Monitor Listings

Your Trainer is controlled by U12, the "read only memory" (ROM). The following is a listing of the program stored in this IC.

Tables at the end of the listing show labels used in the program, keyboard and display addresses, segment codes for characters displayed by the program, and addresses in RAM that are reserved for use by the monitor program.

```
FC00                        ORG     $FC00

                    **      RESET - CLEAR BREAKPOINT TABLE AND INITIALIZE STACK

FC00  8E 00 EB      RESET   LDS     #2*NBR+BKTBL-1
FC03  BD FD 8D              JSR     OUTSTO
FC06  4E 67 3E              FCC     HEXC,LTRP,LTRU,0,LTRU,LTRP+$80
FC0C  CE 00 CB              LDX     #USRSTK
FC0F  DF F2                 STX     USERS           SET UP FOR USER
FC11  86 FF                 LDA A   #$FF
FC13  C6 08                 LDA B   #2*NBR
FC15  36            RESE1   PSH A
FC16  5A                    DEC B
FC17  26 FC                 BNE     RESE1

                    **      MAIN - MAIN MONITOR LOOP
                    *
                    *       HANDLERS RETURN:
                    *               (B) = NUMBER BYTES SUBJECT TO "CHANGE"
                    *               (X) = ADDRESS BYTES SUBJECT TO "CHANGE"
                    *               (A) = 0 ENABLES "FORWARD" AND "BACK"

FC19  97 EE         MAIN    STA A   T1
FC1B  86 19                 LDA A   #-MAIN/256*256+MAIN  LO ORDER RET.ADDR.
FC1D  36                    PSH A
FC1E  86 FC                 LDA A   #MAIN/256            HI ORDER BYTE RET. ADDR.
FC20  36                    PSH A                          RETURN ONTO STACK
FC21  BD FD F4      MAIN1   JSR     INCH                 GET COMMAND
FC24  7D 00 EE              TST     T1
FC27  27 08                 BEQ     MAIN2                FORWARD OR BACK OK
FC29  81 0F                 CMP A   #$F
FC2B  27 F4                 BEQ     MAIN1                ILLEGAL NOW
FC2D  81 0B                 CMP A   #$B
FC2F  27 F0                 BEQ     MAIN1                ALSO ILLEGAL NOW
FC31  DF EC         MAIN2   STX     T0
FC33  CE FF B4              LDX     #CMDTAB-2
FC36  08            MAIN3   INX
FC37  08                    INX                          GET HANDLER ADDRESS
FC38  4A                    DEC A
FC39  2A FB                 BPL     MAIN3
FC3B  A6 01                 LDA A   1,X                  TARGET ADDRESS ONTO STACK
FC3D  36                    PSH A
FC3E  A6 00                 LDA A   0,X
FC40  36                    PSH A
FC41  DE EC                 LDX     T0                   RESTORE X
FC43  96 EE                 LDA A   T1
FC45  39            ZERO    RTS                          JUMP TO HANDLER
```

```
                    **      BKSET - ENTRY FOR BREAKPOINT TABLE
                    *
                    *       ENTRY:  NONE
                    *       EXIT    (B) = 2
                    *               (X) = ADDRESS IN TABLE
                    *       USES: ALL,T0,11

FC46   CE 00 F2  BKSET      LDX     #BKTBL-2
FC49   86 FF                LDA A   #$FF
FC43   C6 04                LDA B   #0DR                FIND SPOT IN TABLE
FC4D   08        BKSE1      INX
FC4E   08                   INX
FC4F   A1 00                CMP A   0,X
FC51   26 04                BNE     BKSE2
FC53   A1 01                CMP A   1,X
FC55   27 0E                REC     BKSE3               EMPTY SPOT
FC57   5A        BKSE2      DEC B
FC58   26 F3                BNE     BKSE1               STILL HOPE

                    *       FULL UP

FC5A   BD FD 8D             JSR     OUTST0
FC5D   00 47 3E 0E          FCB     0,LTRF,LTRU,LTRL,LTRL,$A0
FC63   4C                   INC A
FC64   39                   RTS

FC65   DF EE     BKSE3      STX     T1
FC67   8D 1D                BSR     OUTSTA
FC69   1F 85                FCC     LIBB,LTRR+$80
FC6B   8D 08                BSR     D0PM1
FC6D   4C                   INC A       (A) = 1
FC6E   39                   RTS
```

```
                        **      DOPMT - ACCEPT ADDRESS VALUE WITH 'DO' PROMPT
                        *
                        *       ENTRY:  (X) = ADDRESS TO STORE INPUTTED VALUE
                        *       EXIT:   (B) = 2
                        *               (X) UNCHANGED
                        *       USES:   ALL,T0,T1

FC6F   DF EE      DOPMT STX     T1
FC71   8D 13            BSR     OUTSTA          OUTPUT PROMPT 'DO'
FC73   3D 90            FCC     LTRD,LTR0+$80
FC75   8D 45      DOPM1 BSR     REDIS           RESET DISPLAY
FC77   DE EE            LDX     T1              RESTORE X
FC79   C6 02            LDA B   #2
