off return address
9052: 8D 83 95            STA  PCNT       ;put on prgm counter
9055: 68                  PLA
9056: 8D 84 95            STA  PCNT+1
9059: BA                  TSX             ;save stack
905A: 8E 2B 91            STX  sSAVE
905D: 4C C7 90            JMP  UPDATE     ;next prgm counter & step

9060: AD 83 95  INTJSR    LDA  PCNT       ;increment prgm counter by 2
9063: 69 01               ADC  #1         ;  imitating normal 6502 operation
9065: A8                  TAY             ;since carry was set on entry we save
9066: AD 84 95            LDA  PCNT+1     ;  SOME CODE BY ADDING 1 plus carry
9069: 69 00               ADC  #0         ;  instead of CLC then adding 2
906B: 48                  PHA             ;and push on stack
906C: 98                  TYA
906D: 48                  PHA
906E: BA                  TSX             ;save & continue as 'JMP'
906F: 8E 2B 91            STX  sSAVE
9072: AD 26 91  INTJMP    LDA  NXTINST+1  ;find new PC by looking at
9075: 48                  PHA             ;  opcode address field
9076: AD 27 91            LDA  NXTINST+2
9079: 48                  PHA
907A: 4C 9A 90            JMP  INTJMP10
907D: AD 26 91  INTJMPI   LDA  NXTINST+1  ;find indirect address by finding
9080: 8D 83 95            STA  PCNT       ;  the contents of the address
9083: AD 27 91            LDA  NXTINST+2  ;  specified in the address field
9086: 8D 84 95            STA  PCNT+1
9089: 20 24 95            JSR  ZRESTORE   ;put back ZP in case adrs is there
908C: A2 00               LDX  #0         ;find contents of address
908E: 20 82 95            JSR  PRGGET
9091: 48                  PHA
9092: E8                  INX
9093: 20 82 95            JSR  PRGGET
9096: 48                  PHA
9097: 20 51 95            JSR  ZSAVE      ;swap 0-pages again
909A: 68        INTJMP10  PLA             ;install new address
909B: 8D 84 95            STA  PCNT+1
909E: 68                  PLA
909F: 8D 83 95            STA  PCNT
90A2: 4C 32 8F            JMP  STEPLOOP   ;continue


                * Add PCNT to A , carry clear or set prior to entry

90A5: 6D 83 95  ADDPCNT   ADC  PCNT       ;if carry set, result 1 greater
90A8: 8D 83 95            STA  PCNT
90AB: 90 03               BCC  ADDP10
90AD: EE 84 95            INC  PCNT+1
90B0: 60        ADDP10    RTS


                * All non-interpreted instr. that do not branch come here

90B1: 8D 28 91  NOBRNCH   STA  aSAVE      ;operation completed, save stuff
90B4: 8E 29 91            STX  xSAVE
90B7: 8C 2A 91            STY  ySAVE
90BA: 08                  PHP
90BB: 68                  PLA
90BC: 8D 2C 91            STA  pSAVE
90BF: BA                  TSX
90C0: 8E 2B 91            STX  sSAVE
90C3: 20 51 95            JSR  ZSAVE      ;return my 0-page
90C6: D8                  CLD             ;will cause many problems if set!
90C7: 38        UPDATE    SEC             ;add LENGTH + 1 (SEC) to PC
90C8: A5 2F               LDA  LENGTH
90CA: 20 A5 90            JSR  ADDPCNT    ;add them
90CD: 4C 32 8F            JMP  STEPLOOP   ;continue


                * Possible user responses

90D0: C1 D8 D9  REGNAM    ASC  "AXYS"
90D3: D3
90D4: CE D6 E5  PROCNAM   HEX  CED6E5E2C4C9DAC3 ;NVebDICZ (lower-case not used)
90D7: E2 C4 C9 DA C3
90DC: D2 CA D0  PSUDNAM   ASC  "RJPTEK"
90DF: D4 C5 CB
90E2: 85 8B               HEX  858B       ;ctrl-E, ctrl-K
90E4: 20 03 1A  PROMPT    INV  " CZIDB.VN  S  Y  X  A   ]K[K ]E[E R JPTQ'
90E7: 09 04 02 2E 16 0E 20 20
90EF: 13 20 20 19 20 20 18 20
90F7: 20 01 20 20 20 1D 0B 1B
90FF: 0B 20 1D 05 1B 05 20 12
9107: 20 0A 10 14 11

