```
**********************************************************
*                                                        *
*                      B 2 2 0 S I M                     *
*                                                        *
*                 Burroughs 220 Simulator                *
*                                                        *
*   Written by Michael J. Mahon   -   March 21, 2016     *
*                                                        *
* The B220 is a BCD word-oriented computer with 5000     *
* 11-digit words in the following format:                *
*      ___________________________________________       *
*     | S | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |      *
*     |___|___|___|___|___|___|___|___|___|___|___|      *
*                                                        *
* If the sign digit (S) is even, the number is positive, *
* if odd, negative.  If S is 2, the word is interpreted  *
* as five alphanumeric characters.                       *
*                                                        *
* "Partial fields" may be specified within a word by a   *
* 2-digit partial field specification, sL, where s is    *
* the rightmost digit of the field and L is the length,  *
* extending to the left no further than the Sign digit.  *
*                                                        *
* Decimal floating-point data is stored in this format:  *
*                                                        *
*      ___________________________________________       *
*     | S | E | E | M | M | M | M | M | M | M | M |      *
*     |___|___|___|___|___|___|___|___|___|___|___|      *
*                                                        *
* S is the sign of the mantissa, as for fixed-point data.*
*                                                        *
* EE is the excess-50 power of ten.                      *
*                                                        *
* MMMMMMMM is the fractional, normalized mantissa.       *
*                                                        *
* Instructions have the following format:                *
*                                                        *
*      ___________________________________________       *
*     | S | V | V | V | V | O | P | A | D | D | R |      *
*     |___|___|___|___|___|___|___|___|___|___|___|      *
*                                                        *
* If S is odd, ADDR is modified by the B register before *
* use.                                                   *
*                                                        *
* The Variant field (VVVV) has an op-specific format.    *
*                                                        *
* The OP field is the opcode.                            *
*                                                        *
* The ADDR field is the address part of the instruction  *
* which is augmented by B if the Sign digit is odd.      *
*                                                        *
**********************************************************
```

```

         put   B220HISTORY
**********************************************************
*                                                        *
*                       History                          *
*                                                        *
* 03/29/16 - Ran first B220 op--HLT!  BCD address to MEM *
*            address is OK.                              *
*                                                        *
* 03/31/16 - Began implementing B220 front panel display *
*            in 40-column text mode.                     *
*                                                        *
* 04/02/16 - Front panel complete, adding keyboard cntl. *
*                                                        *
* 04/05/16 - Keyboard control complete, adding opcodes.  *
*                                                        *
* 04/11/16 - Refined error handling. Added B220CODE file *
*            loading. Implemented partial field STA/R/B. *
*                                                        *
* 04/12/16 - Added conditional branches, STx, LDR, LDB,  *
*            LSA, CLx, CLL, SRx, IBB, DBB.               *
*            Revised manual (keyboard) control.          *
*                                                        *
* 04/13/16 - Added non-BCD digit checking for addresses. *
*            Improved macros for B220 code assembly.     *
*            Split source into small 'put' files.        *
*                                                        *
* 04/15/16 - Added SLx and tested all shifts.            *
*                                                        *
* 04/18/16 - Added ADD and SUB and variants.             *
*                                                        *
* 04/19/16 - Added ADL, tested ADD, ADA, SUB, SUA, ADL.  *
*                                                        *
* 04/22/16 - Added simple MUL and a faster, byte-shifting*
*            version (currently FMU).                    *
*                                                        *
* 04/26/16 - Added EXT and RND. Added special cases for  *
*            SRT 10 and  SLT 10.                         *
*                                                        *
* 04/27/16 - Added simple version of DIV.                *
*                                                        *
* 04/29/16 - Added CFA, CFR.                             *
*                                                        *
* 05/02/16 - Added BFA, BFR. Made 'compare' subroutine.  *
*                                                        *
* 05/04/16 - Added RTF, DFL, and DLB.  Split B220EXEC.   *
*                                                        *
* 05/09/16 - Added help redisplay. Paginated EXEC1 & 2.  *
*                                                        *
* 05/12/16 - Moved HLT execution to 'fetch'. Looks good! *
*                                                        *
* 05/15/16 - Fixed bug in 'compare'.  Added simple SPO.  *
*                                                        *
* 05/16/16 - Added Z reset command, revised help.        *
*                                                        *
* 05/18/16 - Added PWR command; first disk command.      *
*                                                        *
* 06/02/16 - Added PRD, PRB commands, removed B220CODE   *
*            pre-load hack.                              *
*                                                        *
* 06/07/16 - Moved FP ops to B220EXEC2. Changed Quit to  *
*            go to full text window and reconnect ProDOS.*
*                                                        *
* 06/19/16 - Fixed STR/STB partial field bug.            *
*                                                        *
* 06/24/16 - Changed PWR to truncate preexisting file.   *
*                                                        *
* 07/01/16 - Added FAD, FSU.                           *
*                                                      *
* 07/21/16 - Added FMU.                                *
*                                                      *
* 07/25/16 - Many small JMP --> Bxx space optimizations. *
*            RTF now moves upward! Generalized 'clear'.  *
*                                                      *
* 07/28/16 - Added FDV. Organized shift subroutines.   *
*                                                      *
* 08/22/16 - Modified 'b220asc' table for ) and %.     *
*                                                      *
* 08/27/16 - Fixed LBC bug--hi byte was high by one.   *
*            Fixed SPO: +, form feed, and 'ignore'.    *
*                                                      *
* 09/01/16 - Implemented B220 "tab" in SPO.            *
*                                                      *
* 09/02/16 - Fixed RTF: rB now incremented when NN = 00. *
*                                                      *
* 09/03/16 - Fixed BCH. Was branching on equal.        *
*                                                      *
* 09/05/16 - Fixed IFL, DFL, DLB: if s odd, zeroed s+1.  *
*                                                      *
* 09/09/16 - Added SOR/SOH op and subset of Mag Tape ops.*
*                                                      *
* 09/10/16 - Split PTUNITn into PTRDRn and PTPCHn.     *
*                                                      *
* 09/11/16 - Combined paper tape and mag tape I/O.     *
*                                                      *
* 09/16/16 - Added MRD B-modification.                 *
*                                                      *
* 09/20/16 - Added MPE as NOP.                         *
*                                                      *
* 09/21/16 - Added MLS for SNAP 1E.                    *
*                                                      *
* 09/23/16 - Added IOM (Interrogate Overflow Mode).    *
*                                                      *
* 09/24/16 - Fixed IFL bug: No Ov if hi field posn even. *
*                                                      *
* 11/12/16 - Several small cleanups. ** RELEASED v1.0 ** *
*                                                      *
* 01/16/17 - Moved MEM to top in prep for IOCFG addition.*
*                                                      *
* 01/17/17 - Added I/O configuration editor.           *
*            Restricted PTRDR and PTPCH units to 0 and 1.*
*                                                      *
* 01/25/17 - Integrated I/O Config Editor into B220SIM.  *
*            Fixed MPB bug.                            *
*                                                      *
* 02/01/17 - Added "v1.1" and I/O Config help line.    *
*            ** RELEASED v1.1 **                       *
*                                                      *
* 04/27/17 - Added 'skipincP' to skip P reg increment if *
*            PRB sign 6/7 instruction executed.        *
*                                                      *
* 05/01/17 - Char code matched to CCONV: 04 = ), 10 = (, *
*            27 = $, 32 = ?, 34 = '                    *
*                                                      *
* 06/27/17 - Fixed bug in 'divide', now RTS on overflow. *
*                                                      *
* 08/09/20 - Fixed align & normalization bugs in FAD/FSU.*
*            Fixed post-normalization bug in FDV.      *
*            Kluged KAD as a HLT for rA modification.  *
*            Added "Quit to BASIC" to help lines.      *
*            Cleaned up SUB code.                      *
*                                                      *
* 08/11/20 - Fixed sign logic bugs in CAD/CAA/CSU/CSA.   *
*                                                      *
* 08/12/20 - Fixed rotate bugs in SLA/SLT/SLS.           *
*                                                        *
* 08/13/20 - Preserve rR sign in FDV.                    *
*            Always clear rR in RND.                     *
*            Force rA sign to 0 or 1 in ADL.             *
*            Post-normalize in FAD.                      *
*            Fix FAD result on exponent overflow.        *
*                                                        *
* 08/14/20 - Fix FMU overflow exit state.                *
*                                                        *
* 08/15/20 - Clear rA sign before ovflow check in DIV.   *
*            Clear rA sign if ovflow in FDV.             *
*                                                        *
* 08/16/20 - Force normal zero result in FAD/FSU.        *
*                                                        *
* 08/23/20 - Rewrote SRx to save code!                   *
*                                                        *
* 08/24/20 - Detect EXP Ovflo before mant srT2 in FDV.   *
*            Clear rA EXP on exponent overflow in FDV,   *
*             except when it occurs in ':shrT2'.         *
*            Carry out of mantissa in FAD is not zero.   *
*                                                        *
* 09/01/20 - Changed 'B220msg' to v1.2.                  *
*            Made B220HISTORY a separate PUT file.       *
*            ** RELEASED v1.2 **                         *
*                                                        *
* 09/14/20 - Restarted v2.0 with sim and MEM in Aux mem, *
*             leaving panel, I/O, and display in Main.   *
*            Fixed FMU pending ovflow in FMU & MUL to    *
*             be compatible with SOR mode.               *
*                                                        *
* 09/15/20 - Rewrote ]prd and PWR to use buffered I/O.   *
*                                                        *
* 09/22/20 - Rewrote B220IO to support paper tape.       *
*                                                        *
* 09/26/20 - Paper tape buffered I/O works.              *
*                                                        *
* 10/11/20 - Replaced 'bcdcor' with 'bcd2bin' table.     *
*                                                        *
* 11/23/20 - Integrated B220IO and B220MT for trial.     *
*                                                        *
* 12/04/20 - Version 2.0 now runs REGRESS.OBJ.           *
*            Debugging Mag Tape operations.              *
*                                                        *
* 01/12/21 - Fixed numerous bugs--still testing.         *
*                                                        *
* 01/29/21 - Rewrote 'putwrd' to write buffer only when  *
*            called and buffer is already full.          *
*                                                        *
* 01/30/21 - Runs SNAP1E assembling itself!              *
*                                                        *
* 02/01/21 - Fixed SPO bug when tabbing beyond col. 39.  *
*                                                        *
* 02/10.21 - Fixed MIW bug if new file and buffer EMPTY. *
*                                                        *
* 02/14/21 - Fixed ADDRerr if 4999 is a BUN.             *
*                                                        *
* 02/15/21 - Runs BALGOL (BAC 220) correctly!            *
*                                                        *
* 02/17/21 - Reroute PWR to SPO if filename is "SPO".    *
*                                                        *
* 02/23/21 - Added 220 trace routines & instruction ctr. *
*            Changed INIT to allow COMMON > one page.    *
*                                                        *
* 02/27/21 - Added I/O device activity indicators.       *
*                                                        *
* 02/28/21 - Saved v2.0 to history directory.            *
*                                                         *
* 03/23/21 - Began integration of VIEW (DEC 340 display  *
*            simulation as implemented at Caltech).      *
*                                                         *
* 03/29/21 - Fixed addressing of 'xl' and 'yl' in VIEW.  *
*                                                         *
* 04/12/21 - VIEW integration complete, with "light pen" *
*            simulation.                                 *
*                                                         *
* 04/13/21 - Replaced "lup" computation of BCDTBL with   *
*            pre-computed table to save assembly time.   *
*            (Regenerate with BCDT if MEM changes.)      *
*                                                         *
* 05/27/21 - Caltech CRT keyboard implemented, with      *
*            partial (reverse-engineered) key table.     *
*                                                         *
*            All Caltech extensions are now complete.    *
*                                                         *
* 06/08/21 - Fixed bug in ]prd: if sign = 7, B-modify    *
*            not needed; fetch will do it.               *
*                                                         *
* 06/09/21 - Fixed bug in VIEW: when displaying, don't   *
*            process a key if in CRT keyboard mode.      *
*                                                         *
* 06/12/21 - Added option to not erase plotted points    *
*            while Closed-Apple key (PB1) is depressed.  *
*                                                         *
* 07/24/21 - Added SPO charset and sound options to      *
*            I/O Configuration screen.                   *
*                                                         *
***********************************************************
```

```
         put   B220MAP
*                 B220SIM  Memory Map
*
*           MAIN Memory        AUX Memory
* $0000 +=================+=================+ $0000
*       |    Page Zero    |                 |
* $0100 +-----------------+                 |
*       |      Stack      |                 |
* $0200 +-----------------+                 |
*       |Input/Cmd Buffer |    (unused)     |
* $0300 +-----------------+                 |
*       |    (unused)     |                 |
* $0400 +-----------------+                 |
*       |   Text Page 1   |                 |
* $0800 +-----------------+-----------------+ $0800
*       |   Common Code   |   Common Code   |
* $0928 +-----------------+-----------------+ $0928
*       |      INIT       |      FETCH      |
*       |      KEYB       |      EXEC1      |
*       |      PANEL      |      EXEC2      |
*       |      I/O        |       MT        |
*       |      PDOS       |      BCDTBL     |
*       |      TRACE      |                 |
*       |      VIEW       +-----------------+ $1CAF
* $2000 +-----------------+                 |
*       |                 |                 |
*       | VIEW map tables |                 |
*       |                 |                 |
* $3880 +-----------------+                 |
*       |    (unused)     |    (unused)     |
* $3B4C +-----------------+                 |
*       |   Paper Tape    |                 |
*       | Reader Buffers  |                 |
* $4000 +-----------------+                 |
*       |      HGR2       +-----------------+ $4AD0
*       |                 |                 |
*       |                 |                 |
* $6000 +-----------------+                 |
*       |  Punch Buffers  |                 |
* $64B4 +-----------------+                 |
*       |                 |                 |
*       |     Mag-Tape    |    5000 Word    |
*       |     Buffers     |   B220 Memory   |
*       |                 |                 |
* $9600 +-----------------+                 |
*       |                 |                 |
*       |                 |                 |
*       |  BASIC.SYSTEM   |                 |
*       |                 |                 |
*       |                 |                 |
* $BF00 +-----------------+                 |
*       |ProDOS Global Pag|                 |
* $C000 +-----------------+-----------------+ $C000
*       |                                   |
*       |             I/O Pages             |
*       |                                   |
* $D000 +-----------------+-----------------+ $D000
*       |                 |                 |
*       |                 |                 |
*       |       ROM       |     ProDOS      |
*       |                 |                 |
*       |                 |                 |
* $FFFF +-----------------+-----------------+ $FFFF
```

```
         use   B220DEFS
* 6502 equates

BCSop    equ   $B0        ; BCS opcode
BNEop    equ   $D0        ; BNE opcode
BPLop    equ   $10        ; BPL opcode
CLCop    equ   $18        ; CLC opcode
SECop    equ   $38        ; SEC opcode
NOPop    equ   $EA        ; NOP opcode
ADCZop   equ   $65        ; ADC zp opcode
BITZop   equ   $24        ; BIT zp opcode
CMPIop   equ   $C9        ; CMP # opcode
LDAIop   equ   $A9        ; LDA # opcode
SBCZop   equ   $E5        ; SBC zp opcode
ADCYop   equ   $79        ; ADC aaaa,y opcode
SBCYop   equ   $F9        ; SBC aaaa,y opcode

* Apple equates

WNDTOP   equ   $22        ; Top line of text window
CH       equ   $24        ; COUT horizontal cursor
BASL     equ   $28        ; Screen base address
IN       equ   $200       ; Keyboard input buffer
VIEWhlp1 equ   $A50       ; Line 20 of mixed HGR2 display
VIEWhlp2 equ   $AD0       ; Line 21 of mixed HGR2 display
VIEWhlp3 equ   $B50       ; Line 22 of mixed HGR2 display
VIEWhlp4 equ   $BD0       ; Line 23 of mixed HGR2 display
KBD      equ   $C000      ; Keyboard port
READMAIN equ   $C002      ; Store to read Main
READAUX  equ   $C003      ; Store to read Aux
WRITMAIN equ   $C004      ; Store to write Main
WRITAUX  equ   $C005      ; Store to write Aux
ALTCHAR  equ   $C00F      ; Store to enable alt charset
KBSTROBE equ   $C010      ; Keyboard strobe reset
CASSOUT  equ   $C020      ; Toggle cassette out
SPKR     equ   $C030      ; Toggle speaker
TEXT     equ   $C050      ; Text mode softswitch
MIXED    equ   $C052      ; Mixed mode softswitch
PAGE2    equ   $C054      ; Page 2 softswitch
HIRES    equ   $C056      ; High-res mode softswitch
PB0      equ   $C061      ; Pushbutton 0 (Open-Apple)
PB1      equ   $C062      ; Pushbutton 1 (Closed-Apple)
PDL      equ   $C064      ; Paddle inputs (0..3)
PTRIG    equ   $C070      ; Paddle timer trigger

OFF      equ   0          ; Softswitch off
ON       equ   1          ; Softswitch on

* Apple entry points

DOSCON   equ   $3D0       ; ProDOS reconnect vector
DOSCMD   equ   $BE03      ; BASIC.SYSTEM PDOS command
BSSTATE  equ   $BE42      ; BASIC.SYSTEM state var
PRBL2    equ   $F94A      ; Print (X) blanks
BASCALC  equ   $FBC1      ; Set BASL to line (A)
BEEP     equ   $FBDD      ; Beep
HOME     equ   $FC58      ; Clear screen
CROUT    equ   $FD8E      ; Output a CR
COUT     equ   $FDED      ; Output char in A
```

```
* Simulation constants

memb     equ   5000*6     ; 5000 6-byte B220 words
MEM      equ   $C000-memb ; Simulated B220 memory in Aux
ndb      equ   6          ; Number of Device Blocks
dispcnt  equ   100        ; Update panel every 100 instrs
nokey    equ   $AF        ; crtkey "empty" code.

* Buffered I/O flag byte definitions

EOF      equ   $EF        ; End-Of-File flag byte
EMPTY    equ   $EE        ; Empty buffer flag byte
EOB      equ   $EB        ; End-Of-Buffer flag byte
PREF     equ   $B0        ; Block prefix sign flag
```

```
               *********************************************************
               *                                                       *
               *                 Page zero variables                   *
               *                                                       *
               *********************************************************


                        dum   $90         ; Start of Page Zero variables

               * B220 memory fields

               S        equ   0           ; Sign digit
               sL       equ   1           ; rC sL specifier
               VV       equ   2           ; rC Variant
               OP       equ   3           ; rC Op code
               ADDR     equ   4           ; rC BCD address
               EXP      equ   1           ; FP exponent
               MANT     equ   2           ; FP mantissa

               * Simulated B220 State Variables

               B220strt equ   *           ; Start of simulated B220 state
0090: 00 00 00 rBx      ds    4           ; 4 const zero byte prefix to rB
0094: 00 00    rB       dw    0           ; BCD B register
0096: 00 00    rP       dw    0           ; BCD P register
0098: 00 00 00 rC       ds    6           ; BCD Control (instruction) reg
009E: 00 00 00 rA       ds    6           ; BCD A register
00A4: 00 00 00 rR       ds    6           ; BCD R register
00AA: 00 00 00 rD       ds    6           ; BCD D register
00B0: 00 00 00 rD10     ds    6           ; BCD D10 reg (rD * 10)
00B6: 00 00 00 CSW      ds    10          ; Control switches (0=off)
00C0: 00       RUN      db    0           ; RUN mode/indicator (0=off)
00C1: 00       ERR      db    0           ; ERR indicator (0=off)
00C2: 00       COMP     db    0           ; Compare lo,eql,hi (<0,0,>0)
00C3: 00       Ov       db    0           ; Overflow indicator (0=off)
00C4: 00       Rp       db    0           ; Repeat indicator (0=off)
00C5: 00       newp     db    0           ; "P changed manually" indicator
00C6: 00       skipincP db    0           ; Skip incP if PRB sign 6/7.
00C7: 00 00 00 instctr  ds    3           ; Count of 'dispctr' turns
00CA: 00       traceflg db    0           ; Trace flag (0=off)
00CB: 00       viewmode db    0           ; DEC CRT VIEW mode (0=off)
00CC: 00       lpen     db    0           ; CRT "light pen" (0=off)
00CD: 00       kbmode   db    0           ; CRT Keyboard mode (0=off)
               B220end  equ   *           ; End of B220 simulated state

               * Simulator page zero variables

00CE: FF       OvHlt    db    $FF         ; OVerflow Halt toggle (0=off)
00CF: 00 00    instptr  dw    0           ; Pointer corresponding to rP
00D1: 00 00    memptr   dw    0           ; Pointer to instruction data
00D3: 00 00    ptr      dw    0           ; Utility pointer
00D5: 00 00    inptr    dw    0           ; 'keyin' register label ptr
00D7: 00       t1       db    0           ; Temp byte
00D8: 00       NN       db    0           ; 2-digit BCD count
00D9: 00       dbx      db    0           ; Device Block index
00DA: 64       dispctr  db    dispcnt     ; Display refresh counter
00DB: 00 00    linev    dw    0           ; Line base for decimal value
00DD: 00 00    line1    dw    0           ; Line base for 1-bits
00DF: 00 00    line2    dw    0           ; Line base for 2-bits
00E1: 00 00    line4    dw    0           ; Line base for 4-bits
00E3: 00 00    line8    dw    0           ; Line base for 8-bits
00E5: AF       crtkey   db    nokey       ; CRT keyboard code (nokey=none)
               zpend    equ   *           ; End of B220SIM zero page.
                        dend
```

```
***********************************************************
*                                                         *
*                  Macro Definitions                      *
*                                                         *
***********************************************************

auxjmp   mac                ; <addr>
         sta   READAUX
         sta   WRITAUX
         jmp   ]1
         eom

auxjsr   mac                ; <addr>
         sta   READAUX
         sta   WRITAUX
         jsr   ]1
         sta   READMAIN
         sta   WRITMAIN
         rts
         eom

mainjmp  mac                ; <addr>
         sta   READMAIN
         sta   WRITMAIN
         jmp   ]1
         eom

mainjsr  mac                ; <addr>
         sta   READMAIN
         sta   WRITMAIN
         jsr   ]1
         jmp   AUXrts
         eom

seti     mac                ; Set indicator
         lda   #$FF
         sta   ]1           ; Set non-zero.
         eom

resi     mac                ; Reset indicator
         lda   #0
         sta   ]1           ; Zero indicator.
         eom

align    mac
         ds    *-1/]1*]1+]1-*
         eom

putat    mac                ; ORG forward without overlap
         err   *-1/]1       ; Error if beyond ]1.
         ds    ]1-*         ; Advance to ]1.
         eom
```

```
*     Caltech CRT keyboard mapping table macros
*
* The 'key' and 'fkey' macros are used to populate two
* 128-byte tables mapping ASCII character codes into a
* specially coded byte corresponding to the two-digit
* octal code and two modifier bits produced by the
* keyboards used with the Caltech CRT display.
*
* The 'fkey' macro is used in a similar way to map keys
* pressed while the Open-Apple key is held down to the
* 32-key Function Key keypad.
*
* The first parameter is the two-digit octal code
* specifying the key pressed, and the second parameter
* specifies whether the key has the Shift modifier
* or not.
*
* 'key' is used to populate the mapping table for normal
* keys (indicated by a high bit of 0) and 'fkey' is used
* to populate the table for Function keys (indicated by
* a high bit of 1).
*
* The format of the table entries is:
*       +----+----+----+----+----+----+----+----+
*       | FK | Lo Octal dig | UC | Hi Octal dig |
*       +----+----+----+----+----+----+----+----+

uc       equ   $08         ; Upper Case bit
lc       equ   0           ; Lower Case
invalid  equ   55          ; Octal for invalid key

key      mac               ; octal;uc
]hd      equ   ]1/10       ; High octal digit
         exp   on
         db    0-]hd*10+]1*16*10+]1/10+]2
         exp   off
         eom

fkey     mac               ; octal;uc
]hd      equ   ]1/10       ; High octal digit
         exp   on
         db    0-]hd*10+]1*16*10+]1/10+$80+]2
         exp   off
         eom
```

```
               60           dsk   /ap/merlin/work/b220/b220sim
               61
               62           org   $800       ; Start of MAIN (& common) code
               63           put   B220COMMON
              >1     **********************************************************
              >2     *                                                        *
              >3     *         B220SIM Common Code (Main and Auxmem)          *
              >4     *                                                        *
              >5     **********************************************************
              >6
              >7     common   equ   *          ; Start of code common to Aux & Main
              >8
              >9     * Entry point and restart vector
              >10
0800: 4C 28 09 >11   B220SIM  jmp   init       ; Initialize simulation
0803: 4C AA 09 >12   RESTART  jmp   restart    ; Restart warm.
              >13
              >14    * Vectors for Main to reference Auxmem routines
              >15
              >16    X_fetch  auxjmp fetch
0806: 8D 03 C0 >16            sta   READAUX
0809: 8D 05 C0 >16            sta   WRITAUX
080C: 4C 52 09 >16            jmp   fetch
              >16             eom
              >17    X_newP   auxjmp newP
080F: 8D 03 C0 >17            sta   READAUX
0812: 8D 05 C0 >17            sta   WRITAUX
0815: 4C 34 09 >17            jmp   newP
              >17             eom
              >18    X_cont   auxjmp ]contin
0818: 8D 03 C0 >18            sta   READAUX
081B: 8D 05 C0 >18            sta   WRITAUX
081E: 4C B1 09 >18            jmp   ]contin
              >18             eom
              >19    X_IOerr  auxjmp IOerr
0821: 8D 03 C0 >19            sta   READAUX
0824: 8D 05 C0 >19            sta   WRITAUX
0827: 4C 43 0A >19            jmp   IOerr
              >19             eom
              >20    X_resetr auxjmp resetran
082A: 8D 03 C0 >20            sta   READAUX
082D: 8D 05 C0 >20            sta   WRITAUX
0830: 4C 8D 19 >20            jmp   resetran
              >20             eom
              >21
              >22    X_incP   auxjsr incP
0833: 8D 03 C0 >22            sta   READAUX
0836: 8D 05 C0 >22            sta   WRITAUX
0839: 20 17 0A >22            jsr   incP
083C: 8D 02 C0 >22            sta   READMAIN
083F: 8D 04 C0 >22            sta   WRITMAIN
0842: 60       >22            rts
              >22             eom
```

```
               >24   * Vectors for Aux to reference Main routines
               >25
               >26   M_keyin  mainjmp keyin
0843: 8D 02 C0 >26            sta   READMAIN
0846: 8D 04 C0 >26            sta   WRITMAIN
0849: 4C 21 0A >26            jmp   keyin
               >26            eom
               >27   M_stop   mainjmp ]stop
084C: 8D 02 C0 >27            sta   READMAIN
084F: 8D 04 C0 >27            sta   WRITMAIN
0852: 4C 40 0A >27            jmp   ]stop
               >27            eom
               >28
               >29   M_disp   mainjsr display
0855: 8D 02 C0 >29            sta   READMAIN
0858: 8D 04 C0 >29            sta   WRITMAIN
085B: 20 44 12 >29            jsr   display
085E: 4C 21 09 >29            jmp   AUXrts
               >29            eom
               >30   M_iosel  mainjsr iosel
0861: 8D 02 C0 >30            sta   READMAIN
0864: 8D 04 C0 >30            sta   WRITMAIN
0867: 20 E5 14 >30            jsr   iosel
086A: 4C 21 09 >30            jmp   AUXrts
               >30            eom
               >31   M_iodsel mainjsr iodsel
086D: 8D 02 C0 >31            sta   READMAIN
0870: 8D 04 C0 >31            sta   WRITMAIN
0873: 20 C8 14 >31            jsr   iodsel
0876: 4C 21 09 >31            jmp   AUXrts
               >31            eom
               >32   M_getwrd mainjsr getwrd
0879: 8D 02 C0 >32            sta   READMAIN
087C: 8D 04 C0 >32            sta   WRITMAIN
087F: 20 20 15 >32            jsr   getwrd
0882: 4C 21 09 >32            jmp   AUXrts
               >32            eom
               >33   M_putwrd mainjsr putwrd
0885: 8D 02 C0 >33            sta   READMAIN
0888: 8D 04 C0 >33            sta   WRITMAIN
088B: 20 59 15 >33            jsr   putwrd
088E: 4C 21 09 >33            jmp   AUXrts
               >33            eom
               >34   M_setlan mainjsr setlan
0891: 8D 02 C0 >34            sta   READMAIN
0894: 8D 04 C0 >34            sta   WRITMAIN
0897: 20 72 16 >34            jsr   setlan
089A: 4C 21 09 >34            jmp   AUXrts
               >34            eom
               >35   M_resetd mainjsr resetdb
089D: 8D 02 C0 >35            sta   READMAIN
08A0: 8D 04 C0 >35            sta   WRITMAIN
08A3: 20 C4 16 >35            jsr   resetdb
08A6: 4C 21 09 >35            jmp   AUXrts
               >35            eom
               >36   M_nxtblk mainjsr nxtblk
08A9: 8D 02 C0 >36            sta   READMAIN
08AC: 8D 04 C0 >36            sta   WRITMAIN
08AF: 20 A5 15 >36            jsr   nxtblk
08B2: 4C 21 09 >36            jmp   AUXrts
               >36            eom
               >37   M_prvblk mainjsr prvblk
08B5: 8D 02 C0 >37            sta   READMAIN
08B8: 8D 04 C0 >37            sta   WRITMAIN
08BB: 20 F9 15 >37            jsr   prvblk
08BE: 4C 21 09 >37            jmp   AUXrts
               >37            eom
               >38   M_readbf mainjsr readbuf
08C1: 8D 02 C0 >38            sta   READMAIN
08C4: 8D 04 C0 >38            sta   WRITMAIN
08C7: 20 9A 15 >38            jsr   readbuf
08CA: 4C 21 09 >38            jmp   AUXrts
               >38            eom
               >39   M_ckspo  mainjsr ckspo
08CD: 8D 02 C0 >39            sta   READMAIN
08D0: 8D 04 C0 >39            sta   WRITMAIN
08D3: 20 9C 16 >39            jsr   ckspo
08D6: 4C 21 09 >39            jmp   AUXrts
               >39            eom
               >40   M_trace  mainjsr prtrace
08D9: 8D 02 C0 >40            sta   READMAIN
08DC: 8D 04 C0 >40            sta   WRITMAIN
08DF: 20 84 18 >40            jsr   prtrace
08E2: 4C 21 09 >40            jmp   AUXrts
               >40            eom
               >41   M_plot   mainjsr b220plot
08E5: 8D 02 C0 >41            sta   READMAIN
08E8: 8D 04 C0 >41            sta   WRITMAIN
08EB: 20 40 1B >41            jsr   b220plot
08EE: 4C 21 09 >41            jmp   AUXrts
               >41            eom
               >42   M_lpread mainjsr lpread
08F1: 8D 02 C0 >42            sta   READMAIN
08F4: 8D 04 C0 >42            sta   WRITMAIN
08F7: 20 70 1C >42            jsr   lpread
08FA: 4C 21 09 >42            jmp   AUXrts
               >42            eom
               >43   M_xdrawc mainjsr xdrawcur
08FD: 8D 02 C0 >43            sta   READMAIN
0900: 8D 04 C0 >43            sta   WRITMAIN
0903: 20 F2 1B >43            jsr   xdrawcur
0906: 4C 21 09 >43            jmp   AUXrts
               >43            eom
               >44   M_COUT   mainjsr COUT
0909: 8D 02 C0 >44            sta   READMAIN
090C: 8D 04 C0 >44            sta   WRITMAIN
090F: 20 ED FD >44            jsr   COUT
0912: 4C 21 09 >44            jmp   AUXrts
               >44            eom
               >45   M_PRBL2  mainjsr PRBL2
0915: 8D 02 C0 >45            sta   READMAIN
0918: 8D 04 C0 >45            sta   WRITMAIN
091B: 20 4A F9 >45            jsr   PRBL2
091E: 4C 21 09 >45            jmp   AUXrts
               >45            eom
               >46
0921: 8D 03 C0 >47   AUXrts   sta   READAUX
0924: 8D 05 C0 >48            sta   WRITAUX
0927: 60       >49            rts
```

```
                endcomm  equ   *          ; End of code common to AUX & MAIN
                         put   B220INIT
                **********************************************************
                *                                                        *
                *                    Initialize B220                     *
                *                                                        *
                **********************************************************

0928: AD CA 09  init     lda   initstk    ; Been here before?
092B: D0 46              bne   :notinit   ; -Yes, skip init copys.
092D: BA                 tsx              ; -No, save initial stk ptr.
092E: 8E CA 09           stx   initstk
0931: 20 BA 09           jsr   swapzp     ; Make B220SIM ZP active.
0934: A9 00              lda   #<common   ; Copy common code from Main-->Aux
0936: 85 D1              sta   memptr
0938: A9 08              lda   #>common
093A: 85 D2              sta   memptr+1
093C: A0 00              ldy   #0
093E: 8D 05 C0           sta   WRITAUX    ; Stores go to AUX memory.
0941: B1 D1     :commlp  lda   (memptr),y
0943: 91 D1              sta   (memptr),y
0945: C8                 iny
0946: D0 F9              bne   :commlp
0948: E6 D2              inc   memptr+1   ; Next page...
094A: A9 09              lda   #>endcomm-1
094C: C5 D2              cmp   memptr+1   ; Move next page?
094E: B0 F1              bcs   :commlp    ; -Yes.
0950: A9 00              lda   #<AUXcode  ; Copy B220SIM to Aux mem
0952: 85 D3              sta   ptr
0954: A9 21              lda   #>AUXcode
0956: 85 D4              sta   ptr+1
0958: A9 28              lda   #<endcomm
095A: 85 D1              sta   memptr
095C: A9 09              lda   #>endcomm
095E: 85 D2              sta   memptr+1
0960: A0 00              ldy   #0         ; Move a page
0962: B1 D3     :auxlp   lda   (ptr),y
0964: 91 D1              sta   (memptr),y
0966: C8                 iny
0967: D0 F9              bne   :auxlp
0969: E6 D2              inc   memptr+1
096B: E6 D4              inc   ptr+1
096D: A5 D4              lda   ptr+1
096F: C9 35              cmp   #>AUXend+$100 ; Past last page?
0971: 90 EF              bcc   :auxlp     ; -No, keep moving.
0973: 8D 05 C0  :notinit sta   WRITAUX    ; Stores go to AUX memory.
0976: A9 D0              lda   #<MEM      ; Initialize B220 memory to 0
0978: 85 D1              sta   memptr
097A: A9 4A              lda   #>MEM
097C: 85 D2              sta   memptr+1
097E: A0 00              ldy   #0
0980: 98        :loop    tya
0981: 91 D1     :pagloop sta   (memptr),y
0983: C8                 iny
0984: D0 FB              bne   :pagloop
0986: E6 D2              inc   memptr+1
0988: A5 D2              lda   memptr+1
098A: C9 96              cmp   #>$9600
098C: 90 F2              bcc   :loop
098E: 8D 04 C0           sta   WRITMAIN   ; Back to Main mem
0991: 20 4D 1A           jsr   HGRinit    ; Init VIEW mode tables.
0994: A2 3D     reset    ldx   #B220end-B220strt-1 ; Clear B220 state
0996: A9 00              lda   #0
0998: 95 90     :regloop sta   B220strt,x
099A: CA                 dex
```

```
099B: 10 FB    >64            bpl   :regloop
099D: 20 B3 16 >65            jsr   resetdbs   ; Rewind all tapes.
               >66            seti  OvHlt      ; Set Ovflow Halt mode.
09A0: A9 FF    >66            lda   #$FF
09A2: 85 CE    >66            sta   OvHlt      ; Set non-zero.
               >66            eom
09A4: A9 AF    >67            lda   #nokey     ; Init crtkey
09A6: 85 E5    >68            sta   crtkey     ;  to empty.
09A8: D0 03    >69            bne   ]restore   ; Skip swapzp (always)
09AA: 20 BA 09 >70   restart  jsr   swapzp     ; Make B220SIM ZP active.
09AD: AE CA 09 >71   ]restore ldx   initstk    ; Restore initial stack ptr.
09B0: 9A       >72            txs
09B1: 20 11 10 >73   back2sim jsr   disppanl   ; Init screen for B220
09B4: 20 44 12 >74            jsr   display    ;  panel & display state.
09B7: 4C 0F 08 >75            jmp   X_newP     ; Start simulation.
               >76
09BA: A2 55    >77   swapzp   ldx   #zpend-B220strt-1 ; Swap BASIC and
09BC: B5 90    >78   :swaplp  lda   B220strt,x ;        B220SIM zero page.
09BE: BC CB 09 >79            ldy   zpsave,x
09C1: 9D CB 09 >80            sta   zpsave,x
09C4: 94 90    >81            sty   B220strt,x
09C6: CA       >82            dex
09C7: D0 F3    >83            bne   :swaplp
09C9: 60       >84            rts
               >85
09CA: 00       >86   initstk  db    0          ; Stack pointer at entry.