FC7B   7E FD 25         JMP     PROMPT          INPUT NEW VALUE


                        **      ADDR - ACCEPT ADDRESS VALUE WITH 'AD' PROMPT
                        *
                        *       ENTRY, EXIT -- SEE 'DOPMT'

FC7E   DF EE      ADDR  STX     T1
FC80   8D 04            BSR     OUTSTA
FC82   77 BD            FCC     HEXA,LTRD+$80
FC84   20 EF            BRA     DOPM1


                        **      OUTSTA - OUTPUT STRING FOR ADDRESS PROMPT
                        *

FC86   CE C1 2F   OUTSTA LDX    #DG2ADD
FC89   7E FE 50          JMP    OUTST1


                        **      DO - RESET USER PC AND RESUME
                        *
                        *       ENTRY:  NONE
                        *       EXIT:   TO 'RESUME'
                        *       USES:   ALL

FC8C   DE F2      DO    LDX     USERS
FC8E   08               INX
FC8F   08               INX
FC90   08               INX
FC91   08               INX
FC92   08               INX
FC93   08               INX                     X TO USER PC
FC94   8D D9            BSR     DOPMT
```

```
                        **      RESUME - RESUME USER PROGRAM
                        *
                        *       1) BLANKS ALL DISPLAYS
                        *       2) INITIALIZES (DIGADD)
                        *       3) STEPS USER CODE PAST BREAKPOINT
                        *       4) INSERTS BREAKPOINTS
                        *       5) PRINTS INSTRUCTION UPON RETURN
                        *       ENTRY:NONE
                        *       EXIT: (B) = 1
                        *             (X) = USERPC
                        *       USES  ALL,T0,T1

FC96  8D 24     RESUME  BSR     REDIS           RESET DISPLAY
FC98  4F                CLR A
FC99  C6 06             LDA B   #6
FC9B  BD FE 3A  RES1    JSR     OUTCH           CLEAR DISPLAYS
FC9E  5A                DEC B
FC9F  26 FA             BNE     RES1
FCA1  8D 19             BSR     REDIS           RESET DISPLAY
FCA3  BD FE 6B  RES2    JSR     SSTEP           STEP PAST BREAKPOINT
FCA6  C6 04             LDA B   #NBR            SET BREAKPOINTS
FCA8  30        RES3    TSX
FCA9  EE 08             LDX     2*NBR,X         GET BREAKPOINT ADDRESS

FCAB  A6 00             LDA A   0,X
FCAD  36                PSH A
FCAE  36                PSH A
FCAF  86 3F             LDA A   #$3F            REPLACE WITH SWI
FCB1  A7 00             STA A   0,X
FCB3  5A                DEC B
FCB4  26 F2             BNE     RES3
FCB6  CE FC CE          LDX     #BKPT
FCB9  7E FE FC          JMP     SWIVE1          GO TO USER CODE

                        **      REDIS - RESET DISPLAYS
                        *
                        *       ENTRY:  NONE
                        *       EXIT:   DIGADD SET TO LEFTMOST DIGIT
                        *       USES:   T0

FCBC  DF EC     REDIS   STX     T0
FCBE  CE C1 6F          LDX     #DG6ADD
FCC1  DF F0             STX     DIGADD
FCC3  DE EC             LDX     T0
FCC5  39                RTS
```

```
                    **      BADDR - BUILD ADDRESS
                    *
                    *       ENTRY: NONE
                    *       EXIT   (X) = ADDRESS

FCC6  CE 00 EE      BADDR   LDX     #T1
FCC9  8D B3                 BSR     ADDR
FCCB  DE EE                 LDX     T1
FCCD  39                    RTS


                    **      BKPT - BREAK POINT RETURN
                    *       1) REMOVE BKPTS FROM USER CODE
                    *       2) CHECK FOR BREAKPOINT HIT AND EETHER
                    *           A)  RESUME IF NO HIT
                    *           B)  PRINT INSTRUCTION AND RETURN IF HIT

FCCE  30            BKPT    TSX
FCCF  9F F2                 STS     USERS
FCD1  A6 06                 LDA A   6,X
FCD3  26 02                 BNE     BKP1            DECREMENT PC ON USERS STACK
FCD5  6A 05                 DEC     5,X
FCD7  4A            BKP1    DEC A
FCD8  A7 06                 STA A   6,X
FCDA  E6 05                 LDA B   5,X
FCDC  D7 EC                 STA B   T0              SAVE FOR COMPARE
FCDE  97 ED                 STA A   T0+1

                    **      NOW CLEAR BREAKPOINTS

FCE0  0C                    CLC                     'C' IS HIT FLAG
FCE1  8E 00 D9      BKP2    LDS     #BKTBL-3-NBR-NBR
FCE4  C6 04                 LDA B   #NBR
FCE6  32            BKP3    PUL A
FCE7  32                    PUL A                   OLD OP CODE INTO A
FCE8  30                    TSX
FCE9  EE 08                 LDX     2*NBR,X
FCEB  9C EC                 CPX     T0              DO WE HAVE A HIT?