                * Cursor tab for user responses

910C: 13 16 19  REGTAB    HEX  1316191C
910F: 1C
9110: 1F 20 21  PROCTAB   HEX  1F2021222324252б
9113: 22 23 24 25 26
9118: 03 02 01  PSDTAB    HEX  030201

                * Masks for pSAVE

911B: 80 40 20  ORMASK    HEX  8040201008040201
911E: 10 08 04 02 01

9123: 00        INDEX     DS   1          ;temporary index for user entry
9124: 00        DFLAG     DS   1          ;multi-purpose flag
9125: 00 00 00  NXTINST   DS   3          ;holds next instruction
9128: 00        aSAVE     DS   1          ;register save
9129: 00        xSAVE     DS   1
912A: 00        ySAVE     DS   1
912B: 00        sSAVE     DS   1
912C: 00        pSAVE     DS   1
912D: 00        rFLAG     DS   1          ;run flag
912E: 00        jFLAG     DS   1          ;JSR flag
912F: 00        pFLAG     DS   1          ;printer flag
9130: 00        tFLAG     DS   1          ;stack flag
9131: 00        mCOUNT    DS   1          ;number of memory loc. (mCOUNT+1)/2
9132: 00        bCOUNT    DS   1          ; "   "  brk. points (bCOUNT+1)/2

9133: 00 00 00  MEMAREA   DS   2*10-1     ;space for 10 memory locations
9136: 00 00 00 00 00 00 00 00
913E: 00 00 00 00 00 00 00 00
9146: 00 00 00  BRKAREA   DS   2*5-1      ;space for 5 breakpoints
```

```
9149: ØØ ØØ ØØ ØØ ØØ ØØ

914F: ØØ ØØ ØØ  PRGZ     DS    $56         ;source program Ø-page storage
9152: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
915A: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
9162: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
916A: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
9172: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
917A: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
9182: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
918A: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
9192: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
919A: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
91A2: ØØ ØØ ØØ
91A5: ØØ ØØ ØØ  MYZ      DS    $56         ;my Ø-page storage
91A8: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
91BØ: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
91B8: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
91CØ: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
91C8: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
91DØ: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
91D8: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
91EØ: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
91E8: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
91FØ: ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
91F8: ØØ ØØ ØØ
91FB: ØØ ØØ ØØ  PRGDOS   DS    4           ;program DOS print parameters
91FE: ØØ
91FF: ØØ ØØ ØØ  MYDOS    DS    4           ;my DOS print parameters
92Ø2: ØØ


                • Display current registers

92Ø3: A2 27     DSPREG   LDX   #39         ;prompt for screen only
92Ø5: BD E4 9Ø  DSPR1Ø   LDA   PROMPT,X
92Ø8: 2Ø FØ FD           JSR   COUT+3
92ØB: CA                 DEX
92ØC: 1Ø F7              BPL   DSPR1Ø
92ØE: A9 BE              LDA   #">
921Ø: 2Ø FØ FD           JSR   COUT+3
9213: A2 Ø2              LDX   #2          ;examine flags
9215: BD 2E 91  DSPR2Ø   LDA   jFLAG,X
9218: FØ Ø3              BEQ   DSPR21
921A: 2Ø 8Ø FE           JSR   SETINV      ;display in inverse if set
921D: BD DD 9Ø  DSPR21   LDA   PSUDNAM+1,X
922Ø: 2Ø FØ FD           JSR   COUT+3      ;display flag letter
9223: 2Ø 84 FE           JSR   SETNORM
9226: CA                 DEX
9227: 1Ø EC              BPL   DSPR2Ø
9229: A9 13              LDA   #19         ;tab right on screen
922B: 85 24              STA   CH
922D: A2 ØØ              LDX   #Ø
922F: BD 28 91  DSPR3Ø   LDA   aSAVE,X     ;get value of registers
9232: 2Ø DA FD           JSR   PRBYTE      ;display
9235: A9 AØ              LDA   #SPACE
9237: 2Ø ED FD           JSR   COUT
923A: E8                 INX
923B: EØ Ø4              CPX   #4          ;done?
923D: 9Ø FØ              BCC   DSPR3Ø      ;no
923F: AD 2C 91           LDA   pSAVE       ;now display processor status
9242: A8                 TAY
9243: A9 8Ø              LDA   #%1ØØØØØØØ ;mask, interested in bits here
9245: 85 3C              STA   A1          ;temporary storage
9247: 98        DSPR4Ø   TYA               ;restore pSAVE value
9248: 25 3C              AND   A1          ;and w/ current mask
924A: FØ Ø3              BEQ   DSPR5Ø      ;Ø value
924C: A9 B1              LDA   #"1         ;on, display '1'
924E: 2C                 HEX   2C          ;'BIT XXXX' -- falls through
924F: A9 BØ     DSPR5Ø   LDA   #"Ø         ;off, display 'Ø'
9251: 2Ø ED FD           JSR   COUT
9254: 46 3C              LSR   A1          ;shift mask -->
9256: DØ EF              BNE   DSPR4Ø      ;do for all bits
9258: A9 8D              LDA   #CR         ;for screen
925A: 2Ø FØ FD           JSR   COUT+3
925D: A9 BE              LDA   #">         ;next line (mark for printer)
925F: 4C ED FD           JMP   COUT


