```
*******************************************************************************
*                                                                             *
*                    6809 Microsystem System Software                         *
*                    --------------------------------                         *
*                                                                             *
*                                                                             *
*                                                                             *
*       Author              : J.A. Eva                                        *
*       Version             : 4                                               *
*       Revision            : 0                                               *
*       Latest release      : 0                                               *
*       Release date        : 6 April 1984                                    *
*                                                                             *
*       Release History     : 0 (6-APR-84)                                    *
*                             First release                                   *
*                                                                             *
*******************************************************************************

*------------------------------------------------------------------------------
*       Equates
*       -------
*

       * Monitor ID numbers
       * ------------------

0034   VERNUM  EQU    '4      Version number
0030   REVNUM  EQU    '0      Revision number
0030   RELNUM  EQU    '0      Release number

       * Addresses and pages
       * --------------------

B000   CHRGEN  EQU    $B000   Start of character generator
C000   BOOTAD  EQU    $C000   Address of disk boot block
CD03   FLXWST  EQU    $CD03   FLEX warm start address
E000   IOBASE  EQU    $E000   IO page base address
E100   DATAST  EQU    $E100   Start of monitor data and CRT parameters
E800   VIDRAM  EQU    $E800   Start address of screen RAM
F000   CODEST  EQU    $F000   Start of monitor code
00E0   IOPAGE  EQU    $E0     Logical IO page address
00E0   VBASE   EQU    $E0

       * Monitor related equates
       * ------------------------

FFF0   DAT     EQU    $FFF0   Start of DAT RAM
A5A5   TSTPAT  EQU    $A5A5   Test pattern for DAT setup

0000   REGCC   EQU    0       Offsets from U pointer for user registers
0001   REGA    EQU    1
0002   REGB    EQU    2
0003   REGDP   EQU    3
0004   REGX    EQU    4
```

```
0006  REGY    EQU   6
0008  REGUS   EQU   8
000A  REGPC   EQU   10

0000  BYTE    EQU   0         Used in tables to indicate byte or word values
0001  WORD    EQU   1

003F  SWIINS  EQU   $3F       SWI instruction opcode
0008  NTRP    EQU   8         Number of traps allowed

      * ASCII and other character equates
      * ---------------------------------

0000  EOTB    EQU   $00       End-of-table character (null)
0004  EOT     EQU   $04       End of text for strings
0007  BEL     EQU   $07       Bell character
0008  BS      EQU   $08       Backspace
0009  HT      EQU   $09       Tab
000A  LF      EQU   $0A       Line feed
000C  FF      EQU   $0C       Form feed
000D  CR      EQU   $0D       Carriage return
001B  ESC     EQU   $1B       Escape
0020  SPACE   EQU   $20       Real ASCII space
003F  ERRCHR  EQU   '?        Monitor error character
00FE  SOE     EQU   $FE       Start-of-entry char
00FF  BLANK   EQU   $FF       Blank screen character

      * Special key codes
      * ------------------

00C1  CUPKEY  EQU   'A+$80    Up arrow
00C2  CDNKEY  EQU   'B+$80    Down arrow
00C4  CLFKEY  EQU   'D+$80    Left arrow
00C3  CRTKEY  EQU   'C+$80    Right arrow
00CB  HOMKEY  EQU   'H+$80    Home key
00CC  EDTKEY  EQU   FF+$80    Edit/SOE key
00CA  RPTKEY  EQU   'J+$80    Repeat line key
00CD  INSKEY  EQU   'M+$80    Insert key
00D0  DELKEY  EQU   'P+$80    Delete key

      * I/O driver equates
      * -------------------

      * Control bits for IO byte 1
0001  FFBIT   EQU   %00000001 FF causes line feeds instead of clrscr
0002  BSBIT   EQU   %00000010 BS does not blank out char
0004  ESCBIT  EQU   %00000100 ESC sequences not recognised
0008  WRFBIT  EQU   %00001000 Overlength lines do not wrap
0010  CTLBIT  EQU   %00010000 Display illegal control chars?
0020  LFDBIT  EQU   %00100000 Clear line during linefeed in middle page
0040  CMVBIT  EQU   %01000000 Ignore cursor move keys in INECHO
0080  BFRBIT  EQU   %10000000 Ignore buffered input keys in INECHO
0002  DEFIOW  EQU   %0000000000000010 IOBYT2+IOBYT1 default values
```

```
0003  GAPCNT  EQU   3          Number of line feeds between pages

0080  BUFLEN  EQU   128        Length of system input buffer

      * CRT Controller equates
      * ----------------------

E0FE  CRTC    EQU   IOBASE+$FE CRTC address register
E0FC  CRTCON  EQU   IOBASE+$FC CRTC mode register

000A  CURTOP  EQU   10         Cursor start register in CRTC
000E  CURREG  EQU   14         Cursor address register in CRTC

0020  NONDSP  EQU   $20        Cursor non-display code for CRTC

      * PIA equates
      * -----------

E024  PIA     EQU   IOBASE+$24 Keyboard PIA
E024  PIADA   EQU   PIA+0
E025  PIASA   EQU   PIA+1
E026  PIADB   EQU   PIA+2
E027  PIASB   EQU   PIA+3

      * Floppy disk controller equates
      * -----------------------------

E014  FDC     EQU   IOBASE+$14 FDC base address
E014  DRVREG  EQU   FDC+0      Drive/density register
E018  CMDREG  EQU   FDC+4      Command register
E018  STSREG  EQU   FDC+4      Status register
E019  TRKREG  EQU   FDC+5      Track register
E01A  SECREG  EQU   FDC+6      Sector register
E01B  DATREG  EQU   FDC+7      Data register

008C  RDSCMD  EQU   %10001100 Read single sector
0009  RSTCMD  EQU   %00001001 Restore head (12ms step rate)
00D0  INTCMD  EQU   %11010000 Force interrupt
0002  FDDRQ   EQU   %00000010 Data request status bit
0001  FDBUSY  EQU   %00000001 Controller busy bit
```

```
                        *-------------------------------------------------------------------------
                        *       Monitor and IO data storage allocation
                        *       ------------------------------------------
                        *
                        *=========================================================================
                        *
                        *       Monitor Data
                        *       ------------
                        *
                        * The monitor data area is divided into two sections:
                        *   1) Data that remains constant from monitor release to release
                        *   2) General scratchpad and variable storage whose position is
                        *       unimportant, as it is not used by anything except the monitor.
                        *

E100                            ORG     DATAST  Normally just above the IO page, i.e. $E100

                        * Constant data (must never be accidentally overwritten!)
                        * ---------------------------------------------------------

E100                    INVEC   RMB     2       Character input without echo
E102                    OUTVEC  RMB     2       Character output
E104                    INEVEC  RMB     2       Character input with echo
E106                    ESCVEC  RMB     2       Escape handling routine (not called!)
E108                    STSVEC  RMB     2       Input status check

E10A                    USERAD  RMB     2       User jump vector
E10C                    IRQAD   RMB     2       IRQ jump vector
E10E                    FIRQAD  RMB     2       FIRQ jump vector
E110                    SWI2AD  RMB     2       SWI2 jump vector
E112                    SWI3AD  RMB     2       SWI3 jump vector

E114                    DATMAP  RMB     16      Copy of DAT RAM contents
(E124                   BITMAP  RMB     8       Bit map of occupied 4K RAM blocks

                E13C    IO      EQU     *+16    Pointer to IO variables - centred for max spread

E12C                    SC_TOP  RMB     2       Top of screen address
E12E                    SC_BOT  RMB     2       Screen end address
E130                    C_ROW   RMB     1       Current cursor row
E131                    C_COL   RMB     1       Current cursor column
E132                    COL_0   RMB     2       Address of column 0 (start of line)
E134                    MAXROW  RMB     1       Maximum number of rows (screen height)
E135                    MAXCOL  RMB     1       Maximum number of columns (screen width)
E136                    W_TOP   RMB     2       Address of top of window
E138                    FW_ROW  RMB     1       First window row
E139                    LW_ROW  RMB     1       Last window row
E13A                    FW_COL  RMB     1       First window column
E13B                    LW_COL  RMB     1       Last window column
E13C                    CURTYP  RMB     1       Cursor type 1 (usually solid line)
E13D                    CURTP2  RMB     1       Cursor type 2 (usually blinking)
E13E                    IOWORD
```

```
E13E                    IOBYT2  RMB     1       IO characteristics byte number 2 (MS byte)
E13F                    IOBYT1  RMB     1       IO characteristics byte number 1
E140                    IOSAVE  RMB     2       Save location for IOWORD
E142                    CHMASK  RMB     1       Mask for character generator

                        * Monitor and IO scratchpad
                        * ------------------------------

E143                    L_ROW   RMB     1       Line start row
E144                    L_COL   RMB     1       Line start column
E145                    LCOL_0  RMB     2       Address of line column 0
E147                    LB_PTR  RMB     2       System line buffer pointer

E149                    ASCBUF  RMB     16      ASCII buffer for D command
E159                    RAMSIZ  RMB     2       Available RAM in system
E15B                    SAVESP  RMB     2       Temporary hold for monitor stack
E15D                    W_HLD1  RMB     1       Temporary holds for new window limits
E15E                    W_HLD2  RMB     1
E15F                    W_HLD3  RMB     1
E160                    W_HLD4  RMB     1
E161                    SV_INE  RMB     2       Holds for INEVEC and STSVEC while buffered
E163                    SV_STS  RMB     2       ... IO is in operation

E165                    TRPTBL  RMB     4*NTRP  NTRP entries, 4 bytes each

E185                    IO_BFR  RMB     5       Buffer to return window and IO words
E18A                    S_LBUF  RMB     BUFLEN  System line buffer

                        * Stacks (right up at the top of LBLK 14)
                        * ------------------------------------------

E7B4                    ORG     (DATAST&$F000)+$07FF-75 Put the stacks right up at the top

        E7B4    USRSTK  EQU     *       User stack can grow as much as it likes
E7B4                    RMB     75      Allow 75 bytes for monitor stack
        E7FF    MONSTK  EQU     *
```

```
                        *-----------------------------------------------------------------------
                        *       MONITOR START
                        *       =============
                        *

F000                            ORG     CODEST  Code start address
                        *=======================================================================
                        *
                        *               System Entry Jump Table
                        *               -----------------------
                        *
                        * These are the system entry jumps, used by utility and applications
                        *  programs for system and IO calls.  They remain identical for
                        *  different releases of the monitor.  Programs which perform
                        *  calls or jumps into the monitor other than at these locations
                        *  will not work for different monitor releases.  Note that IO
                        *  calls INCHR, INCHRE, OUTCHR and KCHECK are done via RAM vectors
                        *  at $E100, which are set up by COLDST and RESIO, and merely provide
                        *  a convenient way of performing IO without using an indirect jump.
                        *

                        *
                        *  1. System related functions
                        *
F000 16  0979                   LBRA    COLDST  Complete system initialisation except DAT
F003 16  09E3                   LBRA    WARMST  Warm start of monitor into command loop
F006 16  0D3E                   LBRA    LRA     Convert logical to real (physical) address
                        *
                        *  2. I/O related functions
                        *
>F009 16  0051                  LBRA    RESIO   Reset the whole IO system to power-on state
 F00C 16  056E                  LBRA    INCHR   Input 1 char w/o echo (Just a JMP [INVEC])
 F00F 16  056F                  LBRA    INCHRE  Input 1 char with echo (Just a JMP [INEVEC])
 F012 16  06E0                  LBRA    STATUS  Check for key pressed (Just a JMP [STSVEC])
 F015 16  05EF                  LBRA    OUTCHR  Output 1 char (Just a JMP [OUTVEC])
                        *
                        *  3. Monitor related functions
                        *
 F018 16  0D7A                  LBRA    PCRLF   Print a CR/LF
 F01B 16  0DE3                  LBRA    PSTRNG  Print a string of chars
 F01E 16  0D7E                  LBRA    PLINE   Print a string, preceded by a CR/LF
 F021 16  0D40                  LBRA    P4HEX   Print X as 4 hex digits
 F024 16  0D49                  LBRA    P2HEX   Print A as 2 hex digits
 F027 16  0D53                  LBRA    P1HEX   Print the ls nibble of A as 1 hex digit
 F02A 16  0D5F                  LBRA    P1SPC   Print one space
 F02D 16  0D5A                  LBRA    P2SPC   Print 2 spaces
                        *
                        *  4. Screen and IO system related functions
                        *
 F030 16  0552                  LBRA    HOMCUR  Home cursor to top of window
 F033 16  04AE                  LBRA    CEOSCR  Clear from cursor to end of window
 F036 16  04A1                  LBRA    CEOLIN  Clear from cursor to end of line
```

Jump Table

```
F039 16   04B2            LBRA   CSMOVE   Move the cursor to (row, col)
F03C 16   06BE            LBRA   STWIND   Set the window limits
F03F 16   0518            LBRA   GTWIND   Get the current window limits
F042 16   06B4            LBRA   STIOWD   Setup the state of the IO word
F045 16   050E            LBRA   GTIOWD   Get the current state of the IO word
```

```
                        *-----------------------------------------------------------------------
                        *       SYSTEM INITIALISATION
                        *       =====================
                        *

                        * Cold start system initialisation - used on reset and power-up.
                        *  Initialises everything - keyboard PIA, char gen, CRTC,
                        *  screen driver variables, interrupt and IO vectors.
F048                    INTSYS
F048 86   3C                    LDA     #$3C        Initialise keyboard PIA
F04A B7   E025                  STA     PIASA
F04D 8D   48                    BSR     CGLOAD      Initialise the character generator
F04F 8E   F9B9                  LDX     #WARMST     Initialise the interrupt vectors
F052 108E E10A                  LDY     #USERAD     Start of vector table
F056 C6   05                    LDB     #5          Number of vectors
F058 AF   A1            INTS05  STX     0,Y++
F05A 5A                         DECB
F05B 26   FB                    BNE     INTS05      Loop till all 5 done
                        *
                        * RESIO entry point - used to restore the IO after it has been
                        *  mucked up.  Initialises the CRTC, screen driver variables and
                        *  IO vectors.
                        *
F05D                    RESIO
F05D 8D   29                    BSR     INTCRT      Initialise the CRT controller
F05F 8D   03                    BSR     INTIO       Then finally the IO variables and vectors
F061 16   0514                  LBRA    HOMCLR      Last of all, clear the screen and return

                        *-----------------------------------------------------------------------
                        * INTIO - initialise the IO variables and vectors
                        *
F064                    INTIO
F064 8E   E18A                  LDX     #S_LBUF     Set up the system buffer
F067 BF   E147                  STX     LB_PTR
F06A 86   0D                    LDA     #CR         Mark the buffer as empty
F06C A7   84                    STA     0,X
F06E 108E F8D4                  LDY     #VECTBL     Set up the vector table and constants
F072 8E   E100                  LDX     #DATAST
F075 C6   0A                    LDB     #VTBLEN     Number of bytes to shift
F077 17   0535                  LBSR    MOVBYT      Copy the bytes
F07A 30   88 22                 LEAX    10+16+8,X   Step over int vectors, DAT table and bitmap
F07D C6   17                    LDB     #IOBLEN     Number of bytes in IO table
F07F 17   054D                  LBSR    MOVBYT      Setup the IO table
F082 86   E3                    LDA     #VBASE+3    Setup the CRT control byte
F084 B7   E0FC                  STA     CRTCON
F087 39                         RTS

                        *-----------------------------------------------------------------------
                        * INTCRT - program the CRT controller
                        *
F088                    INTCRT
F088 4F                         CLRA                Program CRTC - start with reg 0
```

```
F089 8E  F722          LDX   #CRTTAB   CRTC programming constant table
F08C E6  80    INCR10  LDB   0,X+      Get register value
F08E FD  E0FE          STD   CRTC
F091 4C                INCA            Step to next register
F092 81  10            CMPA  #16       Finished yet?
F094 25  F6            BLO   INCR10
F096 39                RTS

                       *-----------------------------------------------------------------------
                       * CGLOAD - load the compressed character generator
                       *
F097                   CGLOAD
F097 86  E2            LDA   #VBASE+2  CRTC control byte - enable chargen
F099 B7  E0FC          STA   CRTCON
F09C 8E  B000          LDX   #CHRGEN   Start of character generator
F09F 6F  80    CGLD05  CLR   0,X+      Clear it all first
F0A1 8C  BFFF          CMPX  #CHRGEN+$0FFF
F0A4 23  F9            BLS   CGLD05
F0A6 8E  B211          LDX   #$B211    Start of used gen area
F0A9 10BE F732         LDY   #CMPRGN   Compressed character generator
F0AD CE  F710          LDU   #SHFTAB   Table of shifted chars
F0B0 CC  0709          LDD   #$0709    Preset counters: 3,S - lines to skip at end of cha
r
F0B3 34  06            PSHS  D         2,S - lines/char
F0B5 CC  0601          LDD   #$0601    1,S - source bits still to shift
F0B8 34  06            PSHS  D         0,S - dest bits still to shift
F0BA 5F                CLRB
F0BB 20  02            BRA   CGLD35    Enter loop late
F0BD                   CGLD30
F0BD 46                RORA            Move one bit into destination byte
F0BE 59                ROLB
F0BF 6A  E4    CGLD35  DEC   0,S       Dec the destination bits count
F0C1 26  2A            BNE   CGLD10    Not finished it yet
F0C3 58                ASLB            Else centre it in the character cell
F0C4 E7  80            STB   0,X+      Put completed byte into memory
F0C6 34  04            PSHS  B         Now create the emphasised version
F0C8 58                ASLB
F0C9 EA  E0            ORB   0,S+
F0CB E7  89 07FF       STB   ($800-1),X And put it in the upper part
F0CF C6  05            LDB   #5        Reset the bit count
F0D1 E7  E4            STB   0,S
F0D3 6A  62            DEC   2,S       Any more rows to do in this char?
F0D5 26  15            BNE   CGLD15    Yes - carry on to next byte
F0D7 C6  07            LDB   #7        Else reset count
F0D9 E7  62            STB   2,S
F0DB E6  63            LDB   3,S       Get number of blank lines on this char
F0DD 3A                ABX             Inc the destination pointer to the next char
F0DE C6  09            LDB   #9        Preset the count of blank lines
F0E0 AC  C4            CMPX  0,U       Is it a shifted char?
F0E2 26  06            BNE   CGLD20    No - leave as is
F0E4 33  42            LEAU  2,U       Else move pointer to next tab entry
F0E6 30  02            LEAX  2,X       Step source pointer to create the shift
F0E8 C6  07            LDB   #7        And reduce the number of trailing blank lines
F0EA E7  63    CGLD20  STB   3,S       Blank line count
```

```
F0EC 5F          CGLD15  CLRB            Clear byte to receive the decompressed bits
F0ED 6A   61     CGLD10  DEC   1,S       Finished source byte yet?
F0EF 26   06             BNE   CGLD25     No - skip on
F0F1 86   08             LDA   #8        Else reset the source bit count
F0F3 A7   61             STA   1,S
F0F5 A6   A0             LDA   0,Y+      And get new source byte
F0F7 8C   E7EF   CGLD25  CMPX  #CHRGEN+('~*16)+15 Done all the chars yet?
F0FA 23   C1             BLS   CGLD30
                 * Now put in the SOE char (this is not compressed)
F0FC 8E   BFE1           LDX   #CHRGEN+(SOE*16)+1
F0FF C6   06             LDB   #6        Number of bytes to move
F101 17   04CB           LBSR  MOVBYT    Transfer the bytes
F104 35   B0             PULS  X,Y,PC    Clean all counters off the stack and return
```

```
                        *-----------------------------------------------------------------------
                        *       KEYBOARD AND SCREEN CONTROL
                        *       ===========================
                        *
                        *-----------------------------------------------------------------------
                        *       Screen output routines
                        *
                        *-----------------------------------------------------------------------
                        * Standard CRT output routine.  The initialisation code will set
                        *  up OUTVEC to point here.