09CB: 00 00 00 >87   zpsave   ds    zpend-B220strt
```

```
                  66            put   B220KEYB
                 >1     *********************************************************
                 >2     *                                                       *
                 >3     *                Keyboard Input Routines                *
                 >4     *                                                       *
                 >5     *********************************************************
                 >6
0A21: 8D 10 C0 >7     keyin    sta   KBSTROBE   ; Clear strobe.
0A24: A6 CB    >8              ldx   viewmode   ; View mode?
0A26: F0 03    >9              beq   :ckview    ; -No, check other hot keys.
0A28: 4C 8C 19 >10             jmp   viewkey    ; -Yes, analyze VIEW keys.
               >11
0A2B: C9 D6    >12    :ckview  cmp   #"V"       ; Switch to VIEW mode?
0A2D: F0 04    >13             beq   :von       ; -Yes, enter VIEW mode.
0A2F: C9 F6    >14    :lcv     cmp   #"v"       ; -No, lower case "V"?
0A31: D0 09    >15             bne   ]ckstop    ; -No, check stop/step.
0A33: 4C 64 19 >16    :von     jmp   viewon     ; -Yes, enter VIEW mode
               >17
0A36: 20 DD FB >18    :bleep   jsr   BEEP       ; Beep
0A39: 4C 18 08 >19    :xeq     jmp   X_cont     ; Execute current OP.
               >20
0A3C: C9 A0    >21    ]ckstop  cmp   #"         " ; Space bar?
0A3E: D0 F6    >22             bne   :bleep     ; -No, beep & continue.
               >23    ]stop    resi  RUN        ; -Yes, reset RUN mode
0A40: A9 00    >23             lda   #0
0A42: 85 C0    >23             sta   RUN        ; Zero indicator.
               >23             eom
0A44: AD 67 05 >24             lda   ERRlab     ; Did I/O error
0A47: C9 C9    >25             cmp   #"I"       ;  get us here?
0A49: F0 30    >26             beq   :edit      ; -Yes, don't flush.
0A4B: D0 2B    >27             bne   :flush     ; -No, flush bufs (always)
               >28
               >29             putat VIEWhlp1   ; Line 20 of HGR2
               >29             err   *-1/VIEWhlp1 ; Error if beyond VIEWhlp1.
0A4D: 00 00 00 >29             ds    VIEWhlp1-* ; Advance to VIEWhlp1.
               >29             eom
0A50: A0 A0 D6 >30             asc   "  VIEW CRT display mode (ESC to exit)   "
               >31
0A78: 20 10 17 >32    :flush   jsr   flushall   ; Flush all buffers.
0A7B: 20 44 12 >33    :edit    jsr   display    ; Update B220 panel
               >34             resi  ERR        ; Reset ERR indicator
0A7E: A9 00    >34             lda   #0
0A80: 85 C1    >34             sta   ERR        ; Zero indicator.
               >34             eom
0A82: AD 00 C0 >35    ]waitkey lda   KBD        ; Get a key.
0A85: 10 FB    >36             bpl   ]waitkey
0A87: 8D 10 C0 >37             sta   KBSTROBE   ; Clear strobe
0A8A: A6 CB    >38             ldx   viewmode   ; View mode?
0A8C: F0 03    >39             beq   ]analyze   ; -No, check stop/step.
0A8E: 4C 8C 19 >40             jmp   viewkey    ; -Yes, analyze VIEW keys.
               >41
0A91: C9 A0    >42    ]analyze cmp   #"         " ; Space bar?
0A93: F0 0E    >43             beq   :step      ; -Yes, step.
0A95: C9 BF    >44             cmp   #"?"       ; Show help?
0A97: F0 1D    >45             beq   :disphlp   ; -Yes, do it.
0A99: 29 DF    >46             and   #$DF       ; Force upper case.
0A9B: C9 C7    >47             cmp   #"G"       ; G = Go?
0A9D: D0 24    >48             bne   :nx1       ; -No, analyze keypress.
               >49             seti  RUN        ; -Yes, set RUN mode
0A9F: A9 FF    >49             lda   #$FF
0AA1: 85 C0    >49             sta   RUN        ; Set non-zero.
               >49             eom
0AA3: A9 F2    >50    :step    lda   #"r"       ; Reset ERRlab on screen
0AA5: 8D 67 05 >51             sta   ERRlab
0AA8: A5 C5    >52             lda   newp       ; rP changed manually?
0AAA: D0 10    >53             bne   :new       ; -Yes, re-fetch.
0AAC: A5 9B    >54             lda   rC+OP      ; -No, is OP a HLT?
```

```
0AAE: D0 89    >55           bne   :xeq        ; -No, execute current OP
0AB0: 20 33 08 >56           jsr   X_incP      ; -Yes, skip HLT
0AB3: 4C 06 08 >57           jmp   X_fetch     ;   and fetch next.
               >58
0AB6: 20 33 12 >59   :disphlp jsr  disphelp    ; Display help lines
0AB9: 4C 82 0A >60           jmp   ]waitkey    ;  and get another key.
               >61
               >62   :new    resi  newp        ; Reset new P indicator
0ABC: A9 00    >62           lda   #0
0ABE: 85 C5    >62           sta   newp        ; Zero indicator.
               >62           eom
0AC0: 4C 0F 08 >63           jmp   X_newP      ;  and re-fetch.
               >64
0AC3: A6 CB    >65   :nx1    ldx   viewmode    ; VIEW mode on?
0AC5: F0 31    >66           beq   :ckkeys     ; -No, check other keys.
0AC7: 4C 88 0B >67           jmp   :beep       ; -Yes, other keys invalid.
               >68
               >69           putat VIEWhlp2    ; Line 21 of HGR2
               >69           err   *-1/VIEWhlp2 ; Error if beyond VIEWhlp2.
0ACA: 00 00 00 >69           ds    VIEWhlp2-* ; Advance to VIEWhlp2.
               >69           eom
0AD0: D0 E1 EE >70           asc   "Pan display: arrow keys, Zoom: + and -  "
               >71
0AF8: C9 D1    >72   :ckkeys cmp   #"Q"        ; Is it Quit?
0AFA: D0 37    >73           bne   :nx2        ; -No, continue.
0AFC: D8       >74           cld               ; -Yes, clear decimal
0AFD: 18       >75           clc               ;   and Carry.
0AFE: 20 BA 09 >76           jsr   swapzp      ; Make BASIC ZP active.
0B01: A9 00    >77           lda   #0          ; Set full-screen
0B03: 85 22    >78           sta   WNDTOP      ;  text window,
0B05: 68       >79           pla               ;  pop return
0B06: 68       >80           pla               ;   address, and
0B07: 4C D0 03 >81           jmp   DOSCON      ;    reconnect ProDOS.
               >82
0B0A: A9 13    >83   :flipsw lda   #$13        ; Set "Sw" label to inverse.
0B0C: 8D 53 05 >84           sta   SWlab
0B0F: A9 77    >85           lda   #$77
0B11: 8D 54 05 >86           sta   SWlab+1
0B14: 20 6E 0C >87           jsr   getdig      ; Get digit or CR
0B17: B0 0D    >88           bcs   :swdone     ; Done if CR.
0B19: AA       >89           tax               ; -No, handle digit.
0B1A: B5 B6    >90           lda   CSW,x       ; Pick up switch,
0B1C: F0 04    >91           beq   :seti       ; -If reset, set it.
0B1E: A9 00    >92           lda   #0          ; -If set, reset it.
0B20: F0 02    >93           beq   :store      ; (always)
               >94
0B22: A9 FF    >95   :seti   lda   #$FF
0B24: 95 B6    >96   :store  sta   CSW,x       ;   put it back.
0B26: A9 D3    >97   :swdone lda   #"S"        ; Set "Sw" label to normal.
0B28: 8D 53 05 >98           sta   SWlab
0B2B: A9 F7    >99           lda   #"w"
0B2D: 8D 54 05 >100          sta   SWlab+1
0B30: 4C 7B 0A >101          jmp   :edit
               >102
0B33: C9 D3    >103  :nx2    cmp   #"S"        ; Toggle switch?
0B35: F0 D3    >104          beq   :flipsw     ; -Yes.
0B37: C9 C1    >105          cmp   #"A"        ; A register?
0B39: F0 73    >106          beq   :inputA     ; -Yes, get input.
0B3B: C9 D2    >107          cmp   #"R"        ; R register?
0B3D: F0 73    >108          beq   :inputR     ; -Yes, get input.
0B3F: C9 C2    >109          cmp   #"B"        ; B register?
0B41: F0 73    >110          beq   :inputB     ; -Yes, get input.
0B43: C9 D0    >111          cmp   #"P"        ; P register?
0B45: F0 77    >112          beq   :inputP     ; -Yes, get input.
0B47: C9 C3    >113          cmp   #"C"        ; C register?
0B49: F0 6F    >114          beq   :inputC     ; -Yes, get input.
0B4B: D0 2B    >115          bne   :ckmore     ; -No, check more (always)
```

```

                        putat VIEWhlp3   ; Line 22 of HGR2
                        err   *-1/VIEWhlp3 ; Error if beyond VIEWhlp3.
0B4D: 00 00 00          ds    VIEWhlp3-* ; Advance to VIEWhlp3.
                        eom
0B50: D4 EF E7          asc   "Toggle Pen: L, Enter Keyboard mode: K   "

0B78: C9 DA    :ckmore  cmp   #"Z"       ; Reset?
0B7A: F0 12             beq   :reset     ; -Yes, clear state.
0B7C: C9 C9             cmp   #"I"       ; I/O configuration?
0B7E: F0 11             beq   :edio      ; -Yes, edit I/O config.
0B80: C9 D6             cmp   #"V"       ; Turn on VIEW mode?
0B82: F0 10             beq   :viewon    ; -Yes.
0B84: C9 D4             cmp   #"T"       ; Toggle trace mode?
0B86: F0 0F             beq   :tracetg   ; -Yes.
0B88: 20 DD FB :beep    jsr   BEEP       ; Unrecognized key, beep
0B8B: 4C 82 0A          jmp   ]waitkey   ;  and get another key.

0B8E: 4C 94 09 :reset   jmp   reset      ; Reset B220 state.

0B91: 4C CC 0D :edio    jmp   ediocfg    ; Edit I/O configuration.

0B94: 4C 64 19 :viewon  jmp   viewon     ; Enter VIEW mode.

0B97: A0 00    :tracetg ldy   #0         ; Toggle traceflg.
0B99: A5 CA             lda   traceflg   ; Is it set?
0B9B: D0 01             bne   :settr     ; -Yes, clear it.
0B9D: 88                dey              ; -No, set it.
0B9E: 84 CA    :settr   sty   traceflg
0BA0: 4C 82 0A          jmp   ]waitkey   ;   and get another key.

0BA3: A0 00    :indone  ldy   #0         ; Flip reg label to normal.
0BA5: B1 D5             lda   (inptr),y
0BA7: 09 80             ora   #$80
0BA9: 91 D5             sta   (inptr),y
0BAB: 4C 7B 0A          jmp   :edit

0BAE: A2 00    :inputA  ldx   #Ain-intabl
0BB0: B0 46             bcs   :inreg     ; (always)

0BB2: A2 10    :inputR  ldx   #Rin-intabl
0BB4: B0 42             bcs   :inreg     ; (always)

0BB6: A2 04    :inputB  ldx   #Bin-intabl
0BB8: B0 3E             bcs   :inreg     ; (always)

0BBA: A2 08    :inputC  ldx   #Cin-intabl
0BBC: B0 3A             bcs   :inreg     ; (always)

0BBE: A2 0C    :inputP  ldx   #Pin-intabl
                        seti  newp       ; Signal manual rP change.
0BC0: A9 FF             lda   #$FF
0BC2: 85 C5             sta   newp       ; Set non-zero.
                        eom
0BC4: B0 32             bcs   :inreg     ; (always)

                        putat VIEWhlp4   ; Line 23 of HGR2
                        err   *-1/VIEWhlp4 ; Error if beyond VIEWhlp4.
0BC6: 00 00 00          ds    VIEWhlp4-* ; Advance to VIEWhlp4.
                        eom
0BD0: C5 EE E1          asc   "Enable Pen detect: Hold Open-Apple down "

               * Input register value from keyboard
               *   On entry: X = intabl index
               *   On exit:  Y = Hi (left) byte of register
               *             X = # of bytes in register - 1

```

```
0BF8: BD 33 0C :inreg   lda   intabl,x   ; Set inptr to reg label
0BFB: 85 D5             sta   inptr
0BFD: BD 34 0C          lda   intabl+1,x
0C00: 85 D6             sta   inptr+1
0C02: BC 35 0C          ldy   intabl+2,x ; Y = hi byte of reg
0C05: 8C 27 0C          sty   :ordig+1   ; Save register address
0C08: 8C 29 0C          sty   :stdig+1
0C0B: BD 36 0C          lda   intabl+3,x
0C0E: AA                tax              ; X = reg length - 1
0C0F: A0 00             ldy   #0
0C11: B1 D5             lda   (inptr),y  ; Flip reg label to inverse.
0C13: 29 7F             and   #$7F
0C15: 91 D5             sta   (inptr),y
0C17: D0 00             bne   :getdig    ; (always)

0C19: 20 6E 0C :getdig  jsr   getdig     ; Get digit or CR
0C1C: B0 85             bcs   :indone    ; CR ==> done.
0C1E: 48                pha              ; Save digit
0C1F: AC 27 0C          ldy   :ordig+1   ; Restore Y
0C22: 20 47 0C          jsr   shleft1    ; Shift register left 1 digit
0C25: 68                pla              ; Recover the digit
0C26: 15 00    :ordig   ora   0*0,x      ; OR in the low digit
0C28: 95 00    :stdig   sta   0*0,x      ;  and store it back.
0C2A: 8A                txa              ; Save X
0C2B: 48                pha
0C2C: 20 44 12          jsr   display    ; Update display
0C2F: 68                pla              ; Restore X
0C30: AA                tax
0C31: D0 E6             bne   :getdig    ; (always)

               intabl   equ   *          ; Table of reg input params
0C33: 83 05    Ain      dw    Alab       ; Address of "A" label
0C35: 9E 05             db    rA,6-1     ; Addr of hi digit, length-1
0C37: AB 05    Bin      dw    Blab
0C39: 94 01             db    rB,2-1
0C3B: BB 05    Cin      dw    Clab
0C3D: 98 05             db    rC,6-1
0C3F: B3 05    Pin      dw    Plab
0C41: 96 01             db    rP,2-1
0C43: 95 05    Rin      dw    Rlab
0C45: A4 05             db    rR,6-1
```

```
               **********************************************************
               *                                                        *
               *         Shift Register left 1 digit (4 bits)           *
               *                                                        *
               * On entry: Y = addr of Hi (left) byte of register       *
               *           X = register byte length - 1                 *
               *                                                        *
               * On exit:  X and Y are unchanged.  If rA, rR, or rC,    *
               *           the high digit of the sign byte is cleared.  *
               *                                                        *
               **********************************************************

0C47: 8C 4F 0C shleft1  sty   :shift+1   ; Save register address
0C4A: 8A                txa              ;  and byte length - 1.
0C4B: A0 04             ldy   #4         ; Digit = 4 bits.
0C4D: 18       :nxshift clc              ; Shift in zeroes.
0C4E: 36 00    :shift   rol   0*0,x      ; Shift 1 bit
0C50: CA                dex              ;  for all bytes.
0C51: 10 FB             bpl   :shift
0C53: AA                tax              ; Restore X
0C54: 88                dey
0C55: D0 F6             bne   :nxshift   ; Shift 4 times.
0C57: AC 4F 0C          ldy   :shift+1   ; Restore Y = reg address.
0C5A: C0 96             cpy   #rP        ; rP has no sign byte,
0C5C: F0 0C             beq   :rts       ;  so skip it.
0C5E: C0 94             cpy   #rB        ; rB has no sign byte,
0C60: F0 08             beq   :rts       ;  so skip it.
0C62: B9 00 00          lda   0,y        ; Clear high digit
0C65: 29 0F             and   #$0F       ;  of sign byte.
0C67: 99 00 00          sta   0,y
0C6A: 60       :rts     rts

               **********************************************************
               *                                                        *
               *                   Get Digit or CR                      *
               *                                                        *
               * On exit: If C = 0, A = digit value                     *
               *          If C = 1, CR received                         *
               *          X and Y unchanged.                            *
               *                                                        *
               **********************************************************

0C6B: 20 DD FB beepget  jsr   BEEP       ; Signal error key
0C6E: AD 00 C0 getdig   lda   KBD        ; Get digit or <Enter>
0C71: 10 FB             bpl   getdig
0C73: 8D 10 C0          sta   KBSTROBE   ; Clear strobe
0C76: C9 8D             cmp   #$8D       ; <Enter>?
0C78: F0 0A             beq   :done      ; Yes, exit.
0C7A: C9 B0             cmp   #"0"       ; -No, less than "0"?
0C7C: 90 ED             bcc   beepget    ; -Yes, get another.
0C7E: C9 BA             cmp   #"9"+1     ; -No, greater than "9"?
0C80: B0 E9             bcs   beepget    ; -Yes, get another.
0C82: 29 0F             and   #$0F       ; -No, isolate digit
0C84: 60       :done    rts              ; C ==> digit, /C ==> CR.
```

```
               ***********************************************************
               *                                                         *
               *            Edit B220SIM I/O Configuration               *
               *                                                         *
               ***********************************************************

               cursor   equ   $57         ; Mousetext checkerboard
               uparrow  equ   $8B         ; Up arrow
               dnarrow  equ   $8A         ; Down arrow
               ltarrow  equ   $88         ; Left arrow
               escape   equ   $9B         ; ESCAPE key
               delete   equ   $FF         ; DELETE key
               iocfgtt  equ   11          ; HTAB for screen title
               rtmargin equ   4           ; Right margin
               fnamecol equ   rtmargin+8  ; File name column
               optline1 equ   18          ; First line of options

               fnx      equ   linev       ; File name index (0..7)
               selected equ   linev+1     ; Selected index (0..7)
               selsave  equ   line1       ; Temp Y storage
               savex    equ   line1+1     ; Temp X storage
               selch    equ   line2       ; Selected fname cursor
               line     equ   line2+1     ; Line number (0..23)
               changed  equ   line4       ; File name changed flg
               selBASL  equ   line8       ; Selected line base (DA.DB)

               iocfgstr equ   *           ; I/O Config Screen string
0C85: C9 AF CF          asc   "I/O Configuration",0D
0C97: 0D                db    $0D
0C98: A0 D5 EE          asc   " Unit    File pathname",0D
0CAF: AD AD AD          asc   "------  ------------------------",0D
0CD0: D0 D4 D2          asc   "PTRDR0",01
0CD7: D0 D4 D2          asc   "PTRDR1",01
0CDE: D0 D4 D0          asc   "PTPCH0",01
0CE5: D0 D4 D0          asc   "PTPCH1",01
0CEC: 0D                db    $0D
0CED: CD D4 D5          asc   "MTU0L0",01
0CF4: CD D4 D5          asc   "MTU0L1",01
0CFB: CD D4 D5          asc   "MTU1L0",01
0D02: CD D4 D5          asc   "MTU1L1",01
0D09: 0D 0D             db    $0D,$0D
0D0B: A0 D4 F9          asc   " Type 'SPO' as punch pathname to",0D
0D2C: A0 A0 F2          asc   "  redirect punch output to SPO.",0D
0D4C: 0D 0D 0D          db    $0D,$0D,$0D,$0D ; Reserve 2 option lines
0D50: A0 A0 A0          asc   "    ESC to return to B220SIM"
0D6C: 00                db    00          ; End of screen

0D6D: D3 D0 CF optlines asc   "SPO charset: "
0D7A: 03       charset  inv   'C' ; Inverse upper case
0D7B: 61 6C 74          asc   'altech' ; Inverse lower case
0D81: A0 A0 A8          asc   "  (ctl-C to toggle)",00
0D95: A0 A0 D3          asc   "  Sound out: "
0DA2: 13       sndport  inv   'S' ; Inverse upper case
0DA3: 70 65 61          asc   'peaker' ; Inverse lower case
0DA9: A0 A0 A8          asc   "  (ctl-S to toggle)",00

0DBD: 02 01 0C newcset  inv   'BALGOL',A0 ; Inverse upper case
0DC4: 03       newsnd   inv   'C' ; Inverse upper case
0DC5: 61 73 73          asc   'assette' ; Inverse lower case

```

```
0DCC: A2 00    >332  ediocfg  ldx   #0          ; Edit I/O Configuration
0DCE: 86 22    >333           stx   WNDTOP      ; Set full screen.
0DD0: 86 DC    >334           stx   selected    ; Select first file name.
0DD2: 20 58 FC >335           jsr   HOME        ; Clear screen
0DD5: 20 7F 0F >336           jsr   disopts     ; Display option lines.
0DD8: A2 00    >337  disiocfg ldx   #0          ; iocfgstr index = 0
0DDA: 86 DB    >338           stx   fnx         ; fname index = 0
0DDC: 86 E0    >339           stx   line        ; Line = 0
0DDE: 8A       >340           txa
0DDF: 20 C1 FB >341           jsr   BASCALC     ; Set BASL for line 0
0DE2: A0 0B    >342           ldy   #iocfgtt    ; HTAB to title
0DE4: BD 85 0C >343  :nxch    lda   iocfgstr,x  ; Next disp string char
0DE7: 10 06    >344           bpl   :command    ; -Command char if +
0DE9: 91 28    >345           sta   (BASL),y    ; -Display if not cmd.
0DEB: C8       >346           iny               ; Advance CH
0DEC: E8       >347  :advance inx               ; Advance str index
0DED: D0 F5    >348           bne   :nxch       ; (always)
               >349
0DEF: F0 48    >350  :command beq   :editfn     ; Screen complete, edit.
0DF1: C9 0D    >351           cmp   #$0D        ; CR?
0DF3: D0 0B    >352           bne   :fname      ; -No, insert file name.
0DF5: E6 E0    >353  :nxtline inc   line        ; -Yes, next line.
0DF7: A5 E0    >354           lda   line        ; Compute new line's
0DF9: 20 C1 FB >355           jsr   BASCALC     ;  base addr (BASL)
0DFC: A0 04    >356           ldy   #rtmargin   ; Set right margin.
0DFE: 10 EC    >357           bpl   :advance    ; (always)
               >358
0E00: 86 DE    >359  :fname   stx   savex       ; Insert file name.
0E02: A9 BA    >360           lda   #":"        ; Insert punctuation.
0E04: 91 28    >361           sta   (BASL),y
0E06: A4 DB    >362           ldy   fnx
0E08: C4 DC    >363           cpy   selected    ; This fname selected?
0E0A: F0 01    >364           beq   :selectd    ; -Yes, C = selected.
0E0C: 18       >365           clc               ; -No, /C = not selected.
0E0D: BE EF 13 >366  :selectd ldx   fnxfn,y     ; Index into fnames
0E10: A0 0C    >367           ldy   #fnamecol   ; Y = 1st char of filename.
0E12: BD 00 14 >368  :nxtchar lda   fnames,x    ; Next file name char.
0E15: F0 0C    >369           beq   :fndone     ; End of file name.
0E17: 90 04    >370           bcc   :store      ; /C ==> keep normal.
0E19: 20 FD 0E >371           jsr   inverse     ; C ==> make inverse
0E1C: 38       >372           sec               ;  and stay selected.
0E1D: 91 28    >373  :store   sta   (BASL),y    ; Display character
0E1F: E8       >374           inx               ; Advance fnames index
0E20: C8       >375           iny               ; Advance CH
0E21: D0 EF    >376           bne   :nxtchar    ; (always)
               >377
0E23: E6 DB    >378  :fndone  inc   fnx         ; Advance fnames index
0E25: A6 DE    >379           ldx   savex       ; Restore string index
0E27: 90 CC    >380           bcc   :nxtline    ; Not selected ==> done.
0E29: A9 57    >381           lda   #cursor     ; Selected ==> add cursor.
0E2B: 91 28    >382           sta   (BASL),y
0E2D: 84 DF    >383           sty   selch       ; Save cursor column.
0E2F: A5 28    >384           lda   BASL        ; Save selected line base
0E31: 85 E3    >385           sta   selBASL
0E33: A5 29    >386           lda   BASL+1
0E35: 85 E4    >387           sta   selBASL+1
0E37: D0 BC    >388           bne   :nxtline    ; (always)
               >389
0E39: A4 DF    >390  :editfn  ldy   selch       ; Cursor col of selected.
0E3B: A9 00    >391           lda   #0          ; Mark unchanged.
0E3D: 85 E1    >392           sta   changed
0E3F: AD 00 C0 >393  ]kbdloop lda   KBD         ; Read key and
0E42: 10 FB    >394           bpl   ]kbdloop    ;  wait for keypress.
0E44: 8D 10 C0 >395           sta   KBSTROBE    ; Clear keyboard strobe.
0E47: C9 93    >396           cmp   #$93        ; ctl-S?
0E49: F0 61    >397           beq   :togsnd     ; -Yes, toggle sound port.
0E4B: C9 83    >398           cmp   #$83        ; -No.  ctl-C?
```

```
0E4D: F0 5A    >399           beq   :togcset   ; -Yes, toggle SPO charset.
0E4F: A6 DC    >400           ldx   selected   ; -No, save index of currently
0E51: 86 DD    >401           stx   selsave    ;  selected file name.
0E53: C9 8B    >402           cmp   #uparrow   ; Uparrow?
0E55: D0 58    >403           bne   :notup     ; -No.  Keep checking.
0E57: C6 DC    >404           dec   selected   ; -Yes, move cursor up
0E59: A5 DC    >405           lda   selected   ;  and wrap around.
0E5B: 29 07    >406           and   #7
0E5D: 85 DC    >407           sta   selected
0E5F: A9 A0    >408  :edited  lda   #"         " ; Blank out cursor
0E61: A4 DF    >409           ldy   selch
0E63: 91 E3    >410           sta   (selBASL),y
0E65: A5 E1    >411           lda   changed    ; Fname changed?
0E67: F0 29    >412           beq   :chkexit   ; -No, exit or redisplay.
0E69: A4 DD    >413           ldy   selsave    ; -Yes, get selected index
0E6B: BE EF 13 >414           ldx   fnxfn,y    ; -Yes, commit new
0E6E: A0 0C    >415           ldy   #fnamecol  ;   file name.
0E70: C4 DF    >416  :copy    cpy   selch      ; End of file name?
0E72: F0 11    >417           beq   :fnend     ; -Yes.
0E74: B1 E3    >418           lda   (selBASL),y
0E76: 09 80    >419           ora   #$80       ; -No. Make normal.
0E78: C9 A0    >420           cmp   #$A0       ; Upper case?
0E7A: B0 02    >421           bcs   :norm      ; -No, already normal.
0E7C: 09 40    >422           ora   #$40       ; -Yes, make normal.
0E7E: 9D 00 14 >423  :norm    sta   fnames,x
0E81: E8       >424           inx
0E82: C8       >425           iny
0E83: D0 EB    >426           bne   :copy      ; (always)
               >427
0E85: A9 00    >428  :fnend   lda   #0         ; Null at end
0E87: 9D 00 14 >429           sta   fnames,x   ;  of fname.
0E8A: A4 DD    >430           ldy   selsave    ; Reset Device Block
0E8C: BE E7 13 >431           ldx   fnxdbx,y   ;  for new file.
0E8F: 20 C4 16 >432           jsr   resetdb
0E92: AD 00 C0 >433  :chkexit lda   KBD        ; Check last key.
0E95: C9 1B    >434           cmp   #escape&$7F ; Was it ESCAPE?
0E97: F0 03    >435           beq   :restart   ; -Yes, back to sim.
0E99: 4C D8 0D >436  :disiocr jmp   disiocfg   ; Redisplay & continue.
               >437
0E9C: 4C B1 09 >438  :restart jmp   back2sim   ; Restart B220SIM.
               >439
0E9F: 84 DD    >440  :beep    sty   selsave    ; Scratch to save Y.
0EA1: 20 DD FB >441           jsr   BEEP       ; Signal invalid key
0EA4: A4 DD    >442           ldy   selsave    ; Restore Y
0EA6: 4C 3F 0E >443  :kbdlpr  jmp   ]kbdloop   ;  and continue.
               >444
0EA9: 4C 0A 0F >445  :togcset jmp   togcset    ; Relay JMP
0EAC: 4C 4D 0F >446  :togsnd  jmp   togsound   ; Relay JMP
               >447
0EAF: C9 8A    >448  :notup   cmp   #dnarrow
0EB1: F0 04    >449           beq   :down
0EB3: C9 8D    >450           cmp   #$8D
0EB5: D0 0A    >451           bne   :notdown   ; Not down arrow or return.
0EB7: E6 DC    >452  :down    inc   selected   ; Move cursor down
0EB9: A5 DC    >453           lda   selected   ;  and wrap around.
0EBB: 29 07    >454           and   #7
0EBD: 85 DC    >455           sta   selected
0EBF: 10 9E    >456           bpl   :edited    ; (always)
               >457
0EC1: C9 9B    >458  :notdown cmp   #escape    ; ESC?
0EC3: F0 9A    >459           beq   :edited    ; -Yes, commit fname.
0EC5: C9 88    >460           cmp   #ltarrow   ; Left arrow?
0EC7: F0 04    >461           beq   :backsp    ; -Yes, backspace.
0EC9: C9 FF    >462           cmp   #delete    ; DELETE?
0ECB: D0 13    >463           bne   :addchar   ; -No, add character.
0ECD: C0 0C    >464  :backsp  cpy   #fnamecol  ; At start?
0ECF: F0 CE    >465           beq   :beep      ; -Yes, complain.
```

```
0ED1: A9 A0    >466           lda   #"         " ; -No, blank cursor
0ED3: 91 E3    >467           sta   (selBASL),y
0ED5: 88       >468           dey             ; Back up.
0ED6: A9 57    >469  :changed lda   #cursor    ; Place cursor.
0ED8: 91 E3    >470           sta   (selBASL),y
0EDA: 84 DF    >471           sty   selch      ; Save cursor column.
0EDC: 85 E1    >472           sta   changed    ; Mark changed & cont.
0EDE: D0 C6    >473           bne   :kbdlpr    ; (always)
               >474
0EE0: A6 E1    >475  :addchar ldx   changed    ; Any prior change?
0EE2: D0 0D    >476           bne   :notfrst   ; -Yes, just add char.
0EE4: AA       >477           tax             ; Save character.
0EE5: A9 A0    >478           lda   #"         " ; Blank out file name.
0EE7: C0 0C    >479  :cloop   cpy   #fnamecol
0EE9: F0 05    >480           beq   :addit
0EEB: 91 E3    >481           sta   (selBASL),y
0EED: 88       >482           dey
0EEE: D0 F7    >483           bne   :cloop     ; (always)
               >484
0EF0: 8A       >485  :addit   txa             ; Restore character.
0EF1: C0 24    >486  :notfrst cpy   #fnamecol+24 ; At end?
0EF3: B0 AA    >487           bcs   :beep      ; -Yes, complain.
0EF5: 20 FD 0E >488           jsr   inverse    ; -No, make inverse.
0EF8: 91 E3    >489           sta   (selBASL),y ;  and add to fname.
0EFA: C8       >490           iny             ; Advance CH
0EFB: D0 D9    >491           bne   :changed   ; (always)
               >492
0EFD: 29 7F    >493  inverse  and   #$7F       ; Make inverse
0EFF: C9 40    >494           cmp   #$40       ; Upper case?
0F01: 90 06    >495           bcc   :rts       ; -No, special char.
0F03: C9 60    >496           cmp   #$60       ; Upper case?
0F05: B0 02    >497           bcs   :rts       ; -No, lower case.
0F07: 29 1F    >498           and   #$1F       ; -Yes, make inverse
0F09: 60       >499  :rts     rts
               >500
0F0A: AD 7A 0D >501  togcset  lda   charset    ; Get current charset.
0F0D: C9 03    >502           cmp   #'C'&$1F   ; Is it Caltech (inverse C)?
0F0F: 8D 05 C0 >503           sta   WRITAUX    ; Prepare to write to AUX mem.
0F12: F0 11    >504           beq   :setbal    ; -Yes, switch to BALGOL set.
0F14: A9 A8    >505           lda   #"("       ; -No, switch to Caltech set.
0F16: 8D 36 17 >506           sta   b220asc+$10
0F19: A9 AB    >507           lda   #"+"
0F1B: 8D 39 17 >508           sta   b220asc+$13
0F1E: A9 A5    >509           lda   #"%"
0F20: 8D 4A 17 >510           sta   b220asc+$24
0F23: D0 0F    >511           bne   :cont      ; (always)
               >512
0F25: A9 AB    >513  :setbal  lda   #"+"       ; Switch to BALGOL set.
0F27: 8D 36 17 >514           sta   b220asc+$10
0F2A: A9 BB    >515           lda   #";"
0F2C: 8D 39 17 >516           sta   b220asc+$13
0F2F: A9 A8    >517           lda   #"("
0F31: 8D 4A 17 >518           sta   b220asc+$24
0F34: 8D 04 C0 >519  :cont    sta   WRITMAIN   ; Back to writing main mem.
0F37: 84 D7    >520           sty   t1         ; Save horizontal position.
0F39: A0 06    >521           ldy   #6         ; Swap option label.
0F3B: B9 BD 0D >522  :chlp    lda   newcset,y
0F3E: BE 7A 0D >523           ldx   charset,y
0F41: 99 7A 0D >524           sta   charset,y
0F44: 8A       >525           txa
0F45: 99 BD 0D >526           sta   newcset,y
0F48: 88       >527           dey
0F49: 10 F0    >528           bpl   :chlp
0F4B: 30 2A    >529           bmi   ]finish    ; (always)
               >530
0F4D: AD A2 0D >531  togsound lda   sndport    ; Get current sound port.
0F50: C9 13    >532           cmp   #'S'&$1F   ; Is it Speaker (inverse S)?
0F52: F0 04    >533          beq   :setcass   ; -Yes, switch to cassette.
0F54: A9 30    >534          lda   #<SPKR     ; -No, switch to speaker.
0F56: D0 02    >535          bne   :contin    ; (always)
               >536
0F58: A9 20    >537  :setcass lda  #<CASSOUT
0F5A: 8D 05 C0 >538  :contin  sta  WRITAUX    ; Prepare to write to AUX mem.
0F5D: 8D 9A 09 >539          sta   ]X_sound+1
0F60: 8D 04 C0 >540          sta   WRITMAIN   ; Back to writing main mem.
0F63: 84 D7    >541          sty   t1         ; Save horizontal position.
0F65: A0 07    >542          ldy   #7         ; Swap option label.
0F67: B9 C4 0D >543  :sndlp   lda  newsnd,y
0F6A: BE A2 0D >544          ldx   sndport,y
0F6D: 99 A2 0D >545          sta   sndport,y
0F70: 8A       >546          txa
0F71: 99 C4 0D >547          sta   newsnd,y
0F74: 88       >548          dey
0F75: 10 F0    >549          bpl   :sndlp
0F77: 20 7F 0F >550  ]finish  jsr  disopts    ; Display updated options.
0F7A: A4 D7    >551          ldy   t1         ; Restore horizontal position.
0F7C: 4C 3F 0E >552          jmp   ]kbdloop   ; Get next key...
               >553
0F7F: A9 12    >554  disopts  lda  #optline1  ; Set BASL to first
0F81: 20 C1 FB >555          jsr   BASCALC    ;  option line.
0F84: A2 00    >556          ldx   #0
0F86: A0 00    >557          ldy   #0
0F88: BD 6D 0D >558  :ln1lp   lda  optlines,x
0F8B: F0 06    >559          beq   :next
0F8D: 91 28    >560          sta   (BASL),y
0F8F: C8       >561          iny
0F90: E8       >562          inx
0F91: D0 F5    >563          bne   :ln1lp     ; (always)
               >564
0F93: A9 13    >565  :next    lda  #optline1+1 ; Set BASL to second
0F95: 20 C1 FB >566          jsr   BASCALC    ;  option line.
0F98: E8       >567          inx              ; Skip first line null.