                • Stop loop and get user input

9262: AD 2D 91  RESPOND  LDA   rFLAG       ;in run mode?
9265: FØ 12              BEQ   RS1Ø        ;no
9267: 2Ø Ø1 95           JSR   BRKCHK      ;yes, check for breakpoint
926A: BØ Ø8              BCS   RSØ5        ;found one so stop
926C: AD ØØ CØ           LDA   KYBOARD     ;in run mode, any key stops
926F: 1Ø 53              BPL   RS7Ø        ;else exit w/out checking further
9271: 8D 1Ø CØ           STA   STROBE      ;clear
9274: A9 ØØ     RSØ5     LDA   #Ø          ;no more run
9276: 8D 2D 91           STA   rFLAG
9279: A9 17     RS1Ø     LDA   #23         ;position cursor
927B: 85 25              STA   CV
927D: A9 11              LDA   #17
927F: 85 24              STA   CH
9281: 2Ø 22 FC           JSR   VTAB
9284: 2Ø ØC FD           JSR   RDKEY       ;get response from user
9287: C9 AØ              CMP   #SPACE      ;next step?
```

```
9289: FØ 39              BEQ   RS7Ø        ;yes, exit
928B: C9 D1              CMP   #"Q         ;quit?
928D: DØ Ø3              BNE   RS2Ø
928F: 4C F7 92           JMP   EXITØ       ;exit one level into JSR's

                • Check if a register response A,X,Y, or S

9292: AØ Ø3     RS2Ø     LDY   #3
9294: D9 DØ 9Ø  RS3Ø     CMP   REGNAM,Y
9297: FØ 26              BEQ   RS6ØR       ;yes
9299: 88                 DEY
929A: 1Ø F8              BPL   RS3Ø

                • Check if one of the processor status bits N,V,D,I,Z, or C

929C: AØ Ø7              LDY   #7
929E: D9 D4 9Ø  RS4Ø     CMP   PROCNAM,Y
92A1: FØ 16              BEQ   RS6ØP       ;yes
92A3: 88                 DEY
92A4: 1Ø F8              BPL   RS4Ø

                • Check if one of the pseudo ops R,E,[E],K,[K],T,P, or J

92A6: AØ Ø7              LDY   #7
92A8: D9 DC 9Ø  RS5Ø     CMP   PSUDNAM,Y
92AB: FØ Ø5              BEQ   RS6ØS       ;yes
92AD: 88                 DEY
92AE: 1Ø F8              BPL   RS5Ø
92BØ: 3Ø C7              BMI   RS1Ø        ;no correct response

                • Handle input: carry set if to proceed to next instruction

92B2: 2Ø 59 93  RS6ØS    JSR   DOPSUEDO    ;handle pseudo ops
92B5: 9Ø C2              BCC   RS1Ø        ;back for more (not 'R' or step)
92B7: BØ ØB              BCS   RS7Ø        ;continue to next step
92B9: 2Ø 1D 93  RS6ØP    JSR   DOPROC      ;processor bits
92BC: 4C 79 92           JMP   RS1Ø        ;always back for more
92BF: 2Ø C5 92  RS6ØR    JSR   DOREG       ;register bytes
92C2: 9Ø B5              BCC   RS1Ø        ;no step
92C4: 6Ø        RS7Ø     RTS