                        *
F106                    CRTOUT
F106 34   56                    PSHS    D,X,U       Save used registers
F108 CE   E13C                  LDU     #IO         Point to IO variables
F10B 17   03FD                  LBSR    CURSOF      Turn the cursor off when outputting
F10E E6   43                    LDB     (IOBYT1-IO),U Get the IO byte
F110 C5   10                    BITB    #CTLBIT     Do we print control chars?
F112 26   04                    BNE     CRTO15      Yes - Just print the char regardless
F114 81   20                    CMPA    #'          Printable character?
F116 25   20                    BLO     CRTO05      No - go process control character
F118                    CRTO15
F118 AE   56                    LDX     (COL_0-IO),U Get cursor address
F11A E6   55                    LDB     (C_COL-IO),U
F11C 3A                         ABX
F11D B8   E142                  EORA    CHMASK      Setup the character generator bit
F120 A7   84                    STA     0,X         Write the character
F122 6C   55                    INC     (C_COL-IO),U Advance current column counter
F124 E6   55                    LDB     (C_COL-IO),U Last character on line?
F126 E1   5F                    CMPB    (LW_COL-IO),U
F128 23   0C                    BLS     CRTO20      No - exit
F12A 6A   55                    DEC     (C_COL-IO),U Undo the inc in case we don't linefeed
F12C A6   43                    LDA     (IOBYT1-IO),U Else test if we must auto-linefeed
F12E 85   08                    BITA    #WRPBIT
F130 26   04                    BNE     CRTO20      No - Just exit
F132 8D   0C                    BSR     CRET        Else perform a CR/LF
F134 8D   0F                    BSR     LFEED
F136 35   D6            CRTO20  PULS    D,X,U,PC    And return
F138                    CRTO05
F138 8E   F8F5                  LDX     #CTLTBL     Table of recognised control chars
F13B 17   0D02                  LBSR    TBSRCH      Search table and execute routine
F13E 35   D6                    PULS    D,X,U,PC    And return

                        * Carriage return processing
F140                    CRET
F140 A6   5E                    LDA     (FW_COL-IO),U
F142 A7   55                    STA     (C_COL-IO),U Set current column pointer
F144 39                         RTS

                        * Line feed processing
F145                    LFEED
```

```
F145 A6   54          LDA   (C_ROW-IO),U Are we on the last row?
F147 A1   5D          CMPA  (LW_ROW-IO),U
F149 1024 04FF        LBHS  SCRL_U    Yes - scroll up and return
F14D 6C   54          INC   (C_ROW-IO),U Else step down to next row
F14F 17   03CD        LBSR  FIXCOL    Fixup the column pointers
F152 A6   43          LDA   (IOBYT1-IO),U Test if we must clear the line
F154 85   20          BITA  #LFDBIT
F156 26   0D          BNE   LFD10     No - Just exit
F158 A6   55          LDA   (C_COL-IO),U Save the current column
F15A 34   02          PSHS  A
F15C 8D   E2          BSR   CRET      Go to the start of the line
F15E 17   0113        LBSR  CLREOL    Clear it out
F161 35   02          PULS  A         Then restore the column pointer
F163 A7   55          STA   (C_COL-IO),U
F165 39       LFD10   RTS             And return

                      * Backspace processing
F166                  BSPACE
F166 A6   55          LDA   (C_COL-IO),U Are we at the start of the line?
F168 A1   5E          CMPA  (FW_COL-IO),U
F16A 22   11          BHI   BSP05     No - Just dec the col pointer
F16C A6   54          LDA   (C_ROW-IO),U Else are we at the top corner?
F16E A1   5C          CMPA  (FW_ROW-IO),U
F170 27   1A          BEQ   BSP10     Yes - can't BS so exit
F172 6A   54          DEC   (C_ROW-IO),U Else go to end of previous line
F174 A6   5F          LDA   (LW_COL-IO),U
F176 A7   55          STA   (C_COL-IO),U
F178 17   03A4        LBSR  FIXCOL    Correct the column pointers
F17B 20   02          BRA   BSP15     And blank the character
F17D          BSP05
F17D 6A   55          DEC   (C_COL-IO),U Backspace the column pointer
F17F A6   43      BSP15 LDA   (IOBYT1-IO),U Must we blank out the char?
F181 85   02          BITA  #BSBIT
F183 26   07          BNE   BSP10     No - Just exit
F185 17   03C3        LBSR  GCADDR    Else get the cursor address
F188 86   20          LDA   #SPACE
F18A A7   84          STA   0,X       And blank the char
F18C 39       BSP10   RTS

                      * Tab processing
F18D                  HTAB
F18D A6   55          LDA   (C_COL-IO),U Get the current column
F18F 1F   89          TFR   A,B       B is used later
F191 A0   5E          SUBA  (FW_COL-IO),U Make it relative to window start
F193 84   F8          ANDA  #%11111000 Step to the next multiple of 8
F195 8B   08          ADDA  #%00001000
F197 AB   5E          ADDA  (FW_COL-IO),U And make it absolute again
F199 A1   5F          CMPA  (LW_COL-IO),U Past the window end?
F19B 22   17          BHI   HTAB05    Yes - don't tab
F19D 34   02          PSHS  A         Else save it for later comparisons
F19F AE   56          LDX   (COL_0-IO),U Get the start of the line
F1A1 3A               ABX             Get the current position
F1A2 A6   80      HTAB15 LDA  0,X+     Get the char on the screen
```

```
F1A4 4C                 INCA            Was it $FF?
F1A5 26  04             BNE     HTAB10  No - leave as is
F1A7 86  20             LDA     #SPACE  Else make it a space
F1A9 A7  1F             STA     -1,X
F1AB 5C         HTAB10  INCB            Step on to next position
F1AC E1  E4             CMPB    0,S     Is it the end?
F1AE 25  F2             BLO     HTAB15  No - keep looping
F1B0 35  02             PULS    A       Else remove the termination value
F1B2 A7  5E             STA     (C_COL-IO),U
F1B4 39         HTAB05  RTS

                * Formfeed processing
F1B5            FFEED
F1B5 A6  43             LDA     (IOBYT1-IO),U Must we clear screen or linefeed?
F1B7 85  01             BITA    #FFBIT
F1B9 26  03             BNE     FFD05   NE - line feeds
F1BB 16  03BA           LBRA    HOMCLR  Else home the cursor, clear the screen and return
F1BE            FFD05
F1BE 86  03             LDA     #GAPCNT Number of lines to skip
F1C0 34  02             PSHS    A
F1C2 8D  81     FFD10   BSR     LFEED   Line feed
F1C4 6A  E4             DEC     0,S
F1C6 26  FA             BNE     FFD10
F1C8 35  82             PULS    A,PC    Return when done

                * Bell processing
F1CA            BELL
F1CA A6  50             LDA     (SC_TOP-IO),U Find out what the screen page is
F1CC 84  F0             ANDA    #%11110000 Isolate the 4K page number
F1CE 8A  0B             ORA     #%00001011 Pulse the ATTBE line (bit 3)
F1D0 B7  E0FC           STA     CRTCON
F1D3 84  F7             ANDA    #%11110111 Reset it again
F1D5 B7  E0FC           STA     CRTCON
F1D8 39                 RTS

                * Escape processing
F1D9            ESCAPE
F1D9 A6  43             LDA     (IOBYT1-IO),U Must we ismore ESC?
F1DB 85  04             BITA    #ESCBIT
F1DD 26  03             BNE     ESC05   Yes - skip
F1DF 17  0521           LBSR    SWPVEC  Else swap the ESC and OUT vectors
F1E2 39         ESC05   RTS

                * Unused controls processing
F1E3            CTLDEF
F1E3 39                 RTS             Just ismore them


                * Escape sequence processing
F1E4            ESCPNT
F1E4 34  56             PSHS    D,X,U   Save registers
F1E6 CE  E13C           LDU     #IO     Point to the IO variables
F1E9 8E  F90D           LDX     #ESCTBL Search the table for valid chars,
```

```
F1EC 17  0C51          LBSR  TBSRCH   executing the routine
F1EF 17  0511          LBSR  SWPVEC   Then swap the vectors again
F1F2 35  D6            PULS  D,X,U,PC

                       *-----------------------------------------------------------------------
                       * Cursor move routines:
                       *
                       * ESC A - move cursor up one place
                       * ESC B - move cursor down one place
                       * ESC C - move cursor right one place
                       * ESC D - move cursor left one place
                       *
F1F4                   CURSUP
F1F4 A6  54                   LDA   (C_ROW-IO),U Get current row
F1F6 A1  5C                   CMPA  (FW_ROW-IO),U At the top?
F1F8 22  03                   BHI   CRUP05   No - Just dec the current row
F1FA A6  5D                   LDA   (LW_ROW-IO),U Else put the cursor at the bottom
F1FC 4C                       INCA            And correct for the DEC
F1FD 4A                CRUP05 DECA
F1FE 20  0A                   BRA   CRDN10   And go update C_ROW and COL_0
                       *
F200                   CURSDN
F200 A6  54                   LDA   (C_ROW-IO),U Get current row
F202 A1  5D                   CMPA  (LW_ROW-IO),U On the bottom?
F204 25  03                   BLO   CRDN05
F206 A6  5C                   LDA   (FW_ROW-IO),U Else put it on the top
F208 4A                       DECA            Correct for the INC
F209 4C                CRDN05 INCA
F20A A7  54            CRDN10 STA   (C_ROW-IO),U Save it
F20C 16  0310                 LBRA  FIXCOL   Fixup COL_0 and return
                       *
F20F                   CURSRT
F20F A6  55                   LDA   (C_COL-IO),U Get current column
F211 A1  5F                   CMPA  (LW_COL-IO),U At the right?
F213 25  03                   BLO   CRRT05
F215 A6  5E                   LDA   (FW_COL-IO),U Put it at the left edge
F217 4A                       DECA
F218 4C                CRRT05 INCA
F219 20  0A                   BRA   CRLF10   And go update C_COL
                       *
F21B                   CURSLF
F21B A6  55                   LDA   (C_COL-IO),U
F21D A1  5E                   CMPA  (FW_COL-IO),U At the left?
F21F 22  03                   BHI   CRLF05
F221 A6  5F                   LDA   (LW_COL-IO),U Put it at the right
F223 4C                       INCA
F224 4A                CRLF05 DECA
F225 A7  55            CRLF10 STA   (C_COL-IO),U
F227 39                       RTS

                       * ESC F - use second character generator
F228                   ALTGEN
F228 86  80                   LDA   #%10000000 Set up bit mask required
```

```
F22A A7   46            STA    (CHMASK-IO),U
F22C 39                 RTS

                * ESC G - use normal character generator
F22D            NRMGEN
F22D 6F   46            CLR    (CHMASK-IO),U Clear character mask byte
F22F 39                 RTS

                * ESC H - home cursor
F230            HOME
F230 A6   5C            LDA    (FW_ROW-IO),U Just set up row and col pointers
F232 E6   5E            LDB    (FW_COL-IO),U
F234 ED   54            STD    (C_ROW-IO),U Update both pointers
F236 16   02E6          LBRA   FIXCOL    Update the column pointers and return

                * ESC I - reverse line feed
F239            RLFEED
F239 A6   54            LDA    (C_ROW-IO),U Are we on the first row?
F23B A1   5C            CMPA   (FW_ROW-IO),U
F23D 1023 03B2          LBLS   SCRL_D    Yes - scroll down and exit
F241 20   B1            BRA    CURSUP    Else just move the cursor up and return

                * ESC J - clear to end of window
F243            CLREOS
F243 8D   2F            BSR    CLREOL    Clear the top line
F245 A6   5D            LDA    (LW_ROW-IO),U Get number of rows to clear
F247 A0   54            SUBA   (C_ROW-IO),U
F249 34   02            PSHS   A         Save the row count
F24B 27   25            BEQ    CEOS05    None - the clear is finished
F24D A6   54            LDA    (C_ROW-IO),U
F24F 4C                 INCA             Point to next row
F250 E6   59            LDB    (MAXCOL-IO),U
F252 3D                 MUL
F253 E3   50            ADDD   (SC_TOP-IO),U
F255 EB   5E            ADDB   (FW_COL-IO),U
F257 89   00            ADCA   #0        Get start address of next line in X
F259 1F   01            TFR    D,X
F25B 34   10    CEOS15  PSHS   X         Save it for the next row
F25D E6   5F            LDB    (LW_COL-IO),U
F25F E0   5E            SUBB   (FW_COL-IO),U
F261 5C                 INCB             Number of chars to clear
F262 86   FF            LDA    #BLANK
F264 A7   80    CEOS10  STA    0,X+      Clear the line
F266 5A                 DECB
F267 26   FB            BNE    CEOS10
F269 35   10            PULS   X         Restore the line pointer
F26B E6   59            LDB    (MAXCOL-IO),U
F26D 3A                 ABX              Point to the next line
F26E 6A   E4            DEC    0,S       Dec the row counter
F270 26   E9            BNE    CEOS15    And loop till all done
F272 35   82    CEOS05  PULS   A,PC       Then drop the row counter and return

                * ESC K - clear to end of line
```

```
F274                    CLREOL
F274 17  02D4                   LBSR    GCADDR    Get the cursor address in X
F277 E6  5F                     LDB     (LW_COL-IO),U
F279 E0  55                     SUBB    (C_COL-IO),U Get number of chars to clear
F27B 5C                         INCB
F27C B6  FF                     LDA     #BLANK    Blank char
F27E A7  80             CEOL05  STA     0,X+      Clear the line
F280 5A                         DECB
F281 26  FB                     BNE     CEOL05
F283 39                         RTS

                        * ESC Q - swap cursor types
F284                    SWPCUR
F284 EC  C4                     LDD     (CURTYP-IO),U Get cursor types into A and B
F286 1E  89                     EXG     A,B       Exchange them
F288 ED  C4                     STD     (CURTYP-IO),U And resave them
F28A 39                         RTS

                        * ESC Y - cursor positioning
F28B                    ESCCMV
F28B 8E  F2D1                   LDX     #E_CMV1   Routine to process the first byte
F28E 20  2F                     BRA     EMSQEX    Return via multiple sequence exit point

                        * ESC 0 - Set the window limits
F290                    E_SETW
F290 8E  F2FB                   LDX     #E_STW1   Routine to process the fw_row
F293 20  2A                     BRA     EMSQEX    Exit via routine for multiple char sequence

                        * ESC 1 - return the window limits via BINCHE
F295                    E_GETW
F295 34  16                     PSHS    D,X
F297 8E  E185                   LDX     #IO_BFR   All values are returned in the IO buffer
F29A BF  E147                   STX     LB_PTR    Set up the pointer
F29D A6  88 E3          EGW05   LDA     (FW_ROW-IO_BFR),X Get the window char
F2A0 8B  20                     ADDA    #$20      Add the ASCII offset
F2A2 A7  80                     STA     0,X+      And put it in the IO buffer
F2A4 8C  E189                   CMPX    #IO_BFR+4 Are we finished yet?
F2A7 25  F4                     BLO     EGW05     No - do all four
F2A9 6F  84                     CLR     0,X       Mark the buffer end
F2AB 17  03EA                   LBSR    SETVEC    Set up INEVEC and STSVEC appropriately
F2AE 35  96                     PULS    D,X,PC    And return

                        * ESC 2 - save the IO word
F2B0                    E_SAVI
F2B0 EC  42                     LDD     (IOWORD-IO),U
F2B2 FD  E140                   STD     IOSAVE
F2B5 39                         RTS

                        * ESC 3 - restore the IO word from the save location
F2B6                    E_RSTI
F2B6 FC  E140                   LDD     IOSAVE
F2B9 ED  42                     STD     (IOWORD-IO),U
F2BB 39                         RTS
```

```
                        * ESC 4 - set/clear individual bits in the IO word
F2EC                    E_SETI
F2EC 8E   F338                  LDX     #E_STI1     Routine to process the set/reset char
F2EF                    EMSGEX
F2EF BF   E102                  STX     OUTVEC      Save new routine address
F2C2 17   043E                  LBSR    SWPVEC      Negate swap on return
F2C5 39                         RTS

                        * Non-recognised escape sequences
F2C6                    DEFESC
F2C6 32   62                    LEAS    2,S         Drop the TBSRCH return address
F2C8 17   0438                  LBSR    SWPVEC      Unswap the vectors again
F2CB 35   56                    PULS    D,X,U       Restore the registers
F2CD 6E   9F E102               JMP     [OUTVEC]    And output the char normally

                        * First char of cursor move sequence (row+$20)
F2D1                    E_CMV1
F2D1 34   56                    PSHS    D,X,U       Save registers
F2D3 CE   E13C                  LDU     #IO         Point to variables
F2D6 17   01C3                  LBSR    ASCLIM      Remove the ASCII ($20) offset
F2D9 E6   55                    LDB     (C_COL-IO),U Get the column into B
F2DB E0   5E                    SUBB    (FW_COL-IO),U Make it absolute too
F2DD 17   0217                  LBSR    CURSMV      And move the cursor there
F2E0 8E   F2E8                  LDX     #E_CMV2     Set up next vector
F2E3 BF   E102                  STX     OUTVEC
F2E6 35   D6                    PULS    D,X,U,PC    And return

                        * Second char of cursor move sequence (col+$20)
F2E8                    E_CMV2
F2E8 34   76                    PSHS    D,X,Y,U     Save registers
F2EA CE   E13C                  LDU     #IO         Point to variables
F2ED 17   01AC                  LBSR    ASCLIM      Make the ASCII value absolute
F2F0 1F   89                    TFR     A,B         Put it into B for cursor move
F2F2 A6   54                    LDA     (C_ROW-IO),U Get the row into A
F2F4 A0   5C                    SUBA    (FW_ROW-IO),U Make it absolute
F2F6 17   01FE                  LBSR    CURSMV      And move there
F2F9 20   61                    BRA     ES_EXT      Then exit from sequence

                        * Process the four bytes of the SETW sequence
F2FB                    E_STW1
F2FB 34   36                    PSHS    D,X,Y
F2FD 108E E15D                  LDY     #W_HLD1     Where to save the value
F301 8E   F306                  LDX     #E_STW2     The routine to process the second char
F304 20   14                    BRA     E_SW05
F306                    E_STW2
F306 34   36                    PSHS    D,X,Y
F308 108E E15E                  LDY     #W_HLD2     Where to save the value
F30C 8E   F311                  LDX     #E_STW3     The routine to process the third char
F30F 20   09                    BRA     E_SW05
F311                    E_STW3
F311 34   36                    PSHS    D,X,Y
F313 108E E15F                  LDY     #W_HLD3     Where to save the value
```

```
F317 8E  F324          LDX   #E_STW4   The routine to process the final char
F31A 17  017F  E_SW05  LBSR  ASCLIM    Remove the ASCII ($20) offset
F31D A7  A4            STA   0,Y       Save it in the buffer
F31F BF  E102          STX   OUTVEC    Set up the pointer to the next routine
F322 35  B6            PULS  D,X,Y,PC  And return
F324             E_STW4
F324 34  76            PSHS  D,X,Y,U
F326 17  0173          LBSR  ASCLIM    Remove the ASCII offset
F329 B7  E160          STA   W_HLD4    Put into the buffer
F32C BE  E15D          LDX   W_HLD1    Now set the buffered values
F32F 10BE E15F         LDY   W_HLD3
F333 17  0390          LBSR  SETWND    And setup the window limits (does checks)
F336 20  24            BRA   ES_EXT    Then exit from the sequence finally

                 * Process char of set/clear sequence
F338             E_STI1
F338 34  76            PSHS  D,X,Y,U   Save registers
F33A 8E  E13E          LDX   #IOBYT2   Preset X by assuming IO byte 2
F33D 85  08            BITA  #%00001000 Is it IO byte 1 or 2?