FCED  26 01                 BNE     BKP4               NO WE DO NOT
FCEF  0D                    SEC                           YES WE DO - SET FLAG
FCF0                BKP4    EQU     *
FCF0  A7 00                 STA A   0,X             FIX USER CODE
FCF2  5A                    DEC B
FCF3  26 F1                 BNE     BKP3
FCF5  24 AC                 BCC     RES2            BREAKPOINT NOT HIT
FCF7  DE EC                 LDX     T0              (X) = USER PC


                    **      MEM - DISPLAY ADDRESS AND DATA
                    *
                    *       ENTRY:  (X) = ADDRESS
                    *       EXIT:   (B) = 1
                    *       USES: A,B,C,T0,T1

FCF9  8D C1         MEM     BSR     REDIS           RESET DISPLAY
FCFB  DF EE                 STX     T1
FCFD  CE 00 EE              LDX     #T1
FD00  C6 02                 LDA B   #2
FD02  8D 03                 BSR     MEM2            DISPLAY ADDRESS
FD04  EE 00                 LDX     0,X
FD06  5A                    DEC B
FD07  7E FD 7B      MEM2    JMP     DSPLAY          OUTPUT DATA


                    **      AUTO - AUTO LOAD OF MEMORY
                    *
                    *       ENTRY: NONE
                    *       EXIT:  NO EXIT POSSIBLE
                    *       USES:  ALL,T0,T1
```

```
FD0A  8D BA     AUTO    BSR     BADDR           BUILD ADDRESS
FD0C  8D EB     AUT1    BSR     MEM
FD0E  8D 0B             BSR     REPLAC
FD10  08                INX
FD11  20 F9             BRA     AUT1            NO EXIT


                **      EXAM - EXAMINE MEMORY
                *
                *       ENTRY:  NONE
                *       EXIT:   (X) = ADDRESS
                *               (B) = 0
                *               (A) = 0
                *       USES:   ALL,T0,T1

FD13  8D B1     EXAM    BSR     BADDR           BUILD ADDRESS
FD15  09                DEX

                **      FOWD - DISPLAY NEXT BYTE
                *
                *       ENTRY:  (X) = OLD ADDRESS
                *       EXIT:   (X) = (XOLD) +1
                *               (B) = 1
                *               (A) = 0
                *       USES:   ALL,T0

FD16  08        FOWD    INX
FD17  08                INX


                **      BACK - DISPLAY PREVIOUS BYTE
                *
                *       ENTRY:  (X) = ADDRESS
                *       EXIT:   (X) = (XOLD) + 1
                *               (B) = 1
                *               (A) = 0
                *       USES:   ALL,T0
FD18  09        BACK    DEX
FD19  20 DE             BRA     MEM             DISPLAY ADDRESS AND DATA


                **      REPLAC - REPLACE DISPLAYED VALUE
                *
                *       'REPLAC' 1) BACKSPACES DISPLAY TO CANCEL DISPLAYED VALUE
                *                2) SENDS PROMPT FOR REPLACEMENT VALUE
                *                3) ACCEPTS AND REPLACES DESIGNATED BYTE(S)
                *       ENTRY:  (X) = ADDRESS OF BYTE(S) TO REPLACE
                *               (B) = NUMBER OF BYTES
                *               (DIGADD)= ADDRESS OF DIGIT TO RIGHT OF DISPLAYED
                *       EXIT:   B,X,DIGADD UNCHANGED
                *       USES:   T0,A,C

FD1B  5D        REPLAC  TST B
FD1C  27 06             BEQ     REPL1           NO BYTES
FD1E  36                PSH A
FD1F  8D 22             BSR     BKSP            BACKSPACE DISPLAYS
FD21  8D 02             BSR     PROMPT
FD23  32                PUL A
FD24  39        REPL1   RTS


                **      PROMPT - PROMPT AND INPUT BYTES
                *
                *       ENTRY:  (X) = ADDRESS TO STORE VALUE
                *               (B) = NUMBER OF BYTES
                *               (DIGADD) = ADDRESS OF FIRST ECHO CHARACTER
                *       EXIT:   B,X UNCHANGED
                *               DIGADD UPDATED
                *       USES:   T0, DIGADD
```

```
FD25  37          PROMPT  PSH B
FD26  86 08               LDA A   #DASH           PROMPT CHARACTER
FD28  58                  ASL B
FD29  BD FE 3A    PROM1   JSR     OUTCH           SEND PROMPT

FD2C  5A                  DEC B
FD2D  26 FA               BNE     PROM1
FD2F  33                  PUL B
FD30  8D 11               BSR     BKSP            BACKSPACE DISPLAYS
FD32  37                  PSH B                   **ALTERNATE ENTRY**
FD33  BD FE 09    PROM2   JSR     INHB            GET BYTE VALUE
FD36  A7 00               STA A   0,X             PLACE INTO MEMORY
FD38  08                  INX                     BUMP POINTER