92C5: 8C 23 91  DOREG    STY   INDEX       ;alter register, Y tells which one
92C8: 2Ø 8Ø FE  R1Ø      JSR   SETINV      ;for emphasis
92CB: 2Ø 94 93           JSR   ONSCREEN    ;insure screen only
92CE: AC 23 91           LDY   INDEX       ;restore index
92D1: B9 ØC 91           LDA   REGTAB,Y    ;get proper tab for display
92D4: 85 24              STA   CH
92D6: B9 28 91           LDA   aSAVE,Y     ;display current value
92D9: 2Ø DA FD           JSR   PRBYTE
92DC: AD 2F 91           LDA   pFLAG       ;return to specified output device
92DF: 29 Ø1              AND   #1          ;1 or Ø
92E1: 2Ø 96 93           JSR   DFLTOUT     ;restore current output
92E4: C6 24              DEC   CH          ;put cursor on top of image
92E6: 2Ø 84 FE           JSR   SETNORM
92E9: 2Ø ØC FD           JSR   RDKEY       ;change it as desired
92EC: C9 AØ              CMP   #SPACE      ;step?
92EE: DØ Ø1              BNE   R3Ø         ;no
92FØ: 6Ø        R2Ø      RTS               ;carry set if space
92F1: C9 D1     R3Ø      CMP   #"Q         ;quit?
92F3: DØ ØA              BNE   R4Ø         ;no
92F5: 68                 PLA               ;exit 2 deep in JSR's
92F6: 68                 PLA
92F7: 68        EXITØ    PLA
92F8: 68                 PLA
92F9: 2Ø 24 95  EXIT     JSR   ZRESTORE    ;restore Ø-page
92FC: 4C 69 FF           JMP   MON         ;exit to monitor

92FF: 49 BØ     R4Ø      EOR   #"Ø         ;determines if hex digit
93Ø1: C9 ØA              CMP   #$ØA        ;digits Ø-9 result in value Ø-9 so
93Ø3: 9Ø Ø6              BCC   R5Ø         ;  carry clear means yes
93Ø5: 69 88              ADC   #$88        ;digits $A-$F to $71-$76, add
93Ø7: C9 FA              CMP   #$FA        ;  $89 (w/carry) to push to $FA-$FF
93Ø9: 9Ø E5              BCC   R2Ø         ;no, exit for more input
93ØB: AØ Ø3     R5Ø      LDY   #3          ;lower nibble of A holds $xØ-$xF
93ØD: ØA                 ASL               ;move low nibble to high
93ØE: ØA                 ASL
93ØF: ØA                 ASL
931Ø: ØA                 ASL
9311: AE 23 91           LDX   INDEX       ;where does this go?
9314: ØA        R6Ø      ASL               ;move into low nibble of storage
9315: 3E 28 91           ROL   aSAVE,X     ;  & move low nibble to high
9318: 88                 DEY
9319: 1Ø F9              BPL   R6Ø
931B: 3Ø AB              BMI   R1Ø         ;always

931D: 8C 23 91  DOPROC   STY   INDEX       ;toggle stat bit, Y points to bit
932Ø: 2Ø 8Ø FE  P1Ø      JSR   SETINV
9323: AC 23 91           LDY   INDEX       ;position cursor
9326: B9 1Ø 91           LDA   PROCTAB,Y
9329: 85 24              STA   CH
932B: AD 2C 91           LDA   pSAVE       ;load entire status & mask
932E: 39 1B 91           AND   ORMASK,Y    ;  for desired bit
9331: FØ Ø3              BEQ   P2Ø         ;it's a Ø, convert to 1
9333: A9 BØ              LDA   #"Ø         ;it's a 1, convert to Ø
9335: 2C                 HEX   2C          ;'BIT XXXX' falls through 3 bytes
9336: A9 B1     P2Ø      LDA   #"1
9338: 2Ø FØ FD           JSR   COUT+3      ;display on screen only
```