F33F 26  02            BNE   ESTI15    IO byte 2 - leave X as is
F341 30  01            LEAX  1,X       Else set IO byte 1 (LS Byte)
F343 34  10    ESTI15  PSHS  X         Save the address on the stack
F345 5F                CLRB
F346 43                COMA            Make bit number 7-0 and set carry
F347 84  07            ANDA  #%00000111 Isolate bit number
F349 56        ESTI05  RORB            Get the bit mask in B
F34A 4A                DECA
F34B 2A  FC            BPL   ESTI05
F34D A6  62            LDA   2,S       Get the char back
F34F 85  10            BITA  #%00010000 Is it set or clear?
F351 26  05            BNE   ESTI10    Set - leave mask as is
F353 53                COMB            Else invert it
F354 E4  F4            ANDB  [0,S]     Clear the bit in the correct byte
F356 20  02            BRA   ESTI20    Then replace it and return
F358             ESTI10
F358 EA  F4            ORB   [0,S]     Set the bit in the appropriate byte
F35A E7  F1    ESTI20  STB   [0,S++]   Replace the byte, then exit the sequence

                 * Multiple character escape sequence exit
F35C             ES_EXT
F35C 17  03A4          LBSR  SWPVEC    Restore the output vector
F35F 8E  F1E4          LDX   #ESCPNT   And reset the escape pointer
F362 BF  E106          STX   ESCVEC
F365 35  F6            PULS  D,X,Y,U,PC


*-----------------------------------------------------------------------
*       Keyboard and buffer input routines
*
*-----------------------------------------------------------------------
* Standard keyboard input routines.  The initialisation code will
*  set up the vectors to point to this code.
*
```

```
                        * Check keyboard status (key pressed).  Note that if buffered input
                        *  is in operation, STSVEC will have been altered to point to BUFSTS.
                        *
F367                    KBDSTS
F367 17   01AC                LBSR   CURSON    Turn the cursor on
F36A                    KBST05
F36A 34   02                  PSHS   A
F36C B6   E025                LDA    PIASA
F36F 85   80                  BITA   #%10000000
F371 35   82                  PULS   A,PC      NE if a key is pressed

F373                    BUFSTS
F373 1C   FB                  CLZ              Return permanent NE status
F375 39                       RTS

                        * Input one character without echo
                        *
F376                    INKBCH
F376 17   019D                LBSR   CURSON
F379                    INKB05
F379 8D   EF                  BSR    KBST05    This routine is not affected by buffering
F37B 27   F9                  BEQ    INKBCH    FIX: BEQ INKB05  27 FC XNOFLY!
F37D B6   E024                LDA    PIADA
F380 39                       RTS

                        * Input one character with echo.  If buffered input is in operation,
                        *  INEVEC will have been altered to point to BINCHE.
                        *
F381                    INKCHE
F381 34   74                  PSHS   B,X,Y,U   Save used registers
F383 CE   E13C                LDU    #IO       Point to IO variables
F386 17   01F4          INKC20 LBSR  INCHR      Get a character without echo
F389 BE   E102                LDX    OUTVEC    Special keys are only allowed if OUTVEC
F38C 8C   F106                CMPX   #CRTOUT   points to CRTOUT - else ismore them
F38F 26   1D                  BNE    INKC10
F391 F6   E13F                LDB    IOBYT1    Are we allowing special keys?
F394 C5   80                  BITB   #BFRBIT
F396 26   06                  BNE    INKC05    No - so check if it's a cursor move key
F398 8E   F964                LDX    #INETBL   Else check if it's an 'active' key
F39B 16   0AAC                LBRA   TBSRCH    Process it if so (it doesn't return)
F39E                    INKC05
F39E E6   43                  LDB    (IOBYT1-IO),U Do we recognise cursor moves?
F3A0 C5   40                  BITB   #CMVBIT
F3A2 26   0A                  BNE    INKC10    No - Just echo the key
F3A4 8E   F96D                LDX    #CMVTBL
F3A7 17   0A96                LBSR   TBSRCH    Else process any cursor movement
F3AA 20   DA                  BRA    INKC20    And loop back for another key
F3AC                    INKC15
F3AC 32   62                  LEAS   2,S       Not a valid cursor key - drop TBSRCH return addres
s
F3AE                    INKC10
F3AE 17   0226                LBSR   OUTCHR    Echo the key
F3B1 35   F4                  PULS   B,X,Y,U,PC Then restore used registers and return
```

```
                        * Replacement for INKCHE when buffered input is in operation.
F3B3                    BINCHE
F3B3 34   10                    PSHS    X
F3B5 BE   E147                  LDX     LB_PTR      Get the buffer pointer
F3B8 A6   80                    LDA     0,X+        Get the char there
F3BA 84   7F                    ANDA    #$7F        Remove the generator selection bit
F3BC 81   0D                    CMPA    #CR         Is it the last char?
F3BE 27   0D                    BEQ     BINC05      Yes - exit after replacing the vectors
F3C0 81   1B                    CMPA    #ESC        ESC is also a possible terminator
F3C2 27   09                    BEQ     BINC05
F3C4 6D   84                    TST     0,X         Is the next char a NUL (also end)?
F3C6 27   05                    BEQ     BINC05      Yes - restore vectors without setting it
F3C8 BF   E147                  STX     LB_PTR      Else just update the pointer
F3CB 35   90                    PULS    X,PC        And return
F3CD                    BINC05
F3CD 17   020B                  LBSR    RSTVEC      Restore the INEVEC and STSVEC vectors
F3D0 35   90                    PULS    X,PC        And return

                        * Enter edit mode, repeating the last buffered line.
F3D2                    RPTLIN
F3D2 8D   42                    BSR     ENTSOE      First output a SOE (not via CRTOUT!)
F3D4 8E   E18A                  LDX     #S_LBUF     Empty the system line buffer
F3D7 20   03                    BRA     RPTL10      Then echo the last line
F3D9 17   FD2A          RPTL15  LBSR    CRTOUT
F3DC A6   80            RPTL10  LDA     0,X+        Now repeat the line
F3DE 81   0D                    CMPA    #CR         End of line yet?
F3E0 27   04                    BEQ     RPTL05      Loop till all done
F3E2 81   1B                    CMPA    #ESC        ESC is also end of line
F3E4 26   F3                    BNE     RPTL15
F3E6 17   FE8B          RPTL05  LBSR    CLREOL      Then clear to end of line...
F3E9 20   02                    BRA     INBF05      ...and enter normal edit mode

                        * Input a buffer into the system buffer as a result of typing
                        *  the EDIT key.  Output a SOE first.
F3EB                    INBUFF
F3EB 8D   29                    BSR     ENTSOE      Output a SOE character
F3ED EC   54            INBF05  LDD     (C_ROW-IO),U Setup the line row & col pointers
F3EF ED   47                    STD     (L_ROW-IO),U
F3F1 17   014D                  LBSR    FIXLCL      Setup LCOL_0
F3F4 17   0186                  LBSR    INCHR       Get a character
F3F7 8E   F943                  LDX     #BUFTBL     Check for valid buffer chars
F3FA 17   0A43                  LBSR    TBSRCH      Process it
F3FD 20   EE                    BRA     INBF05      And loop till all done

                        * Cursor movement key processing routines
                        * ---------------------------------------

                        * The cursor movement routines are elsewhere.

                        * HOME - "Jump" the cursor from start to end of line of non-blanks
F3FF                    HOMLIN
F3FF E6   5E                    LDB     (FW_COL-IO),U
```

```
F401 E1   55              CMPB   (C_COL-IO),U Are we at the start of the line?
F403 26   0E              BNE    HOML05       No - go there
F405 AE   56       HOML10 LDX    (COL_0-IO),U Get COL_0 address
F407 3A                   ABX                 Now step forward to the first BLANK or eol
F408 A6   84              LDA    0,X
F40A 81   FF              CMPA   #BLANK
F40C 27   05              BEQ    HOML05       Found one - stop here
F40E 5C                   INCB                Else step on one
F40F E1   5F              CMPB   (LW_COL-IO),U End of line yet?
F411 25   F2              BLO    HOML10       No - keep looking, else stop
F413 E7   55       HOML05 STB    (C_COL-IO),U
F415 39                   RTS

                   * Buffered input key processing routines
                   * ---------------------------------------

                   * Insert a SOE char at the cursor position
F416               ENTSOE
F416 EC   54              LDD    (C_ROW-IO),U Setup the line pointers
F418 ED   47              STD    (L_ROW-IO),U
F41A 17   0124            LBSR   FIXLCL
F41D 8D   07              BSR    INSPC        Create a space
F41F 86   FE              LDA    #SOE
F421 A7   84              STA    0,X          INSPC returns the cursor address in X
F423 16   FDE9            LBRA   CURSRT       And move past the new SOE and return

                   * Insert a space under the cursor position
F426               INSPC
F426 17   0143            LBSR   GW_BOT       Get the window bottom address
F429 34   06              PSHS   D            Save it for comparisons
F42B 17   00BE     INSP10 LBSR   FWD1CH       Now find the next BLANK char
F42E 81   FF              CMPA   #BLANK       Is it a blank?
F430 27   19              BEQ    INSP05       Yes - we can stop now
F432 AC   E4              CMPX   0,S          Are we at the end of the window?
F434 25   F5              BLO    INSP10       No - step onwards
F436 A6   54              LDA    (C_ROW-IO),U If the cursor is not on the top line...
F438 A1   5C              CMPA   (FW_ROW-IO),U ... then we scroll up one line
F43A 27   0F              BEQ    INSP05       Else we just lose the last char
F43C 17   020D            LBSR   SCRL_U       Scroll the screen up one
F43F 6A   54              DEC    (C_ROW-IO),U Dec the row pointers to allow for the scroll
F441 17   00DB            LBSR   FIXCOL
F444 6A   47              DEC    (L_ROW-IO),U
F446 17   00F8            LBSR   FIXLCL
F449 20   E0              BRA    INSP10       And loop (it will now exit)
                   * The line pointers now point to the last char of the insert block
F44B               INSP05
F44B 17   0117            LBSR   GLADDR       Get the line address
F44E 1F   02              TFR    D,Y          Save it in Y
F450 8D   53       INSP15 BSR    BCK1CH       Go back one char and put it in A
F452 A7   A4              STA    0,Y          And move it
F454 1F   12              TFR    X,Y          Save the new address
F456 EC   47              LDD    (L_ROW-IO),U Find out if we are at the cursor address
F458 10A3 54              CMPD   (C_ROW-IO),U
```

```
F45B 26  F3            BNE   INSF15    No - keep looping
F45D 86  20            LDA   #SPACE    Then put in the final space
F45F A7  84            STA   0,X
F461 35  86            PULS  D,PC      And return, dropping the W_BOT address

                * Delete the char under the cursor
F463            DELCH
F463 17  0106          LBSR  GW_BOT    Get bottom of window address
F466 34  06            PSHS  D         Save it for comparison later
F468 17  00FA  DELC10  LBSR  GLADDR    Get the line pointer address
F46B 1F  02            TFR   D,Y       Save it
F46D BD  4D            BSR   FWD1CH    Step forward one, and fetch the char
F46F A7  A4            STA   0,Y       And save it one back
F471 81  FF            CMPA  #BLANK    Was it the last on the line?
F473 27  08            BEQ   DELC05    Yes - done
F475 AC  E4            CMPX  0,S       Are we at the end of the window?
F477 25  EF            BLO   DELC10    No - keep deleting
F479 86  FF            LDA   #BLANK    Else put in a blank
F47B A7  84            STA   0,X
F47D 35  86    DELC05  PULS  D,PC      And return

                * CR - Accept line entered, from most recent SOE marker, with CR
                *      as the terminating char
F47F            ENTCR
F47F 32  62            LEAS  2,S       Drop the TESRCH return address
F481 17  010B          LBSR  LDBUFR    Load the buffer from the screen
F484 86  0D            LDA   #CR       Put in the terminating CR
F486 20  07            BRA   ENTE05    Put it in and return with the first char

                * ESC - Accept line entered, from most recent SOE marker, with ESC
                *       as the terminating char
F488            ENTESC
F488 32  62            LEAS  2,S       Drop the TESRCH return address
F48A 17  0102          LBSR  LDBUFR    Load the buffer from the screen
F48D 86  1B            LDA   #ESC      Put in the terminating ESC
F48F            ENTE05
F48F A7  A4            STA   0,Y       Put it into the buffer
F491 17  0204          LBSR  SETVEC    Setup INEVEC and STSVEC to for buffered IO
F494 17  FF1C          LBSR  BINCHE    Get the first buffer char...
F497 35  F4            PULS  B,X,Y,U,PC ...and return with it

                * Any other char - Just put it on the screen
F499            DEFBCH
F499 16  FC6A          LBRA  CRTOUT    Print it and then return


                *-----------------------------------------------------------------
                *       General IO service routines
                *

                * Remove the ASCII offset from a byte
F49C            ASCLIM
F49C 81  20            CMPA  #$20      Make it at least $20
```

```
F49E 24 02              BHS   ASCL05
F4A0 86 20              LDA   #$20
F4A2 80 20      ASCL05  SUBA  #$20      Then remove the offset
F4A4 39                 RTS

                * Move line col and row pointers back and forward one place
F4A5            BCK1CH
F4A5 A6 48              LDA   (L_COL-IO),U
F4A7 A1 5E              CMPA  (FW_COL-IO),U Start of window?
F4A9 22 0E              BHI   BCK105    No - Just go back one
F4AB A6 47              LDA   (L_ROW-IO),U Must go up one - are we at the top?
F4AD A1 5C              CMPA  (FW_ROW-IO),U
F4AF 23 1F              BLS   FWD110    Yes - can't go back at all so exit
F4B1 6A 47              DEC   (L_ROW-IO),U Else go up one row
F4E3 17 008B            LBSR  FIXLCL
F4B6 A6 5F              LDA   (LW_COL-IO),U
F4B8 4C                 INCA            Correct for the DEC
F4B9 4A         BCK105  DECA
F4BA 20 14              BRA   FWD110    Go update L_COL and calculate the absolute positio
n

F4BC            FWD1CH
F4BC A6 48              LDA   (L_COL-IO),U
F4BE A1 5F              CMPA  (LW_COL-IO),U End of line?
F4C0 25 0D              BLO   FWD105    No - Just go on one
F4C2 A6 47              LDA   (L_ROW-IO),U Else are we on the last line?
F4C4 A1 5D              CMPA  (LW_ROW-IO),U
F4C6 24 08              BHS   FWD110    Yes - must just exit
F4C8 6C 47              INC   (L_ROW-IO),U Else go down one row
F4CA 8D 75              BSR   FIXLCL
F4CC A6 5E              LDA   (FW_COL-IO),U And step on to first column
F4CE 4A                 DECA
F4CF 4C         FWD105  INCA            Move to next col
F4D0 A7 48      FWD110  STA   (L_COL-IO),U
F4D2 17 0090            LBSR  GLADDR    Get absolute address
F4D5 1F 01              TFR   D,X       X holds line start address
F4D7 A6 84              LDA   0,X       Get the char now at the L position
F4D9 39                 RTS

                * Clear to end of line - user call
F4DA            CEOLIN
F4DA 34 76              PSHS  D,X,Y,U   Save regs
F4DC CE E13C            LDU   #IO
F4DF 17 FD92            LBSR  CLREOL    Clear to line end
F4E2 35 F6              PULS  D,X,Y,U,PC Then return

                * Clear to end of screen - user call
F4E4            CEOSCR
F4E4 34 76              PSHS  D,X,Y,U   Save regs
F4E6 CE E13C            LDU   #IO       Point to IO variables
F4E9 17 FD57            LBSR  CLREOS
F4EC 35 F6              PULS  D,X,Y,U,PC Then return

                * Move cursor - user call
```

```
F4EE                    CSMOVE
F4EE 34   76                    PSHS    D,X,Y,U
F4F0 CE   E13C                  LDU     #IO         Point to IO space
F4F3 8D   02                    BSR     CURSMV      Move the cursor
F4F5 35   F6                    PULS    D,X,Y,U,PC

                        * Move the cursor to row (A) and col (B).
                        *  Note that A, B are binary and not offset, and are addresses within the
                        *  window.
F4F7                    CURSMV
F4F7 AB   5C                    ADDA    (FW_ROW-IO),U Offset into the window
F4F9 EB   5E                    ADDB    (FW_COL-IO),U
F4FB A1   5D                    CMPA    (LW_ROW-IO),U Is A too large?
F4FD 23   02                    BLS     CSMV05      No - skip on
F4FF A6   5D                    LDA     (LW_ROW-IO),U Else replace it with the max
F501 E1   5F            CSMV05  CMPB    (LW_COL-IO),U Repeat for the col
F503 23   02                    BLS     CSMV10
F505 E6   5F                    LDB     (LW_COL-IO),U
F507 ED   54            CSMV10  STD     (C_ROW-IO),U Update the cursor co-ords
F509 20   14                    BRA     FIXCOL      And adjust COL_0 and return


                        * Turn cursor on and off
F50B                    CURSOF
F50B 34   06                    PSHS    D
F50D C6   20                    LDB     #NONDSP     Non-display code
F50F 86   0A            CURS05  LDA     #CURTOP     Cursor top register
F511 FD   E0FE                  STD     CRTC        And set up cursor
F514 35   86                    PULS    D,PC
F516                    CURSON
F516 34   06                    PSHS    D
F518 F6   E13C                  LDB     CURTYP      Get current cursor type
F51B 8D   0F                    BSR     FIXCUR      Update registers
F51D 20   F0                    BRA     CURS05

                        * Update COL_0 pointer
F51F                    FIXCOL
F51F 34   06                    PSHS    D
F521 A6   54                    LDA     (C_ROW-IO),U Use C_ROW, MAXCOL and SC_TOP to
F523 E6   59                    LDB     (MAXCOL-IO),U Find COL_0
F525 3D                         MUL
F526 E3   50                    ADDD    (SC_TOP-IO),U
F528 ED   56                    STD     (COL_0-IO),U
F52A 35   86                    PULS    D,PC

                        * Update cursor address registers
F52C                    FIXCUR
F52C 34   16                    PSHS    D,X
F52E 8D   1B                    BSR     GCADDR      Get the cursor address
F530 34   10                    PSHS    X           Put it on the stack
F532 86   0E                    LDA     #CURREG     Update the CRTC cursor registers
F534 35   04                    PULS    B
F536 FD   E0FE                  STD     CRTC
```

```
F539 4C                 INCA
F53A 35   04            PULS   B
F53C FD   E0FE          STD    CRTC
F53F 35   96            PULS   D,X,PC

                * FIXLCL - Update LCOL_0 according to L_ROW
F541            FIXLCL
F541 A6   47            LDA    (L_ROW-IO),U
F543 E6   59            LDB    (MAXCOL-IO),U
F545 3D                 MUL
F546 E3   50            ADDD   (SC_TOP-IO),U
F548 ED   49            STD    (LCOL_0-IO),U
F54A 39                 RTS

                * Get cursor address
F54B            GCADDR
F54B 34   04            PSHS   B
F54D BE   E132          LDX    COL_0      Use COL_0 and C_COL
F550 F6   E131          LDB    C_COL
F553 3A                 ABX
F554 35   84            PULS   B,PC

                * Get the current state of the IO word into D - user call
F556            GTIOWD
F556 FC   E13E          LDD    IOWORD
F559 39                 RTS

                * Get the window limits - user call
F55A            GTWIND
F55A BE   E138          LDX    FW_ROW
F55D 10BE E13A          LDY    FW_COL     Same format as SETWND
F561 FC   E136          LDD    W_TOP
F564 39                 RTS

                * Get the absolute address of the line row, col position into D
F565            GLADDR
F565 EC   49            LDD    (LCOL_0-IO),U
F567 EB   48            ADDB   (L_COL-IO),U
F569 89   00            ADCA   #0
F56B 39                 RTS

                * Get bottom of window address
F56C            GW_BOT
F56C A6   5D            LDA    (LW_ROW-IO),U
F56E E6   59            LDB    (MAXCOL-IO),U
F570 3D                 MUL
F571 EB   5F            ADDB   (LW_COL-IO),U
F573 89   00            ADCA   #0
F575 E3   50            ADDD   (SC_TOP-IO),U
F577 39                 RTS

                * Home cursor and clear screen - user call
F578            HOMCLR
```

```
F578 BD   0B              BSR    HOMCUR    Home the cursor
F57A 16   FF67            LBRA   CEOSCR    Clear to end of screen and return

                  * Input a char without echo
F57D              INCHR
F57D 6E   9F E100         JMP    [INVEC]

                  * Input a char with echo
F581              INCHRE
F581 6E   9F E104         JMP    [INEVEC]

                  * Home cursor - user call
F585              HOMCUR
F585 34   76              PSHS   D,X,Y,U   Save registers
F587 CE   E13C            LDU    #IO       Point to variables
F58A 17   FCA3            LBSR   HOME      Home the cursor
F58D 35   F6              PULS   D,X,Y,U,PC Then return

                  * Load the buffer from the screen, from the most recent SOE
F58F              LDBUFR
F58F 17   FF79            LBSR   CURSOF    Turn the cursor off
F592 BD   E7              BSR    GCADDR    Get actual address in X
F594 BC   E136    LDBF05  CMPX   W_TOP      At the top of window?