FD39  5A                  DEC B
FD3A  26 F7               BNE     PROM2           MORE TO GO
FD3C  33                  PUL B
FD3D  17                  TBA                     DUPLICATE
FD3E  09          PROM3   DEX                     FIX X
FD3F  4A                  DEC A
FD40  26 FC               BNE     PROM3
FD42  39                  RTS                     EXIT


                  **      BKSP - BACKSPACE DISPLAYS
                  *
                  *       ENTRY:  (B) = NUMBER DIGIT PAIRS TO BACKSPACE
                  *       EXIT:   (DIGADD) = (DIGADD) + 20 * (B)
                  *       USES:    A,C

FD43  37          BKSP    PSH B
FD44  96 F1               LDA A   DIGADD+1        L.S. BYTE
FD46  8B 20       BKSP1   ADD A   #$20            BACKSPACE TWO PLACES
FD48  5A                  DEC B
FD49  26 FB               BNE     BKSP1
FD4B  97 F1               STA A   DIGADD+1
FD4D  33                  PUL B
FD4E  39                  RTS


                  **      REGISTER DISPLAY FUNCTIONS
                  *
                  *       ENTRY:  NONE
                  *       EXIT:   (B) = NUMBER BYTES THIS REGISTER
                  *               (X) = REGISTER ADDRESS ON STACK
                  *               (DIGADD) INITIALIZED TO DIGIT 6
                  *       USES:   ALL,T0

FD4F  8D 3B       REGX    BSR     OUTSTJ          PRINT 'REGX'
FD51  30 95               FCC     LTRI,LTRN+$80
FD53  20 16               BRA     REGX1

FD55  8D 35       REGA    BSR     OUTSTJ          PRINT 'ACCA'
FD57  77 0D 0D            FCC     HEXA,LTRC,LTRC,LTRA+$80
FD5B  20 10               BRA     REGA1

FD5D  8D 2D       REGB    BSR     OUTSTJ          PRINT 'ACCB'
FD5F  77 0D 0D            FCC     HEXA,LTRC,LTRC,LTRB+$80
FD63  20 09               BRA     REGB1

FD65  8D 25       REGP    BSR     OUTSTJ          PRINT 'PC'
FD67  67 8D               FCC     LTRP,LTRC+$80

FD69  4C                  INC A                   (A) = OFFSET INTO STACK
FD6A  4C                  INC A                   (B) = #BYTES THIS REGISTER
FD6B  5C          REGX1   INC B
FD6C  4C                  INC A
FD6D  4C          REGA1   INC A
FD6E  5C          REGB1   INC B
FD6F  8B 02               ADD A   #2
```

```
FD71  DE F2           REG1    LDX     USERS
FD73  08                      INX                     POINT X TO REGISTER
FD74  4A                      DEC A
FD75  26 FC                   BNE     REG1
FD77  8D 02                   BSR     DSPLAY
FD79  4C                      INC A
FD7A  39                      RTS


                      **      DISPLAY - DISPLAY INDEXED BYTES
                      *
                      *       ENTRY: (X) = ADDRESS OF BYTES TO OUTPUT
                      *              (B) = NUMBER OF BYTES TO DISPLAY
                      *       EXIT: X,B UNCHANGED
                      *              (DIGADD) UPDATED
                      *       USES:  ALL, TO

FD7B  37              DSPLAY  PSH B
FD7C  A6 00           DIS1    LDA A   0,X             GET BYTE
FD7E  BD FE 20                JSR     OUTBYT          DISPLAY BYTE
FD81  08                      INX
FD82  5A                      DEC B
FD83  26 F7                   BNE     DIS1
FD85  33                      PUL B
FD86  17                      TBA                     DUPLICATE BYTE COUNT
FD87  09              DIS2    DEX                     RESTORE X
FD88  4A                      DEC A
FD89  26 FC                   BNE     DIS2
FD8B  39                      RTS


                      *       CLEAR B AND JUMP TO OUTSTR

FD8C  5F              OUTSTJ  CLR B
FD8D  CE C1 6F        OUTSTO  LDX     #DG6ADD
FD90  7E FE 50                JMP     OUTST1


                      **      CONDX - DISPLAY CONDITION CODES
                      *
                      *       ENTRY:  DIGADD INITIALIZED
                      *       EXIT:   (B) = 0
                      *       USES:   ALL,TO

FD93  BD FC BC        CONDX   JSR     REDIS           RESET DISPLAYS
FD96  DE F2                   LDX     USERS
FD98  C6 20                   LDA B   #$20
FD9A  4F              CONDO   CLR A
FD9B  E5 01                   BIT B   1,X             MASK DESIRED BIT

FD9D  27 01                   BEQ     COND1           IS A ZERO
FD9F  4C                      INC A                   IS A ONE
FDA0  BD FE 28        COND1   JSR     OUTHEX
FDA3  56                      ROR B
FDA4  26 F4                   BNE     CONDO           MORE TO GO
FDA6  4C                      INC A
FDA7  39                      RTS


                      **      STKPTR - OUTPUT USER STACK POINTER