```
933B: 20 84 FE           JSR   SETNORM
933E: AC 23 91           LDY   INDEX
9341: C9 30              CMP   #"0-$80     ;update status bit (MSB
9343: F0 08              BEQ   P30         ;  cleared due to SETINV)
9345: B9 1B 91           LDA   ORMASK,Y    ;change 0 to 1
9348: 0D 2C 91           ORA   pSAVE
934B: D0 08              BNE   P40         ;always
934D: B9 1B 91 P30       LDA   ORMASK,Y
9350: 49 FF              EOR   #$FF        ;convert for ANDing
9352: 2D 2C 91           AND   pSAVE       ;clear a bit
9355: 8D 2C 91 P40       STA   pSAVE
9358: 60                 RTS

9359: 98       DOPSUEDO  TYA               ;handle all other codes
935A: D0 05              BNE   PS1         ;not run flag
935C: EE 2D 91           INC   rFLAG       ;'R', so set flag
935F: 38                 SEC               ;run on exit
9360: 60                 RTS
9361: C9 04    PS1       CMP   #4          ;J, P, or T flags?
9363: B0 4D              BCS   PSD1        ;no, some sort of display option
9365: AA                 TAX               ;index for correct flag
9366: A9 FF              LDA   #$FF        ;toggle flag
9368: 5D 2D 91           EOR   rFLAG,X     ;X=1, 2, or 3
936B: 9D 2D 91           STA   rFLAG,X
936E: F0 03              BEQ   PSF1        ;now update display, 0=off
9370: 20 80 FE           JSR   SETINV      ;on shown in inverse
9373: C6 25    PSF1      DEC   CV          ;move cursor
9375: 20 22 FC           JSR   VTAB
9378: BD 17 91           LDA   PSDTAB-1,X  ;get cursor tab (X is 1 too large)
937B: 85 24              STA   CH
937D: BD DC 90           LDA   PSUDNAM,X   ;get display letter
9380: 20 F0 FD           JSR   COUT+3      ;screen only
9383: 20 84 FE           JSR   SETNORM
9386: E0 02              CPX   #2          ;printer toggle requested?
9388: D0 26              BNE   PSF4        ;no
938A: AD 2F 91           LDA   pFLAG       ;yes, get current value
938D: 29 01              AND   #1          ;make a 1 or 0
938F: D0 05              BNE   DFLTOUT     ;turn printer on
9391: 20 8E FD           JSR   CROUT       ;rtn to screen, clr any prtr buff
9394: A9 00    ONSCREEN  LDA   #0          ;PR#0
9396: 20 95 FE DFLTOUT   JSR   OUTPORT     ;set up hooks
9399: AD 00 BF           LDA   PRODOS
939C: C9 4C              CMP   #$4C
939E: D0 0D              BNE   HOOKDOS
93A0: A5 36              LDA   $36
93A2: 8D 30 BE           STA   VECTOUT
93A5: A5 37              LDA   $37
93A7: 8D 31 BE           STA   VECTOUT+1
93AA: 4C B0 93           JMP   PSF4
93AD: 20 EA 03 HOOKDOS   JSR   HOOKUP      ;tell DOS
93B0: 18       PSF4      CLC               ;return for more user input
93B1: 60                 RTS