F597 23   0C              BLS    LDBF10    Yes - exit
F599 17   FF09            LBSR   BCK1CH    Go back one char
F59C A6   84              LDA    0,X       Get the char there
F59E 81   FE              CMPA   #SOE      Is it an SOE?
F5A0 26   F2              BNE    LDBF05    No - check if at window top yet
F5A2 17   FF17            LBSR   FWD1CH    Move forward again over the SOE
F5A5 1F   12      LDBF10  TFR    X,Y       Save line pointer address
F5A7 BD   A2              BSR    GCADDR    Get the cursor address
F5A9 34   10              PSHS   X         And put on the stack
F5AB 1F   21              TFR    Y,X       Restore line pointer address
F5AD 108E E18A            LDY    #S_LBUF   Get address of line buffer
F5B1 10BF E147            STY    LB_PTR    Save as line buffer pointer
F5B5 5F                   CLRB             Clear a buffer counter
F5B6 AC   E4      LDBF15  CMPX   0,S       Are we at the cursor address?
F5B8 27   12              BEQ    LDBF20    Yes - exit
F5BA A6   84              LDA    0,X       Get the character
F5BC A7   A0              STA    0,Y+      Put it into the buffer
F5BE 5C                   INCB             Bump up buffer count
F5BF C1   80              CMPB   #BUFLEN   Are we at the max yet?
F5C1 24   09              BHS    LDBF20    Yes - exit
F5C3 34   04              PSHS   B         Else save the counter
F5C5 17   FEF4            LBSR   FWD1CH    Go on 1 char
F5C8 35   04              PULS   B         Restore the counter
F5CA 20   EA              BRA    LDBF15
F5CC 32   62      LDBF20  LEAS   2,S       Drop the cursor address
F5CE 39                   RTS              And return - Y points to the buffer end

                  * Move B bytes from [Y] to [X]
F5CF              MOVBYT
F5CF A6   A0              LDA    0,Y+
```

```
F5D1 A7   80              STA    0,X+
F5D3 5A                   DECB
F5D4 26   F9              BNE    MOVBYT
F5D6 39                   RTS

                  * Output a char
F5D7              OUTCHR
F5D7 6E   9F E102         JMP    [OUTVEC]

                  * Restore the INEVEC and STSVEC vectors if buffered IO was used
F5DB              RSTVEC
F5DB 34   10              PSHS   X
F5DD BE   E104            LDX    INEVEC     Was buffered IO in operation?
F5E0 8C   F3B3            CMPX   #BINCHE
F5E3 26   0C              BNE    RSTV05     No - leave vectors as they are
F5E5 BE   E161            LDX    SV_INE     Else restore INEVEC and STSVEC
F5E8 BF   E104            STX    INEVEC
F5EB BE   E163            LDX    SV_STS
F5EE BF   E108            STX    STSVEC
F5F1 35   90      RSTV05  PULS   X,PC

                  * Scroll window down one line
F5F3              SCRL_D
F5F3 34   20              PSHS   Y
F5F5 A6   5D              LDA    (LW_ROW-IO),U
F5F7 A0   5C              SUBA   (FW_ROW-IO),U Get number of lines to scroll
F5F9 34   02              PSHS   A          Save it
F5FB E6   59              LDB    (MAXCOL-IO),U
F5FD 3D                   MUL
F5FE E3   5A              ADDD   (W_TOP-IO),U
F600 1F   01              TFR    D,X        X now holds start of the last line
F602 E6   5F              LDB    (LW_COL-IO),U
F604 E0   5E              SUBB   (FW_COL-IO),U
F606 5C                   INCB              Get number of chars per line
F607 34   04              PSHS   B          And save that too
F609 3A                   ABX               X now points 1 past end of line
F60A 1F   12              TFR    X,Y        Put it into Y for scroll
F60C 6D   61              TST    1,S        Must we scroll anything?
F60E 27   35              BEQ    SCDN05     No - Just clear the last line
F610 54                   LSRB              Divide by 2 to get word count
F611 34   04      SCDN25  PSHS   B          Save word count
F613 34   04              PSHS   B          And a counter
F615 E6   59              LDB    (MAXCOL-IO),U
F617 50                   NEGB
F618 86   FF              LDA    #-1        D now holds (-MAXCOL)
F61A 83   0002            SUBD   #2         D now holds (-MAXCOL-2)
F61D 34   20              PSHS   Y          Save the row pointer
F61F 6D   62              TST    2,S        Is the word count zero?
F621 27   10              BEQ    SCDN10     Yes - only one char
F623 AE   AB      SCDN15  LDX    D,Y        Copy one line down
F625 AF   A3              STX    0,--Y
F627 6A   62              DEC    2,S
F629 26   F8              BNE    SCDN15
```

```
F62B 5C                    INCB              Make D one more for byte move
F62C A6   64               LDA   4,S         Get char count
F62E 44                    LSRA              Is it odd?
F62F 24   06               BCC   SCDN20      Yes - the line is done
F631 86   FF               LDA   #-1         Else restore D
F633 A6   AB        SCDN10 LDA   D,Y         And move one more byte
F635 A7   A2               STA   0,-Y
F637 86   FF        SCDN20 LDA   #-1         Then restore D again
F639 35   20               PULS  Y           Restore the line pointer
F63B 31   AB               LEAY  D,Y         And update it to next row
F63D 31   21               LEAY  1,Y         Correcting since D is (-MAXCOL-1)
F63F 35   06               PULS  D           Restore the word count
F641 6A   61               DEC   1,S         Finished all rows yet?
F643 26   CC               BNE   SCDN25      No - loop back
F645 35   04        SCDN05 PULS  B           Get the character count
F647 10AE 5A               LDY   (W_TOF-IO),U
F64A 20   43               BRA   SCUF35      Then blank the last line and return

                    * Scroll window up one line
F64C                SCRL_U
F64C 34   20               PSHS  Y
F64E 10AE 5A               LDY   (W_TOF-IO),U Point to top of window
F651 A6   5D               LDA   (W_ROW-IO),U
F653 A0   5C               SUBA  (FW_ROW-IO),U Get number of lines to scroll
F655 34   02               PSHS  A           Save it
F657 E6   5F               LDB   (W_COL-IO),U
F659 E0   5E               SUBB  (FW_COL-IO),U
F65B 5C                    INCB              Get number of chars per line
F65C 34   04               PSHS  B           And save that too
F65E 6D   61               TST   1,S         Must we scroll anything?
F660 27   2B               BEQ   SCUF05      No - Just clear the last line
F662 54                    LSRB              Divide by 2 to get word count
F663 34   04        SCUF25 PSHS  B           Save word count
F665 34   04               PSHS  B           And a counter
F667 E6   59               LDB   (MAXCOL-IO),U
F669 4F                    CLRA              D now holds MAXCOL
F66A 34   20               PSHS  Y           Save the row pointer
F66C 6D   62               TST   2,S         Is the word count zero?
F66E 27   0E               BEQ   SCUP10      Yes - only one char
F670 AE   AB        SCUP15 LDX   D,Y         Copy one line up
F672 AF   A1               STX   0,Y++
F674 6A   62               DEC   2,S
F676 26   F8               BNE   SCUP15
F678 A6   64               LDA   4,S         Get char count
F67A 44                    LSRA              Is it odd?
F67B 24   05               BCC   SCUP20      Yes - the line is done
F67D 4F                    CLRA              Else restore D
F67E A6   AB        SCUP10 LDA   D,Y         And move one more byte
F680 A7   A4               STA   0,Y
F682 4F             SCUP20 CLRA              Then restore D again
F683 35   20               PULS  Y           Restore the line pointer
F685 31   AB               LEAY  D,Y         And update it to next row
F687 35   06               PULS  D           Restore the word count
```

```
F689 6A   61              DEC   1,S       Finished all rows yet?
F68B 26   D6              BNE   SCUF25    No - loop back
F68D 35   04      SCUF05  PULS  B         Get the character count
                  * Blank the last line (also used by SCRL_D)
F68F              SCUF35
F68F 86   FF              LDA   #BLANK
F691 A7   A0      SCUF30  STA   0,Y+
F693 5A                   DECB
F694 26   FB              BNE   SCUF30    Blank out the last line
F696 35   A2              PULS  A,Y,PC    Then return, dropping the row counter

                  * SETVEC - set the INEVEC and STSVEC for buffered input
F698              SETVEC
F698 BE   E104            LDX   INEVEC    Check INEVEC - are we buffering already?
F69B 8C   F3B3            CMPX  #BINCHE
F69E 27   15              BEQ   SETV05    Yes - don't redo it!
F6A0 BF   E161            STX   SV_INE    Else save the old INEVEC
F6A3 8E   F3B3            LDX   #BINCHE   And set it up to BINCHE
F6A6 BF   E104            STX   INEVEC
F6A9 BE   E108            LDX   STSVEC    Also set up STSVEC
F6AC BF   E163            STX   SV_STS    Saving it for later restoration
F6AF 8E   F373            LDX   #BUFSTS
F6B2 BF   E108            STX   STSVEC
F6B5 39           SETV05  RTS

                  * Set the window limits - checking for range
F6B6              SETWND
F6B6 34   30              PSHS  X,Y       Save the limits
F6B8 CE   E13C            LDU   #IO       Set up the U resister for IO pointing
F6BB A6   58              LDA   (MAXROW-IO),U Now check if they are out of range
F6BD A1   61              CMPA  1,S       Check LW_ROW
F6BF 22   03              BHI   SETW05
F6C1 4A                   DECA
F6C2 A7   61              STA   1,S       Set it to MAXROW-1 if too large
F6C4 A6   61      SETW05  LDA   1,S       Now set it as max for FW_ROW
F6C6 A1   E4              CMPA  0,S       Check FW_ROW
F6C8 24   02              BHS   SETW10
F6CA A7   E4              STA   0,S       Set FW_ROW to LW_ROW if it's too large
F6CC A6   59      SETW10  LDA   (MAXCOL-IO),U Now repeat this for the columns
F6CE A1   63              CMPA  3,S       Check LW_COL
F6D0 22   03              BHI   SETW15
F6D2 4A                   DECA            Too large - set it to MAXCOL-1
F6D3 A7   63              STA   3,S
F6D5 A6   63      SETW15  LDA   3,S       Now check FW_COL
F6D7 A1   62              CMPA  2,S
F6D9 24   02              BHS   SETW20
F6DB A7   62              STA   2,S
F6DD 35   30      SETW20  PULS  X,Y       Now restore them and set the limits
F6DF AF   5C              STX   (FW_ROW-IO),U X holds FW_ROW + LW_ROW
F6E1 10AF 5E              STY   (FW_COL-IO),U Y holds FW_COL + LW_COL
F6E4 17   FB49            LBSR  HOME      Put the cursor inside the window
F6E7 A6   5C              LDA   (FW_ROW-IO),U Now calculate W_TOP
F6E9 E6   59              LDB   (MAXCOL-IO),U
```

```
F6EB 3D                     MUL
F6EC EB   5E                ADDB   (FW_COL-IO),U
F6EE 89   00                ADCA   #0
F6F0 E3   50                ADDD   (SC_TOF-IO),U
F6F2 ED   5A                STD    (W_TOF-IO),U
F6F4 39                     RTS                 On return, D holds the absolute window top

                    * Check if a key has been pressed
F6F5                STATUS
F6F5 6E   9F E100           JMP    [STSVEC]

                    * Set the IO word to the contents of D - user call
F6F9                STIOWD
F6F9 FD   E13E              STD    IOWORD    A is IOBYT1, B is IOBYT2
F6FC 39                     RTS

                    * Set window limits - user call
F6FD                STWIND
F6FD 34   70                PSHS   X,Y,U
F6FF 8D   B5                BSR    SETWND
F701 35   F0                PULS   X,Y,U,PC

                    * Swap ESC and OUT vectors
F703                SWPVEC
F703 BE   E106              LDX    ESCVEC
F706 FC   E102              LDD    OUTVEC
F709 FD   E106              STD    ESCVEC
F70C BF   E102              STX    OUTVEC
F70F 39                     RTS


                    *-------------------------------------------------------------------------------
                    *       IO strings and tables
                    *

                    * Shifted character table
                    * ------------------------

F710                SHFTAB
F710 B2C1                   FDB    CHRGEN+(',*16)+1
F712 B3B1                   FDB    CHRGEN+('7*16)+1
F714 B5F1                   FDB    CHRGEN+('_*16)+1
F716 B671                   FDB    CHRGEN+('s*16)+1
F718 B6A1                   FDB    CHRGEN+('J*16)+1
F71A B701                   FDB    CHRGEN+('p*16)+1
F71C B711                   FDB    CHRGEN+('q*16)+1
F71E B791                   FDB    CHRGEN+('y*16)+1
F720 0000                   FDB    0

                    * CRTC Initialisation constants
                    * ---------------------------

F722                CRTTAB
```

```
F722 71          HORTOT  FCB     113
F723 50          HORDIS  FCB     80
F724 5C          HSYPOS  FCB     92
F725 0A          HORWID  FCB     10
F726 1B          VERTOT  FCB     27
F727 04          VERADJ  FCB     4
F728 18          VERDIS  FCB     24
F729 19          VSYPOS  FCB     25
F72A 00          INTERL  FCB     0
F72B 0A          MXSADR  FCB     10
F72C 09          CURSTR  FCB     9
F72D 0A          CUREND  FCB     10
F72E E800        STARTH  FDB     VIDRAM
F730 E800        CURSOR  FDB     VIDRAM

                 * Initial character generator - $21(!) to $7E(~) in compressed form
                 * ------------------------------------------------------------------

F732             CMPRGN
                         OPT     LIS

                 * Initial IO vectors
                 * ------------------

F8D4             VECTBL
F8D4 F376                FDB     INKBCH    invec
F8D6 F106                FDB     CRTOUT    outvec
F8D8 F381                FDB     INKCHE    inevec
F8DA F1E4                FDB     ESCPNT    escvec
F8DC F367                FDB     KBDSTS    stsvec

000A             VTBLEN  EQU     *-VECTBL  Total number of vector bytes

                 * Initial data for IO scratchpad
                 * ------------------------------

F8DE             IODATA
F8DE E800                FDB     VIDRAM    sc_top
F8E0 EF7F                FDB     VIDRAM+24*80-1 sc_bot
F8E2 00                  FCB     0         c_row
F8E3 00                  FCB     0         c_col
F8E4 E800                FDB     VIDRAM    col_0
F8E6 18                  FCB     24        maxrow
F8E7 50                  FCB     80        maxcol
F8E8 E800                FDB     VIDRAM    w_top
F8EA 00                  FCB     0         fw_row
F8EB 17                  FCB     23        lw_row
F8EC 00                  FCB     0         fw_col
F8ED 4F                  FCB     79        lw_col
F8EE 09                  FCB     $9        curtyp
F8EF 69                  FCB     $69       curty2
F8F0 0002                FDB     DEFIOW    iobyt1+iobyt2
F8F2 0002                FDB     DEFIOW    iosave
```

```
F8F4 00                 FCB     0           chmask

            0017  IOBLEN  EQU     *-IODATA    Number of IO initialisation bytes

                  * Valid control characters
                  * ------------------------

F8F5                    CTLTBL
F8F5 0A                 FCB     LF
F8F6 F145               FDB     LFEED       Line feed
F8F8 0D                 FCB     CR
F8F9 F140               FDB     CRET        Carriage return
F8FB 1B                 FCB     ESC
F8FC F1D9               FDB     ESCAPE      Escape
F8FE 0C                 FCB     FF
F8FF F1E5               FDB     FFEED       Formfeed
F901 09                 FCB     HT
F902 F18D               FDB     HTAB        Tab
F904 08                 FCB     BS
F905 F166               FDB     BSPACE      Backspace
F907 07                 FCB     BEL
F908 F1CA               FDB     BELL        Pulse the bell output
F90A 00                 FCB     EOTB
F90B F1E3               FDB     CTLDEF      All other controls

                  * Valid escape sequences
                  * ------------------------

F90D                    ESCTBL
F90D 59                 FCC     'Y'
F90E F28B               FDB     ESCCMV      Cursor position
F910 48                 FCC     'H'
F911 F230               FDB     HOME        Cursor home
F913 4A                 FCC     'J'
F914 F243               FDB     CLREOS      Clear to EOS
F916 4B                 FCC     'K'
F917 F274               FDB     CLREOL      Clear to EOL
F919 49                 FCC     'I'
F91A F239               FDB     RLFEED      Reverse line feed
F91C 41                 FCC     'A'
F91D F1F4               FDB     CURSUP      Cursor up
F91F 42                 FCC     'B'
F920 F200               FDB     CURSDN      Cursor down
F922 43                 FCC     'C'
F923 F20F               FDB     CURSRT      Cursor right
F925 44                 FCC     'D'
F926 F21B               FDB     CURSLF      Cursor left
F928 46                 FCC     'F'
F929 F228               FDB     ALTGEN      Use alternate character generator
F92B 47                 FCC     'G'
F92C F22D               FDB     NRMGEN      Use normal character generator
F92E 51                 FCC     'Q'
F92F F284               FDB     SWPCUR      Swap cursor types
```

```
F931 30                 FCC     '0'
F932 F290               FDB     E_SETW      Set window limits (ASCII offset)
F934 31                 FCC     '1'
F935 F295               FDB     E_GETW      Return current window limits via BINCHE
F937 32                 FCC     '2'
F938 F2B0               FDB     E_SAVI      Save current IO word value
F93A 33                 FCC     '3'
F93B F2B6               FDB     E_RSTI      Restore current IO word value
F93D 34                 FCC     '4'
F93E F2BC               FDB     E_SETI      Set/clear bits in the IO word
F940 00                 FCB     EOTB
F941 F2C6               FDB     DEFESC      All other characters

                * Valid buffer creation characters
                * ----------------------------------------

F943            BUFTBL
F943 C1                 FCB     CUPKEY
F944 F1F4               FDB     CURSUP      Cursor up
F946 C2                 FCB     CDNKEY
F947 F200               FDB     CURSDN      Cursor down
F949 C3                 FCB     CRTKEY
F94A F20F               FDB     CURSRT      Cursor right
F94C C4                 FCB     CLFKEY
F94D F21B               FDB     CURSLF      Cursor left
F94F C8                 FCB     HOMKEY
F950 F3FF               FDB     HOMLIN      Go to start/end of line
F952 8C                 FCB     EDTKEY
F953 F416               FDB     ENTSOE      Enter a SOE char
F955 CD                 FCB     INSKEY
F956 F426               FDB     INSPC       Insert a space
F958 D0                 FCB     DELKEY
F959 F463               FDB     DELCH       Delete a character
F95B 0D                 FCB     CR
F95C F47F               FDB     ENTCR       Enter the buffer, termch = CR
F95E 1B                 FCB     ESC
F95F F488               FDB     ENTESC      Enter the buffer, termch = ESC
F961 00                 FCB     EOTB
F962 F499               FDB     DEFBCH      All other chars

                * Valid 'active' keys in INKCHE
                * ----------------------------------------
F964            INETBL
F964 8C                 FCB     EDTKEY
F965 F3EB               FDB     INBUFF      Enter buffered input mode
F967 CA                 FCB     RPTKEY
F968 F3D2               FDB     RPTLIN      Repeat last entered line
F96A 00                 FCB     EOTB
F96B F39E               FDB     INKC05      All other chars

                * Valid cursor move chars
                * ----------------------------------------
F96D            CMVTBL
```

```
F96D C1                 FCB     CUPKEY
F96E F1F4               FDB     CURSUP  Cursor up
F970 C2                 FCB     CDNKEY
F971 F200               FDB     CURSDN  Cursor down
F973 C3                 FCB     CRTKEY
F974 F20F               FDB     CURSRT  Cursor right
F976 C4                 FCB     CLFKEY
F977 F21B               FDB     CURSLF  Cursor left
F979 00                 FCB     EOTB
F97A F3AC               FDB     INKC15  All others return
```

```
                        *-----------------------------------------------------------------------
                        *       MONITOR CONTROL PROGRAM
                        *       =======================
                        *

                        *-----------------------------------------------------------------------
                        * Cold start entry point - prints signon message, RAM size and
                        * sets up initial user registers on user stack

F97C                    COLDST
F97C 10CE E7FF                  LDS     #MONSTK   Set up the stack pointer
F980 17   F6C5                  LBSR    INITSYS   Initialise the VDU and keyboard PIA
F983 8E   FE7F                  LDX     #SIGNON   Signon message
F986 17   0418                  LBSR    PSTRNG    Print it
F989 B6   E159                  LDA     RAMSIZ    MSB of RAM size
F98C 27   03                    BEQ     COLD05    Zero - skip it
F98E 17   03EC                  LBSR    P1HEX     Else print 1 or 2
F991 B6   E15A          COLD05  LDA     RAMSIZ+1
F994 17   03D9                  LBSR    P2HEX     Print LSB of RAM size (must be >= 16K!)