0F99: A0 00    >568          ldy   #0
0F9B: BD 6D 0D >569  :ln2lp   lda  optlines,x
0F9E: F0 06    >570          beq   :done
0FA0: 91 28    >571          sta   (BASL),y
0FA2: C8       >572          iny
0FA3: E8       >573          inx
0FA4: D0 F5    >574          bne   :ln2lp     ; (always)
0FA6: 60       >575  :done    rts
```

```
         put   B220PANEL
**********************************************************
*                                                        *
*           B220 front panel display routines            *
*                                                        *
**********************************************************

off      equ   " "        ; blank (neon off)
on       equ   "*"        ; asterisk (neon on)

AR8      equ   $580       ; Line 4
AR4      equ   $600       ; Line 5
AR2      equ   $680       ; Line 6
AR1      equ   $700       ; Line 7
ARv      equ   $428       ; Line 9
BPC8     equ   $5A8       ; Line 12
BPC4     equ   $628       ; Line 13
BPC2     equ   $6A8       ; Line 14
BPC1     equ   $728       ; Line 15
BPCv     equ   $450       ; Line 17
STATlin  equ   $550       ; Line 19

B220col  equ   13-1       ; Leftmost title column
Acol     equ   6-1        ; Leftmost digit column of A
Rcol     equ   24-1       ; Leftmost digit column of R
Bcol     equ   6-1        ; Leftmost digit column of B
Pcol     equ   14-1       ; Leftmost digit column of P
Ccol     equ   22-1       ; Leftmost digit column of C
SW1col   equ   7-1        ; SW 1 column
RUNcol   equ   18-1       ; RUN column
ERRcol   equ   22-1       ; ERR column
COMPcol  equ   26-1       ; COMP column
OFLcol   equ   32-1       ; OFL column
RPTcol   equ   35-1       ; RPT column

* Register label addresses

Alab     equ   AR8+3
Rlab     equ   AR8+21
Blab     equ   BPC8+3
Plab     equ   BPC8+11
Clab     equ   BPC8+19
SWlab    equ   STATlin+3
ERRlab   equ   STATlin+ERRcol+2 ; Error type character
```

```
               >45   * Register front panel attributes
               >46
0FA7: 2D 04 05 >47   Aattr    dw    ARv+Acol,AR1+Acol,AR2+Acol,AR4+Acol,AR8+Acol
0FB1: A3       >48            db    rA+5       ; Low byte of rA
0FB2: 0B       >49            db    12-1       ; Display columns - 1
0FB3: 01 00 01 >50            db    1,0,1,1,1,1,1,1,1,1,1,1 ; Column mask
0FBF: 3F 04 17 >51   Rattr    dw    ARv+Rcol,AR1+Rcol,AR2+Rcol,AR4+Rcol,AR8+Rcol
0FC9: A9       >52            db    rR+5       ; Low byte of rR
0FCA: 0B       >53            db    12-1       ; Display columns - 1
0FCB: 01 00 01 >54            db    1,0,1,1,1,1,1,1,1,1,1,1 ; Column mask
0FD7: 55 04 2D >55   Battr    dw    BPCv+Bcol,BPC1+Bcol,BPC2+Bcol,BPC4+Bcol,BPC8+Bcol
0FE1: 95       >56            db    rB+1       ; Low byte of rB
0FE2: 03       >57            db    4-1        ; Display columns - 1
0FE3: 01 01 01 >58            db    1,1,1,1    ; Column mask
0FE7: 5D 04 35 >59   Pattr    dw    BPCv+Pcol,BPC1+Pcol,BPC2+Pcol,BPC4+Pcol,BPC8+Pcol
0FF1: 97       >60            db    rP+1       ; Low byte of rP
0FF2: 03       >61            db    4-1        ; Display columns - 1
0FF3: 01 01 01 >62            db    1,1,1,1    ; Column mask
0FF7: 65 04 3D >63   Cattr    dw    BPCv+Ccol,BPC1+Ccol,BPC2+Ccol,BPC4+Ccol,BPC8+Ccol
1001: 9D       >64            db    rC+5       ; Low byte of rC
1002: 0D       >65            db    14-1       ; Display columns - 1
1003: 01 00 01 >66            db    1,0,1,1,1,1,0,1,1,0,1,1,1,1 ; Column mask
```

```
               >68   ***********************************************************
               >69   *                                                         *
               >70   *              Initialize B220 Front Panel                *
               >71   *                                                         *
               >72   ***********************************************************
               >73
1011: D8       >74   disppanl cld              ; Force binary mode.
1012: A9 15    >75            lda   #21        ; Disable 80-col firmware
1014: 20 ED FD >76            jsr   COUT
1017: A9 00    >77            lda   #0
1019: 85 22    >78            sta   WNDTOP     ; Set full-screen window.
101B: 20 58 FC >79            jsr   HOME       ; Clear 40-col screen
101E: 8D 0F C0 >80            sta   ALTCHAR    ; Select alternate charset
1021: A2 0B    >81            ldx   #B220col-1
1023: 20 4A F9 >82            jsr   PRBL2      ; Space to starting column
1026: A0 00    >83            ldy   #0
1028: B9 CC 10 >84   :titloop lda   B220msg,y  ; Display title and AR top border
102B: F0 06    >85            beq   :AR
102D: 20 ED FD >86            jsr   COUT
1030: C8       >87            iny
1031: D0 F5    >88            bne   :titloop   ; (always)
               >89
1033: 20 A2 10 >90   :AR      jsr   disARmid   ; Display 8-bit line
1036: 20 A2 10 >91            jsr   disARmid   ; Display 4-bit line
1039: 20 A2 10 >92            jsr   disARmid   ; Display 2-bit line
103C: 20 A2 10 >93            jsr   disARmid   ; Display 1-bit line
103F: A0 00    >94            ldy   #0
1041: B9 E1 10 >95   :ARborlp lda   ARbord,y   ; Display AR bottom border
1044: F0 06    >96            beq   :BPC
1046: 20 ED FD >97            jsr   COUT
1049: C8       >98            iny
104A: D0 F5    >99            bne   :ARborlp   ; (always)
               >100
104C: 20 9A 10 >101  :BPC     jsr   blanklin   ; <blank line for reg values>
104F: 20 9A 10 >102           jsr   blanklin   ; <blank line>
1052: 20 B0 10 >103           jsr   disBPCbo   ; Display BPC top border
1055: 20 BE 10 >104           jsr   disBPCmi   ; Display 8-bit line
1058: 20 BE 10 >105           jsr   disBPCmi   ; Display 4-bit line
105B: 20 BE 10 >106           jsr   disBPCmi   ; Display 2-bit line
105E: 20 BE 10 >107           jsr   disBPCmi   ; Display 1-bit line
1061: 20 B0 10 >108           jsr   disBPCbo   ; Display BPC bottom border
1064: 20 9A 10 >109           jsr   blanklin   ; <blank line for values>
1067: 20 9A 10 >110           jsr   blanklin   ; <blank line>
106A: A0 00    >111           ldy   #0         ; Display Status & Help lines
106C: B9 79 11 >112  :STATlp  lda   STAT,y
106F: F0 06    >113           beq   :finish
1071: 20 ED FD >114           jsr   COUT
1074: C8       >115           iny
1075: D0 F5    >116           bne   :STATlp    ; (always)
               >117
1077: A9 81    >118  :finish  lda   #$81       ; "A" label
1079: 8D 83 05 >119           sta   Alab
107C: A9 82    >120           lda   #$82       ; "B" label
107E: 8D AB 05 >121           sta   Blab
1081: A9 83    >122           lda   #$83       ; "C" label
1083: 8D BB 05 >123           sta   Clab
1086: A9 90    >124           lda   #$90       ; "P" label
1088: 8D B3 05 >125           sta   Plab
108B: A9 92    >126           lda   #$92       ; "R" label
108D: 8D 95 05 >127           sta   Rlab
1090: A9 93    >128           lda   #$93       ; "S" of "Sw"
1092: 8D 53 05 >129           sta   SWlab
1095: A9 14    >130           lda   #20        ; Window is last 4 lines.
1097: 85 22    >131           sta   WNDTOP
1099: 60       >132           rts
               >133
109A: A9 A0    >134  blanklin lda   #"         " ; Separate CRs with blank
```

```
109C: 20 ED FD >135           jsr   COUT
109F: 4C 8E FD >136           jmp   CROUT
               >137
10A2: A0 00    >138  disARmid ldy   #0          ; Display AR middle line
10A4: B9 07 11 >139  :loop    lda   ARmid,y
10A7: F0 06    >140           beq   :rts
10A9: 20 ED FD >141           jsr   COUT
10AC: C8       >142           iny
10AD: D0 F5    >143           bne   :loop       ; (always)
               >144
10AF: 60       >145  :rts     rts
               >146
10B0: A0 00    >147  disBPCbo ldy   #0          ; Display BPC border
10B2: B9 2D 11 >148  :loop    lda   BPCbord,y
10B5: F0 06    >149           beq   :rts
10B7: 20 ED FD >150           jsr   COUT
10BA: C8       >151           iny
10BB: D0 F5    >152           bne   :loop       ; (always)
               >153
10BD: 60       >154  :rts     rts
               >155
10BE: A0 00    >156  disBPCmi ldy   #0          ; Display BPC middle line
10C0: B9 53 11 >157  :loop    lda   BPCmid,y
10C3: F0 06    >158           beq   :rts
10C5: 20 ED FD >159           jsr   COUT
10C8: C8       >160           iny
10C9: D0 F5    >161           bne   :loop       ; (always)
               >162
10CB: 60       >163  :rts     rts
               >164
10CC: C2 F5 F2 >165  B220msg  asc   "Burroughs 220 v2.1"8DA08D
10E1: A0 A0 A0 >166  ARbord   asc   "    +-+----------+    +-+----------+",8D00
1107: A0 A0 A0 >167  ARmid    asc   "    | |          |    | |          |",8D00
112D: A0 A0 A0 >168  BPCbord  asc   "    +----+  +----+  +-+----+--+----+",8D00
1153: A0 A0 A0 >169  BPCmid   asc   "    |    |  |    |  | |    |  |    |",8D00
1179: A0 A0 A0 >170  STAT     asc   "   Sw 0123456789 Run Err < = > Ov Rp",8DA08D
11A0: A0 D3 F4 >171  Help1    asc   " Stop/Step: <space>, Go: G, Reset: Z",8D
11C5: A0 D3 E5 >172  Help2    asc   " Set reg: A/R/B/P/C + digits + Return",8D
11EB: A0 D4 EF >173  Help3    asc   " Toggle switch: S + digit, Help: ?",8D
120E: A0 C9 AF >174  Help4    asc   " I/O Config: I, View CRT: V, Quit: Q",00
               >175
1233: 20 58 FC >176  disphelp jsr   HOME        ; Display help lines.
1236: A0 00    >177           ldy   #0          ; (window is last 4 lines)
1238: B9 A0 11 >178  :helplp  lda   Help1,y
123B: F0 06    >179           beq   :done
123D: 20 ED FD >180           jsr   COUT
1240: C8       >181           iny
1241: D0 F5    >182           bne   :helplp     ; (always)
               >183
1243: 60       >184  :done    rts
```

```
               **********************************************************
               *                                                        *
               *                   Display B220 State                   *
               *                                                        *
               **********************************************************

1244: 20 56 12 display  jsr   dispA      ; Display A
1247: 20 5D 12          jsr   dispR      ; Display R
124A: 20 64 12          jsr   dispB      ; Display B
124D: 20 6B 12          jsr   dispP      ; Display P
1250: 20 72 12          jsr   dispC      ; Display C
1253: 4C 79 12          jmp   dispSTAT   ; Disp Status & return.

1256: A9 A7    dispA    lda   #<Aattr    ; Register A attributes
1258: A0 0F             ldy   #>Aattr
125A: 4C 06 13          jmp   dispreg    ; Display the register.

125D: A9 BF    dispR    lda   #<Rattr    ; Register R attributes
125F: A0 0F             ldy   #>Rattr
1261: 4C 06 13          jmp   dispreg    ; Display the register.

1264: A9 D7    dispB    lda   #<Battr    ; Register B attributes
1266: A0 0F             ldy   #>Battr
1268: 4C 06 13          jmp   dispreg    ; Display the register.

126B: A9 E7    dispP    lda   #<Pattr    ; Register P attributes
126D: A0 0F             ldy   #>Pattr
126F: 4C 06 13          jmp   dispreg    ; Display the register.

1272: A9 F7    dispC    lda   #<Cattr    ; Register C attributes
1274: A0 0F             ldy   #>Cattr
1276: 4C 06 13          jmp   dispreg    ; Display the register.

1279: A9 50    dispSTAT lda   #<STATlin  ; Set ptr to STATlin
127B: 85 D3             sta   ptr
127D: A9 05             lda   #>STATlin
127F: 85 D4             sta   ptr+1
1281: A2 00             ldx   #0
1283: A0 06             ldy   #SW1col    ; Start at switch 1
1285: B5 B6    :swloop  lda   CSW,x      ; Is it on?
1287: 20 DD 12          jsr   INDshow    ; Display it's state
128A: E8                inx              ; Next switch
128B: E0 0A             cpx   #10        ; Until done...
128D: 90 F6             bcc   :swloop
128F: A0 11             ldy   #RUNcol
1291: A5 C0             lda   RUN
1293: 20 DD 12          jsr   INDshow
1296: A0 15             ldy   #ERRcol
1298: A5 C1             lda   ERR
129A: 20 DD 12          jsr   INDshow
129D: A0 19             ldy   #COMPcol
129F: A5 C2             lda   COMP       ; <0, 0, >0: < = >
12A1: 30 07             bmi   :lt
12A3: F0 0A             beq   :eq
12A5: A2 0C             ldx   #:gtstr-:ltstr ; Point to > string
12A7: 4C B1 12          jmp   :show

12AA: A2 00    :lt      ldx   #:ltstr-:ltstr ; Point to < string
12AC: 4C B1 12          jmp   :show

12AF: A2 06    :eq      ldx   #:eqstr-:ltstr ; Point to = string
12B1: BD CB 12 :show    lda   :ltstr,x
12B4: F0 06             beq   :next
12B6: 91 D3             sta   (ptr),y
12B8: C8                iny
12B9: E8                inx
12BA: D0 F5             bne   :show      ; (always)
```

```

12BC: A0 1F    :next    ldy   #OFLcol
12BE: A5 C3             lda   Ov          ; Overflow indicator
12C0: 20 DD 12          jsr   INDshow
12C3: A0 22             ldy   #RPTcol
12C5: A5 C4             lda   Rp          ; Repeat indicator
12C7: 20 DD 12          jsr   INDshow
12CA: 60                rts

12CB: 3C       :ltstr   asc   '<' ; Inverse
12CC: A0 BD A0          asc   " = >",00
12D1: BC A0    :eqstr   asc   "< "
12D3: 3D                asc   '=' ; Inverse
12D4: A0 BE 00          asc   " >",00
12D7: BC A0 BD :gtstr   asc   "< = "
12DB: 3E 00             asc   '>',00 ; inverse

               **********************************************************
               *                                                        *
               *     Flip indicator to on (inverse) or off (normal)     *
               *                                                        *
               * A = indicator: 0 is OFF, >0 is ON                      *
               * Y = leftmost column of indicator - 1                   *
               * Exits with Y pointing 1 past last column of indicator  *
               *                                                        *
               **********************************************************

12DD: 18       INDshow  clc               ; >0 ==> inv, 0 ==> norm
12DE: 69 FF             adc   #$FF        ; Set C if >0, reset if 0
12E0: B1 D3    :loop    lda   (ptr),y     ; Get indicator char
12E2: 29 20             and   #$20        ; Is it Upper Case?
12E4: D0 06             bne   :notuc      ; -No, leave it alone.
12E6: B1 D3             lda   (ptr),y     ; -Yes, turn off $40 bit
12E8: 29 BF             and   #$BF        ;   to avoid mousetext.
12EA: D0 02             bne   :switch     ; (always)

12EC: B1 D3    :notuc   lda   (ptr),y     ; Recover original char
12EE: 90 04    :switch  bcc   :norm       ; Set to normal
12F0: 29 7F             and   #$7F        ; Set to inverse
12F2: B0 02             bcs   :store      ; (always)

12F4: 09 80    :norm    ora   #$80        ; Set to normal
12F6: 91 D3    :store   sta   (ptr),y
12F8: C8                iny               ; Advance to next char
12F9: B1 D3             lda   (ptr),y     ;  and examine it.
12FB: 09 80             ora   #$80        ; Force normal
12FD: 49 A0             eor   #"          " ; Space?
12FF: F0 04             beq   :done       ; -Yes, done.
1301: 29 E0             and   #$E0        ; -No, digit?
1303: D0 DB             bne   :loop       ; -No, keep going.
1305: 60       :done    rts               ; -Yes, done.
```

```
                >305  **********************************************************
                >306  *                                                        *
                >307  *         Display a B220 register on front panel         *
                >308  *                                                        *
                >309  * Address of register attributes block is loaded in A,Y  *
                >310  *                                                        *
                >311  **********************************************************
                >312
1306: 85 D3     >313  dispreg  sta   ptr          ; Set register attribute ptr
1308: 84 D4     >314           sty   ptr+1
130A: A0 00     >315           ldy   #0
130C: B1 D3     >316  :cpyattr lda   (ptr),y      ; Copy reg attributes to page 0
130E: 99 DB 00  >317           sta   linev,y
1311: C8        >318           iny
1312: C0 0A     >319           cpy   #10
1314: 90 F6     >320           bcc   :cpyattr
1316: B1 D3     >321           lda   (ptr),y      ; Addr of low byte of register
1318: 8D 2B 13  >322           sta   :reg+1
131B: C8        >323           iny
131C: B1 D3     >324           lda   (ptr),y
131E: A8        >325           tay                ; Set Y = rightmost column
131F: 18        >326           clc
1320: A5 D3     >327           lda   ptr          ; Advance ptr to digit mask
1322: 69 0C     >328           adc   #12
1324: 85 D3     >329           sta   ptr
1326: 90 02     >330           bcc   :reg
1328: E6 D4     >331           inc   ptr+1
132A: A5 00     >332  :reg     lda   0*0          ; Load register byte
132C: CE 2B 13  >333           dec   :reg+1       ;  and move to next highest.
132F: 85 D7     >334           sta   t1           ; Save current reg byte
1331: 20 44 13  >335           jsr   dispdig      ; Display lo digit of reg byte
1334: 88        >336           dey                ; Move left one column.
1335: 30 0C     >337           bmi   :done        ; Quit if done...
1337: 20 44 13  >338           jsr   dispdig      ; Display hi digit of reg byte
133A: 88        >339  :skip    dey                ; Move left.
133B: 30 06     >340           bmi   :done        ; -Display complete.
133D: B1 D3     >341           lda   (ptr),y      ; Check mask
133F: F0 F9     >342           beq   :skip        ; -Skip this screen column
1341: D0 E7     >343           bne   :reg         ; -Keep going...
                >344
1343: 60        >345  :done    rts
                >346
                *********************************************************
                *                                                       *
                *           Display one digit of B220 register          *
                *                                                       *
                *********************************************************

1344: A5 D7     dispdig  lda   t1          ; Get (shifted) reg byte.
1346: 29 0F              and   #$0F        ; Mask low digit,
1348: 09 B0              ora   #$B0        ;  make ASCII digit,
134A: 91 DB              sta   (linev),y   ;   and store it on screen.
134C: 46 D7              lsr   t1          ; 1-bit to Carry
134E: A9 A0              lda   #off
1350: 90 02              bcc   :st1
1352: A9 AA              lda   #on
1354: 91 DD     :st1     sta   (line1),y   ; Store 1-bit state to screen
1356: 46 D7              lsr   t1          ; 2-bit to Carry
1358: A9 A0              lda   #off
135A: 90 02              bcc   :st2
135C: A9 AA              lda   #on
135E: 91 DF     :st2     sta   (line2),y   ; Store 2-bit state to screen
1360: 46 D7              lsr   t1          ; 4-bit to Carry
1362: A9 A0              lda   #off
1364: 90 02              bcc   :st4
1366: A9 AA              lda   #on
1368: 91 E1     :st4     sta   (line4),y   ; Store 4-bit state to screen
136A: 46 D7              lsr   t1          ; 8-bit to Carry
136C: A9 A0              lda   #off
136E: 90 02              bcc   :st8
1370: A9 AA              lda   #on
1372: 91 E3     :st8     sta   (line8),y   ; Store 8-bit state to screen
1374: 60                 rts
```

```
                         put   B220IO
                ********************************************************
                *                                                      *
                *               B220 Buffered I/O Routines             *
                *                                                      *
                ********************************************************

                * File/Buffer Parameters

                fnlen    equ   25         ; File name max length
                ptbfsz   equ   100*6      ; Paper tape buf: 100 words.
                blksize  equ   101*6      ; block = Preface + 100 words.
                mtbfsz   equ   10*blksize ; Mag Tape buf: 6060 bytes.

                db       equ   *          ; Device Information Block

1375: 4C 3B     bfstart  dw    ptrdr0bf   ; Paper tape reader 0 buffer
1377: 4C 3B     bfptr    dw    ptrdr0bf   ; Current buf pointer
1379: A4 3D     bfend    dw    ptrdr0bf+ptbfsz ; End of buffer + 1
137B: 58 02     bfsiz    dw    ptbfsz     ; Buffer size in bytes
137D: 4D 04     bfscrn   dw    $428+37    ; Device activity screen addr
137F: D2        bfclasch asc   "R" ; Device class character
1380: 30        bfunitch asc   '0' ; Device unit character
1381: 00        bffn     db    0*fnlen    ; File name table index
1382: 00 00 00  bfoff    db    0,0,0      ; bfstart file offset
1385: 00        bflane   db    0          ; Mag tape lane = 0 or 1
1386: 00        bfdirty  db    0          ; Buffer contents changed

                dbsz     equ   *-db       ; DB size

1387: A6 3D              dw    ptrdr1bf   ; Paper tape reader 1 buffer
1389: A6 3D              dw    ptrdr1bf
138B: FE 3F              dw    ptrdr1bf+ptbfsz
138D: 58 02              dw    ptbfsz
138F: 4D 04              dw    $428+37
1391: D2                 asc   "R"
1392: 31                 asc   '1'
1393: 19                 db    1*fnlen
1394: 00 00 00           db    0,0,0
1397: 00                 db    0
1398: 00                 db    0

1399: 00 60              dw    ptpch0bf   ; Paper tape punch 0 buffer
139B: 00 60              dw    ptpch0bf
139D: 58 62              dw    ptpch0bf+ptbfsz
139F: 58 02              dw    ptbfsz
13A1: 4D 05              dw    $528+37
13A3: D0                 asc   "P"
13A4: 30                 asc   '0'
13A5: 32                 db    2*fnlen
13A6: 00 00 00           db    0,0,0
13A9: 00                 db    0
13AA: 00                 db    0

13AB: 5A 62              dw    ptpch1bf   ; Paper tape punch 1 buffer
13AD: 5A 62              dw    ptpch1bf
13AF: B2 64              dw    ptpch1bf+ptbfsz
13B1: 58 02              dw    ptbfsz
13B3: 4D 05              dw    $528+37
13B5: D0                 asc   "P"
13B6: 31                 asc   '1'
13B7: 4B                 db    3*fnlen
13B8: 00 00 00           db    0,0,0
13BB: 00                 db    0
13BC: 00                 db    0

13BD: B4 64              dw    mt0bf      ; Mag tape 0 buffer
```

```
13BF: B4 64    >67            dw    mt0bf
13C1: 60 7C    >68            dw    mt0bf+mtbfsz
13C3: AC 17    >69            dw    mtbfsz
13C5: 4D 06    >70            dw    $628+37
13C7: CD       >71            asc   "M"
13C8: 30       >72            asc   '0'
13C9: 64       >73            db    4*fnlen    ; (Lane 0)
13CA: 00 00 00 >74            db    0,0,0
13CD: 00       >75            db    0
13CE: 00       >76            db    0
               >77
13CF: 62 7C    >78            dw    mt1bf      ; Mag tape 1 buffer
13D1: 62 7C    >79            dw    mt1bf
13D3: 0E 94    >80            dw    mt1bf+mtbfsz
13D5: AC 17    >81            dw    mtbfsz
13D7: 4D 06    >82            dw    $628+37
13D9: CD       >83            asc   "M"
13DA: 31       >84            asc   '1'
13DB: 96       >85            db    6*fnlen    ; (Lane 0)
13DC: 00 00 00 >86            db    0,0,0
13DF: 00       >87            db    0
13E0: 00       >88            db    0
               >89
               >90   PTRclass equ   0          ; Paper Tape Reader class
               >91   PTPclass equ   2          ; Paper Tape Punch class
               >92   MTUclass equ   4          ; Mag Tape class
               >93
               >94   * Map Device Class + Unit ==> Device Block index
13E1: 00 12 24 >95   classdbx db    0*dbsz,1*dbsz,2*dbsz
13E4: 36 48 5A >96            db    3*dbsz,4*dbsz,5*dbsz
               >97
               >98   * Map filename index ==> Device Block index
13E7: 00 12 24 >99   fnxdbx   db    0*dbsz,1*dbsz,2*dbsz,3*dbsz
13EB: 48 48 5A >100           db    4*dbsz,4*dbsz,5*dbsz,5*dbsz
               >101
               >102  * Map filename index ==> fn table index
13EF: 00 19 32 >103  fnxfn    db    0*fnlen,1*fnlen,2*fnlen,3*fnlen
13F3: 64 7D 96 >104           db    4*fnlen,5*fnlen,6*fnlen,7*fnlen
               >105
               >106  * I/O buffer definitions
               >107
               >108  ptrdr1bf equ   $4000-ptbfsz-2    ; Two PTRDR buffers
               >109  ptrdr0bf equ   ptrdr1bf-ptbfsz-2 ;  just below HGR2.
               >110
               >111           dum   $6000      ; Buffers in high Main mem
6000: 00 00 00 >112  ptpch0bf ds    ptbfsz+2
625A: 00 00 00 >113  ptpch1bf ds    ptbfsz+2
64B4: 00 00 00 >114  mt0bf    ds    mtbfsz+2
7C62: 00 00 00 >115  mt1bf    ds    mtbfsz+2
               >116           err   */$9600    ; Error if past $9600
               >117           dend
               >119  * File name table
               >120
               >121           align 256        ; Put table on page boundary
13F7: 00 00 00 >121           ds    *-1/256*256+256-*
               >121           eom
               >122
1400: D0 D4 D2 >123  fnames   asc   "PTRDR0",00
1407: 00 00 00 >124           ds    fnlen-7
1419: D0 D4 D2 >125           asc   "PTRDR1",00
1420: 00 00 00 >126           ds    fnlen-7
1432: D0 D4 D0 >127           asc   "PTPCH0",00
1439: 00 00 00 >128           ds    fnlen-7
144B: D0 D4 D0 >129           asc   "PTPCH1",00
1452: 00 00 00 >130           ds    fnlen-7
1464: CD D4 D5 >131           asc   "MTU0L0",00
146B: 00 00 00 >132           ds    fnlen-7
147D: CD D4 D5 >133           asc   "MTU0L1",00
1484: 00 00 00 >134           ds    fnlen-7
1496: CD D4 D5 >135           asc   "MTU1L0",00
149D: 00 00 00 >136           ds    fnlen-7
14AF: CD D4 D5 >137           asc   "MTU1L1",00
14B6: 00 00 00 >138           ds    fnlen-7
```

```
               ***********************************************************
               *                                                         *
               *             iodsel - Deselect I/O device                *
               *                                                         *
               * On entry: dbx = DB index.                               *
               * On exit:  X = DB index, bfptr = ptr.                    *
               *                                                         *
               ***********************************************************


14C8: A6 D9    iodsel   ldx   dbx        ; DB index.
14CA: A5 D3             lda   ptr        ; bfptr = ptr.
14CC: 9D 77 13          sta   bfptr,x
14CF: A5 D4             lda   ptr+1
14D1: 9D 78 13          sta   bfptr+1,x
14D4: BD 7D 13          lda   bfscrn,x   ; Set 'ptr' to device
14D7: 85 D3             sta   ptr        ;  activity screen address.
14D9: BD 7E 13          lda   bfscrn+1,x
14DC: 85 D4             sta   ptr+1
14DE: A0 01             ldy   #1
14E0: A9 A0             lda   #"          " ; Blank device unit
14E2: 91 D3             sta   (ptr),y    ;  activity indicator.
14E4: 60                rts

               ***********************************************************
               *                                                         *
               *              iosel - Select I/O device                  *
               *                                                         *
               * On entry: X = Device Class (0=RDR, 2=PCH, 4=MTP)        *
               * On exit:  X = dbx = DB index, ptr = bfptr,              *
               *           A = (ptr) = sign (flag) byte of next word.    *
               *                                                         *
               ***********************************************************

14E5: A5 99    iosel    lda   rC+sL      ; Get unit number.
14E7: 29 E0             and   #$E0       ; Unit number > 0 or 1?
14E9: D0 6B             bne   ]IOerr1    ; -Yes, I/O error.
14EB: A5 99             lda   rC+sL      ; Get unit number
14ED: 29 10             and   #$10
14EF: F0 01             beq   :zero      ; Unit 0
14F1: E8                inx              ; Unit 1
14F2: BD E1 13 :zero    lda   classdbx,x ; Map class + unit to DB index.
14F5: AA                tax
14F6: 85 D9             sta   dbx        ; Save DB index.
14F8: BD 7D 13          lda   bfscrn,x   ; Set 'ptr' to device
14FB: 85 D3             sta   ptr        ;  activity screen address.
14FD: BD 7E 13          lda   bfscrn+1,x
1500: 85 D4             sta   ptr+1
1502: A0 00             ldy   #0
1504: BD 7F 13          lda   bfclasch,x ; Device class (R, P, M)
1507: 91 D3             sta   (ptr),y
1509: C8                iny
150A: BD 80 13          lda   bfunitch,x ; Device unit number
150D: 91 D3             sta   (ptr),y
150F: BD 77 13 setptr   lda   bfptr,x
1512: 85 D3             sta   ptr        ; ptr = bfptr
1514: BD 78 13          lda   bfptr+1,x
1517: 85 D4             sta   ptr+1
1519: A2 00             ldx   #0
151B: A1 D3             lda   (ptr,x)    ; A = sign byte of next word.
151D: A6 D9             ldx   dbx        ; Restore X.
151F: 60                rts
```

```
               ***********************************************************
               *                                                         *
               *       getwrd - Get next word from buffer into rD        *
               *                                                         *
               * On entry: ptr = pointer to next word in buffer,         *
               *           dbx = DB index.                               *
               * On exit:  rD = next word in buffer, ptr advanced.       *
               *                                                         *
               ***********************************************************

1520: A0 00    getwrd   ldy   #0          ; Sign flag: EOF, EOB/Empty,
1522: B1 D3             lda   (ptr),y     ;  normal/Prefix?
1524: C9 BA    :again   cmp   #PREF+$A    ; Normal or prefix word?
1526: B0 18             bcs   :special    ; -No, EOF, EOB, or EMPTY.
1528: 85 AA             sta   rD+S        ; -Yes, put sign in rD and
152A: A0 05             ldy   #5          ;   copy rest of word to rD.
152C: B1 D3    :getlp   lda   (ptr),y
152E: 99 AA 00          sta   rD,y
1531: 88                dey
1532: D0 F8             bne   :getlp
1534: 18       ]incptr6 clc               ; Increment ptr by 6.
1535: A5 D3             lda   ptr
1537: 69 06             adc   #$6
1539: 85 D3             sta   ptr
153B: 90 02             bcc   :rts
153D: E6 D4             inc   ptr+1
153F: 60       :rts     rts

1540: A6 D9    :special ldx   dbx         ; Point to Device Block.
1542: C9 EF             cmp   #EOF        ; End-Of-File?
1544: F0 10             beq   ]IOerr1     ; -Yes, I/O error.
1546: C9 EE             cmp   #EMPTY      ; -No. Is buffer empty?
1548: F0 06             beq   :load       ; -Yes, load buffer.
154A: 20 23 17          jsr   flushbuf    ; -No, EOB. Flush buf to disk.
154D: 20 4F 16          jsr   advoff      ; Advance buf offset.
1550: 20 9A 15 :load    jsr   readbuf     ; Load the buffer
1553: 4C 24 15          jmp   :again      ;  and try again.

1556: 4C 21 08 ]IOerr1  jmp   X_IOerr     ; I/O error relay.
```

```
                ***********************************************************
                *                                                         *
                *     putwrd - Put rD into next buffer word.              *
                *                                                         *
                * On entry: dbx = DB index, ptr current.                  *
                * On exit:  rD = next word in buffer, ptr advanced.       *
                *                                                         *
                ***********************************************************

1559: A6 D9     putwrd   ldx   dbx         ; DB index.
155B: BD 79 13           lda   bfend,x     ; Is buffer full?
155E: C5 D3              cmp   ptr
1560: D0 15              bne   :notfull    ; -No, check empty.
1562: BD 7A 13           lda   bfend+1,x
1565: C5 D4              cmp   ptr+1
1567: D0 0E              bne   :notfull    ; -No, check empty.
1569: 20 23 17           jsr   flushbuf    ; -Yes, write if dirty,
156C: 20 4F 16           jsr   advoff      ;   advance offset, and
156F: A9 EE              lda   #EMPTY      ;    mark buffer empty.
1571: A0 00              ldy   #0
1573: 91 D3              sta   (ptr),y
1575: F0 08              beq   :ckmtape    ; (always)

1577: A0 00     :notfull ldy   #0
1579: B1 D3              lda   (ptr),y
157B: C9 EE              cmp   #EMPTY      ; Is buffer empty?
157D: D0 0A              bne   :put        ; -No, put word.
157F: BD 7C 13  :ckmtape lda   bfsiz+1,x   ; -Yes, is device
1582: C9 17              cmp   #>mtbfsz       a mag tape?
1584: D0 03              bne   :put        ; -No. Put the word.
1586: 20 9A 15           jsr   readbuf     ; -Yes, load the buffer.
1589: A9 01     :put     lda   #1          ; Mark buffer dirty.
158B: 9D 86 13           sta   bfdirty,x
158E: A0 05              ldy   #5          ; Move rD into buffer.
1590: B9 AA 00  :putlp   lda   rD,y
1593: 91 D3              sta   (ptr),y
1595: 88                 dey
1596: 10 F8              bpl   :putlp
1598: 30 9A              bmi   ]incptr6    ; Inc ptr & return. (always)

                ***********************************************************
                *                                                         *
                *                      readbuf                            *
                *                                                         *
                * On entry: dbx = DB index.                               *
                * On exit:  X = dbx = DB index, Y = 0, ptr = bfstart,     *
                *           A = (ptr) = sign (flag) byte of next word.    *
                *                                                         *
                ***********************************************************

159A: 20 D2 16  readbuf  jsr   emptydb     ; Clear the buffer.
159D: 20 79 17           jsr   doread      ; Fill the buffer.
15A0: A0 00              ldy   #0
15A2: B1 D3              lda   (ptr),y     ; A = sign byte of next word.
15A4: 60        ]rts     rts
```

```
               **********************************************************
               *                                                        *
               *      nxtblk - Advance ptr to point at next block.      *
               *                                                        *
               * On entry: X = DB index, A = (ptr) = sign flag.         *
               * On exit:  X unchanged, (ptr) = next block.             *
               *           I/O error if at EOF (unless op = MPE).       *
               *                                                        *
               **********************************************************

15A5: 20 E3 15 nxtblk   jsr   ckpref     ; Position ptr at block preface.
15A8: C9 EF    :nxt     cmp   #EOF       ; At End-Of-File?
15AA: F0 14             beq   :ckmpe     ; -Yes, check for MPE.
15AC: C9 EE             cmp   #EMPTY     ; -No. Is buffer empty?
15AE: F0 0A             beq   :loadbf    ; -Yes, just load buffer.
15B0: C9 EB             cmp   #EOB       ; -No. At End-Of-Buffer?
15B2: D0 1D             bne   incblk     ; -No, just inc to next block.
15B4: 20 23 17          jsr   flushbuf   ; -Yes, flush the buffer,
15B7: 20 4F 16          jsr   advoff     ;       advance buf offset,
15BA: 20 9A 15 :loadbf  jsr   readbuf    ;        and fill the buffer.
15BD: 4C A8 15          jmp   :nxt       ; Go again in fresh buffer.

15C0: A5 9B    :ckmpe   lda   rC+OP      ; MPE opcode?
15C2: C9 58             cmp   #$58
15C4: D0 90             bne   ]IOerr1    ; -No, I/O error.
15C6: A5 9A             lda   rC+VV      ; MPE variant?
15C8: 29 0F             and   #$0F
15CA: C9 02             cmp   #2
15CC: D0 88             bne   ]IOerr1    ; -No, I/O error.
15CE: B1 D3             lda   (ptr),y    ; -Yes, return with
15D0: 60                rts              ;       flag byte.

15D1: 18       incblk   clc              ; ptr = ptr + blksize.
15D2: A5 D3             lda   ptr
15D4: 69 5E             adc   #<blksize
15D6: 85 D3             sta   ptr
15D8: A5 D4             lda   ptr+1
15DA: 69 02             adc   #>blksize
15DC: 85 D4             sta   ptr+1
15DE: A0 00             ldy   #0
15E0: B1 D3             lda   (ptr),y    ; A = (ptr) = sign/flag byte.
15E2: 60                rts

15E3: A0 00    ckpref   ldy   #0         ; Position ptr to point
15E5: B1 D3    :ck      lda   (ptr),y    ;  at preface of current block.
15E7: C9 B0             cmp   #PREF
15E9: 90 01             bcc   :backup    ; If not there, back up.
15EB: 60                rts

15EC: 38       :backup  sec              ; ptr = ptr - 6.
15ED: A5 D3             lda   ptr
15EF: E9 06             sbc   #6
15F1: 85 D3             sta   ptr
15F3: B0 F0             bcs   :ck        ; No borrow. Check again.
15F5: C6 D4             dec   ptr+1
15F7: D0 EC             bne   :ck        ; Check again. (always)
```

```
               **********************************************************
               *                                                        *
               *      prvblk - Adjust ptr to point at previous block.   *
               *                                                        *
               * On entry: X = DB index.                                *
               * On exit:  X unchanged, A = (ptr) = next block, Y = 0.  *
               *           I/O error if at beginning of file.           *
               *                                                        *
               **********************************************************

15F9: 20 E3 15 prvblk   jsr   ckpref     ; Position ptr at block preface.