93B2: C9 07    PSD1      CMP   #7          ;handle display options, ctrl-K?
93B4: D0 07              BNE   PSD2        ;no
93B6: A9 00              LDA   #0          ;clear all breakpoints
93B8: 8D 32 91           STA   bCOUNT
93BB: F0 F3              BEQ   PSF4        ;always
93BD: C9 06    PSD2      CMP   #6          ;clear memory (ctrl-E)?
93BF: D0 07              BNE   PSD3        ;no
93C1: A9 00              LDA   #0          ;clear all
93C3: 8D 31 91           STA   mCOUNT
93C6: F0 E8              BEQ   PSF4        ;always
93C8: C9 05    PSD3      CMP   #5          ;set breakpoints (K)?
93CA: F0 03              BEQ   BSET        ;yes
93CC: 4C 23 94           JMP   INPMEM      ;display addresses & add to list
93CF: 20 94 93 BSET      JSR   ONSCREEN    ;set up for screen only
93D2: 20 8E FD           JSR   CROUT
93D5: AE 32 91           LDX   bCOUNT      ;bCOUNT=# of entries*2+1
93D8: F0 16              BEQ   BS20        ;no entries/no display
93DA: CA                 DEX
93DB: BD 46 91 BS10      LDA   BRKAREA,X   ;stored low/high
93DE: 20 DA FD           JSR   PRBYTE      ;display
93E1: CA                 DEX
93E2: BD 46 91           LDA   BRKAREA,X
93E5: 20 DA FD           JSR   PRBYTE
93E8: A9 A0              LDA   #SPACE      ;separate
93EA: 20 ED FD           JSR   COUT
93ED: CA                 DEX
93EE: 10 EB              BPL   BS10
93F0: 20 8E FD BS20      JSR   CROUT       ;done with display
93F3: AE 32 91           LDX   bCOUNT      ;full?
93F6: E0 0A              CPX   #10
93F8: B0 1A              BCS   BS30        ;yes, can't add any
93FA: 20 B8 94           JSR   GETDIG      ;get address
93FD: AE 32 91           LDX   bCOUNT
9400: AD 24 91           LDA   DFLAG       ;any entry?
9403: F0 0F              BEQ   BS30        ;no
9405: A5 3C              LDA   A1          ;put new address in memory
9407: 9D 46 91           STA   BRKAREA,X
940A: E8                 INX
940B: A5 3D              LDA   A1+1
940D: 9D 46 91           STA   BRKAREA,X
9410: E8                 INX
9411: 8E 32 91           STX   bCOUNT
9414: AD 2F 91 BS30      LDA   pFLAG       ;restore output device
9417: 29 01              AND   #1
9419: 20 96 93           JSR   DFLTOUT
941C: 68                 PLA               ;pull off JSR's
941D: 68                 PLA
941E: 68                 PLA
941F: 68                 PLA
9420: 4C 32 8F           JMP   STEPLOOP    ;redo current line


               * Display routines: INPMEM = display then add
               *                   DSPMEM = display only

9423: 20 94 93 INPMEM    JSR   ONSCREEN    ;set up for screen only
9426: 20 8E FD           JSR   CROUT
9429: A9 00              LDA   #0          ;put a 0 as input flag
942B: 2C                 HEX   2C          ;'BIT XXXX' 3-byte fall through
942C: A9 BD    DSPMEM    LDA   #"=         ;store and use as display flag
942E: 8D 24 91           STA   DFLAG
9431: AE 31 91           LDX   mCOUNT      ;mCOUNT=# of entries*2+1
9434: F0 55              BEQ   DM50        ;no entries/no display
9436: CA                 DEX
9437: A9 09              LDA   #9          ;10*8 across for printer
9439: 8D 23 91           STA   INDEX       ;makes printer 80 column before C/R
943C: BD 33 91 DM10      LDA   MEMAREA,X   ;stored low/high
943F: 8D 5C 94           STA   DMXX+2      ;for contents display
9442: 20 DA FD           JSR   PRBYTE
9445: CA                 DEX
9446: BD 33 91           LDA   MEMAREA,X
9449: 8D 5B 94           STA   DMXX+1
944C: 20 DA FD           JSR   PRBYTE
944F: AD 24 91           LDA   DFLAG       ;0=address only
9452: F0 11              BEQ   DM20        ;input req. (do not disp contents)
9454: 20 ED FD           JSR   COUT        ;print flag ('=')
9457: 20 24 95           JSR   ZRESTORE    ;pull in proper 0-page
945A: AD FF FF DMXX      LDA   $FFFF       ;read memory contents (avoid
945D: 48                 PHA               ;  0-page conflict)
945E: 20 51 95           JSR   ZSAVE       ;give me back my 0-page
9461: 68                 PLA               ;restore memory contents
9462: 20 DA FD           JSR   PRBYTE      ;print it
9465: CE 23 91 DM20      DEC   INDEX       ;keep up with line len for printer
9468: 10 0A              BPL   DM30
946A: 20 8E FD           JSR   CROUT       ;new line for printer
946D: A9 09              LDA   #9          ;new count for line length
946F: 8D 23 91           STA   INDEX
9472: D0 05              BNE   DM40        ;always
9474: A9 A0    DM30      LDA   #SPACE      ;separate entries w/space
9476: 20 ED FD           JSR   COUT
9479: CA       DM40      DEX               ;next entry
947A: 10 C0              BPL   DM10
947C: 20 8E FD           JSR   CROUT       ;done
947F: AD 24 91           LDA   DFLAG       ;input required?
9482: D0 2C              BNE   DM70        ;no
9484: AE 31 91           LDX   mCOUNT      ;full?
9487: E0 14              CPX   #20
9489: B0 1A              BCS   DM60        ;yes
948B: 20 B8 94 DM50      JSR   GETDIG      ;get new address, reuse DFLAG
948E: AE 31 91           LDX   mCOUNT
9491: AD 24 91           LDA   DFLAG       ;any entry?
9494: F0 0F              BEQ   DM60        ;no
9496: A5 3C              LDA   A1          ;put new address in memory
9498: 9D 33 91           STA   MEMAREA,X
949B: A5 3D              LDA   A1+1
949D: E8                 INX
949E: 9D 33 91           STA   MEMAREA,X
94A1: E8                 INX
94A2: 8E 31 91           STX   mCOUNT
94A5: AD 2F 91 DM60      LDA   pFLAG       ;restore output device
94A8: 29 01              AND   #1
94AA: 20 96 93           JSR   DFLTOUT
94AD: 4C 32 8F           JMP   STEPLOOP    ;redo current line
94B0: AD 2F 91 DM70      LDA   pFLAG       ;restore output device
94B3: 29 01              AND   #1
94B5: 4C 96 93           JMP   DFLTOUT     ;& return to tracing stream