F997 8E   FEA1                  LDX     #SGNON2   Second half of signon msg
F99A 17   0404                  LBSR    PSTRNG
F99D CE   E7B4                  LDU     #USRSTK   Set up initial user stack
F9A0 C6   0C                    LDB     #12       Clear all the user registers
F9A2 6F   C2            COLD10  CLR     0,-U      The PC will be set up a little later
F9A4 5A                 DECB
F9A5 26   FB                    BNE     COLD10
F9A7 86   80                    LDA     #%10000000 Initial value of CC - E bit set
F9A9 A7   C4                    STA     REGCC,U
F9AB 8E   E165                  LDX     #TRPTBL   Trap table
F9AE C6   20                    LDB     #NTRP*4   4 bytes per trap
F9B0 86   FF                    LDA     ##FF      Address $FFFF is the dummy
F9B2 A7   80            COLD15  STA     0,X+      Clear out the trap table
F9B4 5A                 DECB
F9B5 26   FB                    BNE     COLD15
F9B7 20   0A                    BRA     WARM05    Set up the user PC and enter the command loop

                        *-----------------------------------------------------------------------
                        * Warm start entry point - saves all registers and sets up the stack.
                        *       NB: the user PC is unknown, so we use WARMST instead.
                        *

F9B9                    WARMST
F9B9 32   7E                    LEAS    -2,S      Allow room for a PC
F9BB 34   7F                    PSHS    CC,A,B,DP,X,Y,U Save all registers
F9BD 1F   43                    TFR     S,U       Save user stack
F9BF 10CE E7FF                  LDS     #MONSTK   And set up monitor stack
F9C3                    WARM05
F9C3 8E   F9B9                  LDX     #WARMST   Use WARMST as the default PC (why not?)
F9C6 AF   4A                    STX     REGPC,U
F9C8 20   36                    BRA     CMNDLP    Then enter command loop

                        *-----------------------------------------------------------------------
```

```
                        * SWI entry point - all registers are on the stack automatically
                        *

F9CA                    SWIENT
F9CA 1F   43                    TFR     S,U         Save the user stack pointer
F9CC 10CE E7FF                  LDS     #MONSTK     And set up the monitor stack
F9D0 17   0455                  LBSR    REMTRP      Remove all traps from user code
F9D3 AE   4A                    LDX     REGPC,U     Get stacked PC
F9D5 30   1F                    LEAX    -1,X        Get address of SWI instruction
F9D7 17   0489                  LBSR    TSTTRP      Was it a stored trap?
F9DA 26   10                    BNE     SWIE05      No - it wasn't a trap
F9DC AF   4A                    STX     REGPC,U     Yes - we must re-execute that opcode
F9DE 8E   FEBF                  LDX     #TRAPMS     Print trap message
F9E1 17   03BD                  LBSR    PSTRNG
F9E4 1F   98                    TFR     B,A         Get trap number
F9E6 17   0394                  LBSR    P1HEX       And print it
F9E9 17   03C1                  LBSR    REGDSP      Then display registers
F9EC 20   12            SWIE05  BRA     CMNDLP      Move into the command loop

                        *-------------------------------------------------------------------------------
                        * NMI entry point (used as a break) - all registers are on the stack,
                        *       so we just print a break message and do a register dump.

F9EE                    NMIENT
F9EE 1F   43                    TFR     S,U         Save the user stack
F9F0 10CE E7FF                  LDS     #MONSTK     Set up the monitor stack
F9F4 17   0431                  LBSR    REMTRP      Remove traps from code
F9F7 8E   FEB1                  LDX     #BRKMSG     <BREAK> message
F9FA 17   03A4                  LBSR    PSTRNG
F9FD 17   03AD                  LBSR    REGDSP      And display all registers

                        *
                        * Main command loop
                        *

FA00                    CMNDLP
FA00 8E   FEA9                  LDX     #PROMPT     Prompt string
FA03 17   039B                  LBSR    PSTRNG
FA06 17   0331                  LBSR    INUCHR      Get the uppercase command, echoing
FA09 8E   FEC7                  LDX     #CMDTBL     Table of valid command characters
FA0C 17   0431                  LBSR    TBSRCH      Search table and execute command
FA0F 24   EF                    BCC     CMNDLP      And loop for more if no error
FA11 17   0381                  LBSR    PCRLF       Put error char on a new line
FA14 86   3F                    LDA     #ERRCHR     Print error character
FA16 17   FBBE                  LBSR    OUTCHR
FA19 17   FBBF                  LBSR    RSTVEC      Then dump the rest of the buffer if necessary
FA1C 20   E2                    BRA     CMNDLP      Then return for more

                        *-------------------------------------------------------------------------------
                        *       Monitor command execution routines
                        *
                        *-------------------------------------------------------------------------------
                        * Command execute subroutines
```

```
                        *
                        *-----------------------------------------------------------------------
                        * BCMD - boots up from the minidisks.  Syntax:
                        *                B
                        *

                00E0            SETDF   IOPAGE      References are direct for speed

FA1E                    BCMD
FA1E 34   08                    PSHS    DP          Save current DP
FA20 86   E0                    LDA     #IOPAGE
FA22 1F   8B                    TFR     A,DP        Use direct addressing for the boot
FA24 86   D0                    LDA     #INTCMD     Reset/initialise FDC
FA26 17   0256                  LBSR    FDCCMD      Perform the command
FA29 0F   14                    CLR     DRVREG      Restore the head on drive 0
FA2B 86   09                    LDA     #RSTCMD
FA2D 17   024F                  LBSR    FDCCMD      Perform the command
FA30 8E   C000                  LDX     #BOOTAD     We boot up into $C000
FA33 86   01                    LDA     #1          We boot from track 0, sector 1
FA35 97   1A                    STA     SECREG      (Restore sets TRKREG to 0)
FA37 17   0252                  LBSR    FDSTTL      Let it settle
FA3A 86   8C                    LDA     #RDSCMD     Read sector command
FA3C 97   18                    STA     CMDREG      Issue command
FA3E 17   024B                  LBSR    FDSTTL      Wait for controller to stabilise
FA41 20   04                    BRA     BCMD05      Jump into data read loop

FA43 96   1B            BCMD10  LDA     DATREG      Get the data byte
FA45 A7   80                    STA     0,X+        And save it
FA47 96   18            BCMD05  LDA     STSREG      Get status
FA49 85   02                    BITA    #FDDRQ      Data ready?
FA4B 26   F6                    BNE     BCMD10      Yes - read it in
FA4D 85   01                    BITA    #FDBUSY     Still busy?
FA4F 26   F6                    BNE     BCMD05      Yes - loop till done

FA51 35   08                    PULS    DP          Restore the DP register
FA53 85   1C                    BITA    #%00011100  Check successful completion
FA55 27   02                    BEQ     BCMD15      Ok - Jump to start
FA57 43                         COMA                Show error - set carry
FA58 39                         RTS                 And return
FA59                    BCMD15
FA59 17   FE7F                  LBSR    RSTVEC      Drop any remaining buffered input
FA5C 7E   C000                  JMP     BOOTAD      And so execute the booted code

                                SETDF

                        *-----------------------------------------------------------------------
                        * CCMD - copy a block of memory from one place to another. Syntax:
                        *                C <source start> , <source end> , <dest start> <CR>
                        *

FA5F                    CCMD
FA5F 17   0242                  LBSR    GET3AD      Get the addresses
```

```
FA82 23   32              BLS    CCMD05    C or Z set - error
FA84 34   36              PSHS   D,X,Y     Save all addresses
FA86 1F   20              TFR    Y,D       Put source end into D
FA88 A3   62              SUBD   2,S       Subtract source start to get size
FA8A AC   E4              CMPX   0,S       Is the source start > dest start?
FA8C 22   14              BHI    CCMD10    Yes - we must move downwards
FA8E E3   E4              ADDD   0,S       Else get dest end address into D
FA90 1F   02              TFR    D,Y       And put it in Y
FA92 AE   64              LDX    4,S       Source end address
FA94 31   21              LEAY   1,Y       Correct X and Y for pre-decrement
FA96 30   01              LEAX   1,X
FA98 A6   82        CCMD15 LDA    0,-X      Transfer the block
FA9A A7   A2              STA    0,-Y
FA9C AC   62              CMPX   2,S       At the end yet?
FA9E 22   F8              BHI    CCMD15    No - loop
FAA0 20   11              BRA    CCMD20    Else exit - it's done

FAA2 AE   62        CCMD10 LDX    2,S       Start of source
FAA4 10AE E4              LDY    0,S       Start of destination
FAA7 A6   80        CCMD25 LDA    0,X+      Transfer the block
FAA9 A7   A0              STA    0,Y+
FAAB 30   84              LEAX   0,X       This is a two-byte TSTX instruction
FAAD 27   04              BEQ    CCMD20    Zero - we must have passed the end ($FFFF)
FAAF AC   64              CMPX   4,S       Else are we past the end?
FAB1 23   F4              BLS    CCMD25    No - keep going
FAB3              CCMD20
FAB3 4F                   CLRA             Show no error (clear the carry)
FAB4 35   B6              PULS   D,X,Y,PC  Correct stack and return
FAB6 43             CCMD05 COMA             Show error - sets the carry
FAB7 39                   RTS              And return

                  *-------------------------------------------------------------------------------
                  * DCMD - display memory in hex and ASCII
                  *              D <start address> , <end address> <CR>

FAB8              DCMD
FAB8 17   02C00           LBSR   GET2AD    Get start and end addresses
FABB 25   5E              BCS    DCMD05    Error in entry - just exit
FABD 34   30              PSHS   X,Y       Put start and end addresses on stack
FABF 1F   10              TFR    X,D       Put start into D to manipulate
FAC1 C4   F0              ANDB   #$F0      Start D on a 16-byte boundary
FAC3 1F   01              TFR    D,X       Put it into X to print
FAC5 17   02ED      DCMD25 LBSR   PCRLF     Start a new line
FAC8 17   02B9            LBSR   P4HEX     And print the start address
FACB 17   02DC            LBSR   P2SPC     Print two spaces
FACE 108E E159            LDY    #ASCBUF+16 Address of end of ASCII buffer into Y
FAD2 C6   0F              LDB    #15        Number of bytes per line - 1
FAD4 AC   E4        DCMD20 CMPX   0,S       Are we past start?
FAD6 25   13              BLO    DCMD10    No - print spaces for the byte
FAD8 AC   62              CMPX   2,S       Are we past the end?
FADA 22   0F              BHI    DCMD10    Yes - again print spaces
FADC A6   80              LDA    0,X+      Else get the byte
FADE 17   02AF            LBSR   P2HEX     Print it
```

```
FAC1 84   7F              ANDA   #$7F       Remove the parity
FAC3 81   20              CMPA   #'         Is it a control char?
FAC5 24   0B              BHS    DCMD30     No - we print it as is
FAC7 86   2E              LDA    #'.        Else replace it by a period
FAC9 20   07              BRA    DCMD30     Skip to save char in ASCII buffer
FACB                DCMD10
FACB 30   01              LEAX   1,X        Step X on to the next byte
FACD 17   02BA            LBSR   P2SPC      Print 2 spaces for the byte
FAD0 86   20              LDA    #'         And put a space in the ASCII buffer
FAD2 A7   A2        DCMD30 STA    0,-Y       Put character in the buffer in reverse
FAD4 17   02E5            LBSR   P1SPC      And print a space between bytes
FAD7 17   0135            LBSR   BREAK      Test for break and stop if necessary
FADA 25   1C              BCS    DCMD15     Stop if ESC followed by CR
FADC C5   03              BITB   #%00000011 Every 4 bytes print an extra space
FADE 26   03              BNE    DCMD40
FAE0 17   02A9            LBSR   P1SPC      ..to aid counting
FAE3 5A             DCMD40 DECB             Dec the byte count (16 bytes per line)
FAE4 2A   CE              BPL    DCMD20     Loop till they're done
FAE6 C6   0F              LDB    #15        Then print the ASCII buffer in reverse
FAE8 A6   A5        DCMD35 LDA    B,Y        Get the character
FAEA 17   FAEA            LBSR   OUTCHR     Print it
FAED 5A                   DECB
FAEE 2A   F8              BPL    DCMD35     And loop till it's all done
FAF0 30   84              LEAX   0,X        Have we folded over to zero?
FAF2 27   04              BEQ    DCMD15     Yes - the end address must have been $FFFF
FAF4 AC   62              CMPX   2,S        Else have we passed the end address?
FAF6 23   AD              BLS    DCMD25     No - keep looping
FAF8 32   64        DCMD15 LEAS   4,S        Drop the start and end addresses
FAFA 4F                   CLRA              Clear carry to show no error
FAFB 39             DCMD05 RTS
                    *-----------------------------------------------------------
                    * FCMD - fill memory with a data byte.  If data byte not specified,
                    *       use $00 i.e clear memory.  Syntax:
                    *              F <start adr> , <end adr> [ , <data byte> ] <CR>

FAFC                FCMD
FAFC 17   01A5            LBSR   GET3AD     Get the addresses + data
FAFF 25   0D              BCS    FCMD05     Exit if error
FB01 E7   84        FCMD15 STB    0,X        Put in first location
FB03 E1   80              CMPB   0,X+       Did it store?
FB05 26   07              BNE    FCMD05     No - show error
FB07 34   20              PSHS   Y          Are we at the end yet?
FB09 AC   E1              CMPX   0,S++
FB0B 23   F4              BLS    FCMD15     Loop till we are
FB0D 39                   RTS               Then exit - C is clear from above comparison

FB0E 53             FCMD05 COMB             Show error - set carry
FB0F 39                   RTS

                    *-----------------------------------------------------------
                    * GCMD - go from user PC or given address.  Syntax:
                    *              G [<start address>] <CR>
```

```
FB10                    GCMD
FB10 17  017E           GCMD    LBSR    GET1AD      Get an address, CR terminating
FB13 25  10             GCMD05  BCS     GCMD05      Error in address - just exit
FB15 27  02             GCMD10  BEQ     GCMD10      If no address typed, use the user PC
FB17 AF  4A             GCMD15  STX     REGPC,U     Else save the new adr as the PC
FB19 17  FABF           GCMD10  LBSR    RSTVEC      Dump any remaining input
FB1C 17  0276           GCMD20  LBSR    PCRLF       Skip to a new line
FB1F 17  01F0           GCMD25  LBSR    INSTRP      Install all traps (except the one at PC)
FB22 1F  34             GCMD30  TFR     U,S         Set up the user stack pointer
FB24 3B                 GCMD35  RTI                 And go into routine

                        * The GCMD is not a subroutine.  User programs return to the
                        * monitor via a SWI which comes in elsewhere.

FB25 39                 GCMD05  RTS                 Error return to command loop

                        *-----------------------------------------------------------------------
                        * MCMD - examine/change memory contents. Displays contents, followed
                        *        by - prompt. User can type: 2 hex digits (store & goto next)
                        *        space (goto next), ^ (goto previous), CR (start new line &
                        *        display address, ESC (exit).  Anything else just re-echoes
                        *        current location.  Syntax:
                        *              M <start address> <CR>
                        *

FB26                    MCMD
FB26 17  0168           MCMD    LBSR    GET1AD      Get start address
FB29 23  4E             MCMD05  BLS     MCMD05      Exit if C or Z set (error or no address)
FB2B 1F  12             MCMD06  TFR     X,Y         Save the address
FB2D 17  0265           MCMD10  LBSR    PCRLF       Start on new line
FB30 1F  21             MCMD15  TFR     Y,X
FB32 17  022F           MCMD20  LBSR    P4HEX       Print address
FB35 17  0254           MCMD25  LBSR    P1SPC       Add a space
FB38 A6  A4             MCMD30  LDA     0,Y         Get old value
FB3A 17  0233           MCMD35  LBSR    P2HEX       Print it
FB3D 8E  FEAD           MCMD40  LDX     #HYPHEN     " - " string
FB40 17  025E           MCMD45  LBSR    PSTRNG
FB43 17  019E           MCMD50  LBSR    IN4HEX      Get a new value
FB46 24  0D             MCMD55  BCC     MCMD20      Delimiter was SP, comma or CR - store the value
FB48 81  1B             MCMD60  CMPA    #ESC        Was it an ESC?
FB4A 27  09             MCMD65  BEQ     MCMD20      Also legal
FB4C 81  5E             MCMD70  CMPA    #'^         Move to previous location?
FB4E 26  1F             MCMD75  BNE     MCMD35      Yes - it's ok
FB50 5D                 MCMD80  TSTB                Any numbers entered?