                      *
                      *       ENTRY: (DIGADD) INITIALIZED
                      *       EXIT:  (B) = 0
                      *       USES:   ALL, TO

FDA8                  REGS    EQU     *
FDA8  8D E2           STKPTR  BSR     OUTSTJ
FDAA  5B E7                   FCC     LTRS,LTRP+$80
FDAC  D6 F3                   LDA B   USERS+1
FDAE  CB 07                   ADD B   #7
```

```
FDB0  99 F2            ADC A   USERS           CLEAN UP FOR USER
FDB2  8D 6C            BSR     OUTBYT
FDB4  17               TBA
FDB5  5F               CLR B
FDB6  8D 68            BSR     OUTBYT
FDB8  86 01            LDA A   #1
FDBA  39               RTS


                  **     ENCODE - SCAN AND ENCODE KEYBOARD
                  *
                  *      ENTRY: NONE
                  *      EXIT:  (A) = HEX VALUE OF KEY PRESSED
                  *             'C' SET FOR VALID CONDITION
                  *      USES:  A,C,TO

FDBB  37          ENCODE PSH B
FDBC  F6 C0 03           LDA B   COL1            GET KEYBOARD DATA
FDBF  B6 C0 06           LDA A   COL3
FDC2  48                 ASL A                   LEFT JUSTIFY DATA
FDC3  48                 ASL A
FDC4  48                 ASL A
FDC5  59                 ROL B
FDC6  48                 ASL A
FDC7  59                 ROL B
FDC8  48                 ASL A
FDC9  59                 ROL B
FDCA  37                 PSH B
FDCB  F6 C0 05           LDA B   COL2            GET LAST DATA
FDCE  C4 1F              AND B   #$1F            MASK ANY GARBAGE
FDD0  1B                 ABA
FDD1  33                 PUL B
FDD2  43                 COM A
FDD3  53                 COM B

                  *      (BA) IS NOW KEYBOARD PATTERN

FDD4  DF EC              STX     TO
FDD6  CE FF A5           LDX     #HEXTAB-1       TABLE OF POSSIBLE OUTPUTS
FDD9  11                 CBA                     FIND ACTIVE ACCUMULATOR
FDDA  27 11              BEQ     ENC3            ILLEGAL OR NO KEY
FDDC  24 06              BCC     ENC1            A ACTIVE
FDDE  36                 PSH A                   B ACTIVE
FDDF  17                 TBA                         INTERCHANGE B,A
FDE0  33                 PUL B
FDE1  CE FF AD           LDX     #HEXTAB+7
FDE4  5D          ENC1   TST B                   B SHOULD BE ZERO
FDE5  26 06              BNE     ENC3            ILLEGAL
FDE7  08          ENC2   INX                     SCAN FOR ACTIVE BIT
FDE8  48                 ASL A
FDE9  22 FC              BHI     ENC2            NOT ACTIVE BIT
FDEB  27 01              BEQ     ENC4            LEGAL CHARACTER
FDED  0C          ENC3   CLC                     ILLEGAL RETURNS 'C' CLEAR
FDEE  A6 00       ENC4   LDA A   0,X             GET HEX FROM TABLE
FDF0  DE EC              LDX     TO
FDF2  33                 PUL B                   CLEAN UP
FDF3  39                 RTS                       AND RETURN


                  **     INCH - INPUT CHARACTER FROM KEYBOARD
                  *
                  *        'INCH' WAITS FOR A TRANSITION BETWEEN ILLEGAL AND
                  *           LEGAL KEYBOARD CONDITIONS, AND RETURNS HEX VALUE
                  *              OF KEY DEPRESSED
                  *
                  *      ENTRY:  NONE
                  *      EXIT:   (A) = HEX VALUE
                  *      USES:   A,C,TO

FDF4  37          INCH   PSH B
FDF5  C6 7F       INC1   LDA B   #TIME            VIOLATION COUNT
```

```
FDF7  8D C2     INC2    BSR     ENCODE          WAIT FOR ILLEGAL INTERVAL
FDF9  25 FA             BCS     INC1            STILL LEGAL
FDFB  5A                DEC B
FDFC  26 F9             BNE     INC2            NOT A FELONY

                *       NOW WE'RE SURE WE HAVE AN ILLEGAL CONDITION AND
                *       NOT JUST A RELEASE CONTACT BOUNCE

FDFE  C6 7F     INC3    LDA B   #TIME           TIME UNTIL PAROLE
FE00  8D B9     INC4    BSR     ENCODE
FE02  24 FA             BCC     INC3            BAD BEHAVIOR
FE04  5A                DEC B
FE05  26 F9             BNE     INC4            BACK IN THE SLAMMER
FE07  33                PUL B
FE08  39                RTS

                **      IHB - INPUT HEX BYTE AND DISPLAY ON LEDS
                *
                *       ENTRY: NONE
                *       EXIT:  (A) = BYTE VALUE
                *              (DIGADD) UPDATED
                *       USES:  A,TO,C

FE09  8D E9     IHB     BSR     INCH            GET FIRST HALF
FE0B  8D 1B             BSR     OUTHEX          ECHO TO DISPLAYS
FE0D  48                ASL A
FE0E  48                ASL A