               * Get 16-bit digit

94B8: A9 00    GETDIG    LDA   #0          ;clear workspace
94BA: 85 3C              STA   A1
94BC: 85 3D              STA   A1+1
94BE: 8D 24 91           STA   DFLAG       ;set only on a valid entry
94C1: A9 17    GD10      LDA   #23         ;position cursor
94C3: 85 25              STA   CV
94C5: A9 11              LDA   #17
94C7: 85 24              STA   CH
94C9: 20 22 FC           JSR   VTAB
94CC: 20 80 FE           JSR   SETINV      ;for emphasis
94CF: A5 3D              LDA   A1+1        ;print current value
94D1: 20 DA FD           JSR   PRBYTE      ;to scrn only (due to calling rout)
94D4: A5 3C              LDA   A1
94D6: 20 DA FD           JSR   PRBYTE
94D9: C6 24              DEC   CH          ;put cursor on image
94DB: 20 84 FE           JSR   SETNORM
94DE: 20 0C FD           JSR   RDKEY       ;input
94E1: 49 B0              EOR   #"0         ;hex digit? (see R40 above)
94E3: C9 0A              CMP   #$0A
94E5: 90 06              BCC   GD20
```

```
94E7: 69 88               ADC     #$88
94E9: C9 FA               CMP     #$FA
94EB: 9Ø 13               BCC     GD4Ø          ;no
94ED: EE 24 91  GD2Ø      INC     DFLAG         ;set for valid entry (if <> 256!)
94FØ: A2 Ø3               LDX     #3
94F2: ØA                  ASL                   ;shift to high nibble
94F3: ØA                  ASL
94F4: ØA                  ASL
94F5: ØA                  ASL
94F6: ØA        GD3Ø      ASL
94F7: 26 3C               ROL     A1
94F9: 26 3D               ROL     A1+1          ;move carry into storage
94FB: CA                  DEX
94FC: 1Ø F8               BPL     GD3Ø
94FE: 3Ø C1               BMI     GD1Ø          ;more digits
95ØØ: 6Ø        GD4Ø      RTS


                * Check current PCNT against breakpoints

95Ø1: AE 32 91  BRKCHK    LDX     bCOUNT        ;any to check?
95Ø4: FØ 15               BEQ     BC3Ø          ;no
95Ø6: CA                  DEX
95Ø7: BD 46 91  BC1Ø      LDA     BRKAREA,X     ;stored low/high
95ØA: CA                  DEX
95ØB: CD 84 95            CMP     PCNT+1        ;current location
95ØE: DØ Ø8               BNE     BC2Ø          ;no match
951Ø: BD 46 91            LDA     BRKAREA,X
9513: CD 83 95            CMP     PCNT
9516: FØ Ø5               BEQ     BC4Ø          ;match!
9518: CA        BC2Ø      DEX                   ;keep looking
9519: 1Ø EC               BPL     BC1Ø
951B: 18        BC3Ø      CLC                   ;no match
951C: 6Ø                  RTS
951D: A9 87     BC4Ø      LDA     #$87          ;bell
951F: 2Ø ED FD            JSR     COUT
9522: 38                  SEC
9523: 6Ø                  RTS