15FC: A5 D3             lda   ptr        ; Is ptr at start of buffer?
15FE: DD 75 13          cmp   bfstart,x
1601: D0 1A             bne   decblk     ; -No, just decrement ptr.
1603: A5 D4             lda   ptr+1
1605: DD 76 13          cmp   bfstart+1,x
1608: D0 13             bne   decblk     ; -No, just decrement ptr.
160A: 20 23 17          jsr   flushbuf   ; -Yes, flush the buffer,
160D: 20 2F 16          jsr   backoff    ;        back to prev buffer,
1610: 20 9A 15          jsr   readbuf    ;         and fill the buffer.
1613: BD 79 13          lda   bfend,x    ; ptr = bfend.
1616: 85 D3             sta   ptr
1618: BD 7A 13          lda   bfend+1,x
161B: 85 D4             sta   ptr+1
161D: 38       decblk   sec              ; ptr = ptr - blksize
161E: A5 D3             lda   ptr
1620: E9 5E             sbc   #<blksize
1622: 85 D3             sta   ptr
1624: A5 D4             lda   ptr+1
1626: E9 02             sbc   #>blksize
1628: 85 D4             sta   ptr+1
162A: A0 00             ldy   #0         ; A = (ptr) = sign/flag byte.
162C: B1 D3             lda   (ptr),y
162E: 60                rts
```

```
                **********************************************************
                *                                                        *
                *      backoff - Back up bfoff by length of buffer.      *
                *                                                        *
                * On entry: X = DB index                                 *
                * On exit:  X unchanged, bfoff backed up, ptr = bfstart. *
                *           I/O error if offset goes below zero.         *
                *                                                        *
                **********************************************************

162F: 38        backoff  sec               ; bfoff = bfoff - bfsiz.
1630: BD 82 13           lda   bfoff,x
1633: FD 7B 13           sbc   bfsiz,x
1636: 9D 82 13           sta   bfoff,x
1639: BD 83 13           lda   bfoff+1,x
163C: FD 7C 13           sbc   bfsiz+1,x
163F: 9D 83 13           sta   bfoff+1,x
1642: BD 84 13           lda   bfoff+2,x
1645: E9 00              sbc   #0
1647: 9D 84 13           sta   bfoff+2,x
164A: 10 1B              bpl   ]resptr    ; If +, set ptr = bfstart.
164C: 4C 21 08           jmp   X_IOerr    ; Error if offset  0.

                **********************************************************
                *                                                        *
                *      advoff - Advance bfoff by length of buffer.       *
                *                                                        *
                * On entry: X = DB index                                 *
                * On exit:  X unchanged, bfoff advanced, ptr = bfstart.  *
                *                                                        *
                **********************************************************

164F: 18        advoff   clc               ; bfoff = bfoff + bfsiz.
1650: BD 82 13           lda   bfoff,x
1653: 7D 7B 13           adc   bfsiz,x
1656: 9D 82 13           sta   bfoff,x
1659: BD 83 13           lda   bfoff+1,x
165C: 7D 7C 13           adc   bfsiz+1,x
165F: 9D 83 13           sta   bfoff+1,x
1662: 90 03              bcc   ]resptr
1664: FE 84 13           inc   bfoff+2,x
1667: BD 75 13  ]resptr  lda   bfstart,x  ; ptr = bfstart.
166A: 85 D3              sta   ptr
166C: BD 76 13           lda   bfstart+1,x
166F: 85 D4              sta   ptr+1
1671: 60                 rts
```

```
               ***********************************************************
               *                                                         *
               *                setlan - Set MTU lane                    *
               *                                                         *
               * On entry: X = dbx = DB index                            *
               * On exit:  X unchanged, A = filename index               *
               *                                                         *
               ***********************************************************

1672: A5 9A    setlan   lda   rC+VV      ; Isolate lane #.
1674: 29 10             and   #$10
1676: F0 02             beq   :zero      ; Lane 0.
1678: A9 01             lda   #1         ; Lane 1.
167A: DD 85 13 :zero    cmp   bflane,x   ; Lane change?
167D: F0 1C             beq   :done      ; -No, done.
167F: 48                pha              ; -Yes, save new lane,
1680: 20 0F 15          jsr   setptr     ;   ptr = bfptr(dbx).
1683: 20 23 17          jsr   flushbuf   ;    Flush current buffer.
1686: 20 D2 16          jsr   emptydb    ;     and set buffer empty.
1689: 68                pla
168A: 9D 85 13          sta   bflane,x   ; Set new lane
168D: A8                tay              ; Compute new filname index.
168E: EC E5 13          cpx   classdbx+4 ; Mag Tape unit 0 or 1?
1691: F0 02             beq   :unit0     ; -Unit 0 ==> fnx = 4 + lane
1693: C8                iny              ; -Unit 1 ==> fnx = 6 + lane
1694: C8                iny
1695: B9 F3 13 :unit0   lda   fnxfn+4,y  ; Get new lane filename
1698: 9D 81 13          sta   bffn,x     ;  index and save it.
169B: 60       :done    rts

               ***********************************************************
               *                                                         *
               *     ckspo - Check if PWR has been rerouted to SPO.      *
               *                                                         *
               * On entry: dbx = DB index for punch device.              *
               * On exit:  X = 0, flags EQUAL if filename = 'SPO',       *
               *           UNEQUAL otherwise.                            *
               *                                                         *
               ***********************************************************

169C: BC 81 13 ckspo    ldy   bffn,x     ; Get index to filename.
169F: A2 03             ldx   #3         ; Compare filename to 'SPO'.
16A1: BD AF 16 :cmplp   lda   :spostr,x
16A4: D9 03 14          cmp   fnames+3,y
16A7: D0 05             bne   :notspo
16A9: 88                dey
16AA: CA                dex
16AB: 10 F4             bpl   :cmplp
16AD: E8                inx              ; Set flags EQUAL (X = 0).
16AE: 60       :notspo  rts

16AF: D3 D0 CF :spostr  asc   "SPO",00
```

```
               **********************************************************
               *                                                        *
               *                       resetdbs                         *
               *                                                        *
               **********************************************************

16B3: A0 05    resetdbs ldy   #ndb-1     ; Reset all Devices
16B5: BE E1 13 :resetlp ldx   classdbx,y
16B8: 86 D9             stx   dbx        ; DB index
16BA: 20 0F 15          jsr   setptr     ; ptr = bfptr(dbx).
16BD: 20 C4 16          jsr   resetdb
16C0: 88                dey
16C1: 10 F2             bpl   :resetlp
16C3: 60                rts

               **********************************************************
               *                                                        *
               *                       resetdb                          *
               *                                                        *
               * On entry: X = DB index                                 *
               * On exit:  dbx = X = DB index, Y unchanged,             *
               *           Buffer cleared and set to EMPTY.             *
               *                                                        *
               **********************************************************

16C4: 20 23 17 resetdb  jsr   flushbuf   ; Flush buffer.
16C7: A9 00             lda   #0
16C9: 9D 82 13          sta   bfoff,x    ; Set offset = 0
16CC: 9D 83 13          sta   bfoff+1,x
16CF: 9D 84 13          sta   bfoff+2,x
16D2: BD 75 13 emptydb  lda   bfstart,x  ; ptr = bfptr = bfstart.
16D5: 9D 77 13          sta   bfptr,x
16D8: 85 D3             sta   ptr
16DA: BD 76 13          lda   bfstart+1,x
16DD: 9D 78 13          sta   bfptr+1,x
16E0: 85 D4             sta   ptr+1
16E2: 98                tya              ; Save Y.
16E3: 48                pha
16E4: A0 00             ldy   #0
16E6: A9 EE             lda   #EMPTY     ; Mark buffer empty
16E8: D0 02             bne   :store     ; Store EMPTY flag. (always)

16EA: A9 00    :clearlp lda   #0         ; Clear buffer flag bytes.
16EC: 91 D3    :store   sta   (ptr),y    ; Store flag byte.
16EE: 20 34 15          jsr   ]incptr6
16F1: A5 D4             lda   ptr+1      ; At end of buffer?
16F3: DD 7A 13          cmp   bfend+1,x
16F6: 90 F2             bcc   :clearlp   ; -No, keep clearing flags.
16F8: A5 D3             lda   ptr
16FA: DD 79 13          cmp   bfend,x
16FD: D0 EB             bne   :clearlp
16FF: A9 EB             lda   #EOB       ; -Yes, set End-Of-Buffer
1701: 91 D3             sta   (ptr),y    ;   after final block.
1703: BD 75 13          lda   bfstart,x  ; ptr = bfstart.
1706: 85 D3             sta   ptr
1708: BD 76 13          lda   bfstart+1,x
170B: 85 D4             sta   ptr+1
170D: 68                pla              ; Restore Y.
170E: A8                tay
170F: 60                rts
```

```
               **********************************************************
               *                                                        *
               *                      flushall                          *
               *                                                        *
               **********************************************************

1710: A0 05    flushall ldy   #ndb-1     ; Flush all but PTR buffers.
1712: BE E1 13 :flushlp ldx   classdbx,y ; DB index
1715: 86 D9             stx   dbx        ; Set dbx.
1717: 20 0F 15          jsr   setptr     ; ptr = bfptr(dbx)
171A: 20 23 17          jsr   flushbuf   ; Flush a buffer
171D: 88                dey
171E: C0 01             cpy   #1         ; Go until PTR buffers
1720: D0 F0             bne   :flushlp   ;  (1 and 0) are reached.
1722: 60                rts

               **********************************************************
               *                                                        *
               *                      flushbuf                          *
               *                                                        *
               * On entry: X = DB index                                 *
               * On exit:  Buffer clean, ptr, bfptr, bfoff unchanged.   *
               *           X,Y unchanged, A scrambled, dbx = DB index.  *
               *                                                        *
               **********************************************************

1723: 86 D9    flushbuf stx   dbx        ; Set Device Block index.
1725: BD 86 13          lda   bfdirty,x  ; Does buf need to be written?
1728: F0 09             beq   :clean     ; -No, it's clean.
172A: 98                tya              ; -Yes, save Y
172B: 48                pha
172C: 20 34 17          jsr   dowrite    ;   and do it...
172F: 68                pla              ; Restore Y.
1730: A8                tay
1731: A6 D9             ldx   dbx        ; Restore X.
1733: 60       :clean   rts
```

```
               **********************************************************
               *                                                        *
               *                       dowrite                          *
               *                                                        *
               * On entry: dbx = DB index, ptr = current                *
               * On exit:  X = dbx, bfptr = ptr (unchanged), buf clean. *
               *                                                        *
               **********************************************************

1734: A6 D9    dowrite  ldx   dbx        ; Get DB index.
1736: A5 D3             lda   ptr        ; Save 'ptr' in 'bfptr'.
1738: 9D 77 13          sta   bfptr,x
173B: A5 D4             lda   ptr+1
173D: 9D 78 13          sta   bfptr+1,x
1740: A9 18             lda   #>bsave    ; Set for write
1742: A0 26             ldy   #<bsave
1744: 20 B9 17          jsr   PDfae      ; "BSAVE <fn>,A$<bfstart>,E$"
1747: 4C 4D 17          jmp   :ckeof     ; Are we at End-Of-File?

174A: 20 D1 15 :findlp  jsr   incblk     ; Advance to next block.
174D: A0 00    :ckeof   ldy   #0         ; Check prefix sign/flag byte.
174F: B1 D3             lda   (ptr),y
1751: C9 B0             cmp   #PREF      ; Is ptr at block start?
1753: 90 21             bcc   ]IOerr2    ; -No, block sync error.
1755: C9 EF             cmp   #EOF       ; -Yes, are we at End-Of-File?
1757: F0 05             beq   :useptr    ; -Yes, write EOF to file.
1759: C9 EB             cmp   #EOB       ; -No, are we at End-Of-Buffer?
175B: D0 ED             bne   :findlp    ; -No, search forward by block.
175D: 18                clc              ; -Yes, don't write EOB.
175E: A5 D3    :useptr  lda   ptr        ; If not C, use ptr - 1.
1760: E9 00             sbc   #0         ; If C, just use ptr.
1762: A8                tay
1763: A5 D4             lda   ptr+1
1765: E9 00             sbc   #0
1767: 20 F1 17          jsr   PDebx      ; "<ptr>,B$<off>", Execute.
176A: B0 0A             bcs   ]IOerr2
176C: A6 D9             ldx   dbx
176E: A9 00             lda   #0
1770: 9D 86 13          sta   bfdirty,x  ; Mark buffer clean.
1773: 4C 0F 15          jmp   setptr     ; Restore ptr and return.

1776: 4C 21 08 ]IOerr2  jmp   X_IOerr    ; I/O error.
```

```
                >632  *********************************************************
                >633  *                                                       *
                >634  *                       doread                          *
                >635  *                                                       *
                >636  * On entry: dbx = DB index, ptr = current               *
                >637  * On exit:  A = 0, X = dbx, ptr = bfstart, buffer clean. *
                >638  *                                                       *
                >639  *********************************************************
                >640
1779: A9 18     >641  doread   lda   #>bload    ; Set for read.
177B: A0 1F     >642           ldy   #<bload
177D: 20 B9 17  >643           jsr   PDfae      ; "BLOAD <fn>,A$<start>,E$"
1780: BC 79 13  >644           ldy   bfend,x    ; E param is bfend.
1783: BD 7A 13  >645           lda   bfend+1,x
1786: 20 F1 17  >646           jsr   PDebx      ; "<end>,B$<off>", Execute.
1789: A6 D9     >647           ldx   dbx        ; Load DB index.
178B: 90 0F     >648           bcc   :noerr     ; No error.
178D: 29 FE     >649           and   #$FE       ; Fold error 6 & 7 together.
178F: C9 06     >650           cmp   #6         ; "Path Not Found" error?
1791: D0 E3     >651           bne   ]IOerr2    ; -No, IOerr.
1793: 20 67 16  >652           jsr   ]resptr    ; -Yes, set 'ptr' to 'bfstart'
1796: A9 EF     >653           lda   #EOF       ;   and set End-Of-File.
1798: A0 00     >654           ldy   #0
179A: 91 D3     >655           sta   (ptr),y
179C: A0 00     >656  :noerr   ldy   #0
179E: 98        >657           tya
179F: 9D 86 13  >658           sta   bfdirty,x  ; Mark buffer clean.
17A2: BD 79 13  >659           lda   bfend,x    ; ptr = bfend.
17A5: 85 D3     >660           sta   ptr
17A7: BD 7A 13  >661           lda   bfend+1,x
17AA: 85 D4     >662           sta   ptr+1
17AC: B1 D3     >663           lda   (ptr),y
17AE: C9 EF     >664           cmp   #EOF       ; (bfend) = End-Of-File?
17B0: F0 04     >665           beq   :done      ; -Yes, done.
17B2: A9 EB     >666           lda   #EOB       ; -No, set End-Of-Buffer
17B4: 91 D3     >667           sta   (ptr),y    ;       in (bfend).
17B6: 4C 67 16  >668  :done    jmp   ]resptr    ;        reset ptr to bfstart.
```

```
               *********************************************************
               *                                                       *
               *                  PDfae / PDebx                        *
               *                                                       *
               * On entry: dbx = DB index, ptr = current               *
               * On exit:  X = dbx, ptr unchanged.                     *
               *                                                       *
               *********************************************************

               zeroff   equ   line1      ; Zero offset flag

17B9: A2 00    PDfae    ldx   #0         ; Start ProDOS command.
17BB: 20 55 18          jsr   putpdcmd   ; BLOAD or BSAVE.
17BE: A4 D9             ldy   dbx        ; Y = Device Block index.
17C0: B9 7C 13          lda   bfsiz+1,y  ; Init 'zeroff' to 0 to
17C3: 49 02             eor   #>ptbfsz   ;  skip B param if PT unit
17C5: 85 DD             sta   zeroff     ;   and offset = 0.
17C7: B9 81 13          lda   bffn,y     ; (A,Y) --> file name
17CA: A8                tay
17CB: A9 14             lda   #>fnames
17CD: 20 55 18          jsr   putpdcmd   ; Add file name.
17D0: A9 18             lda   #>Aparm
17D2: A0 2D             ldy   #<Aparm
17D4: 20 55 18          jsr   putpdcmd   ; Add ",A$".
17D7: A4 D9             ldy   dbx
17D9: B9 76 13          lda   bfstart+1,y ; address = bfstart
17DC: 48                pha
17DD: B9 75 13          lda   bfstart,y
17E0: A8                tay
17E1: 68                pla
17E2: 20 39 18          jsr   putwdhx    ; Add hex address...
17E5: A9 18             lda   #>Eparm
17E7: A0 31             ldy   #<Eparm
17E9: 20 55 18          jsr   putpdcmd   ; Add ",E$"
17EC: 86 DE             stx   savex      ; Save ProDOS cmd index.
17EE: A6 D9             ldx   dbx
17F0: 60                rts

17F1: A6 DE    PDebx    ldx   savex      ; Restore command index.
17F3: 20 39 18          jsr   putwdhx    ; Add length
17F6: 86 DE             stx   savex      ; Save X before "B" param
17F8: A9 18             lda   #>Bparm
17FA: A0 35             ldy   #<Bparm
17FC: 20 55 18          jsr   putpdcmd   ; Add ",B$"
17FF: A9 03             lda   #3         ; Offset has 3 bytes.
1801: 85 D5             sta   inptr
1803: A4 D9             ldy   dbx
1805: C8                iny              ; Adjust dbx for bfoff+2
1806: C8                iny
1807: B9 82 13 :offlp   lda   bfoff,y    ; MSB of offset first.
180A: F0 02             beq   :zero
180C: 85 DD             sta   zeroff     ; Remember non-zero offset.
180E: 20 3D 18 :zero    jsr   putbyte    ; Add next offset byte.
1811: 88                dey              ; Next-most-sig offset byte.
1812: C6 D5             dec   inptr      ; More offset bytes?
1814: D0 F1             bne   :offlp     ; -Yes, continue.
1816: A5 DD             lda   zeroff     ; -No. Is offset zero?
1818: D0 02             bne   :useB      ; -No, existing file, use B.
181A: A6 DE             ldx   savex      ; -Yes, new file, no B.
181C: 4C 6C 18 :useB    jmp   pdosxeq    ; Execute command and return.

181F: C2 CC CF bload    asc   "BLOAD ",00
1826: C2 D3 C1 bsave    asc   "BSAVE ",00
182D: AC C1 A4 Aparm    asc   ",A$",00
1831: AC C5 A4 Eparm    asc   ",E$",00
1835: AC C2 A4 Bparm    asc   ",B$",00
```

```
1839: 20 3D 18 >737  putwdhx  jsr   putbyte    ; Put first byte in hex
183C: 98       >738           tya              ;  and fall into putbyte.
183D: 48       >739  putbyte  pha              ; Save byte
183E: 4A       >740           lsr
183F: 4A       >741           lsr
1840: 4A       >742           lsr
1841: 4A       >743           lsr
1842: 20 46 18 >744           jsr   :stdig     ; Put hi hex digit
1845: 68       >745           pla              ;  and then lo dig.
1846: 29 0F    >746  :stdig   and   #$0F       ; Isolate digit
1848: 09 B0    >747           ora   #"0"       ; Or in zone
184A: C9 BA    >748           cmp   #$BA       ; >9?
184C: 90 02    >749           bcc   :store     ; -No, store it.
184E: 69 06    >750           adc   #6         ; -Yes, cvt to A..F
1850: 9D 00 02 >751  :store   sta   IN,x       ; Add char to IN buffer.
1853: E8       >752           inx
1854: 60       >753           rts
```

```
                         put   B220PDOS
                **********************************************************
                *                                                        *
                *                      PUTPDCMD                          *
                *                                                        *
                * Append null-terminated string at (A,Y) onto IN,X.     *
                * Command is in hi-ASCII.  A is hi, Y is lo.            *
                *                                                        *
                * Advances X, destroys A, Y, and 'inptr'.               *
                *                                                        *
                **********************************************************

1855: 85 D6     putpdcmd sta   inptr+1    ; Set up string pointer
1857: 84 D5              sty   inptr
1859: A0 00              ldy   #0
185B: B1 D5     :cmdloop lda   (inptr),y  ; Append command string
185D: F0 07              beq   :rts       ;  until null
185F: 9D 00 02           sta   IN,x       ;   to keyboard buffer.
1862: E8                 inx              ; Bump pointers.
1863: C8                 iny
1864: D0 F5              bne   :cmdloop   ; (always)

1866: 60        :rts     rts              ; Return...

                **********************************************************
                *                                                        *
                *                      PDOSCMD                           *
                *                                                        *
                * Execute null-terminated ProDOS command at (A,Y)       *
                * Command is in hi-ASCII.                                *
                *                                                        *
                * Keyboard buffer, sptr, and Y are changed.              *
                * On error, C is set and A contains error code.          *
                *                                                        *
                **********************************************************

1867: A2 00     pdoscmd  ldx   #0         ; Empty kbd buffer.
1869: 20 55 18           jsr   putpdcmd   ; Move in the command
                                          ;  and fall into pdosxeq.

                **********************************************************
                *                                                        *
                *                      PDOSXEQ                           *
                *                                                        *
                * Execute ProDOS command in keyboard buffer after       *
                * appending a carriage return.  Command is in hi-ASCII. *
                *                                                        *
                * On error, C is set and A contains error code.          *
                *                                                        *
                **********************************************************

186C: A9 8D     pdosxeq  lda   #$8D       ; Carriage Return
186E: 9D 00 02           sta   IN,x       ;  at end
1871: AD 42 BE           lda   BSSTATE    ; Save BASIC.SYSTEM
1874: 48                 pha              ;  'state' var & set it
1875: A9 FF              lda   #$FF       ;    to suppress blank
1877: 8D 42 BE           sta   BSSTATE    ;     line.
187A: 20 03 BE           jsr   DOSCMD     ; Then do it...
187D: AA                 tax              ; Save error code.
187E: 68                 pla              ; Restore BASIC.SYSTEM
187F: 8D 42 BE           sta   BSSTATE    ;  state variable.
1882: 8A                 txa              ; A = ProDOS error code.
1883: 60                 rts
```

```
                         put   b220TRACE
                ************************************************************
                *                                                          *
                *                    TRACE Routines                        *
                *                                                          *
                ************************************************************

                PRNTAX   equ   $F941      ; Print A.X in hex
                PRBYTE   equ   $FDDA      ' Print A in hex
                PRHEX    equ   $FDE3      ; Print low nibble of A

1884: A0 96     prtrace  ldy   #rP        ; rP
1886: 20 06 19           jsr   print2     ; Print PPPP + 2 blanks.
1889: A0 98              ldy   #rC        ; rC
188B: 20 E9 18           jsr   printsgn   ; Print sign nibble + blank
188E: C8                 iny
188F: 20 FB 18           jsr   print2c    ; Print VVVV + blank
1892: C8                 iny
1893: C8                 iny
1894: 20 F2 18           jsr   print1     ; Print OP + blank
1897: C8                 iny
1898: 20 06 19           jsr   print2     ; Print AAAA + 2 blanks
189B: A5 D7              lda   t1         ; Does OP have mem ADDR?
189D: 10 08              bpl   :prtmem    ; -Yes, print operand.
189F: A2 0E              ldx   #14        ; -No, print blanks.
18A1: 20 4A F9           jsr   PRBL2
18A4: 4C AC 18           jmp   :prtrB

18A7: A0 AA     :prtmem  ldy   #rD        ; (memptr)
18A9: 20 16 19           jsr   print6
18AC: A0 94     :prtrB   ldy   #rB        ; rB
18AE: 20 06 19           jsr   print2     ; Print BBBB + 2 blanks
18B1: A0 9E              ldy   #rA        ; rA
18B3: 20 16 19           jsr   print6     ; S AAAAAAAAAA + 2 blanks
18B6: A0 A4              ldy   #rR        ; rR
18B8: 20 16 19           jsr   print6     ; S RRRRRRRRRR + 2 blanks
18BB: A2 00              ldx   #0
18BD: A5 C2              lda   COMP       ; Comparison indicator -1,0,+1
18BF: 30 04              bmi   :lt        ; <
18C1: F0 01              beq   :eql       ; =
18C3: E8                 inx              ; >
18C4: E8        :eql     inx
18C5: BD E6 18  :lt      lda   :compch,x  ;     <,=,>
18C8: 20 ED FD           jsr   COUT
18CB: 20 2A 19           jsr   printbl
18CE: A0 C4              ldy   #Rp        ; Repeat indicator
18D0: A9 D2              lda   #"R"
18D2: 20 0C 19           jsr   prind
18D5: A0 C3              ldy   #Ov        ; Overflow indicator
18D7: A9 CF              lda   #"O"
18D9: 20 0C 19           jsr   prind
18DC: A0 CE              ldy   #OvHlt     ; Overflow Halt mode
18DE: A9 C8              lda   #"H"
18E0: 20 0C 19           jsr   prind
18E3: 4C 8E FD           jmp   CROUT      ; Print CR

18E6: BC BD BE  :compch  asc   "<=>" ; Comparison characters

18E9: B9 00 00  printsgn lda   0,y
18EC: 20 E3 FD           jsr   PRHEX      ; Print sign nibble
18EF: 4C 2A 19           jmp   printbl    ;   + blank.

18F2: B9 00 00  print1   lda   0,y
18F5: 20 DA FD           jsr   PRBYTE     ; Print AA
18F8: 4C 2A 19           jmp   printbl    ; Print blank.

18FB: B9 00 00  print2c  lda   0,y
```

```
18FE: B6 01    >67             ldx   1,y
1900: 20 41 F9 >68             jsr   PRNTAX
1903: 4C 2A 19 >69             jmp   printbl
               >70
1906: 20 FB 18 >71   print2    jsr   print2c
1909: 4C 2A 19 >72             jmp   printbl
               >73
190C: B6 00    >74   prind     ldx   0,y         ; Print (A) + blank if on,
190E: F0 17    >75             beq   print2bl    ;  else print 2 blanks.
1910: 20 ED FD >76             jsr   COUT
1913: 4C 2A 19 >77             jmp   printbl
               >78
1916: 20 E9 18 >79   print6    jsr   printsgn    ; Print sign + blank
1919: A2 00    >80             ldx   #0
191B: C8       >81   :bytlp    iny
191C: B9 00 00 >82             lda   0,y
191F: 20 DA FD >83             jsr   PRBYTE      ; Print 5 bytes
1922: E8       >84             inx
1923: E0 05    >85             cpx   #5
1925: D0 F4    >86             bne   :bytlp
1927: 20 2A 19 >87   print2bl jsr   printbl
192A: A9 A0    >88   printbl  lda   #"          "
192C: 4C ED FD >89             jmp   COUT
```

```
         put   B220VIEW
**********************************************************
*                                                        *
*            B220 Display instruction routines           *
*                                                        *
**********************************************************

* Apple ROM zero page equates

GBASL    equ   $26         ; Graphics line base
GBASH    equ   $27
HMASK    equ   $30         ; X bit mask
HPAG     equ   $E6         ; Graphics page (HGR2=$40)
HNDX     equ   $E5         ; X byte

* Local page zero vars

y        equ   line1
scx      equ   line1       ; scale..0 X scale counter
scy      equ   line1+1     ; scale..0 Y scale counter
bcdxy    equ   line2       ; 000..999 BCD X & Y coord
hgrx     equ   line4       ; HGR X value (255 = inv)
hgry     equ   line4+1     ; HGR Y value (255 = inv)

* Applesoft ROM entry points

HGR2     equ   $F3D8       ; Set HGR2 ($4000..$5FFF)
HPOSN    equ   $F411       ; Compute base, byte, & mask
```

```
                * CRT Keyboard mode service routine

192F: C9 9B     kbserve  cmp   #escape    ; ESCape back to VIEW mode?
1931: F0 19              beq   :exitkb    ; -Yes, exit Keyboard mode.
1933: 29 7F              and   #$7F       ; -No.  Turn off high bit.
1935: 8D 52 C0           sta   MIXED+OFF  ; Turn off help lines.
1938: A8                 tay              ; Use key for index.
1939: AD 61 C0           lda   PB0        ; Open-Apple key depressed?
193C: 30 06              bmi   :fkey      ; -Yes, handle Function key.
193E: B9 00 1E           lda   keytbl,y   ; -No, translate regular key
1941: 4C 47 19           jmp   :dokey     ;   and send it to B220.

1944: B9 80 1E  :fkey    lda   fktbl,y    ; Translate Function key,
1947: 85 E5     :dokey   sta   crtkey     ;  save it (for PRD), and
1949: 4C 2A 08           jmp   X_resetr   ;   simulate Reset-Transfer.

                :exitkb  resi  kbmode     ; Exit Keyboard mode
194C: A9 00              lda   #0
194E: 85 CD              sta   kbmode     ; Zero indicator.
                         eom
1950: A9 C9              lda   #"I"       ;  and swap back VIEW help.
1952: 20 B1 1C           jsr   swaphelp
1955: 4C 76 19           jmp   showhelp

                kbmodeon seti  kbmode     ; Turn on CRT Keyboard mode.
1958: A9 FF              lda   #$FF
195A: 85 CD              sta   kbmode     ; Set non-zero.
                         eom
195C: A9 CB              lda   #"K"       ;  and swap in Keyboard help.
195E: 20 B1 1C           jsr   swaphelp
1961: 4C 76 19           jmp   showhelp   ; Display the help lines.

                * VIEW mode switching routines

                viewon   seti  viewmode   ; Turn on VIEW mode switch
1964: A9 FF              lda   #$FF
1966: 85 CB              sta   viewmode   ; Set non-zero.
                         eom
1968: 8D 55 C0           sta   PAGE2+ON
196B: 8D 57 C0           sta   HIRES+ON
196E: 8D 50 C0           sta   TEXT+OFF
1971: A9 C9              lda   #"I"       ; Swap in View mode help
1973: 20 B1 1C           jsr   swaphelp
1976: 8D 53 C0  showhelp sta   MIXED+ON   ; Display help lines.
1979: 4C F4 19           jmp   ]done

                viewoff  resi  viewmode   ; Turn off VIEW mode switch
197C: A9 00              lda   #0
197E: 85 CB              sta   viewmode   ; Zero indicator.
                         eom
1980: 8D 51 C0           sta   TEXT+ON
1983: 8D 56 C0           sta   HIRES+OFF
1986: 8D 54 C0           sta   PAGE2+OFF
1989: 4C F4 19           jmp   ]done

                * View mode keyboard analyzer

198C: A4 CD     viewkey  ldy   kbmode     ; Are we in Keyboard mode?
198E: D0 9F              bne   kbserve    ; -Yes, sevice the keyboard.
1990: 8D 52 C0           sta   MIXED+OFF  ; -No, any key turns off help.
1993: AA                 tax              ; Save raw character.
1994: 29 DF              and   #$DF       ; Force uppercase.
1996: C9 CC              cmp   #"L"       ; Toggle lightpen?
1998: F0 30              beq   :lpen      ; -Yes.
199A: C9 CB              cmp   #"K"       ; -No, keyboard mode?
199C: F0 BA              beq   kbmodeon   ; -Yes, turn it on.
199E: 8A                 txa              ; Restore raw character.
```

```
199F: C9 AD    >84             cmp   #"-"
19A1: F0 5B    >85             beq   :incsc     ; Increment scale.
19A3: C9 AB    >86             cmp   #"+"
19A5: F0 5C    >87             beq   :decsc     ; Decrement scale.
19A7: A2 9A    >88             ldx   #<yl       ; Operate on YL
19A9: C9 8B    >89             cmp   #$8B       ; Up arrow
19AB: F0 7E    >90             beq   :dec
19AD: C9 8A    >91             cmp   #$8A       ; Down arrow
19AF: F0 59    >92             beq   :inc
19B1: A2 98    >93             ldx   #<xl       ; Operate on XL
19B3: C9 95    >94             cmp   #$95       ; Right arrow
19B5: F0 74    >95             beq   :dec
19B7: C9 88    >96             cmp   #$88       ; Left arrow
19B9: F0 4F    >97             beq   :inc
19BB: C9 BF    >98             cmp   #"?"       ; Help
19BD: F0 2D    >99             beq   :help
19BF: C9 9B    >100            cmp   #escape    ; Exit View mode
19C1: F0 B9    >101            beq   viewoff
19C3: A6 C0    >102            ldx   RUN        ; Are we running?
19C5: D0 22    >103            bne   :running   ; -Yes.
19C7: 4C 91 0A >104            jmp   ]analyze   ; -No, analyze regular keys.
               >105
19CA: A5 CC    >106  :lpen     lda   lpen       ; Is pen on?
19CC: D0 0D    >107            bne   :penoff    ; -Yes, turn it off.
               >108            seti  lpen       ; -No, turn it on,
19CE: A9 FF    >108            lda   #$FF
19D0: 85 CC    >108            sta   lpen       ; Set non-zero.
               >108            eom
19D2: 20 70 1C >109            jsr   lpread     ;   read position,
19D5: 20 F2 1B >110            jsr   xdrawcur   ;    and xdraw cursor.
19D8: 4C F4 19 >111            jmp   ]done
               >112
               >113  :penoff   resi  lpen       ; Turn pen off,
19DB: A9 00    >113            lda   #0
19DD: 85 CC    >113            sta   lpen       ; Zero indicator.
               >113            eom
19DF: 20 F2 1B >114            jsr   xdrawcur   ;  erase cursor,
19E2: A9 FF    >115            lda   #$FF       ;   and set px to
19E4: 8D 94 1D >116            sta   px         ;    "invalid".
19E7: D0 0B    >117            bne   ]done
               >118
19E9: 4C 3C 0A >119  :running jmp   ]ckstop    ; Check for stop/step key.
               >120
19EC: 8D 53 C0 >121  :help     sta   MIXED+ON   ; Turn on help lines.
19EF: F0 03    >122            beq   ]done      ;  and continue. (always)
               >123
19F1: 20 93 1A >124  :reinit   jsr   xyinit     ; Regenerate X,Y maps
19F4: A5 C0    >125  ]done     lda   RUN        ; Are we running?
19F6: D0 03    >126            bne   :cont      ; -Yes, continue.
19F8: 4C 82 0A >127            jmp   ]waitkey   ; -No, check regular keys.
               >128
19FB: 4C 18 08 >129  :cont     jmp   X_cont     ; Continue simulation.
               >130
19FE: EE 91 1D >131  :incsc    inc   scale
1A01: D0 EE    >132            bne   :reinit    ; (always)
               >133
1A03: CE 91 1D >134  :decsc    dec   scale
1A06: F0 F6    >135            beq   :incsc     ; Clamp at 1.
1A08: D0 E7    >136            bne   :reinit    ; (always)
               >137
1A0A: F8       >138  :inc      sed
1A0B: 18       >139            clc
1A0C: BD 01 1D >140            lda   xlyl+1,x
1A0F: 6D 93 1D >141            adc   delta+1
1A12: 9D 01 1D >142            sta   xlyl+1,x
1A15: BD 00 1D >143            lda   xlyl,x
1A18: 6D 92 1D >144            adc   delta
```

```
1A1B: 9D 00 1D >145           sta   xlyl,x
1A1E: D8       >146           cld
1A1F: C9 09    >147           cmp   #$09       ; >= 900?
1A21: 90 27    >148           bcc   :rein      ; -No, continue.
1A23: A9 08    >149           lda   #$08       ; -Yes, clamp to
1A25: 9D 00 1D >150           sta   xlyl,x     ;        <900.
1A28: 4C F1 19 >151           jmp   :reinit
               >152
1A2B: F8       >153  :dec     sed
1A2C: 38       >154           sec
1A2D: BD 01 1D >155           lda   xlyl+1,x
1A30: ED 93 1D >156           sbc   delta+1
1A33: 9D 01 1D >157           sta   xlyl+1,x
1A36: BD 00 1D >158           lda   xlyl,x
1A39: ED 92 1D >159           sbc   delta
1A3C: 9D 00 1D >160           sta   xlyl,x
1A3F: D8       >161           cld
1A40: B0 08    >162           bcs   :rein      ; No underflow.
1A42: A9 00    >163           lda   #0         ; Underflow, clamp
1A44: 9D 00 1D >164           sta   xlyl,x     ;  to zero.
1A47: 9D 01 1D >165           sta   xlyl+1,x
1A4A: 4C F1 19 >166  :rein    jmp   :reinit
```

```
                ***********************************************************
                *                                                         *
                *        Initialize map of HGR x,y to address/bit.        *
                *                                                         *
                ***********************************************************

1A4D: A9 40     HGRinit  lda   #$40       ; Set high-res page
1A4F: 85 E6              sta   HPAG       ;  to HGR2.
1A51: 85 D4              sta   ptr+1      ; Clear HGR2 screen.
1A53: A9 00              lda   #0
1A55: 85 D3              sta   ptr
1A57: A8                 tay
1A58: 91 D3     :clrHGR2 sta   (ptr),y
1A5A: C8                 iny
1A5B: D0 FB              bne   :clrHGR2
1A5D: E6 D4              inc   ptr+1      ; Next page.