FE0F  48                ASL A
FE10  48                ASL A
FE11  37                PSH B
FE12  16                TAB
FE13  8D DF             BSR     INCH            GET NEXT HALF
FE15  8D 11             BSR     OUTHEX          ECHO
FE17  1B                ABA
FE18  33                PUL B
FE19  36                PSH A
FE1A  8D 9F     IHB1    BSR     ENCODE          WAIT FOR KEY RELEASE
FE1C  25 FC             BCS     IHB1
FE1E  32                PUL A                   RESTORE LEGAL ENTRY
FE1F  39                RTS


                **      OUTBYT - OUTPUT TWO HEX DIGITS
                *
                *       ENTRY: (A) = BYTE VALUE TO OUTPUT
                *       EXIT:  (DIGADD) UPDATED
                *       USES:  C,TO

FE20  36        OUTBYT  PSH A
FE21  44                LSR A
FE22  44                LSR A
FE23  44                LSR A
FE24  44                LSR A
FE25  8D 01             BSR     OUTHEX          OUTPUT M.S. FOUR BITS
FE27  32                PUL A

-

                **      OUTHEX - OUTPUT HEX DIGIT
                *
                *       ENTRY: (A) = HEX VALUE
                *       EXIT: (DIGADD) UPDATED
                *       USES: C,TO

FE28  36        OUTHEX  PSH A
FE29  84 0F             AND A   #$F             MASK GARBAGE
FE2B  DF EC             STX     TO
FE2D  CE FF 95          LDX     #DISTAB-1       DISPLAY CODE TABLE
FE30  08        OUTH1   INX
FE31  4A                DEC A
FE32  2A FC             BPL     OUTH1
FE34  A6 00             LDA A   0,X             DISPLAY CODE FOR HEX
FE36  8D 04             BSR     OUTO            ALTERNATE ENTRY FOR 'OUTCH'
```

```
FE38  32                  PUL A
FE39  39                  RTS


                  **      OUTCH - OUTPUT CHARACTER TO DISPLAY
                  *
                  *       ENTRY: (A) = SEGMENT CODE
                  *              (DIGADD) = ADDRESS OF DIGIT TO OUTPUT
                  *       EXIT:  (DIGADD) UPDATED
                  *       USES: C,TO

FE3A  DF EC       OUTCH   STX     TO
FE3C  DE F0       OUTO    LDX     DIGADD          **ALTERNATE ENTRY** FROM 'OUTHL'
FE3E  37                  PSH B
FE3F  49                  ROL A
FE40  49                  ROL A                   PRE-ROTATE A
FE41  C6 10               LDA B   #$10            TO GET TO NEXT DIGIT
FE43  49          OUT1    ROL A                   HERE WE MAKE TWO PASSES AT
FE44  A7 00               STA A   0,X                LIGHTING DIGITS--
FE46  09                  DEX                           KING'S X ON FIRST PASS
FE47  5A                  DEC B
FE48  26 F9               BNE     OUT1
FE4A  DF F0               STX     DIGADD          UPDATE 'DIGADD'
FE4C  DE EC               LDX     TO              RESTORES X
FE4E  33                  PUL B
FE4F  39                  RTS


                  **      OUTSTR--OUTPUT IMBEDDED CHARACTER STRING
                  *          CALLING CONVENTION:
                  *                JSR     OUTSTR
                  *                FIRST CHARACTER
                  *                     *
                  *                     *
                  *                LAST CHARACTER (HAS D.P. LIT)
                  *                NEXT INSTRUCTION
                  *
                  *       ENTRY:   NONE
                  *       EXIT:    TO 'NEXT INSTRUCTION'
                  *                (A) = 0
                  *       USES:    A,X,TO

FE50  DF F0       OUTST1  STX     DIGADD          **ALTERNATE ENTRY** SETS UP DIGA
FE52  30          OUTSTR  TSX                     POINT 'X' AT STRING
FE53  EE 00               LDX     0,X
FE55  31                  INS
FE56  31                  INS
FE57  A6 00       OUTST3  LDA A   0,X             GET CHARACTER
FE59  8D DF               BSR     OUTCH           OUTPUT IT TO DISPLAYS
FE5B  08                  INX
FE5C  4D                  TST A                   LAST CHARACTER IS NEGATIVE
FE5D  2A F8               BPL     OUTST3
FE5F  4F                  CLR A
FE60  6E 00               JMP     0,X             RETURN TO 'NEXT INST.'

                  **      STEP - STEP USER CODE
                  *
                  *       ENTRY:   NONE
                  *       EXIT:    (B) = 1
                  *                (X) = USER P.C.
                  *                (A) = 0
                  *       USES:    ALL,TO,T1
FE62  8D 07       STEP    BSR     SSTEP           STEP USER CODE
FE64  DE F2               LDX     USERS           DISPLAY INSTRUCTION
FE66  EE 06               LDX     6,X
FE68  7E FC F9            JMP     MEM