                * Zero-page movers: I only use $55 and below
                *   (Also DOS I/O storage so I/O stuff can be traced)


9524: AØ 55     ZRESTORE LDY      #$55          ;save local, restore source
9526: B9 ØØ ØØ  ZR1       LDA     Ø,Y
9529: 99 A5 91            STA     MYZ,Y
952C: 88                  DEY
952D: 1Ø F7               BPL     ZR1
952F: AØ Ø3               LDY     #3            ;same for DOS memory
9531: B9 59 AA  ZR2       LDA     DOSINOUT,Y
9534: 99 FF 91            STA     MYDOS,Y
9537: 88                  DEY
9538: 1Ø F7               BPL     ZR2
953A: AØ 55               LDY     #$55
953C: B9 4F 91  ZR3       LDA     PRGZ,Y
953F: 99 ØØ ØØ            STA     Ø,Y
9542: 88                  DEY
9543: 1Ø F7               BPL     ZR3
9545: AØ Ø3               LDY     #3
9547: B9 FB 91  ZR4       LDA     PRGDOS,Y
954A: 99 59 AA            STA     DOSINOUT,Y
954D: 88                  DEY
954E: 1Ø F7               BPL     ZR4
955Ø: 6Ø                  RTS

9551: 2Ø 6B 95  ZSAVE     JSR     ZSAVEØ        ;save source, restore local
9554: AØ 55               LDY     #$55
9556: B9 A5 91  ZS3       LDA     MYZ,Y
9559: 99 ØØ ØØ            STA     Ø,Y
955C: 88                  DEY
955D: 1Ø F7               BPL     ZS3
955F: AØ Ø3               LDY     #3
9561: B9 FF 91  ZS4       LDA     MYDOS,Y
9564: 99 59 AA            STA     DOSINOUT,Y
9567: 88                  DEY
9568: 1Ø F7               BPL     ZS4
956A: 6Ø                  RTS

956B: AØ 55     ZSAVEØ    LDY     #$55          ;save source
956D: B9 ØØ ØØ  ZS1       LDA     Ø,Y
957Ø: 99 4F 91            STA     PRGZ,Y
9573: 88                  DEY
9574: DØ F7               BNE     ZS1
9576: AØ Ø3               LDY     #3
9578: B9 59 AA  ZS2       LDA     DOSINOUT,Y
957B: 99 FB 91            STA     PRGDOS,Y
957E: 88                  DEY
957F: 1Ø F7               BPL     ZS2
9581: 6Ø                  RTS


                * Program counter and opcode reader

9582: BD FF FF  PRGGET    LDA     $FFFF,X       ;avoid Ø page conflict
9585: 6Ø                  RTS

                PCNT      =       PRGGET+1      ;simulated program counter
```

END OF LISTING 1

KEY PERFECT 5.Ø
RUN ON
STEPPER

CODE-5.Ø	ADDR# - ADDR#	CODE-4.Ø
73BCF4DE	8FØØ - 8F4F	264B
23BFØ25F	8F5Ø - 8F9F	26DE
5B8431DD	8FAØ - 8FEF	2B45
CF55E927	8FFØ - 9Ø3F	23A4
8C5B12EØ	9Ø4Ø - 9Ø8F	27C1
ØØ722474	9Ø9Ø - 9ØDF	2A36
8869FF3E	9ØEØ - 912F	2AF9
5678BE35	913Ø - 917F	ØØ
5678BE35	918Ø - 91CF	ØØ
Ø36A32FB	91DØ - 921F	ØE4E
5F41BD9Ø	922Ø - 926F	2957
33D9BD61	927Ø - 92BF	299D
E389FDB4	92CØ - 93ØF	24C8
A1921B2B	931Ø - 935F	2AFE
EA9926F6	936Ø - 93AF	256E
C59FACA3	93BØ - 93FF	2A2B
ØØ68F924	94ØØ - 944F	27E2
A6E23A44	945Ø - 949F	2925
8CDBDBE4	94AØ - 94EF	2569
8DA29921	94FØ - 953F	225A
B95B6853	954Ø - 9585	2265
57ADFA97 = PROGRAM TOTAL =		Ø686