1A5F: A6 D4              ldx   ptr+1
1A61: E0 60              cpx   #$60       ; Last page cleared?
1A63: 90 F3              bcc   :clrHGR2   ; -No, keep going.
1A65: 84 DD     :loop    sty   y
1A67: 98                 tya              ; A = Y coordinate
1A68: AA                 tax              ; X = X coordinate
1A69: A0 00              ldy   #0         ; X < 256
1A6B: C9 C0              cmp   #192       ; If Y > 191
1A6D: 90 02              bcc   :yok       ;  force it
1A6F: A9 00              lda   #0         ;   to zero.
1A71: 20 11 F4  :yok     jsr   HPOSN      ; Compute base, byte, & mask.
1A74: A4 DD              ldy   y          ; Recover index
1A76: C0 C0              cpy   #192       ; If Y > 191
1A78: B0 0A              bcs   :skipy     ;  skip storing.
1A7A: A5 26              lda   GBASL      ; Save line base low.
1A7C: 99 00 37           sta   ybasel,y
1A7F: A5 27              lda   GBASH      ; Save line base high.
1A81: 99 C0 37           sta   ybaseh,y
1A84: A5 E5     :skipy   lda   HNDX       ; Save X byte
1A86: 99 00 35           sta   xbyte,y
1A89: A5 30              lda   HMASK      ; Save X bit
1A8B: 29 7F              and   #$7F       ;  (with hi bit off)
1A8D: 99 00 36           sta   xbit,y
1A90: C8                 iny
1A91: D0 D2              bne   :loop      ; Loop 0..255
                * Fall into 'xyinit'.
```

```
                 >211  **********************************************************
                 >212  *                                                        *
                 >213  *        Init tables mapping B220 X,Y to HGR x,y.        *
                 >214  *                                                        *
                 >215  **********************************************************
                 >216
1A93: AD 91 1D >217  xyinit   lda   scale      ; Initialize B220 XY tables
1A96: 85 DD    >218           sta   scx        ;  to HGR XY values, depending
1A98: 85 DE    >219           sta   scy        ;   on xl, yl, and scale.
1A9A: A9 00    >220           lda   #0
1A9C: 85 DF    >221           sta   bcdxy      ; bcdxy = 00 00 (hi,lo).
1A9E: 85 E0    >222           sta   bcdxy+1
1AA0: 85 E1    >223           sta   hgrx       ; HGR X = 0
1AA2: A9 BF    >224           lda   #191
1AA4: 85 E2    >225           sta   hgry       ; HGR Y = 191
1AA6: A6 E1    >226  :xyloop  ldx   hgrx       ; Default X value
1AA8: A5 DF    >227           lda   bcdxy      ; Compare B220 X to xl
1AAA: CD 98 1D >228           cmp   xl
1AAD: 90 09    >229           bcc   :invx      ; B220 X < xl
1AAF: D0 09    >230           bne   :goodx     ; B220 X > xl
1AB1: A5 E0    >231           lda   bcdxy+1    ; Hi dig equal, check lo.
1AB3: CD 99 1D >232           cmp   xl+1
1AB6: B0 02    >233           bcs   :goodx     ; B220 X >= xl
1AB8: A2 FF    >234  :invx    ldx   #$FF       ; B220 X < xl ==> no plot
1ABA: A5 DF    >235  :goodx   lda   bcdxy      ; Set xmap(bcdxy) = x reg
1ABC: 18       >236           clc
1ABD: 69 21    >237           adc   #>xmap     ; Add 0..9 to page
1ABF: 8D C7 1A >238           sta   :staxmap+2 ;  and set sta page.
1AC2: A4 E0    >239           ldy   bcdxy+1
1AC4: 8A       >240           txa
1AC5: 99 00 21 >241  :staxmap sta   xmap+0,y   ; Save mapped HGR X
1AC8: E8       >242           inx              ; Was X invalid? ($FF)
1AC9: F0 11    >243           beq   :doy       ; -Yes, skip X advance.
1ACB: C6 DD    >244           dec   scx        ; -No. Time to adv X?
1ACD: D0 0D    >245           bne   :doy       ; -No.
1ACF: AD 91 1D >246           lda   scale      ; -Yes, reset scx.
1AD2: 85 DD    >247           sta   scx
1AD4: E6 E1    >248           inc   hgrx       ; Advance HGR X
1AD6: D0 04    >249           bne   :doy       ; -Didn't overflow.
1AD8: A2 FF    >250           ldx   #$FF       ; -Overflow. Stick at $FF.
1ADA: 86 E1    >251           stx   hgrx
1ADC: A6 E2    >252  :doy     ldx   hgry       ; Default Y value
1ADE: A5 DF    >253           lda   bcdxy      ; Compare B220 Y to yl
1AE0: CD 9A 1D >254           cmp   yl
1AE3: 90 09    >255           bcc   :invy      ; B220 Y < yl
1AE5: D0 09    >256           bne   :goody     ; B220 Y > yl
1AE7: A5 E0    >257           lda   bcdxy+1    ; Hi dig equal, check lo.
1AE9: CD 9B 1D >258           cmp   yl+1
1AEC: B0 02    >259           bcs   :goody     ; B220 Y >= yl
1AEE: A2 FF    >260  :invy    ldx   #$FF       ; B220 Y < yl ==> no plot
1AF0: A5 E0    >261  :goody   lda   bcdxy+1    ; Tens and Units
1AF2: 29 0F    >262           and   #$0F       ; Units
1AF4: 18       >263           clc
1AF5: 69 2B    >264           adc   #>ymap     ; Add Y map page
1AF7: 8D 10 1B >265           sta   :staymap+2 ;  and modify sta.
1AFA: A5 E0    >266           lda   bcdxy+1    ; Tens and Units
1AFC: 85 D7    >267           sta   t1         ; Save for shift
1AFE: A5 DF    >268           lda   bcdxy      ; Hundreds
1B00: 06 D7    >269           asl   t1         ; Shift in Tens digit
1B02: 2A       >270           rol
1B03: 06 D7    >271           asl   t1
1B05: 2A       >272           rol
1B06: 06 D7    >273           asl   t1
1B08: 2A       >274           rol
1B09: 06 D7    >275           asl   t1
1B0B: 2A       >276           rol
1B0C: A8       >277           tay              ; Y = Hundreds & Tens
```

```
1B0D: 8A       >278           txa
1B0E: 99 00 2B >279  :staymap sta   ymap,y     ; Save HGR Y in ymap.
1B11: E8       >280           inx              ; Was Y invalid? ($FF)
1B12: F0 15    >281           beq   :incxy     ; -Yes, skip Y advance.
1B14: C6 DE    >282           dec   scy        ; -No. Time to adv Y?
1B16: D0 11    >283           bne   :incxy     ; -No.
1B18: AD 91 1D >284           lda   scale      ; -Yes. Reset scy.
1B1B: 85 DE    >285           sta   scy
1B1D: C6 E2    >286           dec   hgry       ; Advance HGR Y
1B1F: A6 E2    >287           ldx   hgry
1B21: E0 C0    >288           cpx   #192       ; Underflow?
1B23: 90 04    >289           bcc   :incxy     ; -No.
1B25: A2 FF    >290           ldx   #$FF       ; -Yes, saturate at $FF.
1B27: 86 E2    >291           stx   hgry
1B29: F8       >292  :incxy   sed              ; Increment bcdxy
1B2A: 18       >293           clc
1B2B: A5 E0    >294           lda   bcdxy+1
1B2D: 69 01    >295           adc   #1
1B2F: 85 E0    >296           sta   bcdxy+1
1B31: D8       >297           cld
1B32: 90 09    >298           bcc   :xyrelay   ; No carry, loop.
1B34: E6 DF    >299           inc   bcdxy      ; Propagate carry.
1B36: A6 DF    >300           ldx   bcdxy      ; Bigger than 9?
1B38: E0 0A    >301           cpx   #$0A
1B3A: 90 01    >302           bcc   :xyrelay   ; -No, continue.
1B3C: 60       >303           rts
               >304
1B3D: 4C A6 1A >305  :xyrelay jmp   :xyloop
```

```
               **********************************************************
               *                                                        *
               *    Plot B220 point in rA on HGR screen, and erase      *
               *    the point plotted 256 points earlier.               *
               *                                                        *
               **********************************************************

1B40: A5 A2    b220plot lda   rA+4       ; Y Units, X Hundreds
1B42: 29 0F             and   #$0F       ; X Hundreds
1B44: 18                clc
1B45: 69 21             adc   #>xmap     ; Add X map page
1B47: 8D 4E 1B          sta   :xload+2
1B4A: A4 A3             ldy   rA+5       ; Tens & Units
1B4C: B9 00 21 :xload   lda   xmap+0,y   ; Mapped HGR X coordinate.
1B4F: 85 D7             sta   t1         ; Save HGR X coordinate.
1B51: C9 FF             cmp   #$FF       ; X invalid?
1B53: F0 11             beq   :skipy     ; -Yes, skip Y eval.
1B55: A5 A2             lda   rA+4       ; Y 100's & 10's digits
1B57: 4A                lsr              ; Align Y Units right
1B58: 4A                lsr
1B59: 4A                lsr
1B5A: 4A                lsr
1B5B: 18                clc
1B5C: 69 2B             adc   #>ymap     ; Add Y map page
1B5E: 8D 65 1B          sta   :yload+2   ; Modify the lda
1B61: A4 A1             ldy   rA+3       ; Y Hundreds and Tens
1B63: B9 00 2B :yload   lda   ymap+0,y   ; Mapped HGR Y coordinate.
1B66: AE 90 1D :skipy   ldx   hx         ; History array index
1B69: 9D 00 20          sta   histy,x    ; Save new Y coord
1B6C: A8                tay              ;  and put in Y reg.
1B6D: A5 D7             lda   t1         ; Recover X coordinate
1B6F: 9D 00 1F          sta   histx,x    ;  and save it.
1B72: EE 90 1D          inc   hx         ; Increment hx mod 256
1B75: C9 FF             cmp   #$FF       ; X invalid?
1B77: F0 54             beq   :erase     ; -Yes, just erase oldest.
1B79: C0 FF             cpy   #$FF       ; Y invalid?
1B7B: F0 50             beq   :erase     ; -Yes, just erase oldest.
1B7D: AA                tax              ; X coord to X reg.
1B7E: A5 CC             lda   lpen       ; Is light pen "on"
1B80: 2D 61 C0          and   PB0        ;  and PB0 pressed?
1B83: 10 12             bpl   :keychk    ; -No, continue.
1B85: 8A                txa              ; -Yes, test X for hit
1B86: 4A                lsr
1B87: CD 94 1D          cmp   px         ; X hit?
1B8A: D0 0B             bne   :keychk    ; -No, continue.
1B8C: 98                tya              ; Test Y for hit
1B8D: 4A                lsr
1B8E: CD 95 1D          cmp   py         ; Y hit?
1B91: D0 04             bne   :keychk    ; -No, continue.
1B93: A9 00             lda   #0         ; -Yes, signal LP hit.
1B95: F0 16             beq   :quit      ;   and quit. (always)

1B97: A5 CD    :keychk  lda   kbmode     ; If in keyboard mode,
1B99: D0 16             bne   :cont      ;  don't process the key.
1B9B: AD 00 C0          lda   KBD        ; Is a key pending?
1B9E: 10 11             bpl   :cont      ; -No, continue.
1BA0: C9 BA             cmp   #$BA       ; -Yes, is it > "9"?
1BA2: B0 0D             bcs   :cont      ; -Yes, leave it pending.
1BA4: C9 B0             cmp   #$B0       ; -No, is it < "0"?
1BA6: 90 09             bcc   :cont      ; -Yes, leave it pending.
1BA8: 8D 10 C0          sta   KBSTROBE   ; -No, mark key taken and
1BAB: 29 0F             and   #$0F       ;   put 0..9 in rA sign.
1BAD: 85 9E    :quit    sta   rA+S
1BAF: 38                sec              ; Signal button press
1BB0: 60                rts              ;  and return.

1BB1: 18       :cont    clc
```

```
1BB2: B9 00 37 >374           lda   ybasel,y   ; Set line base from Y
1BB5: 7D 00 35 >375           adc   xbyte,x    ;  and add in byte offset.
1BB8: 85 D3    >376           sta   ptr        ; (never carries)
1BBA: B9 C0 37 >377           lda   ybaseh,y
1BBD: 85 D4    >378           sta   ptr+1
1BBF: BD 00 36 >379           lda   xbit,x     ; X bit mask
1BC2: A2 00    >380           ldx   #0
1BC4: 01 D3    >381           ora   (ptr,x)    ; Plot the new point.
1BC6: 81 D3    >382           sta   (ptr,x)
1BC8: AD 62 C0 >383           lda   PB1        ; If Closed-Apple pressed
1BCB: 30 23    >384           bmi   :done      ;  don't erase points.
1BCD: AE 90 1D >385  :erase   ldx   hx         ; Recover oldest point.
1BD0: BC 00 20 >386           ldy   histy,x
1BD3: BD 00 1F >387           lda   histx,x
1BD6: AA       >388           tax              ; X coord to X reg.
1BD7: 18       >389           clc
1BD8: B9 00 37 >390           lda   ybasel,y   ; Set line base from Y
1BDB: 7D 00 35 >391           adc   xbyte,x    ;  and add in byte offset.
1BDE: 85 D3    >392           sta   ptr        ; (never carries)
1BE0: B9 C0 37 >393           lda   ybaseh,y
1BE3: 85 D4    >394           sta   ptr+1
1BE5: BD 00 36 >395           lda   xbit,x     ; X bit mask
1BE8: A2 00    >396           ldx   #0
1BEA: 49 FF    >397           eor   #$FF       ; Complement for AND
1BEC: 21 D3    >398           and   (ptr,x)    ; Unplot oldest point.
1BEE: 81 D3    >399           sta   (ptr,x)
1BF0: 18       >400  :done    clc              ; Signal no light pen sensed.
1BF1: 60       >401           rts
```

```
                >403  ***********************************************************
                >404  *                                                         *
                >405  *         xdrawcur - XDRAW "light pen" cursor             *
                >406  *                                                         *
                >407  *    On entry: px, py = "light pen" paddle readings.      *
                >408  *                                                         *
                >409  ***********************************************************
                >410
1BF2: AD 94 1D >411  xdrawcur lda   px          ; "Light pen" x: 0-127
1BF5: 38       >412           sec               ; Shift in "1".
1BF6: 2A       >413           rol               ; A = px * 2 + 1 (green)
1BF7: 8D 96 1D >414           sta   xx          ; Save HGR x coordinate
1BFA: B0 73    >415           bcs   :exit       ; px >127 means "no erase"
1BFC: 18       >416           clc
1BFD: 69 06    >417           adc   #6          ; A = px * 2 + 7
1BFF: AA       >418           tax               ; X = 2 * px + 7 (green)
1C00: AD 95 1D >419           lda   py          ; "Light pen" y: 0-95
1C03: 0A       >420           asl               ; A = py * 2
1C04: A8       >421           tay               ; Y = py * 2
1C05: 8C 97 1D >422           sty   yy          ; Save HGR y coordinate
1C08: A9 07    >423           lda   #7          ; Xdraw horizontal crosshair line
1C0A: 85 D7    >424           sta   t1
1C0C: EC 96 1D >425  :clipx   cpx   xx          ; Overflow?
1C0F: B0 06    >426           bcs   :xgo        ; -No, xdraw line.
1C11: CA       >427           dex               ; -Yes, clip.
1C12: CA       >428           dex
1C13: C6 D7    >429           dec   t1
1C15: D0 F5    >430           bne   :clipx      ; (always)
               >431
1C17: B9 C0 37 >432  :xgo     lda   ybaseh,y
1C1A: 85 D4    >433           sta   ptr+1
1C1C: 18       >434           clc
1C1D: AC 97 1D >435  :xloop   ldy   yy
1C20: B9 00 37 >436           lda   ybasel,y
1C23: 7D 00 35 >437           adc   xbyte,x     ; Never carries.
1C26: 85 D3    >438           sta   ptr
1C28: BD 00 36 >439           lda   xbit,x
1C2B: A0 00    >440           ldy   #0
1C2D: 51 D3    >441           eor   (ptr),y
1C2F: 91 D3    >442           sta   (ptr),y
1C31: CA       >443           dex               ; Clip if X = 1?
1C32: F0 05    >444           beq   :xdone      ; -Yes.
1C34: CA       >445           dex               ; -No, go on.
1C35: C6 D7    >446           dec   t1
1C37: D0 E4    >447           bne   :xloop
1C39: A9 07    >448  :xdone   lda   #7          ; Xdraw 7 dot vertical line.
1C3B: 85 D7    >449           sta   t1
1C3D: AD 97 1D >450           lda   yy
1C40: 69 06    >451           adc   #6          ; A = py * 2 + 6
1C42: A8       >452           tay               ; Y = py * 2 + 6
1C43: C0 C0    >453  :clipy   cpy   #192        ; Clip Y?
1C45: 90 06    >454           bcc   :yloop      ; -No.
1C47: 88       >455           dey               ; -Yes.
1C48: 88       >456           dey
1C49: C6 D7    >457           dec   t1
1C4B: D0 F6    >458           bne   :clipy      ; (always)
               >459
1C4D: B9 C0 37 >460  :yloop   lda   ybaseh,y    ; Xdraw vertical crosshair line
1C50: 85 D4    >461           sta   ptr+1
1C52: AE 96 1D >462           ldx   xx
1C55: B9 00 37 >463           lda   ybasel,y
1C58: 7D 00 35 >464           adc   xbyte,x     ; Never carries.
1C5B: 85 D3    >465           sta   ptr
1C5D: BD 00 36 >466           lda   xbit,x
1C60: A2 00    >467           ldx   #0
1C62: 41 D3    >468           eor   (ptr,x)
1C64: 81 D3    >469           sta   (ptr,x)
```

```
1C66: 98                 tya               ; Clip Y < 0?
1C67: F0 06              beq   :exit       ; -Yes.
1C69: 88                 dey               ; -No, go on.
1C6A: 88                 dey
1C6B: C6 D7              dec   t1
1C6D: D0 DE              bne   :yloop
1C6F: 60       :exit     rts

               **********************************************************
               *                                                        *
               *   lpread - Read paddles 0 & 1 for "light pen" cursor   *
               *                                                        *
               *    On exit: px, py = 0-127, 0-95 paddle readings.      *
               *                                                        *
               **********************************************************

1C70: AD 64 C0 lpread    lda   PDL+0       ; Wait for both paddles
1C73: 0D 65 C0           ora   PDL+1       ;  to time out before
1C76: 30 F8              bmi   lpread      ;   re-triggering.
1C78: 8D 70 C0           sta   PTRIG       ; Read paddles for
1C7B: A2 00              ldx   #0          ;  "light pen" coordinates.
1C7D: A0 A9              ldy   #LDAIop     ; "lda immediate" opcode
1C7F: AD 64 C0 :lploop   lda   PDL+0       ; Paddle 0 timed out?
1C82: 10 1D    :xtest    bpl   :gotx       ; -Yes, save px.
1C84: AD 65 C0 :cky      lda   PDL+1       ; -No. Paddle 1 timed out?
1C87: 10 20    :ytest    bpl   :goty       ; -Yes, save py
1C89: 85 D7              sta   t1          ; Waste 3 cycles.
1C8B: E8       :resume   inx               ; Keep count...
1C8C: 10 F1              bpl   :lploop     ;  until = 128.
1C8E: A9 5F              lda   #95
1C90: CD 95 1D           cmp   py          ; Is py > 95?
1C93: B0 03              bcs   :le95       ; -No, it's good.
1C95: 8D 95 1D           sta   py          ; -Yes, clamp at 95.
1C98: A9 10    :le95     lda   #BPLop      ; Restore tests.
1C9A: 8D 82 1C           sta   :xtest
1C9D: 8D 87 1C           sta   :ytest
1CA0: 60                 rts

1CA1: 8E 94 1D :gotx     stx   px          ; Save pen x coordinate
1CA4: 8C 82 1C           sty   :xtest      ; & disable further hits.
1CA7: 10 DB              bpl   :cky        ; Check y. (always)

1CA9: 8E 95 1D :goty     stx   py          ; Save pen y coordinate
1CAC: 8C 87 1C           sty   :ytest      ;  & disable further hits.
1CAF: 10 DA              bpl   :resume     ; Back to timing loop (always)
```

```
                **********************************************************
                *                                                        *
                *                      swaphelp                          *
                *                                                        *
                * Swaps in the four help lines specified by the content  *
                * of the accumulator on entry.  The character in A is    *
                * the fourth character of the first line of the desired  *
                * set of help lines: "I" for VIEW mode help, and "K"     *
                * for Keyboard mode help.                                *
                *                                                        *
                **********************************************************

1CB1: CD 53 0A  swaphelp cmp   VIEWhlp1+3 ; Does 4th char match request?
1CB4: F0 39              beq   :done      ; -Yes, no swap needed.
1CB6: A0 27              ldy   #40-1      ; -No, swap the help lines.
1CB8: B9 F0 1C  :swaplp  lda   kbhelp1,y  ; Swap first line.
1CBB: BE 50 0A           ldx   VIEWhlp1,y
1CBE: 99 50 0A           sta   VIEWhlp1,y
1CC1: 8A                 txa
1CC2: 99 F0 1C           sta   kbhelp1,y
1CC5: B9 18 1D           lda   kbhelp2,y  ; Swap second line.
1CC8: BE D0 0A           ldx   VIEWhlp2,y
1CCB: 99 D0 0A           sta   VIEWhlp2,y
1CCE: 8A                 txa
1CCF: 99 18 1D           sta   kbhelp2,y
1CD2: B9 40 1D           lda   kbhelp3,y  ; Swap third line.
1CD5: BE 50 0B           ldx   VIEWhlp3,y
1CD8: 99 50 0B           sta   VIEWhlp3,y
1CDB: 8A                 txa
1CDC: 99 40 1D           sta   kbhelp3,y
1CDF: B9 68 1D           lda   kbhelp4,y  ; Swap fourth line.
1CE2: BE D0 0B           ldx   VIEWhlp4,y
1CE5: 99 D0 0B           sta   VIEWhlp4,y
1CE8: 8A                 txa
1CE9: 99 68 1D           sta   kbhelp4,y
1CEC: 88                 dey
1CED: 10 C9              bpl   :swaplp
1CEF: 60        :done    rts

                * Variables and tables

                * The following help lines are swapped with the VIEWhlp
                * lines in the Auxiliary Text page (assembled with the
                * Keyboard Input module.  They are swapped back when
                * exiting Keyboard mode.

1CF0: A0 A0 A0  kbhelp1  asc   "   Keyboard Input mode (ESC to exit)    "
1D18: C6 EF F2  kbhelp2  asc   "For Function Keys, hold Open-Apple down."
1D40: D0 F2 EF  kbhelp3  asc   "Program Control keys: Attention: ctl-@  "
1D68: D2 E5 F3  kbhelp4  asc   "Restart: ctl-R, Cold Restart: ctl-C     "

1D90: 00        hx       db    0          ; History index (mod 256)
1D91: 05        scale    db    5          ; 1..5 ==> 1/n scaling
1D92: 00 50     delta    db    $00,$50    ; BCD 50
1D94: FF        px       db    $FF        ; "light pen" x = 0-127
1D95: 00        py       db    0          ; "light pen" y = 0-95
1D96: 00        xx       db    0          ; Pen HGR x coordinate
1D97: 00        yy       db    0          ; Pen HGR y coordinate

                xlyl     equ   */256*256  ; Base page for xl and yl
1D98: 00 00     xl       db    00,00      ; BCD left window edge
1D9A: 00 00     yl       db    00,00      ; BCD lower window edge
```

```
*       Caltech CRT keyboard mapping tables
*
* The 'key' and 'fkey' macros are used to populate two
* 128-byte tables mapping ASCII character codes into a
* specially coded byte corresponding to the two-digit
* octal code and two modifier bits produced by the
* keyboards used with the Caltech CRT display.
*
* The 'fkey' macro is used in a similar way to map keys
* pressed while the Open-Apple key is held down to the
* 32-key Function Key keypad.
*
* The first parameter is the two-digit octal code
* specifying the key pressed, and the second parameter
* specifies whether the key has the Shift modifier
* or not.
*
* 'key' is used to populate the mapping table for normal
* keys (indicated by a high bit of 0) and 'fkey' is used
* to populate the table for Function keys (indicated by
* a high bit of 1).
*
* The format of the table entries is:
*       +----+----+----+----+----+----+----+----+
*       | FK | Lo Octal dig | UC | Hi Octal dig |
*       +----+----+----+----+----+----+----+----+

               >608           align 256        ; Page align
1D9C: 00 00 00 >608           ds    *-1/256*256+256-*
               >608           eom
               >609
               >610  keytbl   key   00;$80     ; $00 ^@  "Attention" key
1E00: 80       >610           db    0-]hd*10+00*16*10+00/10+$80
               >611           key   invalid;0  ; $01 ^A
1E01: 55       >611           db    0-]hd*10+invalid*16*10+invalid/10+0
               >612           key   invalid;0  ; $02 ^B
1E02: 55       >612           db    0-]hd*10+invalid*16*10+invalid/10+0
               >613           key   00;uc      ; $03 ^C  "Cold Restart" key
1E03: 08       >613           db    0-]hd*10+00*16*10+00/10+uc
               >614           key   invalid;0  ; $04 ^D
1E04: 55       >614           db    0-]hd*10+invalid*16*10+invalid/10+0
               >615           key   invalid;0  ; $05 ^E
1E05: 55       >615           db    0-]hd*10+invalid*16*10+invalid/10+0
               >616           key   invalid;0  ; $06 ^F
1E06: 55       >616           db    0-]hd*10+invalid*16*10+invalid/10+0
               >617           key   invalid;0  ; $07 ^G
1E07: 55       >617           db    0-]hd*10+invalid*16*10+invalid/10+0
               >618           key   75;lc      ; $08 Right arrow
1E08: 57       >618           db    0-]hd*10+75*16*10+75/10+lc
               >619           key   invalid;0  ; $09 ^I
1E09: 55       >619           db    0-]hd*10+invalid*16*10+invalid/10+0
               >620           key   invalid;0  ; $0A ^J
1E0A: 55       >620           db    0-]hd*10+invalid*16*10+invalid/10+0
               >621           key   invalid;0  ; $0B ^K
1E0B: 55       >621           db    0-]hd*10+invalid*16*10+invalid/10+0
               >622           key   invalid;0  ; $0C ^L
1E0C: 55       >622           db    0-]hd*10+invalid*16*10+invalid/10+0
               >623           key   77;lc      ; $0D Enter
1E0D: 77       >623           db    0-]hd*10+77*16*10+77/10+lc
               >624           key   invalid;0  ; $0E ^N
1E0E: 55       >624           db    0-]hd*10+invalid*16*10+invalid/10+0
               >625           key   invalid;0  ; $0F ^O
1E0F: 55       >625           db    0-]hd*10+invalid*16*10+invalid/10+0
               >626           key   invalid;0  ; $10 ^P
1E10: 55       >626           db    0-]hd*10+invalid*16*10+invalid/10+0
               >627           key   invalid;0  ; $11 ^Q
1E11: 55       >627           db    0-]hd*10+invalid*16*10+invalid/10+0
               >628           key   00;0       ; $12 ^R  "Restart" key
1E12: 00       >628           db    0-]hd*10+00*16*10+00/10+0
               >629           key   invalid;0  ; $13 ^S
1E13: 55       >629           db    0-]hd*10+invalid*16*10+invalid/10+0
               >630           key   invalid;0  ; $14 ^T
1E14: 55       >630           db    0-]hd*10+invalid*16*10+invalid/10+0
               >631           key   76;lc      ; $15 Left arrow
1E15: 67       >631           db    0-]hd*10+76*16*10+76/10+lc
               >632           key   invalid;0  ; $16 ^V
1E16: 55       >632           db    0-]hd*10+invalid*16*10+invalid/10+0
               >633           key   invalid;0  ; $17 ^W
1E17: 55       >633           db    0-]hd*10+invalid*16*10+invalid/10+0
               >634           key   invalid;0  ; $18 ^X
1E18: 55       >634           db    0-]hd*10+invalid*16*10+invalid/10+0
               >635           key   invalid;0  ; $19 ^Y
1E19: 55       >635           db    0-]hd*10+invalid*16*10+invalid/10+0
               >636           key   invalid;0  ; $1A ^Z
1E1A: 55       >636           db    0-]hd*10+invalid*16*10+invalid/10+0
               >637           key   invalid;0  ; $1B Escape
1E1B: 55       >637           db    0-]hd*10+invalid*16*10+invalid/10+0
               >638           key   invalid;0  ; $1C ^\
1E1C: 55       >638           db    0-]hd*10+invalid*16*10+invalid/10+0
               >639           key   invalid;0  ; $1D ^]
1E1D: 55       >639           db    0-]hd*10+invalid*16*10+invalid/10+0
               >640           key   invalid;0  ; $1E ^^
1E1E: 55       >640           db    0-]hd*10+invalid*16*10+invalid/10+0
               >641           key   invalid;0  ; $1F ^_
```

```
1E1F: 55        >641            db    0-]hd*10+invalid*16*10+invalid/10+0
                >642            key   60;lc      ; $20 Space
1E20: 06        >642            db    0-]hd*10+60*16*10+60/10+lc
                >643            key   invalid;0  ; $21 !
1E21: 55        >643            db    0-]hd*10+invalid*16*10+invalid/10+0
                >644            key   invalid;0  ; $22 "
1E22: 55        >644            db    0-]hd*10+invalid*16*10+invalid/10+0
                >645            key   invalid;0  ; $23 #
1E23: 55        >645            db    0-]hd*10+invalid*16*10+invalid/10+0
                >646            key   53;lc      ; $24 $
1E24: 35        >646            db    0-]hd*10+53*16*10+53/10+lc
                >647            key   invalid;0  ; $25 %
1E25: 55        >647            db    0-]hd*10+invalid*16*10+invalid/10+0
                >648            key   invalid;0  ; $26 &
1E26: 55        >648            db    0-]hd*10+invalid*16*10+invalid/10+0
                >649            key   06;uc      ; $27 '
1E27: 68        >649            db    0-]hd*10+06*16*10+06/10+uc
                >650            key   11;uc      ; $28 (
1E28: 19        >650            db    0-]hd*10+11*16*10+11/10+uc
                >651            key   12;uc      ; $29 )
1E29: 29        >651            db    0-]hd*10+12*16*10+12/10+uc
                >652            key   10;uc      ; $2A *
1E2A: 09        >652            db    0-]hd*10+10*16*10+10/10+uc
                >653            key   20;uc      ; $2B +
1E2B: 0A        >653            db    0-]hd*10+20*16*10+20/10+uc
                >654            key   73;lc      ; $2C ;
1E2C: 37        >654            db    0-]hd*10+73*16*10+73/10+lc
                >655            key   40;lc      ; $2D -
1E2D: 04        >655            db    0-]hd*10+40*16*10+40/10+lc
                >656            key   33;lc      ; $2E .
1E2E: 33        >656            db    0-]hd*10+33*16*10+33/10+lc
                >657            key   61;lc      ; $2F /
1E2F: 16        >657            db    0-]hd*10+61*16*10+61/10+lc
                >658            key   12;lc      ; $30 0
1E30: 21        >658            db    0-]hd*10+12*16*10+12/10+lc
                >659            key   01;lc      ; $31 1
1E31: 10        >659            db    0-]hd*10+01*16*10+01/10+lc
                >660            key   02;lc      ; $32 2
1E32: 20        >660            db    0-]hd*10+02*16*10+02/10+lc
                >661            key   03;lc      ; $33 3
1E33: 30        >661            db    0-]hd*10+03*16*10+03/10+lc
                >662            key   04;lc      ; $34 4
1E34: 40        >662            db    0-]hd*10+04*16*10+04/10+lc
                >663            key   05;lc      ; $35 5
1E35: 50        >663            db    0-]hd*10+05*16*10+05/10+lc
                >664            key   06;lc      ; $36 6
1E36: 60        >664            db    0-]hd*10+06*16*10+06/10+lc
                >665            key   07;lc      ; $37 7
1E37: 70        >665            db    0-]hd*10+07*16*10+07/10+lc
                >666            key   10;lc      ; $38 8
1E38: 01        >666            db    0-]hd*10+10*16*10+10/10+lc
                >667            key   11;lc      ; $39 9
1E39: 11        >667            db    0-]hd*10+11*16*10+11/10+lc
                >668            key   invalid;0  ; $3A :
1E3A: 55        >668            db    0-]hd*10+invalid*16*10+invalid/10+0
                >669            key   invalid;0  ; $3B ;
1E3B: 55        >669            db    0-]hd*10+invalid*16*10+invalid/10+0
                >670            key   invalid;0  ; $3C <
1E3C: 55        >670            db    0-]hd*10+invalid*16*10+invalid/10+0
                >671            key   01;uc      ; $3D =
1E3D: 18        >671            db    0-]hd*10+01*16*10+01/10+uc
                >672            key   invalid;0  ; $3E >
1E3E: 55        >672            db    0-]hd*10+invalid*16*10+invalid/10+0
                >673            key   61;uc      ; $3F ?