FB51 26  02             MCMD85  BNE     MCMD20      Yes - carry on as normal
FB53 31  3F             MCMD90  LEAY    -1,Y        Else do an extra decrement on the address
FB55 5D                 MCMD20  TSTB                Any chars entered at all?
FB56 1E  10             MCMD21  EXG     X,D         Put the entered value in B, saving A
FB58 27  02             MCMD22  BEQ     MCMD25      No digits (from the TSTB) - don't store
FB5A E7  A4             MCMD23  STB     0,Y         Else store the entered byte and step on
FB5C 1E  10             MCMD25  EXG     X,D         Restore the delimiter to A
FB5E 31  21             MCMD26  LEAY    1,Y         Step to next address
FB60 81  5E             MCMD27  CMPA    #'^         Was the delimiter an ^?
```

```
FB62 26 02          BNE   MCMD30    No - step on
FB64 31 3F          LEAY  -1,Y      Else step back one to the same or previous locatio
n
FB66 81 0D    MCMD30 CMPA  #CR       Was it a CR?
FB68 27 C3          BEQ   MCMD10    Yes - so back and restart loop, printing address
FB6A 81 1B          CMPA  #ESC      Finally, was it an ESC?
FB6C 26 D5          BNE   MCMD15    No - it was a space or comma so just loop
FB6E 39             RTS             Else return with C clear
FB6F          MCMD35
FB6F 86 3F          LDA   #ERRCHR   Show the error
FB71 17 FA63        LBSR  OUTCHR
FB74 17 FA64        LBSR  RSTVEC    Restore the vectors to kill the buffer
FB77 20 B4          BRA   MCMD10    And loop again
FB79          MCMD05
FB79 43             COMA            Show the error
FB7A 39             RTS
```

```
*-----------------------------------------------------------------------------
* QCMD - quick memory test. Syntax:
*                Q <start address> , <end address> <CR>
```

```
FB7B          QCMD
FB7B 17 011D        LBSR  GET2AD    Get the start and end addresses
FB7E 34 30          PSHS  X,Y       Save them (even if error)
FB80 25 35          BCS   QCMD05    Exit if error
FB82 8E DFFF        LDX   #IOBASE-1 Trying to test past the IO page?
FB85 AC 62          CMPX  2,S
FB87 25 2E          BLO   QCMD05    Error exit if so (C is set)
FB89 17 0209        LBSR  PCRLF     Start a new line first
FB8C AE E4    QCMD15 LDX  0,S       Restore the start address
FB8E 1F 98    QCMD25 TFR  B,A       Get the test byte in A (random start value)
FB90 A7 84          STA   0,X       Put into memory
FB92 A8 84          EORA  0,X       Is it the same?
FB94 27 0C          BEQ   QCMD10    Yes - ok
FB96 17 01FC        LBSR  PCRLF     Else start a new line
FB99 17 01C8        LBSR  P4HEX     Print the address in error
FB9C 17 01ED        LBSR  P1SPC     Add a space
FB9F 17 01CE        LBSR  P2HEX     Print the bits different
FBA2 30 01    QCMD10 LEAX 1,X       Step to next location
FBA4 17 00B8        LBSR  BREAK     Test for any key pressed to exit or pause
FBA7 26 0D          BNE   QCMD20    C could be set or clear
FBA9 5C             INCB            Bump up the test byte
FBAA AC 62          CMPX  2,S       At the end yet?
FBAC 23 E0          BLS   QCMD25    No - loop till we are
FBAE 86 2B          LDA   #'+       Print a plus for each pass
FBB0 17 FA24        LBSR  OUTCHR
FBB3 5C             INCB            Make the test byte different
FBB4 20 D6          BRA   QCMD15    And keep looping
FBB6          QCMD20
FBB6 4F             CLRA            Clear carry
FBB7 35 B0    QCMD05 PULS  X,Y,PC    And return, cleaning off the stack
```

```
*-----------------------------------------------------------------------------
* RCMD - examine/change user registers. Syntax:
```

```
                        *           R [<register identifier>] <CR>

FEB9                    RCMD
FEB9 17  017E                LBSR   INUCHR    Get the identifier/CR, echoing
FEBC 8E  FEEE                LDX    #REGTBL   Point to table of valid identifiers
FEBF 17  027E                LBSR   TBSRCH    Search and execute routine
FEC2 39                      RTS               Then return

                        *-------------------------------------------------------------------------
                        * SCMD - performs a call to a user subroutine, returning via an RTS
                        *        instruction.  User PC used if no address given.  Syntax:
                        *              S [<address>] <CR>

FEC3                    SCMD
FEC3 17  00CB                LBSR   GET1AD    Get an address, CR terminating
FEC6 25  27                  BCS    SCMD05    Error in address - just exit
FEC8 26  02                  BNE    SCMD10    If address typed, use it
FECA AE  4A                  LDX    REGPC,U   Else use the PC
FECC 17  FA0C        SCMD10  LBSR   RSTVEC    Dump any remaining buffered input
FECF 17  01C3                LBSR   PCRLF     Skip to a new line
FED2 17  013D                LBSR   INSTRP    Install all traps
FED5 10FF E15B               STS    SAVESP    Save the monitor stack pointer
FED9 AF  4A                  STX    REGPC,U   Setup the subroutine start address
FEDB 1F  34                  TFR    U,S       Setup the user stack pointer again
FEDD 35  7F                  PULS   CC,A,B,DP,X,Y,U Restore user registers
FEDF AD  F4                  JSR    [0,S]     And move in to user subroutine

                        * The user subroutine returns here with a RTS.  Note that the
                        * PC on the stack must not have been corrupted.

FEE1 1A  80                  ORCC   #%10000000 Set E to keep stacking correct
FEE3 34  7F                  PSHS   CC,A,B,DP,X,Y,U Save user registers again
FEE5 1F  43                  TFR    S,U       Save the stack pointer for addressing
FEE7 10FE E15B               LDS    SAVESP    And restore the monitor stack
FEEB 17  023A                LBSR   REMTRP    Remove all traps
FEEE 4F                      CLRA             Show no error in SCMD
FEEF 39              SCMD05  RTS              SCMD done

                        *-------------------------------------------------------------------------
                        * TCMD - set/clear/display traps (breakpoints). Syntax:
                        *              T <CR>              - displays all traps
                        *              T <nn> <CR>         - clears trap n
                        *              T <n> , <addr> <CR> - sets trap n at address
                        *

FEF0                    TCMD
FEF0 17  00F1                LBSR   IN4HEX    Get the trap number
FEF3 25  46                  BCS    TCMD05    Error - exit
FEF5 27  0F                  BEQ    TCMD10    <CR> term - either display or clear trap
FEF7 5D                      TSTB             Else space or comma - must be a number
FEF8 27  41                  BEQ    TCMD05    No digits - error
FEFA 17  0251                LBSR   TRAPAD    Else get the trap address in Y
FEFD 25  3C                  BCS    TCMD05    Illegal number - error exit
```

```
FBFF 17 00BF          LBSR  GET1AD   Get the trap address
FC02 23 37            BLS   TCMD05   Error or no address entered - error
FC04 20 0B            BRA   TCMD25   Else go insert it

FC06 5D         TCMD10 TSTB         Any digits entered?
FC07 27 0B            BEQ   TCMD30   No - display all traps
FC09 17 0242          LBSR  TRAPAD   Get the address in Y
FC0C 25 2D            BCS   TCMD05   Error in number - exit
FC0E 8E FFFF          LDX   #$FFFF   Dummy trap address
FC11 AF A4      TCMD25 STX  0,Y     Then save trap address
FC13 39               RTS           And exit successfully (C is clear)

FC14 108E E165  TCMD30 LDY  #TRPTBL  Trap table
FC18 5F               CLRB          Start with trap #0
FC19 AE A1      TCMD45 LDX  0,Y++    Get trap address
FC1B 8C FFFF          CMPX  #$FFFF   Is it dummy?
FC1E 27 13            BEQ   TCMD40   Yes - skip
FC20 17 0172          LBSR  PCRLF    Else print the trap
FC23 86 54            LDA   #'T      Print in form "Tn aaaa"
FC25 17 F9AF          LBSR  OUTCHR
FC28 1F 98            TFR   B,A      Get trap number
FC2A 17 0150          LBSR  P1HEX    Print trap number
FC2D 17 015C          LBSR  P1SPC    Print a space
FC30 17 0131          LBSR  P4HEX    Print trap address
FC33 31 22      TCMD40 LEAY 2,Y     Step over opcode field and installed flag
FC35 5C               INCB          Step to next
FC36 C1 08            CMPB  #NTRP    At the end yet?
FC38 25 DF            BLO   TCMD45   Loop till all are done
FC3A 39               RTS           Then exit successfully (C is clear)

FC3B 43         TCMD05 COMA         Show an error
FC3C 39               RTS

*-----------------------------------------------------------------------
* XCMD - Jumps to FLEX warm start address. Syntax:
*                X <CR>
*

FC3D            XCMD
FC3D 17 F99B          LBSR  RSTVEC   Drop all remaining input
FC40 16 D0C0          LBRA  FLXWST   FLEX warm start

*-----------------------------------------------------------------------
* NULCMD - no action at all
*

FC43            NULCMD
FC43 4F               CLRA          Show no error
FC44 39               RTS           And return

*-----------------------------------------------------------------------
* Unrecognised command letters all processed here
*
```

```
FC45                    DEFCMD
FC45 43                         COMA            Just set carry to show error
FC46 39                         RTS             Error message will be printed on return

                        *-------------------------------------------------------------------
                        *       General monitor service routines
                        *
                        *-------------------------------------------------------------------
                        * General subroutines
                        *

                        *-------------------------------------------------------------------
                        * ASCBIN - convert ASCII hex digit in A to binary in A. C set if not
                        *          hex.
                        *

FC47                    ASCBIN
FC47 81   30                    CMPA    #'0     Is it < 0?
FC49 25   11                    BLO     ASCB05  Yes - not hex
FC4B 81   39                    CMPA    #'9     Is it > 9?
FC4D 23   0A                    BLS     ASCB10  No - it's valid so convert it
FC4F 81   41                    CMPA    #'A     Is it < A?
FC51 25   09                    BLO     ASCB05  Yes - not hex
FC53 81   46                    CMPA    #'F     Is it > F?
FC55 22   05                    BHI     ASCB05  Yes - not hex
FC57 80   07                    SUBA    #'A-'9-1 Subtract offset between A and 9
FC59 80   30            ASCB10  SUBA    #'0     Then make it binary - C is cleared
FC5B 39                         RTS
FC5C 1A   01            ASCB05  SEC             Error - A is unaltered
FC5E 39                         RTS

                        *-------------------------------------------------------------------
                        * BREAK  - tests for pause in output, i.e. ESC typed.  Waits for next
                        *          ESC, then returns.  If CR typed after first ESC, returns
                        *          with C set.  Returns EQ if no keys pressed or <ESC> <ESC>
                        *          entered, and NE if key typed.

FC5F                    BREAK
FC5F 34   02                    PSHS    A       Save A
FC61 17   FA91                  LBSR    STATUS  Key pressed?
FC64 27   15                    BEQ     BRK05   No - exit with C clear and EQ status
FC66 17   F914                  LBSR    INCHR   Else set the char (without echo!)
FC69 81   1B                    CMPA    #ESC    Is it ESC?
FC6B 26   0E                    BNE     BRK05   No - Just exit with C clear, NE status
FC6D 17   F90D          BRK10   LBSR    INCHR   Else wait for next char
FC70 81   1B                    CMPA    #ESC    Is it also ESC?
FC72 27   07                    BEQ     BRK05   Yes - exit with C clear, EQ status
FC74 81   0D                    CMPA    #CR     Else is it CR?
FC76 26   F5                    BNE     BRK10   No - wait for either of these
FC78 43                         COMA            Else set C to show abort
FC79 35   82                    PULS    A,PC    And return with C set, Z clear
FC7B                    BRK05
```

```
FC7B 1C   FE              CLC             Show no error - leave Z flag as is
FC7D 35   82              PULS   A,PC

                  *-------------------------------------------------------------
                  * FDCCMD - perform a FDC command and wait for completion
                  *

FC7F              FDCCMD
FC7F B7   E018            STA    CMDREG   Issue the command
FC82 8D   08              BSR    FDSTTL   Allow FDC to settle
FC84 B6   E018    FDCC05  LDA    STSREG
FC87 85   01              BITA   #FDBUSY  Wait till its finished
FC89 26   F9              BNE    FDCC05
FC8B 39                   RTS

                  *-------------------------------------------------------------
                  * FDSTTL - waits for +- 30 us to allow FDC to settle
                  *

FC8C              FDSTTL
FC8C 8D   00              BSR    FDST05
FC8E 8D   00      FDST05  BSR    FDST10
FC90 39           FDST10  RTS

                  *-------------------------------------------------------------
                  * GET1AD - gets one 16-bit address from the keyboard, terminated by
                  *          a CR.  Carry is set if the character was invalid, and Z is
                  *          set (EQ) if there were no digits entered (just a CR).  B is
                  *          zero if there were no valid digits before the terminating
                  *          char.  In all cases, A holds the terminating char.
                  *

FC91              GET1AD
FC91 8D   51              BSR    IN4HEX   Get 4 hex digits
FC93 25   04              BCS    G1AD05   Exit if error
FC95 26   02              BNE    G1AD05   Not terminated by CR - error
FC97 5D                   TSTB            Set Z if no digits entered
FC98 39                   RTS             C is clear from IN4HEX
FC99 53           G1AD05  COMB            Show the error
FC9A 39                   RTS             And return

                  *-------------------------------------------------------------
                  * GET2AD - gets two 16-bit addresses from the keyboard, separated by
                  *          a space or comma.  Two addresses must be entered, and if
                  *          the first is larger than the second, they are set equal.
                  *

FC9B              GET2AD
FC9B 8D   27              BSR    IN2AD    Get 2 addresses (does most of the work)
FC9D 25   03              BCS    G2AD05   Error
FC9F 26   01              BNE    G2AD05   No CR terminating - exit
FCA1 39                   RTS             Else just return
FCA2 53           G2AD05  COMB            Show error
```

```
FCA3 39                 RTS

                        *-----------------------------------------------------------------------
                        * GET3AD - gets 3 addresses.  First two as for GET2AD (except
                        *          terminated by space or , or CR) and third as for GET1AD.
                        *          Returned in X, Y and D respectively.

FCA4                    GET3AD
FCA4 8D   1E                    BSR     IN2AD       Get 2 addresses
FCA6 25   14                    BCS     G3AD05      Error - exit
FCA8 27   0D                    BEQ     G3AD15      Term'd by CR ( <adr>,<adr><CR> )
FCAA 34   30                    PSHS    X,Y         Else save them
FCAC 8D   E3                    BSR     GET1AD      Get another address
FCAE 25   0A                    BCS     G3AD10      Error - exit
FCB0 5D                 G3AD10  TSTB                Were there any digits?
FCB1 27   07            G3AD10  BEQ     G3AD05      No (i.e. <adr>,<adr>,<CR> ) - error
FCB3 1F   10            G3AD10  TFR     X,D         Else put into D
FCB5 35   B0            G3AD10  PULS    X,Y,PC      And return with others - C clear, Z clear
FCB7 4F                 G3AD15  CLRA                Set D to zero (clears C, sets Z)
FCB8 5F                 G3AD15  CLRB
FCB9 39                 G3AD15  RTS                 And return - adrs in X and Y, last is 0
FCBA                    G3AD10
FCBA 32   64            G3AD10  LEAS    4,S         Drop first two
FCBC 53                 G3AD05  COMB                Show error and return
FCBD 39                 G3AD05  RTS

                        *-----------------------------------------------------------------------
                        * IN1CH  - inputs one character from the keyboard with echo.
                        *          Strips parity, character returned in A.

FCBE                    IN1CH
FCBE 17   F8C0          IN1CH   LBSR    INCHRE      Get the char
FCC1 84   7F            IN1CH   ANDA    #$7F
FCC3 39                 IN1CH   RTS

                        *-----------------------------------------------------------------------
                        * IN2AD  - gets 2 addresses into X and Y.  If 1st > 2nd, then
                        *          2nd := 1st.  On exit: C set = error, Z set = CR terminated.

FCC4                    IN2AD
FCC4 8D   1E            IN2AD   BSR     IN4HEX      Get address, terminated by space, comma or CR
FCC6 23   1A            IN2AD   BLS     I2AD05      If C set (CS) or Z set (EQ) - exit
FCC8 5D                 IN2AD   TSTB                Were there any digits at all?
FCC9 27   17            IN2AD   BEQ     I2AD05      No - also exit
FCCB 34   10            IN2AD   PSHS    X           Else it's valid - save it
FCCD 8D   15            IN2AD   BSR     IN4HEX      Get the next address
FCCF 34   01            IN2AD   PSHS    CC          Save C and Z (indicates terminator)
FCD1 25   0D            IN2AD   BCS     I2AD10      Invalid hex digit - exit
FCD3 5D                 IN2AD   TSTB                Number of bytes received
FCD4 27   0A            IN2AD   BEQ     I2AD10      No characters entered - exit
FCD6 AC   61            IN2AD   CMPX    1,S         Is it smaller than the last number?
FCD8 24   02            IN2AD   BHS     I2AD15      No - ok
FCDA AE   61            IN2AD   LDX     1,S         Yes - make them equal to the first
```

```
FCDC 1F  12      I2AD15  TFR   X,Y       Put the last address in Y
FCDE 35  91              PULS  CC,X,PC   Get the codes and the first no. and return

FCE0 32  63      I2AD10  LEAS  3,S       Get the first value and CC off the stack
FCE2 53          I2AD05  COMB             Set the carry to show an error
FCE3 39                  RTS              And return

                 *-----------------------------------------------------------------
                 * IN4HEX - inputs a word into X, terminated by space, comma or CR.
                 *          Outputs are: X is entered word, C=1 if error, C=0 if no
                 *          error.  Z=1 (EQ) if terminator is a CR, Z=0 if a space
                 *          or comma.  B=0 if no digits entered before the delim. In
                 *          all cases, A contains the character that terminated the
                 *          number.
                 *

FCE4             IN4HEX
FCE4 6F  E2              CLR   0,-S      Digit counter
FCE6 8E  0000            LDX   #0        Clear X (the entered number)
FCE9 8D  4F      IN4H10  BSR   INUCHR    Get a character
FCEB 17  FF59            LBSR  ASCBIN    Convert it to binary if possible
FCEE 25  17              BCS   IN4H05    Not hex - number is finished
FCF0 34  02              PSHS  A         Else save binary value
FCF2 1F  10              TFR   X,D       Put running total in D
FCF4 58                  ASLB             Move it left by one nibble
FCF5 49                  ROLA
FCF6 58                  ASLB
FCF7 49                  ROLA
FCF8 58                  ASLB
FCF9 49                  ROLA
FCFA 58                  ASLB
FCFB 49                  ROLA
FCFC EB  E0              ADDB  0,S+      And add it new value
FCFE 1F  01              TFR   D,X       And put it back in X
FD00 6C  E4              INC   0,S       Inc the digit counter
FD02 26  E5              BNE   IN4H10    Still less than 256 digits - loop back
FD04 53          IN4H15  COMB             Else give an error
FD05 35  84              PULS  B,PC      And return
FD07             IN4H05
FD07 17  016A            LBSR  VALDL     Was it a valid delimiter?
FD0A 26  F8              BNE   IN4H15    No - error exit
FD0C 81  0D              CMPA  #CR       Was it a CR?
FD0E 1C  FE              CLC             No error - Z is set/cleared by above compare
FD10 35  84              PULS  B,PC      Put the digit count in B and return

                 *-----------------------------------------------------------------
                 * INSTRP - installs all set traps from the trap table.