                  **      SSTEP - PERFORM SINGLE STEP.
                  *
```

```
FE6B  9F EE      SSTEP   STS     TEMP            WE'LL USE THIS WHEN WE RETURN
FE6D  DE F2              LDX     USERS
FE6F  A6 07              LDA A   7,X             PUSHING USER PC ONTO MONITOR
FE71  36                 PSH A                     STACK
FE72  A6 06              LDA A   6,X
FE74  36                 PSH A
FE75  EE 06              LDX     6,X             NOW GET USER PC INTO X
FE77  86 3F              LDA A   #$3F            SWI'S ARE NORMAL EXIT FROM
FE79  36                 PSH A                     SCRATCHPAD EXECUTION
FE7A  36                 PSH A
FE7B  A6 02              LDA A   2,X             NOW WE ARE COPYING THREE BYTES
FE7D  36                 PSH A                     OF INSTRUCTION
FE7E  A6 01              LDA A   1,X
FE80  36                 PSH A
FE81  A6 00              LDA A   0,X             THIS IS THE OP CODE SO
FE83  36         BYTCNT  PSH A                     SCRUTINIZE CAREFULLY
FE84  16                 TAB
FE85  CE FF 75           LDX     #OPTAB-1
FE88  08         BYT1    INX
FE89  C0 08              SUB B   #8
FE8B  24 FB              BCC     BYT1
FE8D  A6 00              LDA A   0,X
FE8F  46         BYT2    ROR A
FE90  5C                 INC B
FE91  26 FC              BNE     BYT2
FE93  32                 PUL A
FE94  36                 PSH A
FE95  25 1E              BCS     BYT7
FE97  81 30              CMP A   #$30            CHECK FOR BRANCH
FE99  24 04              BCC     BYT3
FE9B  81 20              CMP A   #$20
FE9D  24 14              BCC     BYT5            IT IS A BRANCH
FE9F  81 60      BYT3    CMP A   #$60
FEA1  25 11              BCS     BYT6            IT IS ONE BYTE
FEA3  81 8D              CMP A   #$8D
FEA5  27 0C              BEQ     BYT5            IT IS BSR
FEA7  84 BD              AND A   #$BD
FEA9  81 8C              CMP A   #$8C
FEAB  27 04              BEQ     BYT4            IS X OR SP IMMEDIATE
FEAD  84 30              AND A   #$30            CHECK FOR THREE BYTES
FEAF  81 30              CMP A   #$30
FEB1  C2 FF      BYT4    SBC B   #$FF
FEB3  5C         BYT5    INC B
FEB4  5C         BYT6    INC B
FEB5  27 70      BYT7    BEQ     RSTRD
FEB7  30                 TSX
FEB8  25 02              BCS     STEP1
FEBA  E7 01              STA B   1,X             BRANCH OFFSET TO 2
FEBC  86 01      STEP1   LDA A   #1
FEBE  C1 02              CMP B   #2
FEC0  2E 06              BGT     STEP3
FEC2  27 02              BEQ     STEP2           TWO BYTES
FEC4  A7 01              STA A   1,X             FOR ONE BYTERS
FEC6  A7 02      STEP2   STA A   2,X             NOT FOR THREE BYTERS
FEC8  4F         STEP3   CLR A                   NOW ADD BYTE COUNT TO PC
FEC9  EB 06              ADD B   6,X
FECB  A9 05              ADC A   5,X
FECD  A7 05              STA A   5,X
FECF  E7 06              STA B   6,X

                 *       DOES THE INSTRUCTION INVOLVE THE PC?  IF SO THEN I.
                 *         MUST BE INTERPRETED

FED1  DE F2      SRCHOP  LDX     USERS
FED3  A7 06              STA A   6,X
FED5  E7 07              STA B   7,X             UPDATE PC ON USER STACK
FED7  C6 06              LDA B   #6
FED9  32                 PUL A
FEDA  36                 PSH A                   GET COPY OF OPCODE
```

```
FEDB  84 CF          AND A   #$CF            IS THIS A SUBROUTINE CALL?
FEDD  81 8D          CMP A   #$8D
FEDF  32             PUL A
FEE0  27 48          BEQ     BSRH
FEE2  81 6E          CMP A   #$6E
FEE4  27 5B          BEQ     JPXH            IT IS INDEXED JUMP
FEE6  81 7E          CMP A   #$7E
FEE8  27 5E          BEQ     JMPH            IT IS EXTENDED JUMP
FEEA  81 39          CMP A   #$39
FEEC  27 62          BEQ     RTSH            IT IS RTS
FEEE  81 3B          CMP A   #$3B
FEF0  27 6C          BEQ     RTIH            IT IS RTI
FEF2  81 3F          CMP A   #$3F
FEF4  27 6E          BEQ     SWIH            IT IS SWI
FEF6  AF 06          STS     6,X             AIM USER PC AT SCRATCH AREA
FEF8  36             PSH A                   REPLACE OPCODE
FEF9  CE FF 05       LDX     #SSRET

                **   SWIVE1 - SET UP BREAKPOINT RETURN AND JUMP TO USER CODE
                *
                *    ENTRY:  (X) = SWI VECTOR
                *    EXIT:   TO USER PROGRAM

FEFC  86 7E     SWIVE1  LDA A   #$7E             JUMP OP CODE
FEFE  97 F4             STA A   SYSSWI
FF00  DF F5             STX     SYSSWI+1
FF02  9E F2             LDS     USERS
FF04  3B                RTI


                *    THE FOLLOWING CODE IS EXECUTED AFTER A SINGLE STEP
                *      OF AN OUT-OF-PLACE INSTRUCTION.  NOW CHECK TO SEE
                *         IF BRANCH OCCURRED, MODIFY THE USER PC ACCORDINGLY

FF05  30        SSRET   TSX                     GET SWI HIT LOCATION INTO X
FF06  EE 05             LDX     5,X