1E3F: 1E        >673            db    0-]hd*10+61*16*10+61/10+uc
                >674            key   invalid;0  ; $40 @
1E40: 55        >674            db    0-]hd*10+invalid*16*10+invalid/10+0
```

```
               >675           key   21;uc      ; $41 A
1E41: 1A       >675           db    0-]hd*10+21*16*10+21/10+uc
               >676           key   22;uc      ; $42 B
1E42: 2A       >676           db    0-]hd*10+22*16*10+22/10+uc
               >677           key   23;uc      ; $43 C
1E43: 3A       >677           db    0-]hd*10+23*16*10+23/10+uc
               >678           key   24;uc      ; $44 D
1E44: 4A       >678           db    0-]hd*10+24*16*10+24/10+uc
               >679           key   25;uc      ; $45 E
1E45: 5A       >679           db    0-]hd*10+25*16*10+25/10+uc
               >680           key   26;uc      ; $46 F
1E46: 6A       >680           db    0-]hd*10+26*16*10+26/10+uc
               >681           key   27;uc      ; $47 G
1E47: 7A       >681           db    0-]hd*10+27*16*10+27/10+uc
               >682           key   30;uc      ; $48 H
1E48: 0B       >682           db    0-]hd*10+30*16*10+30/10+uc
               >683           key   31;uc      ; $49 I
1E49: 1B       >683           db    0-]hd*10+31*16*10+31/10+uc
               >684           key   41;uc      ; $4A J
1E4A: 1C       >684           db    0-]hd*10+41*16*10+41/10+uc
               >685           key   42;uc      ; $4B K
1E4B: 2C       >685           db    0-]hd*10+42*16*10+42/10+uc
               >686           key   43;uc      ; $4C L
1E4C: 3C       >686           db    0-]hd*10+43*16*10+43/10+uc
               >687           key   44;uc      ; $4D M
1E4D: 4C       >687           db    0-]hd*10+44*16*10+44/10+uc
               >688           key   45;uc      ; $4E N
1E4E: 5C       >688           db    0-]hd*10+45*16*10+45/10+uc
               >689           key   46;uc      ; $4F O
1E4F: 6C       >689           db    0-]hd*10+46*16*10+46/10+uc
               >690           key   47;uc      ; $50 P
1E50: 7C       >690           db    0-]hd*10+47*16*10+47/10+uc
               >691           key   50;uc      ; $51 Q
1E51: 0D       >691           db    0-]hd*10+50*16*10+50/10+uc
               >692           key   51;uc      ; $52 R
1E52: 1D       >692           db    0-]hd*10+51*16*10+51/10+uc
               >693           key   62;uc      ; $53 S
1E53: 2E       >693           db    0-]hd*10+62*16*10+62/10+uc
               >694           key   63;uc      ; $54 T
1E54: 3E       >694           db    0-]hd*10+63*16*10+63/10+uc
               >695           key   64;uc      ; $55 U
1E55: 4E       >695           db    0-]hd*10+64*16*10+64/10+uc
               >696           key   65;uc      ; $56 V
1E56: 5E       >696           db    0-]hd*10+65*16*10+65/10+uc
               >697           key   66;uc      ; $57 W
1E57: 6E       >697           db    0-]hd*10+66*16*10+66/10+uc
               >698           key   67;uc      ; $58 X
1E58: 7E       >698           db    0-]hd*10+67*16*10+67/10+uc
               >699           key   70;uc      ; $59 Y
1E59: 0F       >699           db    0-]hd*10+70*16*10+70/10+uc
               >700           key   71;uc      ; $5A Z
1E5A: 1F       >700           db    0-]hd*10+71*16*10+71/10+uc
               >701           key   invalid;0  ; $5B [
1E5B: 55       >701           db    0-]hd*10+invalid*16*10+invalid/10+0
               >702           key   invalid;0  ; $5C \
1E5C: 55       >702           db    0-]hd*10+invalid*16*10+invalid/10+0
               >703           key   invalid;0  ; $5D ]
1E5D: 55       >703           db    0-]hd*10+invalid*16*10+invalid/10+0
               >704           key   invalid;0  ; $5E ^
1E5E: 55       >704           db    0-]hd*10+invalid*16*10+invalid/10+0
               >705           key   invalid;0  ; $5F _
1E5F: 55       >705           db    0-]hd*10+invalid*16*10+invalid/10+0
               >706           key   invalid;0  ; $60 `
1E60: 55       >706           db    0-]hd*10+invalid*16*10+invalid/10+0
               >707           key   21;lc      ; $61 a
1E61: 12       >707           db    0-]hd*10+21*16*10+21/10+lc
               >708           key   22;lc      ; $62 b
```

```
1E62: 22        >708            db    0-]hd*10+22*16*10+22/10+lc
                >709            key   23;lc      ; $63 c
1E63: 32        >709            db    0-]hd*10+23*16*10+23/10+lc
                >710            key   24;lc      ; $64 d
1E64: 42        >710            db    0-]hd*10+24*16*10+24/10+lc
                >711            key   25;lc      ; $65 e
1E65: 52        >711            db    0-]hd*10+25*16*10+25/10+lc
                >712            key   26;lc      ; $66 f
1E66: 62        >712            db    0-]hd*10+26*16*10+26/10+lc
                >713            key   27;lc      ; $67 g
1E67: 72        >713            db    0-]hd*10+27*16*10+27/10+lc
                >714            key   30;lc      ; $68 h
1E68: 03        >714            db    0-]hd*10+30*16*10+30/10+lc
                >715            key   31;lc      ; $69 i
1E69: 13        >715            db    0-]hd*10+31*16*10+31/10+lc
                >716            key   41;lc      ; $6A j
1E6A: 14        >716            db    0-]hd*10+41*16*10+41/10+lc
                >717            key   42;lc      ; $6B k
1E6B: 24        >717            db    0-]hd*10+42*16*10+42/10+lc
                >718            key   43;lc      ; $6C l
1E6C: 34        >718            db    0-]hd*10+43*16*10+43/10+lc
                >719            key   44;lc      ; $6D m
1E6D: 44        >719            db    0-]hd*10+44*16*10+44/10+lc
                >720            key   45;lc      ; $6E n
1E6E: 54        >720            db    0-]hd*10+45*16*10+45/10+lc
                >721            key   46;lc      ; $6F o
1E6F: 64        >721            db    0-]hd*10+46*16*10+46/10+lc
                >722            key   47;lc      ; $70 p
1E70: 74        >722            db    0-]hd*10+47*16*10+47/10+lc
                >723            key   50;lc      ; $71 q
1E71: 05        >723            db    0-]hd*10+50*16*10+50/10+lc
                >724            key   51;lc      ; $72 r
1E72: 15        >724            db    0-]hd*10+51*16*10+51/10+lc
                >725            key   62;lc      ; $73 s
1E73: 26        >725            db    0-]hd*10+62*16*10+62/10+lc
                >726            key   63;lc      ; $74 t
1E74: 36        >726            db    0-]hd*10+63*16*10+63/10+lc
                >727            key   64;lc      ; $75 u
1E75: 46        >727            db    0-]hd*10+64*16*10+64/10+lc
                >728            key   65;lc      ; $76 v
1E76: 56        >728            db    0-]hd*10+65*16*10+65/10+lc
                >729            key   66;lc      ; $77 w
1E77: 66        >729            db    0-]hd*10+66*16*10+66/10+lc
                >730            key   67;lc      ; $78 x
1E78: 76        >730            db    0-]hd*10+67*16*10+67/10+lc
                >731            key   70;lc      ; $79 y
1E79: 07        >731            db    0-]hd*10+70*16*10+70/10+lc
                >732            key   71;lc      ; $7A z
1E7A: 17        >732            db    0-]hd*10+71*16*10+71/10+lc
                >733            key   invalid;0  ; $7B {
1E7B: 55        >733            db    0-]hd*10+invalid*16*10+invalid/10+0
                >734            key   invalid;0  ; $7C |
1E7C: 55        >734            db    0-]hd*10+invalid*16*10+invalid/10+0
                >735            key   invalid;0  ; $7D }
1E7D: 55        >735            db    0-]hd*10+invalid*16*10+invalid/10+0
                >736            key   invalid;0  ; $7E ~
1E7E: 55        >736            db    0-]hd*10+invalid*16*10+invalid/10+0
                >737            key   invalid;0  ; $7F Delete
1E7F: 55        >737            db    0-]hd*10+invalid*16*10+invalid/10+0
```

```
              >739  fktbl    fkey  invalid;0  ; $00 ^@
1E80: D5      >739           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >740           fkey  invalid;0  ; $01 ^A
1E81: D5      >740           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >741           fkey  invalid;0  ; $02 ^B
1E82: D5      >741           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >742           fkey  invalid;0  ; $03 ^C
1E83: D5      >742           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >743           fkey  invalid;0  ; $04 ^D
1E84: D5      >743           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >744           fkey  invalid;0  ; $05 ^E
1E85: D5      >744           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >745           fkey  invalid;0  ; $06 ^F
1E86: D5      >745           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >746           fkey  invalid;0  ; $07 ^G
1E87: D5      >746           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >747           fkey  invalid;0  ; $08 Right arrow
1E88: D5      >747           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >748           fkey  invalid;0  ; $09 ^I
1E89: D5      >748           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >749           fkey  invalid;0  ; $0A ^J
1E8A: D5      >749           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >750           fkey  invalid;0  ; $0B ^K
1E8B: D5      >750           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >751           fkey  invalid;0  ; $0C ^L
1E8C: D5      >751           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >752           fkey  invalid;0  ; $0D Enter
1E8D: D5      >752           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >753           fkey  invalid;0  ; $0E ^N
1E8E: D5      >753           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >754           fkey  invalid;0  ; $0F ^O
1E8F: D5      >754           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >755           fkey  invalid;0  ; $10 ^P
1E90: D5      >755           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >756           fkey  invalid;0  ; $11 ^Q
1E91: D5      >756           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >757           fkey  invalid;0  ; $12 ^R
1E92: D5      >757           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >758           fkey  invalid;0  ; $13 ^S
1E93: D5      >758           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >759           fkey  invalid;0  ; $14 ^T
1E94: D5      >759           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >760           fkey  invalid;0  ; $15 Left arrow
1E95: D5      >760           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >761           fkey  invalid;0  ; $16 ^V
1E96: D5      >761           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >762           fkey  invalid;0  ; $17 ^W
1E97: D5      >762           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >763           fkey  invalid;0  ; $18 ^X
1E98: D5      >763           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >764           fkey  invalid;0  ; $19 ^Y
1E99: D5      >764           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >765           fkey  invalid;0  ; $1A ^Z
1E9A: D5      >765           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >766           fkey  invalid;0  ; $1B Escape
1E9B: D5      >766           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >767           fkey  invalid;0  ; $1C ^\
1E9C: D5      >767           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >768           fkey  invalid;0  ; $1D ^]
1E9D: D5      >768           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >769           fkey  invalid;0  ; $1E ^^
1E9E: D5      >769           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >770           fkey  invalid;0  ; $1F ^_
1E9F: D5      >770           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >771           fkey  invalid;0  ; $20 Space
1EA0: D5      >771           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
              >772           fkey  01;uc      ; $21 !
```

```
1EA1: 98        >772            db    0-]hd*10+01*16*10+01/10+$80+uc
                >773            fkey  invalid;0  ; $22 "
1EA2: D5        >773            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >774            fkey  03;uc      ; $23 #
1EA3: B8        >774            db    0-]hd*10+03*16*10+03/10+$80+uc
                >775            fkey  04;uc      ; $24 $
1EA4: C8        >775            db    0-]hd*10+04*16*10+04/10+$80+uc
                >776            fkey  05;uc      ; $25 %
1EA5: D8        >776            db    0-]hd*10+05*16*10+05/10+$80+uc
                >777            fkey  07;uc      ; $26 &
1EA6: F8        >777            db    0-]hd*10+07*16*10+07/10+$80+uc
                >778            fkey  invalid;0  ; $27 '
1EA7: D5        >778            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >779            fkey  09;uc      ; $28 (
1EA8: 18        >779            db    0-]hd*10+09*16*10+09/10+$80+uc
                >780            fkey  10;uc      ; $29 )
1EA9: 89        >780            db    0-]hd*10+10*16*10+10/10+$80+uc
                >781            fkey  08;uc      ; $2A *
1EAA: 08        >781            db    0-]hd*10+08*16*10+08/10+$80+uc
                >782            fkey  invalid;0  ; $2B +
1EAB: D5        >782            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >783            fkey  invalid;0  ; $2C ;
1EAC: D5        >783            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >784            fkey  invalid;0  ; $2D -
1EAD: D5        >784            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >785            fkey  invalid;0  ; $2E .
1EAE: D5        >785            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >786            fkey  invalid;0  ; $2F /
1EAF: D5        >786            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >787            fkey  12;lc      ; $30 0
1EB0: A1        >787            db    0-]hd*10+12*16*10+12/10+$80+lc
                >788            fkey  01;lc      ; $31 1
1EB1: 90        >788            db    0-]hd*10+01*16*10+01/10+$80+lc
                >789            fkey  02;lc      ; $32 2
1EB2: A0        >789            db    0-]hd*10+02*16*10+02/10+$80+lc
                >790            fkey  03;lc      ; $33 3
1EB3: B0        >790            db    0-]hd*10+03*16*10+03/10+$80+lc
                >791            fkey  04;lc      ; $34 4
1EB4: C0        >791            db    0-]hd*10+04*16*10+04/10+$80+lc
                >792            fkey  05;lc      ; $35 5
1EB5: D0        >792            db    0-]hd*10+05*16*10+05/10+$80+lc
                >793            fkey  06;lc      ; $36 6
1EB6: E0        >793            db    0-]hd*10+06*16*10+06/10+$80+lc
                >794            fkey  07;lc      ; $37 7
1EB7: F0        >794            db    0-]hd*10+07*16*10+07/10+$80+lc
                >795            fkey  10;lc      ; $38 8
1EB8: 81        >795            db    0-]hd*10+10*16*10+10/10+$80+lc
                >796            fkey  11;lc      ; $39 9
1EB9: 91        >796            db    0-]hd*10+11*16*10+11/10+$80+lc
                >797            fkey  invalid;0  ; $3A :
1EBA: D5        >797            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >798            fkey  invalid;0  ; $3B ;
1EBB: D5        >798            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >799            fkey  invalid;0  ; $3C <
1EBC: D5        >799            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >800            fkey  invalid;0  ; $3D =
1EBD: D5        >800            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >801            fkey  invalid;0  ; $3E >
1EBE: D5        >801            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >802            fkey  invalid;0  ; $3F ?
1EBF: D5        >802            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >803            fkey  02;uc      ; $40 @
1EC0: A8        >803            db    0-]hd*10+02*16*10+02/10+$80+uc
                >804            fkey  13;uc      ; $41 A
1EC1: B9        >804            db    0-]hd*10+13*16*10+13/10+$80+uc
                >805            fkey  14;uc      ; $42 B
1EC2: C9        >805            db    0-]hd*10+14*16*10+14/10+$80+uc
```

```
                >806            fkey  15;uc      ; $43 C
1EC3: D9        >806            db    0-]hd*10+15*16*10+15/10+$80+uc
                >807            fkey  16;uc      ; $44 D
1EC4: E9        >807            db    0-]hd*10+16*16*10+16/10+$80+uc
                >808            fkey  17;uc      ; $45 E
1EC5: F9        >808            db    0-]hd*10+17*16*10+17/10+$80+uc
                >809            fkey  20;uc      ; $46 F
1EC6: 8A        >809            db    0-]hd*10+20*16*10+20/10+$80+uc
                >810            fkey  21;uc      ; $47 G
1EC7: 9A        >810            db    0-]hd*10+21*16*10+21/10+$80+uc
                >811            fkey  22;uc      ; $48 H
1EC8: AA        >811            db    0-]hd*10+22*16*10+22/10+$80+uc
                >812            fkey  23;uc      ; $49 I
1EC9: BA        >812            db    0-]hd*10+23*16*10+23/10+$80+uc
                >813            fkey  24;uc      ; $4A J
1ECA: CA        >813            db    0-]hd*10+24*16*10+24/10+$80+uc
                >814            fkey  25;uc      ; $4B K
1ECB: DA        >814            db    0-]hd*10+25*16*10+25/10+$80+uc
                >815            fkey  26;uc      ; $4C L
1ECC: EA        >815            db    0-]hd*10+26*16*10+26/10+$80+uc
                >816            fkey  27;uc      ; $4D M
1ECD: FA        >816            db    0-]hd*10+27*16*10+27/10+$80+uc
                >817            fkey  30;uc      ; $4E N
1ECE: 8B        >817            db    0-]hd*10+30*16*10+30/10+$80+uc
                >818            fkey  31;uc      ; $4F O
1ECF: 9B        >818            db    0-]hd*10+31*16*10+31/10+$80+uc
                >819            fkey  32;uc      ; $50 P
1ED0: AB        >819            db    0-]hd*10+32*16*10+32/10+$80+uc
                >820            fkey  33;uc      ; $51 Q
1ED1: BB        >820            db    0-]hd*10+33*16*10+33/10+$80+uc
                >821            fkey  34;uc      ; $52 R
1ED2: CB        >821            db    0-]hd*10+34*16*10+34/10+$80+uc
                >822            fkey  35;uc      ; $53 S
1ED3: DB        >822            db    0-]hd*10+35*16*10+35/10+$80+uc
                >823            fkey  36;uc      ; $54 T
1ED4: EB        >823            db    0-]hd*10+36*16*10+36/10+$80+uc
                >824            fkey  37;uc      ; $55 U
1ED5: FB        >824            db    0-]hd*10+37*16*10+37/10+$80+uc
                >825            fkey  40;uc      ; $56 V
1ED6: 8C        >825            db    0-]hd*10+40*16*10+40/10+$80+uc
                >826            fkey  invalid;0  ; $57 W
1ED7: D5        >826            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >827            fkey  invalid;0  ; $58 X
1ED8: D5        >827            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >828            fkey  invalid;0  ; $59 Y
1ED9: D5        >828            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >829            fkey  invalid;0  ; $5A Z
1EDA: D5        >829            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >830            fkey  invalid;0  ; $5B [
1EDB: D5        >830            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >831            fkey  invalid;0  ; $5C \
1EDC: D5        >831            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >832            fkey  invalid;0  ; $5D ]
1EDD: D5        >832            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >833            fkey  06;uc      ; $5E ^
1EDE: E8        >833            db    0-]hd*10+06*16*10+06/10+$80+uc
                >834            fkey  invalid;0  ; $5F _
1EDF: D5        >834            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >835            fkey  invalid;0  ; $60 `
1EE0: D5        >835            db    0-]hd*10+invalid*16*10+invalid/10+$80+0
                >836            fkey  13;lc      ; $61 a
1EE1: B1        >836            db    0-]hd*10+13*16*10+13/10+$80+lc
                >837            fkey  14;lc      ; $62 b
1EE2: C1        >837            db    0-]hd*10+14*16*10+14/10+$80+lc
                >838            fkey  15;lc      ; $63 c
1EE3: D1        >838            db    0-]hd*10+15*16*10+15/10+$80+lc
                >839            fkey  16;lc      ; $64 d
```

```
1EE4: E1       >839           db    0-]hd*10+16*16*10+16/10+$80+lc
               >840           fkey  17;lc      ; $65 e
1EE5: F1       >840           db    0-]hd*10+17*16*10+17/10+$80+lc
               >841           fkey  20;lc      ; $66 f
1EE6: 82       >841           db    0-]hd*10+20*16*10+20/10+$80+lc
               >842           fkey  21;lc      ; $67 g
1EE7: 92       >842           db    0-]hd*10+21*16*10+21/10+$80+lc
               >843           fkey  22;lc      ; $68 h
1EE8: A2       >843           db    0-]hd*10+22*16*10+22/10+$80+lc
               >844           fkey  23;lc      ; $69 i
1EE9: B2       >844           db    0-]hd*10+23*16*10+23/10+$80+lc
               >845           fkey  24;lc      ; $6A j
1EEA: C2       >845           db    0-]hd*10+24*16*10+24/10+$80+lc
               >846           fkey  25;lc      ; $6B k
1EEB: D2       >846           db    0-]hd*10+25*16*10+25/10+$80+lc
               >847           fkey  26;lc      ; $6C l
1EEC: E2       >847           db    0-]hd*10+26*16*10+26/10+$80+lc
               >848           fkey  27;lc      ; $6D m
1EED: F2       >848           db    0-]hd*10+27*16*10+27/10+$80+lc
               >849           fkey  30;lc      ; $6E n
1EEE: 83       >849           db    0-]hd*10+30*16*10+30/10+$80+lc
               >850           fkey  31;lc      ; $6F o
1EEF: 93       >850           db    0-]hd*10+31*16*10+31/10+$80+lc
               >851           fkey  32;lc      ; $70 p
1EF0: A3       >851           db    0-]hd*10+32*16*10+32/10+$80+lc
               >852           fkey  33;lc      ; $71 q
1EF1: B3       >852           db    0-]hd*10+33*16*10+33/10+$80+lc
               >853           fkey  34;lc      ; $72 r
1EF2: C3       >853           db    0-]hd*10+34*16*10+34/10+$80+lc
               >854           fkey  35;lc      ; $73 s
1EF3: D3       >854           db    0-]hd*10+35*16*10+35/10+$80+lc
               >855           fkey  36;lc      ; $74 t
1EF4: E3       >855           db    0-]hd*10+36*16*10+36/10+$80+lc
               >856           fkey  37;lc      ; $75 u
1EF5: F3       >856           db    0-]hd*10+37*16*10+37/10+$80+lc
               >857           fkey  40;lc      ; $76 v
1EF6: 84       >857           db    0-]hd*10+40*16*10+40/10+$80+lc
               >858           fkey  invalid;0  ; $77 w
1EF7: D5       >858           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
               >859           fkey  invalid;0  ; $78 x
1EF8: D5       >859           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
               >860           fkey  invalid;0  ; $79 y
1EF9: D5       >860           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
               >861           fkey  invalid;0  ; $7A z
1EFA: D5       >861           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
               >862           fkey  invalid;0  ; $7B {
1EFB: D5       >862           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
               >863           fkey  invalid;0  ; $7C |
1EFC: D5       >863           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
               >864           fkey  invalid;0  ; $7D }
1EFD: D5       >864           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
               >865           fkey  invalid;0  ; $7E ~
1EFE: D5       >865           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
               >866           fkey  invalid;0  ; $7F Delete
1EFF: D5       >866           db    0-]hd*10+invalid*16*10+invalid/10+$80+0
               >867
               >869
1F00: 00 00 00 >870  histx    ds    256        ; x history table
2000: 00 00 00 >871  histy    ds    256        ; y history table
               >872
               >873           dum   *          ; VIEW BCD X,Y --> HGR x,y tables
2100: 00 00 00 >874  xmap     ds    10*256     ; B220 X (0..999) --> HGR X (0..254)
2B00: 00 00 00 >875  ymap     ds    10*256     ; B220 Y (0..999) --> HGR Y (0..191)
3500: 00 00 00 >876  xbyte    ds    256        ; X byte offset
3600: 00 00 00 >877  xbit     ds    256        ; X bit in byte
```

```
3700: 00 00 00 >878  ybasel   ds    192        ; Y line base lo
37C0: 00 00 00 >879  ybaseh   ds    192        ; Y line base hi
               >880
               >881  VIEWend  equ   *          ; End of VIEW module
               >882           dend
```

```
                MAINend  equ   VIEWend    ; End of MAIN seg (below buffers)
                         err   MAINend/ptrdr0bf ; Can't overrun buffers.

                AUXcode  equ   *          ; Start of Aux code
                         org   endcomm    ; Aux mem origin
                         put   B220FETCH
                **********************************************************
                *                                                        *
                *            Simulate next B220 Instruction              *
                *                                                        *
                **********************************************************

0928: 4C 3F 0A  ADDRerrR jmp   ADDRerr    ; Relay branch
092B: 4C 49 0A  UNDIGerR jmp   UNDIGerr   ; Relay branch
092E: 4C 43 08  keyinR   jmp   M_keyin    ; Relay branch to Main
0931: 4C 4C 08  stopR    jmp   M_stop     ; Relay branch to Main

                * Convert rP to instruction address

0934: A6 97     newP     ldx   rP+1       ; Low 2 BCD digits of rP
0936: E0 9A              cpx   #$99+1     ; Undigits?
0938: B0 F1              bcs   UNDIGerR   ; -Yes, error.
093A: A4 96              ldy   rP         ; High 2 BCD digits of rP
093C: C0 4A              cpy   #$49+1     ; ADDR error?
093E: B0 E8              bcs   ADDRerrR   ; -Yes, stop.
0940: BD E8 1A           lda   BCDLadrl,x ; -No, compute 'instptr'
0943: 79 1C 1C           adc   BCDHadrl,y
0946: 85 CF              sta   instptr    ; Low byte of instr address
0948: BD 82 1B           lda   BCDLadrh,x
094B: 79 66 1C           adc   BCDHadrh,y
094E: B0 DB              bcs   UNDIGerR   ; Carry out ==> undigit(s)
0950: 85 D0              sta   instptr+1  ; High byte of instr address
0952: A0 00     fetch    ldy   #0         ; Fetch next instruction.
0954: 84 C6              sty   skipincP   ; Don't skip incP
0956: A5 96              lda   rP
0958: C9 4A              cmp   #$49+1     ; rP >= 5000?
095A: B0 CC              bcs   ADDRerrR   ; -Yes, address error.
095C: B1 CF              lda   (instptr),y ;-No, fetch instruction.
095E: 85 98              sta   rC+S       ; Sign
0960: C8                 iny
0961: B1 CF              lda   (instptr),y
0963: 85 99              sta   rC+sL      ; (field) start, Length
0965: C8                 iny
0966: B1 CF              lda   (instptr),y
0968: 85 9A              sta   rC+VV      ; Variants
096A: C8                 iny
096B: B1 CF              lda   (instptr),y
096D: 85 9B              sta   rC+OP      ; OPcode
096F: C8                 iny
0970: B1 CF              lda   (instptr),y
0972: 85 9C              sta   rC+ADDR    ; High 2 digits of ADDR
0974: C8                 iny
0975: B1 CF              lda   (instptr),y
0977: 85 9D              sta   rC+ADDR+1  ; Low 2 digits of ADDR
0979: A5 98     execute  lda   rC+S       ; Is Sign negative?
097B: 29 01              and   #1
097D: F0 0F              beq   :noBmod    ; -No, skip rB modification
097F: F8                 sed              ; / Decimal mode
0980: 18                 clc
0981: A5 9D              lda   rC+ADDR+1  ; Add rB to rC+ADDR
0983: 65 95              adc   rB+1
0985: 85 9D              sta   rC+ADDR+1
0987: A5 9C              lda   rC+ADDR
0989: 65 94              adc   rB
098B: 85 9C              sta   rC+ADDR
098D: D8                 cld              ; \ Back to binary mode
098E: AD 00 C0  :noBmod  lda   KBD        ; User interaction?
```

```
0991: 30 9B             bmi   keyinR     ; -Yes, handle it.
0993: A5 C0             lda   RUN        ; RUN mode off
0995: 25 9B             and   rC+OP      ;  or HLT instruction?
0997: F0 98             beq   stopR      ; -Yes, stop.
0999: 8D 30 C0 ]X_sound sta   SPKR       ; -No, toggle speaker.
099C: C6 DA             dec   dispctr    ; Update display every
099E: 10 11             bpl   ]contin    ;  'dispcnt' instructions.
09A0: A9 64             lda   #dispcnt   ; Reset counter
09A2: 85 DA             sta   dispctr
09A4: E6 C7             inc   instctr    ; Count 'dispctr' resets.
09A6: D0 06             bne   :disp
09A8: E6 C8             inc   instctr+1
09AA: D0 02             bne   :disp
09AC: E6 C9             inc   instctr+2
09AE: 20 55 08 :disp    jsr   M_disp
09B1: A4 9B    ]contin  ldy   rC+OP      ; Op code
09B3: C0 60             cpy   #$60       ; OP out of range?
09B5: B0 7C             bcs   OPerr      ; -Yes, stop.
09B7: A5 C3             lda   Ov         ; -No, is Overflow set
09B9: 25 CE             and   OvHlt      ;   and Ovflo Halt mode?
09BB: F0 04             beq   :ok        ; -No, continue.
09BD: C0 31             cpy   #$31       ; -Yes, is OP BOF?
09BF: D0 76             bne   OFLerr     ; -No, Overflow error.
09C1: B9 5C 0A :ok      lda   optabl,y   ; -Yes, get execute address.
09C4: 8D 0E 0A          sta   :go+1
09C7: B9 B6 0A          lda   optabh,y   ; High bit set?
09CA: 85 D7             sta   t1         ; Save "no address" for trace.
09CC: 30 42             bmi   :noADDR    ; -Yes, ignore ADDR
09CE: 8D 0F 0A          sta   :go+2      ; -No, save execute address
09D1: A6 9D             ldx   rC+ADDR+1  ; Low 2 BCD ADDR digits
09D3: E0 9A             cpx   #$99+1     ; Undigits?
09D5: B0 72             bcs   UNDIGerr   ; -Yes, error.
09D7: A4 9C             ldy   rC+ADDR    ; High 2 BCD ADDR digits
09D9: C0 4A             cpy   #$49+1     ; ADDR error?
09DB: B0 62             bcs   ADDRerr    ; -Yes, stop.
09DD: BD E8 1A          lda   BCDLadrl,x ; -No, compute 'memptr'
09E0: 79 1C 1C          adc   BCDHadrl,y
09E3: 85 D1             sta   memptr     ; Low byte of memory address
09E5: BD 82 1B          lda   BCDLadrh,x
09E8: 79 66 1C          adc   BCDHadrh,y
09EB: B0 5C             bcs   UNDIGerr   ; Carry out ==> undigit(s).
09ED: 85 D2             sta   memptr+1   ; High byte of memory address
09EF: A5 CA    :xeq     lda   traceflg   ; Tracing?
09F1: F0 0D             beq   :notrace   ; -No.
09F3: A0 05             ldy   #5         ; -Yes, copy (memptr) to rD
09F5: B1 D1    :copylp  lda   (memptr),y
09F7: 99 AA 00          sta   rD,y
09FA: 88                dey
09FB: 10 F8             bpl   :copylp
09FD: 20 D9 08          jsr   M_trace    ;   and print trace info.
0A00: A5 C6    :notrace lda   skipincP   ; Skip increment P?
0A02: D0 03             bne   :skip      ; -Yes, PRB hit sign 6/7.
0A04: 20 17 0A          jsr   incP       ; -No, inc rP and instptr.
0A07: A0 00    :skip    ldy   #0         ; Enter execute with Y=0
0A09: B1 D1             lda   (memptr),y ;  & operand sign in A & rD+S.
0A0B: 85 AA             sta   rD+S
0A0D: 4C 00 00 :go      jmp   0*0        ; Go to execute routine.

0A10: 29 7F    :noADDR  and   #$7F       ; Turn off "noADDR" bit
0A12: 8D 0F 0A          sta   :go+2      ;  and save execute address.
0A15: D0 D8             bne   :xeq       ; (always)

               * Increment rP and instptr

0A17: F8       incP     sed              ; / BCD mode arithmetic
0A18: 18                clc
0A19: A5 97             lda   rP+1       ; Increment rP by 1
```

```
0A1B: 69 01              adc   #1
0A1D: 85 97              sta   rP+1
0A1F: 90 06              bcc   :nocar     ; Hi digits don't change.
0A21: A5 96              lda   rP         ; Propagate carry.
0A23: 69 00              adc   #0
0A25: 85 96              sta   rP
0A27: D8        :nocar   cld              ; \ Back to binary.
0A28: A5 CF              lda   instptr    ; Inc 'instptr' by 6
0A2A: 69 06              adc   #6
0A2C: 85 CF              sta   instptr
0A2E: 90 02              bcc   :nocarry
0A30: E6 D0              inc   instptr+1
0A32: 60        :nocarry rts
```

```
               * B220 error routines

0A33: A9 CF    OPerr    lda   #"O"       ; OPcode error
0A35: D0 14             bne   ]err       ; (always)

0A37: A9 D6    OFLerr   lda   #"V"       ; Overflow error
0A39: D0 10             bne   ]err       ; (always)

0A3B: A9 C6    FIELDerr lda   #"F"       ; Field error
0A3D: D0 0C             bne   ]err       ; (always)

0A3F: A9 C1    ADDRerr  lda   #"A"       ; Address error
0A41: D0 08             bne   ]err       ; (always)

0A43: 85 00    IOerr    sta   0          ; Save I/O err code
0A45: A9 C9             lda   #"I"       ; I/O error
0A47: D0 02             bne   ]err

0A49: A9 D8    UNDIGerr lda   #"X"       ; Non-BCD digit error
0A4B: 8D 04 C0 ]err     sta   WRITMAIN   ; Store to text screen
0A4E: 8D 67 05          sta   ERRlab     ; Show on screen.
0A51: 8D 05 C0          sta   WRITAUX    ; Back to Auxmem
0A54: 85 C1             sta   ERR        ; Set error indicator,
0A56: 20 DD FB          jsr   BEEP       ;  sound beep,
0A59: 4C 4C 08          jmp   M_stop     ;   and stop...
```

```
                 78             put   B220EXEC1
                >1    * OPcode execute phase dispatch table
                >2
                >3    optabl    equ   *           ; Low byte of execute routines
0A5C: 10        >4              db    <HLT        ; S ---- 00 ---- HaLT
0A5D: 10        >5              db    <NOP        ; S ---- 01 ---- No OP
0A5E: 33        >6              db    <OPerr      ;        02
0A5F: 13        >7              db    <PRD        ; S unnv 03 ADDR Pap tape RD
0A60: 57        >8              db    <PRB        ; S u--v 04 ADDR Pap tape Rd, Br
0A61: E5        >9              db    <PRI        ; S unnv 05 ADDR Pap tape Rd, Inv
0A62: E8        >10             db    <PWR        ; S unn- 06 ADDR Pap tape WR
0A63: 2F        >11             db    <PWI        ; S u--- 07 ADDR Pap tape Wr, Int
0A64: 40        >12             db    <KAD        ; S ---- 08 ---- Keyboard ADd
0A65: 32        >13             db    <SPO        ; S dnnv 09 ADDR Sup Print Out
0A66: 33 33 33 >14              db    <OPerr,<OPerr,<OPerr,<OPerr,<OPerr,<OPerr
0A6C: C6        >15             db    <CAD        ; S ---v 10 ADDR Clear ADd (Abs)
0A6D: B1        >16             db    <CSU        ; S ---v 11 ADDR Clear SUb (Abs)
0A6E: 1C        >17             db    <ADD        ; S ---v 12 ADDR ADD (Abs)
0A6F: AE        >18             db    <SUB        ; S ---v 13 ADDR SUBtract (Abs)
0A70: C4        >19             db    <MUL        ; S ---- 14 ADDR MULtiply
0A71: 4D        >20             db    <DIV        ; S ---- 15 ADDR DIVide
0A72: C8        >21             db    <RND        ; S ---- 16 ---- RouND
0A73: EA        >22             db    <EXT        ; S ---- 17 ADDR EXTract
0A74: 12        >23             db    <CFA        ; S sLfv 18 ADDR Comp Fld A (R)
0A75: 8C        >24             db    <ADL        ; S ---- 19 ADDR ADd to Location
0A76: 33 33 33 >25              db    <OPerr,<OPerr,<OPerr,<OPerr,<OPerr,<OPerr
0A7C: 78        >26             db    <IBB        ; S nnnn 20 ADDR Increase B, Br
0A7D: 8B        >27             db    <DBB        ; S nnnn 21 ADDR Decrease B, Br
0A7E: D0        >28             db    <FAD        ; S n--v 22 ADDR Float ADd (Abs)
0A7F: DD        >29             db    <FSU        ; S n--v 23 ADDR Float SUb (Abs)
0A80: F2        >30             db    <FMU        ; S ---- 24 ADDR Float MUltiply
0A81: 8D        >31             db    <FDV        ; S ---- 25 ADDR Float DiVide
0A82: 10        >32             db    <IFL        ; S sLnn 26 ADDR Inc Fld Loc
0A83: 56        >33             db    <DFL        ; S sLnn 27 ADDR Dec Fld Loc
0A84: 66        >34             db    <DLB        ; S sLnn 28 ADDR Dec fld loc,Ld B
0A85: 12        >35             db    <RTF        ; S -nn- 29 ADDR Record TransFer
0A86: 33 33 33 >36              db    <OPerr,<OPerr,<OPerr,<OPerr,<OPerr,<OPerr
0A8C: E1        >37             db    <BUN        ; S ---- 30 ADDR Branch UNcond
0A8D: 9E        >38             db    <BOF        ; S ---- 31 ADDR Branch OverFlow
0A8E: AB        >39             db    <BRP        ; S ---- 32 ADDR Branch RePeat
0A8F: B1        >40             db    <BSA        ; S ---n 33 ADDR Branch Sign A
0A90: BB        >41             db    <BCH        ; S ---v 34 ADDR Br Comp Hi (Lo)
0A91: CF        >42             db    <BCE        ; S ---v 35 ADDR Br Comp Eq (Un)
0A92: F8        >43             db    <BFA        ; S sLnn 36 ADDR Branch Field A
0A93: F4        >44             db    <BFR        ; S sLnn 37 ADDR Branch Field R
0A94: 47        >45             db    <BCS        ; S u--- 38 ADDR Br Control Sw
0A95: 54        >46             db    <SOR        ; S ---V 39 ---- Set Ov Remember
0A96: 33 33 33 >47              db    <OPerr,<OPerr,<OPerr,<OPerr,<OPerr,<OPerr
0A9C: 68        >48             db    <STA        ; S sLfv 40 ADDR STore A (R/B)
0A9D: DB        >49             db    <LDR        ; S ---- 41 ADDR LoaD R
0A9E: E7        >50             db    <LDB        ; S ---v 42 ADDR LoaD B (Comp)
0A9F: 0D        >51             db    <LSA        ; S ---n 43 ---- Load Sign A
0AA0: 16        >52             db    <STP        ; S ---- 44 ADDR STore P
0AA1: 2B        >53             db    <CLA        ; S ---v 45 ---- CLr A/R/AR/B/AB/T
0AA2: 4C        >54             db    <CLL        ; S ---- 46 ADDR CLear Location
0AA3: 33        >55             db    <OPerr      ;        47
0AA4: 57        >56             db    <SRA        ; S ---v 48 --nn Shft Rt A (AR/AS)
0AA5: 8C        >57             db    <SLA        ; S ---v 49 --nn Shft Lt A (AR/AS)
0AA6: 33 33 33 >58              db    <OPerr,<OPerr,<OPerr,<OPerr,<OPerr,<OPerr
0AAC: B0        >59             db    <MTS        ; S uhhv 50 addr Mag Tape Search
0AAD: 30        >60             db    <MTC        ; S uhhK 51 addr Mag Tape sCan
0AAE: AD        >61             db    <MRD        ; S un-v 52 addr Mag tape ReaD
0AAF: AC        >62             db    <MRR        ; S un-v 53 addr Mt Read Record
0AB0: B0        >63             db    <MIW        ; S unkk 54 addr Mt Init Write
```

```
0AB1: AF        >64             db    <MIR        ; S un-- 55 addr Mt Init wr Rec
0AB2: 35        >65             db    <MOW        ; S unkk 56 addr Mt OverWrite
0AB3: 34        >66             db    <MOR        ; S un-- 57 addr Mt Overwr Rec
0AB4: 97        >67             db    <MPF        ; S un-v 58 ---- Mt Pos Fwd
0AB5: D4        >68             db    <MIB        ; S u--v 59 addr Mt Interr Branch
```

```
               >70   noAD     equ   $8000      ; Hi bit means "ignore ADDR"
               >71   operr    equ   OPerr+noAD ; Ignore ADDR on illegal OPs.