                 *

FD12             INSTRP
FD12 34  10              PSHS  X         Save the X register
FD14 108E E165           LDY   #TRPTBL   Table of set traps
```

```
FD18 C6   08          LDB   #NTRP     Number of entries
FD1A 86   FF    INST10 LDA  #$FF
FD1C A7   23          STA   3,Y       Mark trap as not installed
FD1E AE   A1          LDX   0,Y++     Get trap address
FD20 8C   FFFF        CMPX  #$FFFF    Is it a dummy?
FD23 27   0E          BEQ   INST05    Yes - skip this entry
FD25 AC   4A          CMPX  REGPC,U   Is it the same as the PC?
FD27 27   0A          BEQ   INST05    Yes - don't install it
FD29 A6   84          LDA   0,X       Else get old instruction
FD2B A7   A4          STA   0,Y       Put into table
FD2D 86   3F          LDA   #SWIINS   Opcode for SWI
FD2F A7   84          STA   0,X       And put into code
FD31 6F   21          CLR   1,Y       Set the installed flag
FD33 31   22    INST05 LEAY 2,Y       Skip past opcode byte and installed flag
FD35 5A               DECB            Dec entry count
FD36 26   E2          BNE   INST10    Not done all yet - loop
FD38 35   90          PULS  X,PC      Else just return

*-----------------------------------------------------------------------
* INUCHR - inputs one character. Lower case alpha
*          characters are converted to upper case.

FD3A                  INUCHR
FD3A 8D   82          BSR   INICH     Get a character
FD3C 81   61          CMPA  #'a       Is it lower case?
FD3E 25   06          BLO   INUC05    No - skip on
FD40 81   7A          CMPA  #'z       Is it a-z?
FD42 22   02          BHI   INUC05    No - skip on
FD44 80   20          SUBA  #('a-'A)  Make it upper case
FD46 39               INUC05 RTS      Then return with it

*-----------------------------------------------------------------------
* LRA    - takes logical address in X, returns physical address
*          in A:X
FD47                  LRA
FD47 34   14          PSHS  B,X       Put logical address on stack
FD49 1F   10          TFR   X,D
FD4B 1F   89          TFR   A,B       A and B hold the MS byte
FD4D 84   0F          ANDA  #%00001111 Isolate the LS nibble
FD4F A7   61          STA   1,S       And replace the stacked MS byte with it
FD51 86   10          LDA   #16       Now shift the MSByte 4 to the left
FD53 3D               MUL             ... into A
FD54 8E   E114        LDX   #DATMAP   And use it to index into the DAT map
FD57 A6   86          LDA   A,X       A now holds the top 6 bits
FD59 88   3F          EORA  #%00111111 Remove the effect of the inverters
FD5B C6   10          LDB   #16       Now do the cunning shift again
FD5D 3D               MUL             A now has the MS 2 bits, B has the MS nibble
FD5E EB   61          ADDB  1,S       Add in the top nibble
FD60 E7   61          STB   1,S       Replace it on the stack
FD62 35   94          PULS  B,X,PC    And return with A:X holding the physical addr

*-----------------------------------------------------------------------
* P4HEX
```

```
                        * P2HEX
                        * P1HEX  - prints 4, 2 or 1 hex digits.  P4HEX prints X register,
                        *          P2HEX prints A and P1HEX prints LS nibble of A.
                        *

FD64                    P4HEX
FD64 34   06                    PSHS    D       Save A and B registers
FD66 1F   10                    TFR     X,D     Value to be printed in D
FD68 8D   06                    BSR     P2HEX   Print A (the MSB)
FD6A 1E   89                    EXG     A,B     Now print the LSB
FD6C 8D   02                    BSR     P2HEX
FD6E 35   86                    PULS    D,PC    Restore A and B and return

FD70                    P2HEX
FD70 34   06                    PSHS    D       Save A and B
FD72 C6   10                    LDB     #16
FD74 3D                         MUL             This is equivalent to LSRA four times
FD75 8D   06                    BSR     P1HEX
FD77 A6   E4                    LDA     0,S     Get A off the stack again
FD79 8D   02                    BSR     P1HEX   Print LS digit
FD7B 35   86                    PULS    D,PC    Restore registers and return

FD7D                    P1HEX
FD7D 84   0F                    ANDA    #%00001111 Isolate LS nibble
FD7F 8B   30                    ADDA    #'0     Make it into hex
FD81 81   39                    CMPA    #'9
FD83 23   02                    BLS     P1HX05  0-9, print it
FD85 8B   07                    ADDA    #'A-'9-1 Else make it A-F
FD87 16   F84D          P1HX05  LBRA    OUTCHR  Then print it and return

                        *-------------------------------------------------------------------
                        * P1SPC
                        * P2SPC  - prints 1 or 2 spaces to the terminal
                        *

FD8A                    P2SPC
FD8A 8D   00                    BSR     P1SPC
FD8C                    P1SPC
FD8C 34   02                    PSHS    A
FD8E 86   20                    LDA     #'      Just print a space
FD90 17   F844                  LBSR    OUTCHR
FD93 35   82                    PULS    A,PC    And return

                        *-------------------------------------------------------------------
                        * PCRLF  - print a carriage return, line feed
                        *

FD95                    PCRLF
FD95 86   0D                    LDA     #CR     Print a CR
FD97 17   F83D                  LBSR    OUTCHR
FD9A 86   0A                    LDA     #LF     Then a LF
FD9C 16   F838                  LBRA    OUTCHR  ...and return afterwards
```

```
                        *-----------------------------------------------------------
                        * FLINE  - Prints a string preceded by a CR/LF
                        * FSTRNG - Prints a string terminated by EOT ($04)
                        *
FD9F                    FLINE
FD9F 8D   F4                    BSR     PCRLF
FDA1                    FSTRNG
FDA1 A6   80                    LDA     0,X+            Get character
FDA3 81   04                    CMPA    #EOT            End of string?
FDA5 27   05                    BEQ     PSTR05
FDA7 17   F82D                  LBSR    OUTCHR          Else print it
FDAA 20   F5                    BRA     PSTRNG
FDAC 39                 PSTR05  RTS

                        *-----------------------------------------------------------
                        * REGDSP - displays all registers
                        *

FDAD                    REGDSP
FDAD 108E FF0C                  LDY     #REGSTR         Table of register ids and offsets
FDB1 8D   E2                    BSR     PCRLF
FDB3 A6   A0            RGDS10  LDA     0,Y+            Get first char of id
FDB5 17   F81F                  LBSR    OUTCHR          Print it
FDB8 A6   A0                    LDA     0,Y+            Get second char of id
FDBA 27   03                    BEQ     RGDS25          If null, skip it
FDBC 17   F818                  LBSR    OUTCHR          Else print it
FDBF 86   3D            RGDS25  LDA     #'=             Followed by an equals
FDC1 17   F813                  LBSR    OUTCHR
FDC4 A6   A0                    LDA     0,Y+            Get offset for register
FDC6 E6   A0                    LDB     0,Y+            Get byte/word flag
FDC8 26   06                    BNE     RGDS05          Word - skip on
FDCA A6   C6                    LDA     A,U             Else set byte to print
FDCC 8D   A2                    BSR     P2HEX           Print the register value
FDCE 20   06                    BRA     RGDS20          Then 1 space and loop
FDD0 2B   08            RGDS05  BMI     RGDS15          Finished table - must just print user SP
FDD2 AE   C6                    LDX     A,U             Else set word to print
FDD4 8D   8E                    BSR     P4HEX           Print it
FDD6 8D   B4            RGDS20  BSR     P1SPC           Followed by 1 space
FDD8 20   D9                    BRA     RGDS10
FDDA 1F   31            RGDS15  TFR     U,X             Get user SP
FDDC 8D   86                    BSR     P4HEX
FDDE 4F                         CLRA                    No error
FDDF 39                         RTS

                        *-----------------------------------------------------------
                        * DEFREG
                        * RxCHNG - Sets the new value of the appropriate register
                        *          according to the entered value.  DEFREG is for illegal
                        *          identifiers.

FDE0                    DEFREG
FDE0 43                         COMA                    Show an error
FDE1 39                         RTS
```

```
FDE2                    RACHNG
FDE2 86   01                    LDA     #REGA       Get register A offset
FDE4 20   1C                    BRA     RCHG05      Go change it
FDE6                    RBCHNG
FDE6 86   02                    LDA     #REGB       Repeat for reg B
FDE8 20   18                    BRA     RCHG05
FDEA                    RCCHNG
FDEA 86   00                    LDA     #REGCC      Repeat for reg CC
FDEC 20   14                    BRA     RCHG05
FDEE                    RDCHNG
FDEE 86   03                    LDA     #REGDP      Repeat for reg DP
FDF0 20   10                    BRA     RCHG05
FDF2                    RXCHNG
FDF2 86   04                    LDA     #REGX       Fetch register X offset
FDF4 20   21                    BRA     RCHG15
FDF6                    RYCHNG
FDF6 86   06                    LDA     #REGY       Fetch register Y
FDF8 20   1D                    BRA     RCHG15
FDFA                    RUCHNG
FDFA 86   08                    LDA     #REGUS      Fetch register US
FDFC 20   19                    BRA     RCHG15
FDFE                    RPCHNG
FDFE 86   0A                    LDA     #REGPC      Fetch register PC
FE00 20   15                    BRA     RCHG15
FE02                    RCHG05
FE02 34   02                    PSHS    A           Save the offset
FE04 17   FEBA                  LBSR    GET1AD      Get the replacement value
FE07 23   1B                    BLS     RCHG10      If EQ or CS, then no good
FE09 1F   10                    TFR     X,D         Put the lower byte in B
FE0B 35   02                    PULS    A
FE0D E7   C6                    STB     A,U         Save new value
FE0F A6   C4                    LDA     REGCC,U
FE11 8A   80                    ORA     #%10000000  Set the E bit in the CC value
FE13 A7   C4                    STA     REGCC,U
FE15 20   0B                    BRA     RCHG20      Then exit successfully
FE17                    RCHG15
FE17 34   02                    PSHS    A           Save the offset
FE19 17   FE75                  LBSR    GET1AD      Get the replacement
FE1C 23   06                    BLS     RCHG10      If EQ or CS, then no good
FE1E 35   02                    PULS    A           Restore offset
FE20 AF   C6                    STX     A,U         Set up new value
FE22                    RCHG20
FE22 4F                         CLRA                No error
FE23 39                         RTS
FE24                    RCHG10
FE24 35   02                    PULS    A           Remove the offset from the stack
FE26 43                         COMA                Show error
FE27 39                         RTS

                        *-----------------------------------------------------------
                        * REMTRP - remove all set traps from memory (in reverse order)
                        *
```

```
FE28                    REMTRP
FE28 108E E184                  LDY     #TRPTBL+(NTRP*4)-1 End of trap table
FE2C C6   08                    LDB     #NTRP       Number of entries
FE2E AE   3D            REMT10  LDX     -3,Y        Get trap address
FE30 6D   A4                    TST     0,Y         Is the installed flag set?
FE32 26   06                    BNE     REMT05      No - skip this trap
FE34 63   A4                    COM     0,Y         Mark it as not installed again
FE36 A6   3F                    LDA     -1,Y        Get opcode
FE38 A7   84                    STA     0,X         And put it back into memory
FE3A 31   3C            REMT05  LEAY    -4,Y        Step back to previous entry
FE3C 5A                 REMT10  DECB                Done all yet?
FE3D 26   EF                    BNE     REMT10      No - loop
FE3F 39                         RTS                 Else return

                        *-------------------------------------------------------------------------
                        * TBSRCH - searches a table for a byte contained in A.  If found,
                        *         jumps to the address following the match - if not found,
                        *         the last address in the table is used.
                        *

FE40                    TBSRCH
FE40 A1   80                    CMPA    0,X+        Does it match?
FE42 27   08                    BEQ     TSCH05      Yes - jump to it
FE44 30   02                    LEAX    2,X         Step over address
FE46 6D   84                    TST     0,X         End of table?
FE48 26   F6                    BNE     TBSRCH      No - carry on
FE4A 30   01                    LEAX    1,X         Else step to default address
FE4C                    TSCH05
FE4C 6E   94                    JMP     [0,X]       Jump to it

                        *-------------------------------------------------------------------------
                        * TRAPAD - The number of the trap is passed in the LS nibble of X,
                        *         and is converted to the appropriate address (in Y) in
                        *         the trap table.
                        *

FE4E                    TRAPAD
FE4E 1F   10                    TFR     X,D         Put trap number into B
FE50 53                         COMB                Invert it (make 0-F become F-0)
FE51 C4   0F                    ANDB    #%00001111 Isolate LS nibble
FE53 C1   08                    CMPB    #NTRP       Now NTRP-F is legal
FE55 25   0B                    BLO     TPAD05      0-(NTRP-1) is illegal - sets carry!
FE57 CA   F0                    ORB     #%11110000 Make it $FF-NTRP - $FF
FE59 108E E161                  LDY     #TRPTBL-4 4 entries per trap
FE5D 31   24            TPAD10  LEAY    4,Y         Now step up the table..
FE5F 5C                         INCB                Due to inversion of B
FE60 26   FB                    BNE     TPAD10      Loop till all done
FE62 39                 TPAD05  RTS                 Exit with C set or clear

                        *-------------------------------------------------------------------------
                        * TSTTRP - tests if the value in X is a trap address - if so, returns
                        *         EQ with trap number in B. NB: does not test if trap was
```

```
                        *       installed - if the address is the same, we assume it was.
                        *

FE63                    TSTTRP
FE63 103E E165                  LDY     #TRPTBL     Table of traps
FE67 5F                         CLRB                Start with trap #0
FE68 AC   A1            TSTT10  CMPX    0,Y++       Is it the same?
FE6A 27   07                    BEQ     TSTT05      Yes - exit with EQ set
FE6C 31   22                    LEAY    2,Y         Step over opcode field and installed flag
FE6E 5C                         INCB                Move to next trap
FE6F C1   07                    CMPB    #NTRP-1     Past the last one yet?
FE71 23   F5                    BLS     TSTT10      No - keep looking
FE73 39                 TSTT05  RTS                 Else return with Z clear (NE)

                        *-------------------------------------------------------------------------------
                        * VALDL  - check if the character in A is a valid delimiter
                        *

FE74                    VALDL
FE74 81   20                    CMPA    #'          Is it a space?
FE76 27   06                    BEQ     VALD05      Yes - OK
FE78 81   2C                    CMPA    #',         Is it a comma?
FE7A 27   02                    BEQ     VALD05      Yes - OK
FE7C 81   0D                    CMPA    #CR         Is it a CR?
FE7E 39                 VALD05  RTS                 EQ means valid, NE is not valid

                        *=============================================================================
                        *       Strings and tables for the monitor
                        *
                        *=============================================================================
                        * Strings and tables
                        *

FE7F 36 38 30 39        SIGNON  FCC     "6809 Microsystem Monitor "
FE83 20 4D 69 63
FE87 72 6F 73 79
FE8B 73 74 65 6D
FE8F 20 4D 6F 6E
FE93 69 74 6F 72
FE97 20
FE98 56 34 2E 30                FCC     'V',VERNUM,'.',REVNUM,':',RELNUM,CR,LF,EOT
FE9C 3A 30 0D 0A
FEA0 04
FEA1 4B 20 52 41        SGNON2  FCC     "K RAM",CR,LF,EOT
FEA5 4D 0D 0A 04

FEA9 0D 0A 3E 04        PROMPT  FCC     CR,LF,">",EOT

FEAD 20 2D 20 04        HYPHEN  FCC     " - ",EOT

FEB1 0D 0A 3C 4E        BRKMSG  FCC     CR,LF,"<NMI Break>",EOT
FEB5 4D 49 20 42
FEB9 72 65 61 6B
```

```
FEBD 3E 04

FEBF 0D 0A 54 72   TRAPMS  FCC    CR,LF,"Trap ",EOT
FEC3 61 70 20 04

                   *-----------------------------------------------------------------------
                   * Table of valid (recognised) command letters
                   *

FEC7               CMDTBL
FEC7 42                   FCB    'B          Boot from disk
FEC8 FA1E                 FDB    BCMD
FECA 43                   FCB    'C          Copy block of memory
FECB FA5F                 FDB    CCMD
FECD 44                   FCB    'D          Display memory
FECE FA98                 FDB    DCMD
FED0 46                   FCB    'F          Fill/clear memory
FED1 FAFC                 FDB    FCMD
FED3 47                   FCB    'G          Go from address
FED4 FB10                 FDB    GCMD
FED6 4D                   FCB    'M          Memory examine/change
FED7 FB26                 FDB    MCMD
FED9 51                   FCB    'Q          Quick test memory
FEDA FB7B                 FDB    QCMD
FEDC 52                   FCB    'R          Register examine/change
FEDD FBB9                 FDB    RCMD
FEDF 53                   FCB    'S          Subroutine call
FEE0 FBC3                 FDB    SCMD
FEE2 54                   FCB    'T          Trap (breakpoint) set/clear/display
FEE3 FBF0                 FDB    TCMD
FEE5 58                   FCB    'X          Jump to FLEX warm start
FEE6 FC3D                 FDB    XCMD
FEE8 0D                   FCB    CR          CR - null command
FEE9 FC43                 FDB    NULCMD
FEEB 00                   FCB    EOTB        All other characters
FEEC FC45                 FDB    DEFCMD

                   *-----------------------------------------------------------------------
                   * Table of valid register identifiers for R command
                   *

FEEE               REGTBL
FEEE 41                   FCB    'A          Accumulator A
FEEF FDE2                 FDB    RACHNG
FEF1 42                   FCB    'B          Accumulator B
FEF2 FDE6                 FDB    RBCHNG
FEF4 43                   FCB    'C          Condition codes
FEF5 FDEA                 FDB    RCCHNG
FEF7 44                   FCB    'D          Direct page
FEF8 FDEE                 FDB    RDCHNG
FEFA 58                   FCB    'X          Index X
FEFB FDF2                 FDB    RXCHNG
FEFD 59                   FCB    'Y          Index Y
```

```
FEFE FDF6                       FDB     RYCHNG
FF00 55                         FCB     'U          Stack U
FF01 FDFA                       FDB     RUCHNG
FF03 50                         FCB     'P          Program counter
FF04 FDFE                       FDB     RPCHNG
FF06 0D                         FCB     CR          Display all regs
FF07 FDAD                       FDB     REGDSP
FF09 00                         FCB     EOTB        Illegal identifier
FF0A FDE0                       FDB     DEFREG

                        *-------------------------------------------------------------------------
                        * Table of register ID strings, the associated register's offset
                        * from U, and whether it is a byte or word
                        *

FF0C                    REGSTR
FF0C 41 00                      FCC     'A',0       ID string (nul byte for single-letter IDs)
FF0E 01 00                      FCB     REGA,BYTE Offset from U + byte/word flag
FF10 42 00                      FCC     'B',0
FF12 02 00                      FCB     REGB,BYTE
FF14 43 43                      FCC     'CC'
FF16 00 00                      FCB     REGCC,BYTE
FF18 44 50                      FCC     'DP'
FF1A 03 00                      FCB     REGDP,BYTE
FF1C 58 00                      FCC     'X',0
FF1E 04 01                      FCB     REGX,WORD
FF20 59 00                      FCC     'Y',0
FF22 06 01                      FCB     REGY,WORD
FF24 55 53                      FCC     'US'
FF26 08 01                      FCB     REGUS,WORD
FF28 50 43                      FCC     'PC'
FF2A 0A 01                      FCB     REGPC,WORD
FF2C 53 50                      FCC     'SP'
FF2E 00 81                      FCB     0,WORD+$80 Negative marks end of table

                        *=========================================================================
                        *       DAT initialisation and vectors
                        *
                        *=========================================================================
                        * Power-up/reset entry point - must be in the top 256 bytes.