FF08  08                INX
FF09  4F                CLR A
FF0A  5F                CLR B
FF0B  9C EE             CPX     TEMP
FF0D  26 0C             BNE     BCHNTK

                *    ADD THE BRANCH OFFSET TO THE USER PC

FF0F  09                DEX
FF10  EE 00             LDX     0,X             X WILL NOW POINT AT USERPC
FF12  09                DEX                     SAVED VALUE OF PC INTO X
FF13  E6 00             LDA B   0,X             PREPARE TO FETCH BRANCH OFFSET
FF15  2A 01             BPL     PLUS
FF17  43                COM A                   A IS SIGN EXTENSION OF B
FF18  30        PLUS    TSX                     LO COST WAY TO POINT TO USERPC
FF19  EE 05             LDX     5,X
FF1B  EB 01     BCHNTK  ADD B   1,X             ADD BRANCH OFFSET OR ZERO TO PC
FF1D  A9 00             ADC A   0,X
FF1F  30                TSX                     PLACE NEW USERPC ONTO STACK
FF20  A7 05             STA A   5,X
FF22  E7 06             STA B   6,X
FF24  09                DEX                     NOW X AND SP ARE EQUAL
FF25  DF F2     STOX    STX     USERS
FF27  9E EE     BSTRD   LDS     TEMP
FF29  39                RTS                     RETURN TO CALLING ROUTINE

                **   SPECIAL HANDLERS

                **   BSR HANDLER

FF2A  81 8D     BSRH    CMP A   #$8D            IS IT BSR
FF2C  26 02             BNE     JSRH
FF2E  86 5F             LDA A   #$5F            THIS CONVERTS BSR(8D) TO JSR(BD)
```

```
                        **      JSR HANDLER

FF30  80 3F     JSRH    SUB A   #$3F            JSR'S TO JUMPS
FF32  36                PSH A                   CORRECTED OPCODE ONTO STACK
FF33  09                DEX
FF34  09                DEX
FF35  DF F2             STX     USERS
FF37  A6 03     JSRH1   LDA A   3,X
FF39  A7 01             STA A   1,X             MOVE USER REGISTERS
FF3B  08                INX
FF3C  5A                DEC B
FF3D  2A F8             BPL     JSRH1
FF3F  20 90             BRA     SRCHOP          NOW EXECUTE JUMP INSTRUCT

                        **      JPXH - INDEXED JUMP HANDLER.

FF41  33        JPXH    PUL B                   GET OFFSET
FF42  4F                CLR A
FF43  EB 05             ADD B   5,X
FF45  A9 04             ADC A   4,X
FF47  8C                FCB     $8C             CPX#: ONE BYTE BRA NEWPC

                        **      JMP HANDLER

FF48  32        JMPH    PUL A
FF49  33                PUL B
FF4A  A7 06     NEWPC   STA A   6,X
FF4C  E7 07             STA B   7,X
FF4E  20 D5             BRA     STOX            RETURN TO CALLER

                        **      RTS HANDLER

FF50  08        RTSH    INX
FF51  08                INX
FF52  DF F2             STX     USERS           NET PULL OF TWO BYTES
FF54  A6 03     RTS1    LDA A   3,X             MOVE FIVE BYTES
FF56  A7 05             STA A   5,X
FF58  09                DEX
FF59  5A                DEC B
FF5A  2E F8             BGT     RTS1
FF5C  20 C9             BRA     BSTRD

                        **      RTI HANDLER

FF5E  08        RTIH    INX
FF5F  5A                DEC B
FF60  2A FC             BPL     RTIH
FF62  20 C1             BRA     STOX

                        **      SWI HANDLER

FF64  A6 07     SWIH    LDA A   7,X
FF66  A7 00             STA A   0,X
FF68  09                DEX
FF69  5A                DEC B
FF6A  2A F8             BPL     SWIH
FF6C  8A 10             ORA A   #^00010000      SET INTERUPT MASK
FF6E  A7 01             STA A   1,X
FF70  C6 FA             LDA B   #-USWI/256*256+USWI     USWI LO ORDER
FF72  86 00             LDA A   #USWI/256
FF74  20 D4             BRA     NEWPC           PATCH IN UIRQ
```

```
                        **      OPTAB - LEGAL OP-CODE LOOKUP TABLE

FF76  9C 00 3C  OPTAB   FDB     $9C00,$3CAF,$4000,$00AC,$6412,$6412,$6410,$6410
FF86  11 01 10          FDB     $1101,$1004,$1000,$1000,$110D,$100C,$100C,$100C

                        **      HEX DISPLAY CODE TABLE

FF96  7E 30 6D  DISTAB  FCC     HEX0,HEX1,HEX2,HEX3,HEX4,HEX5,HEX6,HEX7
FF9E  7F 7B 77          FCC     HEX8,HEX9,HEXA,HEXB,HEXC,HEXD,HEXE,HEXF

                        **      KEY VALUE TABLE

FFA6  07 0A 0D  HEXTAB  FCC     7,10,13,2,5,8,11,14
FFAE  03 06 09          FCC     3,6,9,12,15,0,1,4

                        **      COMMAND HANDLER ENTRY POINT TABLE

FFB6  FC 45 FD  CMDTAB  FDB     ZERO,REGA,REGB,REGP,REGX,CONDX,REGS,RESUME,STEP
FFC8  FC 46 FD          FDB     BKSET,AUTO,BACK,REPLAC,DO,EXAM,FOWD



FFF8                    ORG     $FFF8

                        **      INTERRUPT VECTORS.

FFF8  00 F7             FDB     UIRQ            USER IRQ HANDLER
FFFA  00 F4             FDB     SYSSWI          SYSTEM SWI HANDLER
FFFC  00 FD             FDB     UNMI            USER NMI HANDLER
FFFE  FC 00             FDB     RESET

0000                    END


STATEMENTS =970

FREE BYTES =24298

NO ERRORS DETECTED
```