               >72
               >73   optabh   equ   *          ; High byte of execute routines
0AB6: 8B       >74            db    >HLT+noAD  ; S ---- 00 ---- HaLT
0AB7: 8B       >75            db    >NOP+noAD  ; S ---- 01 ---- No OP
0AB8: 8A       >76            db    >operr     ;        02
0AB9: 0B       >77            db    >PRD       ; S unnv 03 ADDR Pap tape RD
0ABA: 0B       >78            db    >PRB       ; S u--v 04 ADDR Pap tape Rd, Br
0ABB: 0B       >79            db    >PRI       ; S unnv 05 ADDR Pap tape Rd, Inv
0ABC: 0B       >80            db    >PWR       ; S unn- 06 ADDR Pap tape WR
0ABD: 0C       >81            db    >PWI       ; S u--- 07 ADDR Pap tape Wr, Int
0ABE: 8A       >82            db    >KAD+noAD  ; S ---- 08 ---- Keyboard ADd
0ABF: 0C       >83            db    >SPO       ; S dnnv 09 ADDR Sup Print Out
0AC0: 8A 8A 8A >84            db    >operr,>operr,>operr,>operr,>operr,>operr
0AC6: 0C       >85            db    >CAD       ; S ---v 10 ADDR Clear ADd (Abs)
0AC7: 0C       >86            db    >CSU       ; S ---v 11 ADDR Clear SUbtr (Abs)
0AC8: 0D       >87            db    >ADD       ; S ---v 12 ADDR ADD (Abs)
0AC9: 0D       >88            db    >SUB       ; S ---v 13 ADDR SUBtract (Abs)
0ACA: 0D       >89            db    >MUL       ; S ---- 14 ADDR MULtiply
0ACB: 0E       >90            db    >DIV       ; S ---- 15 ADDR DIVide
0ACC: 8E       >91            db    >RND+noAD  ; S ---- 16 ---- RouND
0ACD: 0E       >92            db    >EXT       ; S ---- 17 ADDR EXTract
0ACE: 0F       >93            db    >CFA       ; S sLfv 18 ADDR Comp Fld A (R)
0ACF: 0D       >94            db    >ADL       ; S ---- 19 ADDR ADd to Location
0AD0: 8A 8A 8A >95            db    >operr,>operr,>operr,>operr,>operr,>operr
0AD6: 13       >96            db    >IBB       ; S nnnn 20 ADDR Increase B, Br
0AD7: 13       >97            db    >DBB       ; S nnnn 21 ADDR Decrease B, Br
0AD8: 0F       >98            db    >FAD       ; S n--v 22 ADDR Float ADd (Abs)
0AD9: 10       >99            db    >FSU       ; S n--v 23 ADDR Float SUb (Abs)
0ADA: 10       >100           db    >FMU       ; S ---- 24 ADDR Float MUltiply
0ADB: 11       >101           db    >FDV       ; S ---- 25 ADDR Float DiVide
0ADC: 12       >102           db    >IFL       ; S sLnn 26 ADDR Inc Fld Loc
0ADD: 12       >103           db    >DFL       ; S sLnn 27 ADDR Dec Fld Loc
0ADE: 12       >104           db    >DLB       ; S sLnn 28 ADDR Dec fld loc,Ld B
0ADF: 13       >105           db    >RTF       ; S -nn- 29 ADDR Record TransFer
0AE0: 8A 8A 8A >106           db    >operr,>operr,>operr,>operr,>operr,>operr
0AE6: 13       >107           db    >BUN       ; S ---- 30 ADDR Branch UNcond
0AE7: 13       >108           db    >BOF       ; S ---- 31 ADDR Branch OverFlow
0AE8: 13       >109           db    >BRP       ; S ---- 32 ADDR Branch RePeat
0AE9: 13       >110           db    >BSA       ; S ---n 33 ADDR Branch Sign A
0AEA: 13       >111           db    >BCH       ; S ---v 34 ADDR Br Comp Hi (Lo)
0AEB: 13       >112           db    >BCE       ; S ---v 35 ADDR Br Comp Eq (Un)
0AEC: 13       >113           db    >BFA       ; S sLnn 36 ADDR Branch Field A
0AED: 13       >114           db    >BFR       ; S sLnn 37 ADDR Branch Field R
0AEE: 14       >115           db    >BCS       ; S u--- 38 ADDR Br Control Sw
0AEF: 14       >116           db    >SOR       ; S ---v 39 ---- Set Ov Remember
0AF0: 8A 8A 8A >117           db    >operr,>operr,>operr,>operr,>operr,>operr
0AF6: 14       >118           db    >STA       ; S sLfv 40 ADDR STore A (R/B)
0AF7: 14       >119           db    >LDR       ; S ---- 41 ADDR LoaD R
0AF8: 14       >120           db    >LDB       ; S ---v 42 ADDR LoaD B (Comp)
0AF9: 95       >121           db    >LSA+noAD  ; S ---n 43 ---- Load Sign A
0AFA: 15       >122           db    >STP       ; S ---- 44 ADDR STore P
0AFB: 95       >123           db    >CLA+noAD  ; S ---v 45 ---- CLr A/R/AR/B/AB/T
0AFC: 15       >124           db    >CLL       ; S ---- 46 ADDR CLear Location
0AFD: 8A       >125           db    >operr     ;        47
0AFE: 95       >126           db    >SRA+noAD  ; S ---v 48 --nn Shft Rt A (AR/AS)
0AFF: 95       >127           db    >SLA+noAD  ; S ---v 49 --nn Shft Lt A (AR/AS)
0B00: 8A 8A 8A >128           db    >operr,>operr,>operr,>operr,>operr,>operr
0B06: 17       >129           db    >MTS       ; S uhhv 50 addr Mag Tape Search
0B07: 18       >130           db    >MTC       ; S uhhk 51 addr Mag Tape sCan
0B08: 18       >131           db    >MRD       ; S un-v 52 addr Mag tape ReaD
0B09: 18       >132           db    >MRR       ; S un-v 53 addr Mt Read Record
```

```
0B0A: 19       >133           db    >MIW       ; S unkk 54 addr Mt Init Write
0B0B: 19       >134           db    >MIR       ; S un-- 55 addr Mt Init wr Rec
0B0C: 1A       >135           db    >MOW       ; S unkk 56 addr Mt OverWrite
0B0D: 1A       >136           db    >MOR       ; S un-- 57 addr Mt Overwr Rec
0B0E: 9A       >137           db    >MPF+noAD  ; S un-v 58 ---- Mt Pos Fwd
0B0F: 1A       >138           db    >MIB       ; S u--v 59 addr Mt Interr Branch
```

```
               **********************************************************
               *                                                        *
               *            B220 Instruction Execute Routines           *
               *                                                        *
               * For all OPs with ADDR = memory address, Y = 0          *
               *  and A and rD+S = sign of MEM operand.                 *
               *                                                        *
               **********************************************************

               HLT      equ   *          ; Halt is executed in 'fetch'.

0B10: 4C 52 09 NOP      jmp   fetch      ; Do nothing.

0B13: A5 99    PRD      lda   rC+sL      ; Paper tape ReaD
0B15: C9 70             cmp   #$70       ; Caltech keyboard read?
0B17: D0 35             bne   :prd       ; -No, regular PRD.
0B19: A5 9A             lda   rC+VV      ; -Yes, check length and variant.
0B1B: C9 14             cmp   #$14       ;   Is it: 7014 03 aaaa?
0B1D: D0 2F             bne   :prd       ; -No, regular PRD.
0B1F: A5 E5             lda   crtkey     ; -Yes, it's a keyboard read.
0B21: C9 AF             cmp   #nokey     ; Is a keypress waiting?
0B23: F0 2F             beq   :ioerr     ; -No, I/O error!
0B25: AA                tax              ; -Yes, save key in X.
0B26: 29 F0             and   #$F0       ; F key bit (sign) & lo code bits
0B28: 10 02             bpl   :notfk     ; Function key bit off.
0B2A: 49 81             eor   #$81       ; Move F key bit to low bit.
0B2C: A0 05    :notfk   ldy   #5
0B2E: 91 D1             sta   (memptr),y ; Save 0ddd000f
0B30: 8A                txa              ; Recover key byte.
0B31: 29 08             and   #$08       ; Is Upper Case bit on?
0B33: F0 06             beq   :notuc     ; -No.
0B35: B1 D1             lda   (memptr),y ; -Yes, move it.
0B37: 09 02             ora   #$02       ;   to 2-bit of low digit.
0B39: 91 D1             sta   (memptr),y ;    and put it back.
0B3B: 8A       :notuc   txa              ; Recover key: fddduddd
0B3C: 29 07             and   #$07       ; Isolate high octal digit
0B3E: 88                dey              ;  of crtkey code
0B3F: 91 D1             sta   (memptr),y ;   and store it.
0B41: A9 00             lda   #0
0B43: 88       :clrlp   dey              ; Clear the high digits
0B44: 91 D1             sta   (memptr),y ;  of the B220 word.
0B46: D0 FB             bne   :clrlp
0B48: A9 AF             lda   #nokey     ; Clear the keypress.
0B4A: 85 E5             sta   crtkey
0B4C: D0 03             bne   :fetch     ; (always)

0B4E: 20 6B 0B :prd     jsr   ]prd       ; Paper tape ReaD
0B51: 4C 52 09 :fetch   jmp   fetch

0B54: 4C 43 0A :ioerr   jmp   IOerr

0B57: A5 99    PRB      lda   rC+sL      ; Paper tape Read & Branch
0B59: 29 F0             and   #$F0       ; Fake NN = 00 (100 words)
0B5B: 85 99             sta   rC+sL
0B5D: A5 9A             lda   rC+VV
0B5F: 29 0F             and   #$0F
0B61: 09 01             ora   #$01       ;  and xeq sign 6/7.
0B63: 85 9A             sta   rC+VV
0B65: 20 6B 0B :read    jsr   ]prd       ; Read "tape" until
0B68: 4C 65 0B          jmp   :read      ;  sign 6/7 terminates.

               Bmodflg  equ   linev      ; B-modification flag
               xeqflg   equ   linev+1    ; Sign 6/7 execute flag

0B6B: 20 75 16 ]prd     jsr   midNN      ; Get word count (1..100)
0B6E: 85 D8             sta   NN         ;  in binary.
0B70: A5 9A             lda   rC+VV      ; Examine variant digit
```

```
0B72: 29 08    >207           and   #$08       ; 8-bit on?
0B74: 85 DB    >208           sta   Bmodflg    ; Set B-modify mask.
0B76: A5 9A    >209           lda   rC+VV      ; Variant again...
0B78: 29 01    >210           and   #$01       ; Execute 6/7 sign?
0B7A: F0 02    >211           beq   :noxeq     ; -No, ignore 6/7 sign.
0B7C: A9 06    >212           lda   #6         ; -Yes, set xeq mask.
0B7E: 85 DC    >213  :noxeq   sta   xeqflg
0B80: A2 00    >214           ldx   #PTRclass  ; PTRDR device class
0B82: 20 61 08 >215           jsr   M_iosel    ; Select device.
0B85: 20 79 08 >216  :readlp  jsr   M_getwrd   ; Next word to rD.
0B88: A5 AA    >217           lda   rD+S       ; Sign digit 8/9?
0B8A: 25 DB    >218           and   Bmodflg    ; Variant 8-bit
0B8C: F0 05    >219           beq   :noBmod    ; -No B modification.
0B8E: 20 B9 0B >220           jsr   BmodrD     ; -B-modify address
0B91: 10 08    >221           bpl   :store     ; (always)
               >222
0B93: A5 AA    >223  :noBmod  lda   rD+S       ; Re-fetch sign digit
0B95: 25 DC    >224           and   xeqflg     ; Apply xeq mask (0/6)
0B97: C9 06    >225           cmp   #6         ; Sign = 6 or 7?
0B99: F0 0B    >226           beq   :xeq       ; -Yes, execute it.
0B9B: 20 CF 0B >227  :store   jsr   storerD    ; -No, store rD & adv memptr.
0B9E: C6 D8    >228           dec   NN         ; More words?
0BA0: D0 E3    >229           bne   :readlp    ; -Yes, continue scan.
0BA2: 20 6D 08 >230           jsr   M_iodsel   ; -No, deselect device
0BA5: 60       >231           rts              ;       and return.
               >232
0BA6: A2 05    >233  :xeq     ldx   #5         ; Execute input word.
0BA8: B5 AA    >234  :xeqlp   lda   rD,x       ; Copy rD to rC.
0BAA: 95 98    >235           sta   rC,x
0BAC: CA       >236           dex
0BAD: 10 F9    >237           bpl   :xeqlp
0BAF: 86 C6    >238           stx   skipincP   ; Don't inc P reg.
0BB1: 20 6D 08 >239           jsr   M_iodsel   ; Deselect device.
0BB4: 68       >240           pla              ; No return.
0BB5: 68       >241           pla
0BB6: 4C 79 09 >242           jmp   execute    ; Execute instruction.
               >243
0BB9: F8       >244  BmodrD   sed              ; / Decimal mode.
0BBA: 18       >245           clc
0BBB: A5 AF    >246           lda   rD+ADDR+1  ; Add rB to rD ADDR.
0BBD: 65 95    >247           adc   rB+1
0BBF: 85 AF    >248           sta   rD+ADDR+1
0BC1: A5 AE    >249           lda   rD+ADDR
0BC3: 65 94    >250           adc   rB
0BC5: 85 AE    >251           sta   rD+ADDR
0BC7: D8       >252           cld              ; \ Binary mode.
0BC8: A5 AA    >253           lda   rD+S       ; Turn off
0BCA: 29 01    >254           and   #$01       ;  8-bit of sign.
0BCC: 85 AA    >255           sta   rD+S       ; (return w/ >=)
0BCE: 60       >256           rts
               >257
0BCF: A0 05    >258  storerD  ldy   #5         ; Store rD
0BD1: B9 AA 00 >259  :stlp    lda   rD,y
0BD4: 91 D1    >260           sta   (memptr),y
0BD6: 88       >261           dey
0BD7: 10 F8    >262           bpl   :stlp
0BD9: 18       >263  incmem   clc              ; Advance memptr
0BDA: A5 D1    >264           lda   memptr     ;  to next word.
0BDC: 69 06    >265           adc   #6
0BDE: 85 D1    >266           sta   memptr
0BE0: 90 02    >267           bcc   :nocarry
0BE2: E6 D2    >268           inc   memptr+1   ; Propagate carry.
0BE4: 60       >269  :nocarry rts
               >270
0BE5: 4C 33 0A >271  PRI      jmp   OPerr      ; Unimplemented
```

```
0BE8: 20 75 16 >273  PWR      jsr   midNN      ; Get word count
0BEB: 85 D8    >274           sta   NN         ;  in binary.
0BED: A2 02    >275           ldx   #PTPclass  ; PTPCH device class.
0BEF: 20 61 08 >276           jsr   M_iosel    ; Select device.
0BF2: 20 CD 08 >277           jsr   M_ckspo    ; PWR rerouted to SPO?
0BF5: D0 0C    >278           bne   :wrdlp     ; -No, do PWR.
0BF7: 20 6D 08 >279           jsr   M_iodsel   ; -Yes, deselect punch,
0BFA: A5 9A    >280           lda   rC+VV      ;   force 0 SPO variant,
0BFC: 29 F0    >281           and   #$F0
0BFE: 85 9A    >282           sta   rC+VV
0C00: 4C 32 0C >283           jmp   SPO        ;    and execute SPO.
               >284
0C03: 20 22 0C >285  :wrdlp   jsr   loadrD     ; (memptr) word --> rD
0C06: 20 85 08 >286           jsr   M_putwrd   ; Put rD in buffer.
0C09: C6 D8    >287           dec   NN         ; More words?
0C0B: D0 F6    >288           bne   :wrdlp     ; -Yes, go again.
0C0D: A9 EF    >289           lda   #EOF       ; -No, set EOF flag.
0C0F: A0 00    >290           ldy   #0
0C11: 8D 04 C0 >291           sta   WRITMAIN
0C14: 91 D3    >292           sta   (ptr),y
0C16: 8D 05 C0 >293           sta   WRITAUX
0C19: 20 6D 08 >294           jsr   M_iodsel   ; Deselect device.
0C1C: 4C 52 09 >295           jmp   fetch
               >296
0C1F: 4C 43 0A >297  :ioerr   jmp   IOerr      ; Relay jump.
               >298
0C22: A0 05    >299  loadrD   ldy   #5         ; Load (memptr) into rD.
0C24: B1 D1    >300  :ldlp    lda   (memptr),y
0C26: 99 AA 00 >301           sta   rD,y
0C29: 88       >302           dey
0C2A: 10 F8    >303           bpl   :ldlp
0C2C: 4C D9 0B >304           jmp   incmem     ; Adv to next word & return.
               >305
0C2F: 4C 33 0A >306  PWI      jmp   OPerr      ; Unimplemented
               >307
               >308  KAD      equ   ]stop      ; Kluge to allow rA mod.
```

```
0C32: 20 75 16  SPO      jsr   midNN      ; Get count (NN) in A
0C35: 85 D8              sta   NN         ; NN = binary word count.
0C37: A0 00     :nxword  ldy   #0
0C39: B1 D1              lda   (memptr),y ; Get sign
0C3B: C9 02              cmp   #2         ; Alphanumeric?
0C3D: D0 35              bne   :num       ; -No, numeric.
0C3F: C8        :nxchar  iny              ; -Yes, print alpha.
0C40: B1 D1              lda   (memptr),y ; Get next char
0C42: C9 26              cmp   #$26       ; "Tab" code?
0C44: F0 11              beq   :tab       ; -Yes, do tab.
0C46: C9 02              cmp   #$02       ; -No, "Ignore" code?
0C48: F0 07              beq   :ignore    ; -Yes, skip it.
0C4A: AA                 tax              ; -No, translate B220
0C4B: BD 26 17           lda   b220asc,x  ;       char to ASCII.
0C4E: 20 09 09           jsr   M_COUT     ;        and print it.
0C51: C0 05     :ignore  cpy   #5         ; Word complete?
0C53: D0 EA              bne   :nxchar    ; -No, keep going.
0C55: F0 49              beq   :done      ; -Yes, word done (always)

0C57: A2 00     :tab     ldx   #0
0C59: A5 24              lda   CH
0C5B: DD AC 0C  :nxtab   cmp   tabs,x     ; Find first tab
0C5E: 90 07              bcc   :gottab    ;  greater than CH.
0C60: E8                 inx
0C61: E0 05              cpx   #5
0C63: D0 F6              bne   :nxtab
0C65: F0 EA              beq   :ignore    ; (always) Skip if past tabs.

0C67: 38        :gottab  sec              ; Compute tab - CH.
0C68: BD AC 0C           lda   tabs,x
0C6B: E5 24              sbc   CH
0C6D: AA                 tax              ; X = tab - CH
0C6E: 20 15 09           jsr   M_PRBL2    ; Print X blanks
0C71: 4C 51 0C           jmp   :ignore    ;  and continue...

0C74: A2 A0     :num     ldx   #"          " ; Print blank if sign 0
0C76: C9 00              cmp   #0
0C78: F0 09              beq   :prtsign
0C7A: A2 AD              ldx   #"-"       ; Print - if sign 1
0C7C: C9 01              cmp   #1
0C7E: F0 03              beq   :prtsign
0C80: 09 B0              ora   #"0"       ; Else print sign digit.
0C82: AA                 tax
0C83: 8A        :prtsign txa
0C84: 20 09 09           jsr   M_COUT
0C87: C8        :nxbyte  iny              ; Print rest of number.
0C88: B1 D1              lda   (memptr),y
0C8A: 48                 pha
0C8B: 4A                 lsr
0C8C: 4A                 lsr
0C8D: 4A                 lsr
0C8E: 4A                 lsr              ; Hi digit in A
0C8F: 09 B0              ora   #"0"       ; OR in zone
0C91: 20 09 09           jsr   M_COUT     ;  and print digit.
0C94: 68                 pla              ; Recover low digit
0C95: 29 0F              and   #$0F       ; Isolate it
0C97: 09 B0              ora   #"0"       ;  add zone
0C99: 20 09 09           jsr   M_COUT     ;   and print it.
0C9C: C0 05              cpy   #5         ; End of word?
0C9E: D0 E7              bne   :nxbyte    ; -No, continue.
0CA0: C6 D8     :done    dec   NN         ; -Yes, more words?
0CA2: F0 05              beq   :quit      ; -No, all done.
0CA4: 20 D9 0B           jsr   incmem     ; -Yes, increment memptr.
0CA7: D0 8E              bne   :nxword    ; (always)

0CA9: 4C 52 09  :quit    jmp   fetch

```

```
0CAC: 09 11 19  tabs     db    9,17,25,33,41 ; SPO tab table

0CB1: A5 9A     CSU      lda   rC+VV      ; CSU/CSA
0CB3: 29 0F              and   #$0F       ; Isolate variant digit.
0CB5: C9 01              cmp   #$01       ; CSA?
0CB7: D0 06              bne   :csu       ; -No, CSU.
0CB9: A5 AA              lda   rD+S       ; -Yes, CSA.
0CBB: 09 01              ora   #$01       ;   Force sign negative.
0CBD: D0 17              bne   ]loadrA    ; (always)

0CBF: A5 AA     :csu     lda   rD+S       ; CSU
0CC1: 49 01              eor   #$01       ; Flip the 1-bit
0CC3: 4C D6 0C           jmp   ]loadrA    ;  and complete the load.


0CC6: A5 9A     CAD      lda   rC+VV      ; CAD/CAA
0CC8: 29 0F              and   #$0F       ; Isolate variant digit.
0CCA: C9 01              cmp   #$01       ; CAA?
0CCC: F0 47              beq   CAA        ; -Yes.
0CCE: A5 9A              lda   rC+VV      ; -No.  Is it Caltech's
0CD0: 29 10              and   #$10       ;   DISplay instruction?
0CD2: D0 11              bne   :DIS       ; -Yes!
0CD4: A5 AA              lda   rD+S       ; -No, CAD.  Sign unchanged.
0CD6: 85 9E     ]loadrA  sta   rA+S       ; Set rA sign.
0CD8: A0 05              ldy   #5
0CDA: B1 D1     :cpyloop lda   (memptr),y
0CDC: 99 9E 00           sta   rA,y
0CDF: 88                 dey
0CE0: D0 F8              bne   :cpyloop
0CE2: 4C 52 09           jmp   fetch

0CE5: A5 CB     :DIS     lda   viewmode   ; In VIEW mode
0CE7: 25 CC              and   lpen       ;  and light pen on?
0CE9: F0 09              beq   :DISloop   ; -No, just display.
0CEB: 20 FD 08           jsr   M_xdrawc   ; -Yes, erase cursor,
0CEE: 20 F1 08           jsr   M_lpread   ;   read new position,
0CF1: 20 FD 08           jsr   M_xdrawc   ;    and redraw cursor.
0CF4: A0 05     :DISloop ldy   #5         ; Copy operand to rA.
0CF6: B1 D1     :lp      lda   (memptr),y
0CF8: 99 9E 00           sta   rA,y
0CFB: 88                 dey
0CFC: 10 F8              bpl   :lp
0CFE: A5 9F     :display lda   rA+1       ; Is "display continue"
0D00: 29 01              and   #$01       ;  bit on?
0D02: F0 0E              beq   :fetch     ; -No, DISplay ended.
0D04: 20 E5 08           jsr   M_plot     ; -Yes, plot point in rA.
0D07: B0 06              bcs   :penbutn   ; Handle light pen or button.
0D09: 20 D9 0B           jsr   incmem     ; Advance to next point
0D0C: 4C F4 0C           jmp   :DISloop   ;  and continue plotting.

0D0F: 20 17 0A  :penbutn jsr   incP       ; Skip next instruction
0D12: 4C 52 09  :fetch   jmp   fetch      ;  if pen or button sensed.

0D15: A5 AA     CAA      lda   rD+S       ; CAA
0D17: 29 FE              and   #$FE       ; Force sign positive
0D19: 4C D6 0C           jmp   ]loadrA    ;  and complete the load.
```

```
0D1C: A5 9A    >434  ADD      lda   rC+VV      ; ADD, ADA
0D1E: 29 0F    >435           and   #$0F
0D20: C9 01    >436           cmp   #1         ; ADA?
0D22: D0 04    >437           bne   :add       ; -No, ADD.
0D24: A9 00    >438           lda   #0         ; -Yes, force MEM sign +
0D26: 85 AA    >439           sta   rD+S
0D28: 20 2E 0D >440  :add     jsr   ]add       ; Do the add.
0D2B: 4C 52 09 >441           jmp   fetch
               >442
0D2E: A5 9E    >443  ]add     lda   rA+S
0D30: 29 01    >444           and   #$01
0D32: 85 9E    >445           sta   rA+S       ; Force sign 0 (+) or 1 (-)
0D34: 45 AA    >446           eor   rD+S       ; Signs same or different?
0D36: 29 01    >447           and   #$01
0D38: D0 18    >448           bne   :subtr     ; -Different, subtract.
0D3A: A0 05    >449           ldy   #5         ; -Same, add.
0D3C: F8       >450           sed              ; / Decimal mode.
0D3D: 18       >451           clc
0D3E: B9 9E 00 >452  :addloop lda   rA,y       ; Do the addition...
0D41: 71 D1    >453           adc   (memptr),y
0D43: 99 9E 00 >454           sta   rA,y
0D46: 88       >455           dey
0D47: D0 F5    >456           bne   :addloop
0D49: D8       >457           cld              ; \ Back to binary.
0D4A: 90 3F    >458           bcc   :done      ; Done.
               >459           seti  Ov         ; Signal Overflow
0D4C: A9 FF    >459           lda   #$FF
0D4E: 85 C3    >459           sta   Ov         ; Set non-zero.
               >459           eom
0D50: D0 39    >460           bne   :done      ; (always)
               >461
0D52: A0 01    >462  :subtr   ldy   #1         ; Compare magnitudes.
0D54: B9 9E 00 >463  :comloop lda   rA,y
0D57: D1 D1    >464           cmp   (memptr),y
0D59: F0 04    >465           beq   :cont      ; Equal, keep comparing.
0D5B: B0 07    >466           bcs   :Abig      ; rA is bigger
0D5D: 90 16    >467           bcc   :Asmall    ; rA is smaller
               >468
0D5F: C8       >469  :cont    iny
0D60: C0 06    >470           cpy   #6
0D62: D0 F0    >471           bne   :comloop   ; If =, fall into :Abig.
0D64: A0 05    >472  :Abig    ldy   #5         ; Subtract MEM from rA.
0D66: F8       >473           sed              ; / Decimal mode.
0D67: B9 9E 00 >474  :subloop lda   rA,y
0D6A: F1 D1    >475           sbc   (memptr),y
0D6C: 99 9E 00 >476           sta   rA,y
0D6F: 88       >477           dey
0D70: D0 F5    >478           bne   :subloop
0D72: D8       >479           cld              ; \ Back to binary.
0D73: F0 16    >480           beq   :done      ; (always)
               >481
0D75: A5 AA    >482  :Asmall  lda   rD+S       ; MEM - rA ==> rA
0D77: 29 01    >483           and   #$01       ; rA sign = MEM sign.
0D79: 85 9E    >484           sta   rA+S
0D7B: A0 05    >485           ldy   #5
0D7D: F8       >486           sed              ; / Decimal mode.
0D7E: 38       >487           sec
0D7F: B1 D1    >488  :sloop   lda   (memptr),y
0D81: F9 9E 00 >489           sbc   rA,y
0D84: 99 9E 00 >490           sta   rA,y
0D87: 88       >491           dey
0D88: D0 F5    >492           bne   :sloop
0D8A: D8       >493           cld              ; \ Back to binary.
0D8B: 60       >494  :done    rts
```

```
0D8C: A5 9E    >496  ADL      lda   rA+S        ; Force rA sign
0D8E: 29 01    >497           and   #$01        ;  to 0 or 1.
0D90: 85 9E    >498           sta   rA+S
0D92: A2 FA    >499           ldx   #-6         ; MEM + rA ==> MEM
0D94: B5 A4    >500  :pushlp  lda   rA+6,x      ; Push rA
0D96: 48       >501           pha
0D97: E8       >502           inx
0D98: D0 FA    >503           bne   :pushlp
0D9A: 20 2E 0D >504           jsr   ]add        ; rA + MEM ==> rA
0D9D: A0 05    >505           ldy   #5          ; rA ==> MEM
0D9F: B9 9E 00 >506  :mvloop  lda   rA,y
0DA2: 91 D1    >507           sta   (memptr),y
0DA4: 68       >508           pla               ;  and pop rA.
0DA5: 99 9E 00 >509           sta   rA,y
0DA8: 88       >510           dey
0DA9: 10 F4    >511           bpl   :mvloop
0DAB: 4C 52 09 >512           jmp   fetch
               >513
0DAE: A5 9A    >514  SUB      lda   rC+VV       ; SUB, SUA
0DB0: 29 0F    >515           and   #$0F
0DB2: C9 01    >516           cmp   #1          ; SUA?
0DB4: F0 06    >517           beq   :setsign    ; -Yes, force operand neg.
0DB6: A5 AA    >518  :sub     lda   rD+S        ; -No, SUB.
0DB8: 29 01    >519           and   #$01        ; Invert
0DBA: 49 01    >520           eor   #$01        ;  operand
0DBC: 85 AA    >521  :setsign sta   rD+S        ;   sign
0DBE: 20 2E 0D >522           jsr   ]add        ;    and add.
0DC1: 4C 52 09 >523           jmp   fetch
```

```
0DC4: 20 CA 0D  MUL      jsr   multiply   ; Multiply
0DC7: 4C 52 09           jmp   fetch

0DCA: 45 9E     multiply eor   rA+S       ; Multiply subroutine
0DCC: 29 01              and   #$01
0DCE: 48                 pha              ; Save result sign
0DCF: A2 00              ldx   #0
0DD1: A0 05              ldy   #5
0DD3: B1 D1     :init    lda   (memptr),y ; rD = multiplicand
0DD5: 99 AA 00           sta   rD,y
0DD8: 99 B0 00           sta   rD10,y     ; rD10 = multiplicand
0DDB: B9 9E 00           lda   rA,y       ; rR = multiplier
0DDE: 99 A4 00           sta   rR,y
0DE1: 96 9E              stx   rA,y       ; rA = 0 (including sign)
0DE3: 88                 dey
0DE4: 10 ED              bpl   :init
0DE6: A5 C3              lda   Ov         ; FMU overflow pending?
0DE8: C9 80              cmp   #$80
0DEA: D0 02              bne   :cont      ; -No, continue.
0DEC: 68                 pla              ; -Yes, discard result sign
0DED: 60                 rts              ;   and return.

0DEE: 86 AA     :cont    stx   rD+S       ; Clear rD sign
0DF0: 86 B0              stx   rD10+S     ;  and rD10 sign.
0DF2: A0 04              ldy   #4         ; 4 bits/digit.
0DF4: 18        :shloop  clc              ; Shift in zeros.
0DF5: 26 B5              rol   rD10+5     ; Multiply rD10 by 10.
0DF7: 26 B4              rol   rD10+4
0DF9: 26 B3              rol   rD10+3
0DFB: 26 B2              rol   rD10+2
0DFD: 26 B1              rol   rD10+1
0DFF: 26 B0              rol   rD10
0E01: 88                 dey
0E02: D0 F0              bne   :shloop
0E04: A9 05              lda   #5         ; Set multiplier byte
0E06: 85 D7              sta   t1         ;  count = 5.
0E08: F8                 sed              ; / Decimal mode.
0E09: A5 A9     :ckadd1  lda   rR+5
0E0B: 29 0F              and   #$0F       ; Low digit of multiplier
0E0D: F0 10              beq   :ckadd10   ; Skip add1 if zero.
0E0F: A8                 tay              ; Y = add1 count.
0E10: A2 05     :add1    ldx   #5
0E12: 18                 clc              ; rA = rA + rD
0E13: B5 9E     :add1lp  lda   rA,x
0E15: 75 AA              adc   rD,x
0E17: 95 9E              sta   rA,x
0E19: CA                 dex
0E1A: 10 F7              bpl   :add1lp
0E1C: 88                 dey              ; More adds?
0E1D: D0 F1              bne   :add1      ; -Yes.
0E1F: A5 A9     :ckadd10 lda   rR+5       ; Low multiplier byte
0E21: 29 F0              and   #$F0       ; High digit of byte
0E23: F0 14              beq   :shift     ; Skip add10 if zero.
0E25: 4A                 lsr
0E26: 4A                 lsr
0E27: 4A                 lsr
0E28: 4A                 lsr
0E29: A8                 tay              ; Y = add10 count.
0E2A: A2 05     :add10   ldx   #5
0E2C: 18                 clc              ; rA = rA + rD10
0E2D: B5 9E     :add10lp lda   rA,x
0E2F: 75 B0              adc   rD10,x
0E31: 95 9E              sta   rA,x
0E33: CA                 dex
0E34: 10 F7              bpl   :add10lp
0E36: 88                 dey              ; More adds?
0E37: D0 F1              bne   :add10     ; -Yes.
```

```
0E39: 20 1D 16 >592  :shift   jsr   srT2       ; -No, shift |rA| & |rR|
0E3C: A5 9E    >593           lda   rA+S       ;  right 2 digits
0E3E: 85 9F    >594           sta   rA+1       ;   including rA sign.
0E40: 86 9E    >595           stx   rA+S       ; Clear rA sign.
0E42: C6 D7    >596           dec   t1         ; Keep going if more
0E44: D0 C3    >597           bne   :ckadd1    ;  multiplier digits.
0E46: D8       >598           cld              ; \ Back to binary.
0E47: 68       >599           pla              ; Recover product sign
0E48: 85 9E    >600           sta   rA+S       ;  and set rA & rR signs.
0E4A: 85 A4    >601           sta   rR+S
0E4C: 60       >602           rts
```

```
0E4D: 20 53 0E >604  DIV      jsr   divide     ; DIVide
0E50: 4C 52 09 >605           jmp   fetch
               >606
0E53: 45 9E    >607  divide   eor   rA+S
0E55: 29 01    >608           and   #$01
0E57: 48       >609           pha              ; Sign of quotient
0E58: A5 9E    >610           lda   rA+S
0E5A: 85 A4    >611           sta   rR+S       ; Sign of remainder
0E5C: C8       >612           iny              ; Y = 1: skip signs.
0E5D: B9 9E 00 >613  :comp    lda   rA,y       ; Compare rA magnitude
0E60: D1 D1    >614           cmp   (memptr),y ;  with divisor magnitude.
0E62: 90 0D    >615           bcc   :divide    ; rA < MEM, so divide.
0E64: D0 05    >616           bne   :oflow     ; rA > MEM, overflow.
0E66: C8       >617           iny
0E67: C0 06    >618           cpy   #6
0E69: D0 F2    >619           bne   :comp
               >620  :oflow   seti  Ov         ; Signal overflow
0E6B: A9 FF    >620           lda   #$FF
0E6D: 85 C3    >620           sta   Ov         ; Set non-zero.
               >620           eom
0E6F: 68       >621           pla              ; Drop result sign
0E70: 60       >622           rts              ;  and return.
               >623
0E71: A0 0A    >624  :divide  ldy   #10        ; Quotient digit count = 10.
0E73: 84 D7    >625           sty   t1
0E75: A0 05    >626           ldy   #5
0E77: B1 D1    >627  :div2rD  lda   (memptr),y ; Move divisor to rD
0E79: 99 AA 00 >628           sta   rD,y
0E7C: 88       >629           dey
0E7D: D0 F8    >630           bne   :div2rD
0E7F: 84 9E    >631           sty   rA+S       ; Clear sign of rA
0E81: 84 AA    >632           sty   rD+S       ;  and rD.
0E83: F8       >633           sed              ; / Decimal mode.
0E84: A0 04    >634  :shift   ldy   #4         ; 4 bits/digit.
0E86: 18       >635  :shiftlp clc              ; Shift AR left 1 digit
0E87: 20 31 16 >636           jsr   slT        ;  shifting in zeros.
0E8A: 26 9E    >637           rol   rA+S       ;   (include sign in A)
0E8C: 88       >638           dey
0E8D: D0 F7    >639           bne   :shiftlp
0E8F: A2 00    >640           ldx   #0
0E91: B5 9E    >641  :complp  lda   rA,x       ; Compare A with divisor
0E93: D5 AA    >642           cmp   rD,x
0E95: 90 25    >643           bcc   :zero      ; Speed up quotient zeros.
0E97: D0 05    >644           bne   :sub       ; A > divisor
0E99: E8       >645           inx
0E9A: E0 06    >646           cpx   #6
0E9C: D0 F3    >647           bne   :complp
0E9E: A2 05    >648  :sub     ldx   #5         ; A(ext) = A(ext) - D(ext).
0EA0: 38       >649           sec
0EA1: B5 9E    >650  :sublp   lda   rA,x
0EA3: F5 AA    >651           sbc   rD,x
0EA5: 95 9E    >652           sta   rA,x
0EA7: CA       >653           dex
0EA8: 10 F7    >654           bpl   :sublp
0EAA: 90 04    >655           bcc   :restore   ; Restore if underflow
0EAC: E6 A9    >656           inc   rR+5       ; Increment quotient digit.
0EAE: D0 EE    >657           bne   :sub       ; (always)
               >658
0EB0: A2 05    >659  :restore ldx   #5         ; Add divisor back to A.
0EB2: 18       >660           clc
0EB3: B5 9E    >661  :restlp  lda   rA,x
0EB5: 75 AA    >662           adc   rD,x
0EB7: 95 9E    >663           sta   rA,x
0EB9: CA       >664           dex
0EBA: 10 F7    >665           bpl   :restlp
0EBC: C6 D7    >666  :zero    dec   t1         ; Quotient complete?
0EBE: D0 C4    >667           bne   :shift     ; -No, keep dividing.
```

```
0EC0: 20 46 16 >668           jsr   exchAR     ; -Yes, exchange A and R
0EC3: D8       >669           cld              ; \ Back to binary.
0EC4: 68       >670           pla
0EC5: 85 9E    >671           sta   rA+S       ; Set quotient sign.
0EC7: 60       >672           rts
               >673
0EC8: A5 A5    >674  RND      lda   rR+1       ; Hi digit of rR
0ECA: C9 50    >675           cmp   #$50       ; C=1 if hi digit >= 5.
0ECC: A2 A4    >676           ldx   #rR        ; Clear rR.
0ECE: 20 68 16 >677           jsr   clear      ; (Doesn't disturb C)
0ED1: 90 14    >678           bcc   :done      ; Done if hi digit < 5.