                        * Sets up the DAT RAM, the DAT map and the memory bitmap.
                        *


                        * NB: There must be RAM at PBLK 62 ($3E000-$3EFFF) as this must
                        * correspond to the logical IO address (LBLK 14, $E000-$EFFF) if the
                        * system is to work.  If no RAM is present at PBLK 62, the system
                        * will just crash in grand style.  Assuming LBLK 14 is all right,
                        * we can set up the DAT and DAT map.  Initially they are all
                        * set to just "pass the address through" i.e. no translation
                        * is performed ($0000-$FFFF => $30000-$3FFFF in the 256K
                        * system, => $0000-$FFFF in the 64K system).
```

```
FF30                    POWRUP
FF30 8E   E114                  LDX   #DATMAP   Initialise the DAT and DAT map
FF33 86   0F                    LDA   #$0F      Start with LBLK 0
FF35 A7   89 1EDC       PWR15   STA   (DAT-DATMAP),X Put it into the DAT RAM
FF39 A7   80                    STA   0,X+      And into the DAT map
FF3B 4A                         DECA
FF3C 2A   F7                    BPL   PWR15     Repeat for LBLK 0-15 ($0F - $00)
FF3E 86   08                    LDA   #8        Now clear the memory bitmap
FF40 6F   80            PWR10   CLR   0,X+
FF42 4A                         DECA
FF43 26   FB                    BNE   PWR10

                        * Now we check to see whether we have a 64K or 256K system. If
                        * it is 64K, we only scan PBLKs 48-62 for RAM, for a 256K
                        * system we scan PBLKs 0-62 to count up how much RAM is in the
                        * system, and set up the DAT, the DAT map and the memory bitmap
                        * accordingly.

FF45 108E A5A5                  LDY   #TSTPAT   Set up the test pattern
FF49 FE   E100                  LDU   $E100     LBLK 14 is set up - we use it to check
FF4C 10BF E100                  STY   $E100     for foldback every 64K - i.e. a 64K system
FF50 86   31                    LDA   #%00110001 This is the DAT byte for PBLK 14
FF52 B7   FFFE                  STA   DAT+14    This remaps LBLK 14
FF55 BE   E100                  LDX   $E100     Save old contents first
FF58 7F   E100                  CLR   $E100     Now we clear the first byte
FF5B 86   01                    LDA   #%00000001 Reset the DAT to PBLK 62
FF5D B7   FFFE                  STA   DAT+14
FF60 C6   3F                    LDB   #$3F      Preset B with PBLK 0 (for a 256K system)
FF62 10BC E100                  CMPY  $E100     Does it still match?
FF66 27   02                    BEQ   PWR40     Yes - it's a 256K system
FF68 C6   0F                    LDB   #$0F      Else it's a 64K system - start with PBLK 48
FF6A FF   E100          PWR40   STU   $E100     Now restore original contents of PBLK 62
FF6D 86   31                    LDA   #%00110001
FF6F B7   FFFE                  STA   DAT+14
FF72 BF   E100                  STX   $E100     Restore the contents of PBLK 14
FF75 86   01                    LDA   #%00000001 And finally restore the DAT again
FF77 B7   FFFE                  STA   DAT+14

                        * Now we can initialise the DAT and perform RAM sizing

FF7A 8E   FFF0                  LDX   #DAT+0    Point to LBLK 0 in the DAT
FF7D 7F   E159                  CLR   RAMSIZ    Clear the RAM size locations
FF80 7F   E15A                  CLR   RAMSIZ+1
FF83 F7   FFFF          PWR20   STB   DAT+15    Use LBLK 15 as test location
FF86 108E A5A5                  LDY   #TSTPAT   Restore the test pattern to Y
FF8A FE   F100                  LDU   $F100     Save the old data
FF8D 10BF F100                  STY   $F100     Put in the test pattern
FF91 10BC F100                  CMPY  $F100     Is it there?
FF95 26   3C                    BNE   PWR25     No - move onwards
FF97 FF   F100                  STU   $F100     Restore old data
FF9A B6   E15A                  LDA   RAMSIZ+1  Get RAM count
FF9D 8B   04                    ADDA  #4        Add 4K
FF9F 19                         DAA             The RAM count is in BCD
```

```
FFA0 B7  E15A          STA   RAMSIZ+1
FFA3 24  03            BCC   PWR30
FFA5 7C  E159          INC   RAMSIZ     No need to DAA - it only goes to 2
FFA8 CE  E124  PWR30   LDU   #BITMAP    Put the bitmap address in U
FFAB 1F  02            TFR   D,Y        Save D for the bitmap calculation
FFAD C8  3F            EORB  #$3F       Invert the bits
FFAF 54                LSRB
FFB0 54                LSRB
FFB1 54                LSRB             And divide by 8 to get the bitmap byte
FFB2 33  C5            LEAU  B,U        U now holds the byte address
FFB4 1F  20            TFR   Y,D        Restore D
FFB6 4F                CLRA             A will hold the bit mask
FFB7 53                COMB             Invert to get the true block number (sets C)
FFB8 C4  07            ANDB  #%00000111 Isolate the lower 3 bits (bit number)
FFBA 49        PWR35   ROLA             Rotate C to the right place
FFBB 5A                DECB
FFBC 2A  FC            BPL   PWR35
FFBE AA  C4            ORA   0,U        Now set the bit in the bitmap
FFC0 A7  C4            STA   0,U
FFC2 1F  20            TFR   Y,D        Then restore D
FFC4 8C  FFFE          CMPX  #DAT+14    Were we at LBLK 14?
FFC7 27  0A            BEQ   PWR25      Yes - skip since LBLK 14 is already set up
FFC9 C1  01            CMPB  #%00000001 Is this PBLK 62?
FFCB 27  06            BEQ   PWR25      Yes - skip since it must correspond to LBLK 14
FFCD E7  89 E124       STB   (DATMAP-DAT),X Update the DAT map
FFD1 E7  80            STB   0,X+       Update the DAT RAM too
FFD3 5A        PWR25   DECB             Move on to the next PBLK
FFD4 26  AD            BNE   PWR20      If we just did PBLK 62, then stop

                * Now all maps are set up.  We can now clean up, then set
                * the stack pointer and proceed into the monitor.

FFD6 7F  FFFF          CLR   DAT+15     LBLK 15 (the monitor) points to PBLK 63
FFD9 16  F9A0          LBRA  COLDST     And enter the monitor cold start point

                *=================================================================
                * SWI2, SWI3, FIRQ, IRQ and the user vector go via RAM vectors.
                *
FFDC 6E  9F E110 JSWI2V JMP   [SWI2AD]
FFE0 6E  9F E112 JSWI3V JMP   [SWI3AD]
FFE4 6E  9F E10E JFIRQV JMP   [FIRQAD]
FFE8 6E  9F E10C JIRQV  JMP   [IRQAD]
FFEC 6E  9F E10A JUSERV JMP   [USERAD]

                *=================================================================
                * Interrupt and reset vectors : RESET, NMI and SWI are used by the
                * monitor for cold start, <break> and software traps (breakpoints)
                * respectively.  The rest are vectored via RAM locations.
                *

                * Check location counter and re-ORG or give warning as appropriate
```

```
FFF0                    ORG     $FFF0       Vectors are at very top of memory

FFF0 FFEC               FDB     JUSERV      User vector
FFF2 FFE0               FDB     JSWI3V      SWI3
FFF4 FFDC               FDB     JSWI2V      SWI2
FFF6 FFE4               FDB     JFIRQV      FIRQ
FFF8 FFEB               FDB     JIRQV       IRQ
FFFA F9CA               FDB     SWIENT      SWI - re-enter monitor
FFFC F9EE               FDB     NMIENT      NMI - user break
FFFE FF30               FDB     POWRUP      Reset


                        END
```

ERROR(S) DETECTED

SYMBOL TABLE:

ALTGEN	F228	ASCB05	FC5C	ASCB10	FC59	ASCBIN	FC47
ASCBUF	E149	ASCL05	F4A2	ASCLIM	F49C	BCK105	F4B9
BCK1CH	F4A5	BCMD	FA1E	BCMD05	FA47	BCMD10	FA43
BCMD15	FA59	BEL	0007	BELL	F1CA	BFRBIT	0080
BINC05	F3CD	BINCHE	F3B3	BITMAP	E124	BLANK	00FF
BOOTAD	C000	BREAK	FC5F	BRK05	FC7B	BRK10	FC6D
BRKMSG	FEB1	BS	0008	BSBIT	0002	BSP05	F17D
BSP10	F18C	BSP15	F17F	BSPACE	F166	BUFLEN	0080
BUFSTS	F373	BUFTBL	F943	BYTE	0000	CCMD	FA5F
CCMD05	FA96	CCMD10	FA82	CCMD15	FA78	CCMD20	FA93
CCMD25	FA87	CDNKEY	00C2	CEOL05	F27E	CEOL1N	F4DA
CEOS05	F272	CEOS10	F264	CEOS15	F25B	CEOSCR	F4E4
CGLD05	F09F	CGLD10	F0ED	CGLD15	F0EC	CGLD20	F0EA
CGLD25	F0F7	CGLD30	F0BD	CGLD35	F0BF	CGLOAD	F097
CHMASK	E142	CHRGEN	B000	CLFKEY	00C4	CLREOL	F274
CLREOS	F243	CMDREG	E018	CMDTBL	FEC7	CMNDLP	FA00
CMPRGN	F732	CMVBIT	0040	CMVTBL	F96D	CODEST	F000
COLD05	F991	COLD10	F9A2	COLD15	F9B2	COLDST	F97C
COL_0	E132	CR	000D	CRDN05	F209	CRDN10	F20A
CRET	F140	CRLF05	F224	CRLF10	F225	CRRT05	F218
CRTC	E0FE	CRTCON	E0FC	CRTKEY	00C3	CRTO05	F138
CRTO15	F118	CRTO20	F136	CRTOUT	F106	CRUP05	F1FD
CSMOVE	F4EE	CSMV05	F501	CSMV10	F507	CTLBIT	0010
CTLDEF	F1E3	CTL_TBL	F8F5	CUFKEY	00C1	CUREND	F72D
CURREG	000E	CURS05	F50F	CURSDN	F200	CURSLF	F21B
CURSMV	F4F7	CURSOF	F50B	CURSON	F516	CURSOR	F730
CURSRT	F20F	CURSTR	F72C	CURSUP	F1F4	CURTOP	000A
CURTP2	E13D	CURTYP	E13C	C_COL	E131	C_ROW	E130
DAT	FFF0	DATAST	E100	DATMAP	E114	DATREG	E01B
DCMD	FA98	DCMD05	FAFB	DCMD10	FACB	DCMD15	FAF8
DCMD20	FAB4	DCMD25	FAA5	DCMD30	FAD2	DCMD35	FAE8
DCMD40	FAE3	DEFBCH	F499	DEFCMD	FC45	DEFESC	F2C6
DEFIOW	0002	DEFREG	FDE0	DELC05	F47D	DELC10	F468
DELCH	F463	DELKEY	00D0	DRVREG	E014	EDTKEY	008C
EGW05	F29D	EMSDEX	F2BF	ENTCR	F47F	ENTE05	F48F
ENTESC	F488	ENTSDE	F416	EOT	0004	EOTB	0000
ERRCHR	003F	ESC	001B	ESC05	F1E2	ESCAPE	F1D9
ESCBIT	0004	ESCCMV	F28B	ESCPNT	F1E4	ESCTBL	F90D
ESCVEC	E106	ESTI05	F349	ESTI10	F358	ESTI15	F343
ESTI20	F35A	ES_EXT	F35C	E_CMV1	F2D1	E_CMV2	F2E8
E_GETW	F295	E_RSTI	F2B6	E_SAVI	F2B0	E_SETI	F2BC
E_SETW	F290	E_STI1	F338	E_STW1	F2FB	E_STW2	F306
E_STW3	F311	E_STW4	F324	E_SW05	F31A	FCMD	FAFC
FCMD05	FB0E	FCMD15	FB01	FDBUSY	0001	FDC	E014
FDCC05	FC84	FDCCMD	FC7F	FDDRQ	0002	FDST05	FC8E
FDST10	FC90	FDSTTL	FC8C	FF	000C	FFBIT	0001
FFD05	F1BE	FFD10	F1C2	FFEED	F1B5	FIRQAD	E10E
FIXCOL	F51F	FIXCUR	F52C	FIXLCL	F541	FLXWST	CD03
FWD105	F4CF	FWD110	F4D0	FWD1CH	F4BC	FW_COL	E13A
FW_ROW	E138	G1AD05	FC99	G2AD05	FCA2	G3AD05	FCBC
G3AD10	FCBA	G3AD15	FCE7	GAPCNT	0003	GCADDR	F54B
GCMD	FB10	GCMD05	FB25	GCMD10	FB19	GET1AD	FC91
GET2AD	FC9B	GET3AD	FCA4	GLADDR	F565	GTIOWD	F556
GTWIND	F55A	GW_EOT	F56C	HOMCLR	F578	HOMCUR	F585
HOME	F230	HOMKEY	00C8	HOML05	F413	HOML10	F405
HOMLIN	F3FF	HORDIS	F723	HORTOT	F722	HORWID	F725
HSYPOS	F724	HT	0009	HTAB	F18D	HTAB05	F1B4
HTAB10	F1AB	HTAB15	F1A2	HYPHEN	FEAD	I2AD05	FCE2
I2AD10	FCE0	I2AD15	FCDC	IN1CH	FCBE	IN2AD	FCC4
IN4H05	FD07	IN4H10	FCE9	IN4H15	FD04	IN4HEX	FCE4
INBF05	F3ED	INBUFF	F3EB	INCHR	F57D	INCHRE	F581

INCR10	F08C	INETBL	F964	INEVEC	E104	INKB05	F379	INKBCH	F376
INKC05	F39E	INKC10	F3AE	INKC15	F3AC	INKC20	F3B6	INKCHE	F381
INSKEY	00CD	INSP05	F44B	INSP10	F42B	INSP15	F450	INSPC	F426
INST05	FD33	INST10	FD1A	INSTRF	FD12	INTCMD	00D0	INTCRT	F088
INTERL	F72A	INTIO	F064	INTS05	F058	INTSYS	F048	INUC05	FD46
INUCHR	FD3A	INVEC	E100	IO	E13C	IOBASE	E000	IOBLEN	0017
IOBYT1	E13F	IOBYT2	E13E	IODATA	F8DE	IOPAGE	00E0	IOSAVE	E140
IOWORD	E13E	IO_BFR	E185	IRQAD	E10C	JFIRQV	FFE4	JIRQV	FFE8
JSWI2V	FFFC	JSWI3V	FFE0	JUSERV	FFEC	KBDSTS	F367	KEST05	F36A
LB_PTR	E147	LCOL_0	E145	LDBF05	F594	LDBF10	F5A5	LDBF15	F5B6
LDBF20	F5CC	LDBUFR	F58F	LF	000A	LFD10	F165	LFDBIT	0020
LFEED	F145	LRA	FD47	LW_COL	E13B	LW_ROW	E139	L_COL	E144
L_ROW	E143	MAXCOL	E135	MAXROW	E134	MCMD	FB26	MCMD05	FB79
MCMD10	FB2D	MCMD15	FB43	MCMD20	FB55	MCMD25	FB5C	MCMD30	FB66
MCMD35	FB6F	MONSTK	E7FF	MOVBYT	F5CF	MXSADR	F72B	NMIENT	F9EE
NONDSP	0020	NRMGEN	F22D	NTRP	0008	NULCMD	FC43	OUTCHR	F5D7
OUTVEC	E102	P1HEX	FD7D	P1HX05	FD87	P1SPC	FD8C	P2HEX	FD70
P2SPC	FD8A	P4HEX	FD64	PCRLF	FD95	PIA	E024	PIADA	E024
PIADB	E026	PIASA	E025	PIASB	E027	PLINE	FD9F	POWRUP	FF30
PROMPT	FEA9	PSTR05	FDAC	PSTRNG	FDA1	PWR10	FF40	PWR15	FF35
PWR20	FF83	PWR25	FFD3	PWR30	FFA8	PWR35	FFBA	PWR40	FF6A
QCMD	FB7B	QCMD05	FBB7	QCMD10	FBA2	QCMD15	FB8C	QCMD20	FBB6
QCMD25	FB8E	RACHNG	FDE2	RAMSIZ	E159	RBCHNG	FDE6	RCCHNG	FDEA
RCHG05	FE02	RCHG10	FE24	RCHG15	FE17	RCHG20	FE22	RCMD	FBB9
RDCHNG	FDEE	RDSCMD	008C	REGA	0001	REGB	0002	REGCC	0000
REGDP	0003	REGDSP	FDAD	REGPC	000A	REGSTR	FF0C	REGTBL	FEEE
REGUS	0008	REGX	0004	REGY	0006	RELNUM	0030	REMT05	FE3A
REMT10	FE2E	REMTRP	FE28	RESIO	F05D	REVNUM	0030	RGDS05	FDD0
RGDS10	FDE3	RGDS15	FDDA	RGDS20	FDD6	RGDS25	FDBF	RLFEED	F239
RPCHNG	FDFE	RPTKEY	00CA	RPTL05	F3E6	RPTL10	F3DC	RPTL15	F3D9
RPTLIN	F3D2	RSTCMD	0009	RSTV05	F5F1	RSTVEC	F5DB	RUCHNG	FDFA
RXCHNG	FDF2	RYCHNG	FDF6	SAVESP	E15B	SCDN05	F645	SCDN10	F633
SCDN15	F623	SCDN20	F637	SCDN25	F611	SCMD	FBC3	SCMD05	FBEF
SCMD10	FBCC	SCRL_D	F5F3	SCRL_U	F64C	SCUP05	F68D	SCUP10	F67E
SCUP15	F670	SCUP20	F682	SCUP25	F663	SCUP30	F691	SCUP35	F68F
SC_BOT	E12E	SC_TOP	E12C	SECREG	E01A	SETV05	F6B5	SETVEC	F698
SETW05	F6C4	SETW10	F6CC	SETW15	F6D5	SETW20	F6DD	SETWND	F6B6
SGNON2	FEA1	SHFTAB	F710	SIGNON	FE7F	SOE	00FE	SPACE	0020
STARTH	F72E	STATUS	F6F5	STIOWD	F6F9	STSREG	E018	STSVEC	E108
STWIND	F6FD	SV_INE	E161	SV_STS	E163	SWI2AD	E110	SWI3AD	E112
SWIE05	F9EC	SWIENT	F9CA	SWIINS	003F	SWPCUR	F284	SWPVEC	F703
S_LBUF	E19A	TBSRCH	FE40	TCMD	FBF0	TCMD05	FC3B	TCMD10	FC06
TCMD25	FC11	TCMD30	FC14	TCMD40	FC33	TCMD45	FC19	TPAD05	FE62
TPAD10	FE5D	TRAPAD	FE4E	TRAPMS	FEBF	TRKREG	E019	TRPTBL	E165
TSCH05	FE4C	TSTPAT	A5A5	TSTT05	FE73	TSTT10	FE68	TSTTRP	FE63
USERAD	E10A	USRSTK	E7B4	VALD05	FE7E	VALDL	FE74	VBASE	00E0
VECTBL	FBD4	VERADJ	F727	VERDIS	F728	VERNUM	0034	VERTOT	F726
VIDRAM	E800	VSYPOS	F729	VTBLEN	000A	WARM05	F9C3	WARMST	F9B9
WORD	0001	WRPBIT	0008	W_HLD1	E15D	W_HLD2	E15E	W_HLD3	E15F
W_HLD4	E160	W_TOP	E136	XCMD	FC3D				