0ED3: F8       >679           sed              ; / Decimal mode.
0ED4: 38       >680           sec              ; Add 1 to rA.
0ED5: A2 05    >681           ldx   #5
0ED7: B5 9E    >682  :rndloop lda   rA,x
0ED9: 69 00    >683           adc   #0
0EDB: 95 9E    >684           sta   rA,x
0EDD: CA       >685           dex
0EDE: D0 F7    >686           bne   :rndloop
0EE0: D8       >687           cld              ; \ Back to binary.
0EE1: 90 04    >688           bcc   :done
               >689           seti  Ov         ; Signal Overflow.
0EE3: A9 FF    >689           lda   #$FF
0EE5: 85 C3    >689           sta   Ov         ; Set non-zero.
               >689           eom
0EE7: 4C 52 09 >690  :done    jmp   fetch
               >691
0EEA: A0 05    >692  EXT      ldy   #5         ; Extract digits from rA
0EEC: B1 D1    >693  :extlp   lda   (memptr),y ;  where MEM digits are odd.
0EEE: 29 11    >694           and   #$11       ; Isolate odd bits
0EF0: AA       >695           tax              ; $00, $01, $10, $11.
0EF1: BD 00 0F >696           lda   :exttbl,x  ; $00, $0F, $F0, $FF.
0EF4: 39 9E 00 >697           and   rA,y       ; Mask rA digits
0EF7: 99 9E 00 >698           sta   rA,y
0EFA: 88       >699           dey
0EFB: 10 EF    >700           bpl   :extlp
0EFD: 4C 52 09 >701           jmp   fetch
               >702
0F00: 00 0F    >703  :exttbl  db    $00,$0F    ; Indices $00, $01 used
0F02: 03 02 01 >704  signtbl  db    3,2,1,0,7,6,5,4,8,9 ; CFx sign order
0F0C: 00 00 00 >705           db    0,0,0,0    ; (filler)
0F10: F0 FF    >706           db    $F0,$FF    ; Indices $10, $11 used.
               >707
0F12: A5 9A    >708  CFA      lda   rC+VV      ; CFA, CFR
0F14: A2 A4    >709           ldx   #rR
0F16: 29 01    >710           and   #$01       ; CFR?
0F18: D0 02    >711           bne   :cfr       ; -Yes.
0F1A: A2 9E    >712           ldx   #rA        ; No, CFA.
0F1C: A5 9A    >713  :cfr     lda   rC+VV      ; Reload variant
0F1E: 29 10    >714           and   #$10       ; Partial field bit
0F20: A8       >715           tay              ;  to Y.
0F21: A9 D0    >716           lda   #BNEop     ; Do signed compare.
0F23: 20 32 0F >717           jsr   compare
0F26: 85 C2    >718           sta   COMP       ; Set COMPare indicator
0F28: A5 C1    >719           lda   ERR        ; Error detected?
0F2A: D0 03    >720           bne   :err       ; -Yes, report it.
0F2C: 4C 52 09 >721           jmp   fetch
               >722
0F2F: 4C 4B 0A >723  :err     jmp   ]err
```

```
                >725  **********************************************************
                >726  *                                                        *
                >727  * Compare register with (memptr), whole or partial field.*
                >728  *                                                        *
                >729  * Entry: X = Register addr, (memptr) = comparand addr    *
                >730  *        Y = Whole (0) or partial (not 0)                *
                >731  *        A = BNE (signed comp) or BCS (unsigned comp)    *
                >732  *                                                        *
                >733  *  Exit: A = COMP indicator state (<0, 0, >0)            *
                >734  *                                                        *
                >735  **********************************************************
                >736
0F32: 8D 5C 0F >737  compare  sta   :magonly   ; Signed/unsigned (BNE, BCS)
0F35: B5 00    >738           lda   0,x        ; Save register sign
0F37: 8D 5F 0F >739           sta   :cmpsign+1 ;  for compare.
0F3A: 8E 8E 0F >740           stx   :comp1+1   ; And save register
0F3D: 8E B9 0F >741           stx   :comp2+1   ;  address for loads.
0F40: 8E C4 0F >742           stx   :byte+1
0F43: 84 D8    >743           sty   NN         ; Save whole/partial.
0F45: C0 00    >744           cpy   #0         ; Whole/partial (0, not 0)
0F47: D0 06    >745           bne   :partial   ; -Yes.
0F49: A9 00    >746           lda   #0         ; -No, fake 0:0 field
0F4B: A2 0B    >747           ldx   #11        ;   and compare signs.
0F4D: D0 0F    >748           bne   :cmpsign   ; (always)
               >749
0F4F: 20 54 16 >750  :partial jsr   splitsL    ; Split sL: A = s and X = L.
0F52: 18       >751           clc              ; A = low digit, 1..10
0F53: 69 01    >752           adc   #1         ;     low dig + 1, 2..11
0F55: 38       >753           sec
0F56: 86 D7    >754           stx   t1         ; Digit length
0F58: E5 D7    >755           sbc   t1         ; A = hi digit #
0F5A: 90 18    >756           bcc   :flderr    ; <0 ==> Field error.
0F5C: D0 1F    >757  :magonly bne   :comp      ; >0 ==> Comp magnitudes.
0F5E: A0 00    >758  :cmpsign ldy   #0*0       ; =0 ==> Compare signs.
0F60: C4 AA    >759           cpy   rD+S       ; Reg sign = MEM sign?
0F62: F0 15    >760           beq   :nosign    ; -Yes, comp magnitudes.
0F64: B9 02 0F >761           lda   signtbl,y  ; -No, translate reg sign
0F67: A4 AA    >762           ldy   rD+S       ; MEM sign
0F69: BE 02 0F >763           ldx   signtbl,y  ;  translated.
0F6C: 86 D7    >764           stx   t1
0F6E: C5 D7    >765           cmp   t1         ; Compare signs.
0F70: E6 D8    >766           inc   NN         ; Force no flip.
0F72: D0 26    >767           bne   :neql      ; (always) Sign determines.
               >768
0F74: A5 C6    >769  :flderr  lda   "F"        ; Signal Field error.
0F76: 85 C1    >770           sta   ERR
0F78: 60       >771           rts
               >772
0F79: 18       >773  :nosign  clc              ; Exclude sign from field
0F7A: 69 01    >774           adc   #1         ; Field start + 1
0F7C: CA       >775           dex              ; Field length - 1
0F7D: 18       >776  :comp    clc
0F7E: 69 01    >777           adc   #1
0F80: 4A       >778           lsr              ; A = hi byte for compare
0F81: A8       >779           tay              ; Y = hi byte index
0F82: B0 2E    >780           bcs   :lodigit   ; C ==> lo digit of hi byte.
0F84: CA       >781  :hidigit dex              ; Next digit, too?
0F85: D0 3C    >782           bne   :byte      ; -Yes, comp whole byte.
0F87: B1 D1    >783           lda   (memptr),y ; MEM byte
0F89: 29 F0    >784           and   #$F0       ; -No, final digit.
0F8B: 85 D7    >785           sta   t1
0F8D: B9 00 00 >786  :comp1   lda   0*0,y      ; Reg byte
0F90: 29 F0    >787           and   #$F0       ; Hi digit
0F92: C5 D7    >788  :final   cmp   t1         ; Compare final digit.
0F94: D0 04    >789  :done    bne   :neql      ; =?
0F96: A9 00    >790           lda   #0         ; -Yes, A = 0.
0F98: F0 06    >791           beq   :fin       ; (always)
```

```
                  >792
0F9A: A9 01       >793  :neql    lda   #1
0F9C: B0 02       >794           bcs   :fin        ; >
0F9E: A9 FF       >795           lda   #-1         ; <
0FA0: A4 D8       >796  :fin     ldy   NN          ; Recover whole/partial
0FA2: D0 0D       >797           bne   :noflip     ; Partial ==> no flip
0FA4: A6 AA       >798           ldx   rD+S        ; Original sign
0FA6: F0 09       >799           beq   :noflip     ; + if 0.
0FA8: E0 04       >800           cpx   #4          ; Collate as + or -?
0FAA: B0 05       >801           bcs   :noflip     ; + if >= 4.
0FAC: AA          >802           tax               ; - if 1, 2, or 3.
0FAD: F0 02       >803           beq   :noflip     ; Comp =, no flip.
0FAF: 49 80       >804           eor   #$80        ; Exchange > and <.
0FB1: 60          >805  :noflip  rts
                  >806
0FB2: B1 D1       >807  :lodigit lda   (memptr),y  ; MEM byte
0FB4: 29 0F       >808           and   #$0F        ; Lo digit
0FB6: 85 D7       >809           sta   t1          ; Save for compare.
0FB8: B9 00 00    >810  :comp2   lda   0*0,y       ; Reg byte
0FBB: 29 0F       >811           and   #$0F        ; Lo digit
0FBD: C5 D7       >812           cmp   t1          ; Compare digits.
0FBF: D0 D3       >813           bne   :done       ; Done if unequal.
0FC1: F0 07       >814           beq   :nxbyte     ; Else continue (always)
                  >815
0FC3: B9 00 00    >816  :byte    lda   0*0,y       ; Reg byte
0FC6: D1 D1       >817           cmp   (memptr),y  ; Compare w MEM.
0FC8: D0 CA       >818           bne   :done       ; Done if unequal.
0FCA: C8          >819  :nxbyte  iny               ; Advance byte index and
0FCB: CA          >820           dex               ;  decrement digit count
0FCC: D0 B6       >821           bne   :hidigit    ; Continue if digits left,
0FCE: F0 C4       >822           beq   :done       ;  else done. (always)
```

```
                        put   B220EXEC2
0FD0: 29 01    FAD      and   #$01       ; Standardize sign of
0FD2: 85 AA             sta   rD+S       ;  MEM operand (0/1).
0FD4: A5 9A             lda   rC+VV      ; FAD or FAA?
0FD6: 29 0F             and   #$0F
0FD8: 49 01             eor   #$01
0FDA: D0 02             bne   ]fad       ; -FAD, continue.
0FDC: 85 AA             sta   rD+S       ; -FAA, force +.
0FDE: A5 99    ]fad     lda   rC+sL      ; Get normalization limit.
0FE0: 4A                lsr
0FE1: 4A                lsr
0FE2: 4A                lsr
0FE3: 4A                lsr
0FE4: D0 02             bne   :nonzero
0FE6: A9 0A             lda   #10
0FE8: 85 D8    :nonzero sta   NN         ; Save binary norm limit.
0FEA: A5 9E             lda   rA+S       ; Standardize rA sign (0/1)
0FEC: 29 01             and   #$01
0FEE: 85 9E             sta   rA+S
0FF0: A0 05             ldy   #5         ; Copy MEM operand to rD.
0FF2: B1 D1    :mem2rD  lda   (memptr),y
0FF4: 99 AA 00          sta   rD,y
0FF7: 88                dey
0FF8: D0 F8             bne   :mem2rD    ; (rD sign already set)
0FFA: 84 D7             sty   t1         ; Init t1 = 0
0FFC: A2 01             ldx   #EXP       ; Compare rA & rD magnitudes
0FFE: B5 9E    :complp  lda   rA,x
1000: D5 AA             cmp   rD,x
1002: 90 3B             bcc   :Alt       ; rA < rD.
1004: D0 05             bne   :Age       ; rA > rD.
1006: E8                inx              ; rA = rD so far...
1007: E0 06             cpx   #6
1009: D0 F3             bne   :complp
100B: F8       :Age     sed              ; / Decimal mode.
100C: A5 9F             lda   rA+EXP     ; rA >= rD. C = 1.
100E: E5 AB             sbc   rD+EXP     ; Operand misalignment
1010: F0 3D             beq   :doarith   ; Misalignment = 0, go.
1012: C9 08             cmp   #8         ; Is misalignment > 7?
1014: B0 7E             bcs   :done      ; -Yes, rA unchanged.
1016: 4A                lsr              ; -No, div by 2, C = odd.
1017: 90 0E             bcc   :bytesh    ; Even, so shift bytes.
1019: A2 04             ldx   #4         ; Odd.  4 bits / digit.
101B: 18       :digsh   clc              ; Shift rD right 1 digit.
101C: 66 AC             ror   rD+MANT
101E: 66 AD             ror   rD+MANT+1
1020: 66 AE             ror   rD+MANT+2
1022: 66 AF             ror   rD+MANT+3
1024: CA                dex
1025: D0 F4             bne   :digsh
1027: A8       :bytesh  tay              ; Byte shift count
1028: F0 25             beq   :doarith   ; -Ready to go.
102A: A5 AE    :bytenxt lda   rD+MANT+2  ; -Shift right 2 digits
102C: 85 AF             sta   rD+MANT+3
102E: A5 AD             lda   rD+MANT+1
1030: 85 AE             sta   rD+MANT+2
1032: A5 AC             lda   rD+MANT
1034: 85 AD             sta   rD+MANT+1
1036: A9 00             lda   #0
1038: 85 AC             sta   rD+MANT
103A: 88                dey
103B: D0 ED             bne   :bytenxt
103D: F0 10             beq   :doarith   ; (always)

103F: A2 05    :Alt     ldx   #5         ; Exchange rA and rD
1041: B5 9E    :exchAD  lda   rA,x       ;  so |rA| > |rD|.
1043: B4 AA             ldy   rD,x
1045: 94 9E             sty   rA,x
```

```
1047: 95 AA    >67             sta   rD,x
1049: CA       >68             dex
104A: 10 F5    >69             bpl   :exchAD
104C: 38       >70             sec               ; Now |rA| >= |rD|.
104D: B0 BC    >71             bcs   :Age        ; (always)
               >72
104F: A5 9E    >73   :doarith lda   rA+S        ; Compare signs.
1051: C5 AA    >74             cmp   rD+S
1053: D0 43    >75             bne   :subtr      ; -Different, subtract.
1055: A2 03    >76             ldx   #3          ; -Same, add.
1057: 18       >77             clc
1058: B5 A0    >78   :add      lda   rA+MANT,x  ; rA mantissa =
105A: 75 AC    >79             adc   rD+MANT,x  ;  rA mantissa +
105C: 95 A0    >80             sta   rA+MANT,x  ;   rD mantissa.
105E: 05 D7    >81             ora   t1          ; Summarize zero
1060: 85 D7    >82             sta   t1          ;  mantissa.
1062: CA       >83             dex
1063: 10 F3    >84             bpl   :add
1065: B0 06    >85             bcs   :carry      ; Carry out of mantissa.
1067: A5 D7    >86             lda   t1          ; Result mantissa = 0?
1069: F0 41    >87             beq   :clrexp     ; -Yes, Result = 0.
106B: D0 43    >88             bne   :norm       ; -No, normalize. (always)
               >89
106D: A5 9F    >90   :carry    lda   rA+EXP      ; -Carry into EXP field.
106F: C9 99    >91             cmp   #$99        ; Is EXP = 99 (max)?
1071: D0 0A    >92             bne   :adj        ; -No, shift right.
1073: A9 01    >93             lda   #$01        ; -Yes, force EXP
1075: 85 9F    >94             sta   rA+EXP      ;   to 01 (unshifted sum)
1077: A9 00    >95             lda   #0          ;    and force rA sign
1079: 85 9E    >96             sta   rA+S        ;     to 0.
107B: F0 13    >97             beq   :ovflo      ;      and overflow. (always)
               >98
107D: 38       >99   :adj      sec               ; Restore the carry out.
107E: A2 04    >100            ldx   #4          ; 4 bits / digit.
1080: 20 04 16 >101  :srloop   jsr   srAM        ; -Shift mant 1 dig right.
1083: 18       >102            clc               ; Shift in zeroes.
1084: CA       >103            dex
1085: D0 F9    >104            bne   :srloop
1087: 18       >105            clc
1088: A5 9F    >106            lda   rA+EXP      ; Increment rA exponent.
108A: 69 01    >107            adc   #1
108C: 85 9F    >108            sta   rA+EXP
108E: 90 04    >109            bcc   :done       ; -No overflow.
               >110  :ovflo    seti  Ov          ; -Signal exponent overflow.
1090: A9 FF    >110            lda   #$FF
1092: 85 C3    >110            sta   Ov          ; Set non-zero.
               >110            eom
1094: D8       >111  :done     cld               ; \ Back to binary.
1095: 4C 52 09 >112            jmp   fetch
               >113
1098: A2 03    >114  :subtr    ldx   #3          ; Subtract.
109A: 38       >115            sec
109B: B5 A0    >116  :sub      lda   rA+MANT,x  ; rA mantissa =
109D: F5 AC    >117            sbc   rD+MANT,x  ;  rA mantissa -
109F: 95 A0    >118            sta   rA+MANT,x  ;   rD mantissa.
10A1: 05 D7    >119            ora   t1          ; Summarize zero
10A3: 85 D7    >120            sta   t1          ;  mantissa.
10A5: CA       >121            dex
10A6: 10 F3    >122            bpl   :sub
10A8: A5 D7    >123            lda   t1          ; Result mantissa = 0?
10AA: D0 04    >124            bne   :norm       ; -No, normalize.
10AC: 85 9F    >125  :clrexp   sta   rA+EXP      ; -Yes, exponent = 0.
10AE: F0 E4    >126            beq   :done       ; (always)
               >127
10B0: A5 A0    >128  :norm     lda   rA+MANT     ; Normalize result.
10B2: 29 F0    >129            and   #$F0        ; Hi digit = 0?
10B4: D0 DE    >130            bne   :done       ; -No, all done.
```

```
10B6: A2 04    >131           ldx   #4          ; -Yes, shift left 1 dig.
10B8: 18       >132  :diglp   clc               ; Shift in zeroes.
10B9: 26 A3    >133           rol   rA+MANT+3
10BB: 26 A2    >134           rol   rA+MANT+2
10BD: 26 A1    >135           rol   rA+MANT+1
10BF: 26 A0    >136           rol   rA+MANT
10C1: CA       >137           dex
10C2: D0 F4    >138           bne   :diglp
10C4: C6 D8    >139           dec   NN          ; Norm limit exceeded?
10C6: 10 04    >140           bpl   :ok         ; -No, continue.
               >141           resi  RUN         ; -Limit exceeded, halt.
10C8: A9 00    >141           lda   #0
10CA: 85 C0    >141           sta   RUN         ; Zero indicator.
               >141           eom
10CC: 38       >142  :ok      sec
10CD: A5 9F    >143           lda   rA+EXP      ; Decrement rA exponent
10CF: E9 01    >144           sbc   #1
10D1: 85 9F    >145           sta   rA+EXP
10D3: B0 DB    >146           bcs   :norm
10D5: A2 9E    >147           ldx   #rA         ; Exponent underflow,
10D7: 20 68 16 >148           jsr   clear       ;  clear rA.
10DA: 4C 94 10 >149           jmp   :done
               >150
10DD: 29 01    >151  FSU      and   #$01        ; Standardize sign of
10DF: 85 AA    >152           sta   rD+S        ;  MEM operand (0/1).
10E1: A5 9A    >153           lda   rC+VV       ; FSU or FSA?
10E3: 29 0F    >154           and   #$0F
10E5: C9 01    >155           cmp   #1
10E7: F0 04    >156           beq   :setneg     ; -FSA, set operand -.
10E9: A5 AA    >157           lda   rD+S        ; -FSU.
10EB: 49 01    >158           eor   #$01        ;   Complement sign
10ED: 85 AA    >159  :setneg  sta   rD+S        ;    of operand,
10EF: 4C DE 0F >160           jmp   ]fad        ;     and do FAD.
```

```
10F2: 18       >162  FMU      clc              ; Floating MUltiply
10F3: C8       >163           iny              ; Y = 1 (exponent field)
10F4: F8       >164           sed              ; / Decimal mode.
10F5: B1 D1    >165           lda   (memptr),y ; Operand exponent
10F7: 85 D3    >166           sta   ptr        ; Save for restoration.
10F9: 65 9F    >167           adc   rA+EXP     ;  + rA exponent
10FB: 90 0A    >168           bcc   :notov     ; No overflow.
10FD: C9 50    >169           cmp   #$50       ; Sum < 150?
10FF: 90 0A    >170           bcc   :ok        ; -Yes, no overflow.
1101: A9 80    >171           lda   #$80       ; -No, signal pending
1103: 85 C3    >172           sta   Ov         ;   FMU overflow
1105: B0 09    >173           bcs   :cont      ;    and continue a bit.
               >174
1107: C9 50    >175  :notov   cmp   #$50       ; Sum < 50?
1109: 90 71    >176           bcc   :unflow    ; -Yes, underflow.
110B: 38       >177  :ok      sec              ; -No, subtract extra
110C: E9 50    >178           sbc   #$50       ;   excess 50 and
110E: 85 D8    >179           sta   NN         ;    save result exponent.
1110: A9 00    >180  :cont    lda   #0         ; Clear operand and
1112: 91 D1    >181           sta   (memptr),y ;  rA exponents.
1114: 85 9F    >182           sta   rA+EXP
1116: A5 A0    >183           lda   rA+MANT    ; Is rA unnormalized?
1118: 29 F0    >184           and   #$F0
111A: F0 60    >185           beq   :unflow    ; -Yes, underflow.
111C: C8       >186           iny              ; Y = 2 (mantissa)
111D: B1 D1    >187           lda   (memptr),y ; Is memory operand
111F: 29 F0    >188           and   #$F0       ;  unnormalized?
1121: F0 59    >189           beq   :unflow    ; -Yes, underflow.
1123: A5 AA    >190           lda   rD+S       ; Recover operand sign.
1125: 20 CA 0D >191           jsr   multiply   ; Do the multiply.
1128: A5 C3    >192           lda   Ov         ; FMU overflow pending?
112A: C9 80    >193           cmp   #$80
112C: F0 47    >194           beq   :ovflow    ; -Yes, quit.
112E: A2 02    >195           ldx   #2         ; -No, shift rA & rR
1130: B5 9F    >196  :shloop  lda   rA+1,x     ;  left one byte.
1132: 95 9E    >197           sta   rA,x
1134: E8       >198           inx
1135: E0 06    >199           cpx   #6         ; Skip rR sign byte.
1137: D0 05    >200           bne   :notsign
1139: A5 A5    >201           lda   rR+1
113B: 85 A3    >202           sta   rA+5
113D: E8       >203           inx
113E: E0 0B    >204  :notsign cpx   #11        ; Done?
1140: D0 EE    >205           bne   :shloop    ; -No, continue.
1142: A9 00    >206           lda   #0         ; -Yes, clear
1144: 85 A9    >207           sta   rR+5       ;   low byte of rR.
1146: A5 A0    >208           lda   rA+MANT    ; Is rA normalized?
1148: 29 F0    >209           and   #$F0
114A: D0 13    >210           bne   :normal    ; -Yes.
114C: A0 04    >211           ldy   #4         ; -No, shift rA & rR
114E: 18       >212  :shdig   clc              ;   left one digit.
114F: 20 31 16 >213           jsr   slT
1152: 88       >214           dey
1153: D0 F9    >215           bne   :shdig
1155: A5 D8    >216           lda   NN         ; Recover result exp
1157: F0 23    >217           beq   :unflow    ; Underflow if 0.
1159: F8       >218           sed              ; / Decimal mode.
115A: 38       >219           sec
115B: E9 01    >220           sbc   #1         ; Compensate for shift.
115D: 85 D8    >221           sta   NN
115F: A5 D8    >222  :normal  lda   NN
1161: 85 9F    >223           sta   rA+EXP     ; Set result exponent.
1163: D8       >224  :done    cld              ; \ Binary mode.
1164: A5 C3    >225           lda   Ov         ; Pending FMU ovflow?
1166: F0 04    >226           beq   :noOv      ; -No.
               >227           seti  Ov         ; -Yes, standardize it.
1168: A9 FF    >227           lda   #$FF
```

```
116A: 85 C3    >227           sta   Ov          ; Set non-zero.
               >227           eom
116C: A0 01    >228  :noOv    ldy   #1          ; Restore memory
116E: A5 D3    >229           lda   ptr         ;  operand's exponent.
1170: 91 D1    >230           sta   (memptr),y
1172: 4C 52 09 >231           jmp   fetch
               >232
1175: A9 00    >233  :ovflow  lda   #0
1177: 85 A4    >234           sta   rR+S        ; Clear rR sign
1179: 4C 63 11 >235           jmp   :done       ;  and clean up.
               >236
117C: 20 82 11 >237  :unflow  jsr   clearAR     ; Clear rA and rR
117F: 4C 63 11 >238           jmp   :done       ;  and clean up.
               >239
1182: A2 9E    >240  clearAR  ldx   #rA         ; Clear rA.
1184: 20 68 16 >241           jsr   clear
1187: A2 A4    >242           ldx   #rR         ; Clear rR.
1189: 20 68 16 >243           jsr   clear
118C: 60       >244           rts
```

```
118D: C8       >246  FDV      iny              ; Floating DiVide (Y==>EXP)
118E: B1 D1    >247           lda   (memptr),y ; Save MEM exponent
1190: 85 D3    >248           sta   ptr        ;  for restoration
1192: A9 00    >249           lda   #0         ;   and clear it for
1194: 91 D1    >250           sta   (memptr),y ;    for divide.
1196: C8       >251           iny              ; Y ==> MEM mantissa
1197: B1 D1    >252           lda   (memptr),y ; Hi byte of mant
1199: 29 F0    >253           and   #$F0       ; Divisor normalized?
119B: F0 5D    >254           beq   :denorm    ; -No, overflow.
119D: A5 A0    >255           lda   rA+MANT    ; Hi byte of rA mant
119F: 29 F0    >256           and   #$F0       ; Dividend normalized?
11A1: F0 67    >257           beq   :unflo     ; -No, underflow.
11A3: F8       >258           sed              ; /Decimal mode.
11A4: 38       >259           sec
11A5: A5 9F    >260           lda   rA+EXP     ; Dividend exponent
11A7: E5 D3    >261           sbc   ptr        ;  - divisor exponent.
11A9: B0 07    >262           bcs   :chkov     ; *dend >= *isor, ck ovflo.
11AB: 38       >263           sec              ; *dend <  *isor, ck unflo.
11AC: E9 50    >264           sbc   #$50       ; Restore excess-50
11AE: 90 5A    >265           bcc   :unflo     ; Exponent underflow.
11B0: B0 05    >266           bcs   :ok        ; (always)
               >267
11B2: 18       >268  :chkov   clc
11B3: 69 50    >269           adc   #$50       ; Restore excess-50
11B5: B0 3F    >270           bcs   :ovflo     ; Exponent overflow.
11B7: 85 D8    >271  :ok      sta   NN         ; Save result exponent.
11B9: A9 00    >272           lda   #0         ; Clear rA exponent
11BB: 85 9F    >273           sta   rA+EXP     ;  for divide.
11BD: A0 04    >274           ldy   #4         ; 4 bits/digit.
11BF: 18       >275  :shrt    clc              ; Shift in zeros.
11C0: 20 0F 16 >276           jsr   srAMR      ; Shift rA mant & rR
11C3: 88       >277           dey              ;  right one digit.
11C4: D0 F9    >278           bne   :shrt
11C6: A5 A4    >279           lda   rR+S       ; Save original rR sign
11C8: 48       >280           pha
11C9: A5 AA    >281           lda   rD+S       ; Y=0, A=MEM sign
11CB: 20 53 0E >282           jsr   divide     ; Divide clears decimal mode.
11CE: 68       >283           pla              ; Restore original rR sign
11CF: 85 A4    >284           sta   rR+S
11D1: A5 9F    >285           lda   rA+1       ; Hi byte of quotient.
11D3: 29 F0    >286           and   #$F0       ; Is hi digit = 0?
11D5: D0 0C    >287           bne   :shrT2     ; -No, shift right 2 digs.
11D7: A0 04    >288           ldy   #4         ; -Yes, shift right 1 dig.
11D9: 18       >289  :shloop  clc              ; Shift in zeros.
11DA: 20 0D 16 >290           jsr   srT        ; Shift |rA| & |rR|
11DD: 88       >291           dey              ;  right one digit.
11DE: D0 F9    >292           bne   :shloop
11E0: 18       >293           clc              ; Indicate no overflow.
11E1: F0 0D    >294           beq   :setexp    ; (always)
               >295
11E3: F8       >296  :shrT2   sed              ; / Decimal mode.
11E4: 18       >297           clc
11E5: A5 D8    >298           lda   NN
11E7: 69 01    >299           adc   #1         ; EXP = EXP + 1
11E9: 85 D8    >300           sta   NN
11EB: B0 0D    >301           bcs   :denorm    ; Exponent overflow
11ED: 20 1D 16 >302           jsr   srT2       ; Make room for exponent
11F0: A5 D8    >303  :setexp  lda   NN         ; Set quotient exponent.
11F2: 85 9F    >304           sta   rA+EXP
11F4: 90 0A    >305           bcc   :done      ; (always)
               >306
11F6: A9 00    >307  :ovflo   lda   #0         ; On exponent overflow
11F8: 85 9F    >308           sta   rA+EXP     ;  clear result exponent.
11FA: 85 9E    >309  :denorm  sta   rA+S       ; Clear rA sign and
               >310           seti  Ov         ;  set Overflow indicator.
11FC: A9 FF    >310           lda   #$FF
11FE: 85 C3    >310           sta   Ov         ; Set non-zero.
```

```
                         eom
1200: A5 D3        :done lda   ptr         ; Recover MEM exponent
1202: A0 01              ldy   #1          ;  and put it back into
1204: 91 D1              sta   (memptr),y ;   divisor in memory.
1206: D8                 cld               ; \ Binary mode.
1207: 4C 52 09           jmp   fetch

120A: 20 82 11    :unflo jsr   clearAR     ; Clear rA and rR
120D: 4C 00 12           jmp   :done       ;  and finish up.
```

```
1210: A9 18    >320  IFL      lda   #CLCop     ; Patch ]dfl for IFL
1212: 8D AF 12 >321           sta   ]clc
1215: A9 65    >322           lda   #ADCZop
1217: 8D BE 12 >323           sta   ]adc
121A: A9 C9    >324           lda   #CMPIop
121C: 8D C0 12 >325           sta   ]cmp
121F: A9 EA    >326           lda   #NOPop
1221: 8D E8 12 >327           sta   ]nop
1224: A9 79    >328           lda   #ADCYop
1226: 8D EB 12 >329           sta   ]sub
1229: A9 C3    >330           lda   #Ov
122B: 8D 0A 13 >331           sta   ]Ov+3
122E: 20 7B 12 >332           jsr   ]dfl       ; Do the IFL.
1231: A9 C4    >333           lda   #Rp        ; Patch ]dfl back.
1233: 8D 0A 13 >334           sta   ]Ov+3
1236: A9 F9    >335           lda   #SBCYop
1238: 8D EB 12 >336           sta   ]sub
123B: A9 38    >337           lda   #SECop
123D: 8D E8 12 >338           sta   ]nop
1240: A9 24    >339           lda   #BITZop
1242: 8D C0 12 >340           sta   ]cmp
1245: A9 E5    >341           lda   #SBCZop
1247: 8D BE 12 >342           sta   ]adc
124A: A9 EA    >343           lda   #NOPop
124C: 8D AF 12 >344           sta   ]clc
124F: A5 C1    >345           lda   ERR        ; Error detected?
1251: D0 10    >346           bne   ]errpt     ; -Yes, report it.
1253: 4C 52 09 >347  ]fetch4  jmp   fetch
               >348
               >349  DFL      resi  Rp         ; Reset Repeat indicator.
1256: A9 00    >349           lda   #0
1258: 85 C4    >349           sta   Rp         ; Zero indicator.
               >349           eom
125A: 20 7B 12 >350           jsr   ]dfl       ; Decrease FieLd
125D: A5 C1    >351           lda   ERR        ; Error detected?
125F: D0 02    >352           bne   ]errpt     ; -Yes, report it.
1261: F0 F0    >353           beq   ]fetch4    ; (always)
               >354
1263: 4C 4B 0A >355  ]errpt   jmp   ]err
               >356
               >357  DLB      resi  Rp         ; Reset Repeat indicator.
1266: A9 00    >357           lda   #0
1268: 85 C4    >357           sta   Rp         ; Zero indicator.
               >357           eom
126A: 20 7B 12 >358           jsr   ]dfl       ; Decrease Field
126D: A5 AD    >359           lda   rD+3       ; Load rB from rD 8:4.
126F: 85 94    >360           sta   rB
1271: A5 AE    >361           lda   rD+4
1273: 85 95    >362           sta   rB+1
1275: A5 C1    >363           lda   ERR        ; Error detected?
1277: D0 EA    >364           bne   ]errpt     ; -Yes, report it.
1279: F0 D8    >365           beq   ]fetch4    ; (always)
```

```
127B: A2 AA    ]dfl     ldx   #rD        ; Clear rD.
127D: 20 68 16          jsr   clear
1280: A2 B0             ldx   #rD10      ; Clear rD10.
1282: 20 68 16          jsr   clear
1285: 20 54 16          jsr   splitsL    ; A = s, X = L
1288: 18                clc
1289: 69 01             adc   #1         ; A = s + 1
128B: 4A                lsr              ; A = (s+1)/2, C = even dig
128C: 08                php              ; Push Carry status.
128D: A8                tay              ; Y = low byte index
128E: A5 9A             lda   rC+VV      ; NN
1290: 99 B0 00          sta   rD10,y     ; rD10 = subtrahend
1293: B0 16             bcs   :subtr     ; Even dig first, no shift.
1295: 86 D7             stx   t1         ; Save X
1297: 98                tya              ; Move Y to X.
1298: AA                tax
1299: 16 B0             asl   rD10,x     ; Odd dig first, shift
129B: 36 AF             rol   rD10-1,x   ;  1 digit left.
129D: 16 B0             asl   rD10,x
129F: 36 AF             rol   rD10-1,x
12A1: 16 B0             asl   rD10,x
12A3: 36 AF             rol   rD10-1,x
12A5: 16 B0             asl   rD10,x
12A7: 36 AF             rol   rD10-1,x
12A9: A6 D7             ldx   t1         ; Restore X.
12AB: 28       :subtr   plp              ; Pop C.
12AC: F8                sed              ; / Decimal mode.
12AD: 90 39             bcc   ]nop       ; Not C = odd dig first.
12AF: EA       ]clc     nop              ; <Patch to CLC for IFL>
12B0: CA       :evendig dex              ; Both even and odd digs?
12B1: D0 36             bne   :byte      ; -Yes, subtr whole byte.
12B3: B9 B0 00          lda   rD10,y     ; -No, subtr final digit.
12B6: 29 0F             and   #$0F       ; Isolate even digit
12B8: 85 D7             sta   t1         ;  and save for subtract.
12BA: B1 D1             lda   (memptr),y ; MEM byte
12BC: 29 0F             and   #$0F       ; Isolate even digit
12BE: E5 D7    ]adc     sbc   t1         ;  & subtr. <ADC for IFL>
12C0: 24 10    ]cmp     bit   $10        ; CMP# if IFL (to set C)
12C2: 29 0F             and   #$0F       ; Mask result
12C4: 85 D7             sta   t1         ;  and save it.
12C6: B1 D1             lda   (memptr),y ; Recover MEM byte,
12C8: 29 F0             and   #$F0       ;  mask out even digit,
12CA: 05 D7             ora   t1         ;   OR in difference,
12CC: 91 D1             sta   (memptr),y ;    and put it back.
12CE: A4 AE             ldy   rD+4       ; Save high 4 digits of
12D0: 84 AF             sty   rD+5       ;  difference in rD 8:4.
12D2: A4 AD             ldy   rD+3
12D4: 84 AE             sty   rD+4
12D6: 85 AD             sta   rD+3
12D8: 08                php              ; Push Carry status.
12D9: A2 04             ldx   #4         ; 4 bits/digit
12DB: 26 AF    :shlp    rol   rD+5       ; Shift rD left 1 digit
12DD: 26 AE             rol   rD+4       ;  to line up with rB.
12DF: 26 AD             rol   rD+3
12E1: CA                dex
12E2: D0 F7             bne   :shlp
12E4: 28                plp              ; Pop Carry status.
12E5: 4C 04 13          jmp   :done

12E8: 38       ]nop     sec              ; <Patch to NOP for IFL>
12E9: B1 D1    :byte    lda   (memptr),y ; MEM byte
12EB: F9 B0 00 ]sub     sbc   rD10,y     ;  minus subtrahend
12EE: 91 D1             sta   (memptr),y ;   back to MEM.
12F0: 84 D7             sty   t1         ; Save Y
12F2: A4 AE             ldy   rD+4       ; Save 4 hi digits of
12F4: 84 AF             sty   rD+5       ;  difference in rD 8:4.
12F6: A4 AD             ldy   rD+3
```

```
12F8: 84 AE    >434            sty   rD+4
12FA: 85 AD    >435            sta   rD+3
12FC: A4 D7    >436            ldy   t1          ; Restore Y
12FE: 88       >437            dey
12FF: 30 0B    >438            bmi   :flderr     ; Field error.
1301: CA       >439            dex               ; More digits?
1302: D0 AC    >440            bne   :evendig    ; -Yes, keep subtracting.
1304: D8       >441  :done     cld               ; \ -No. Back to binary.
1305: 90 04    >442            bcc   :noRpt      ; Underflow ==> no Rpt
               >443  ]Ov       seti  Rp          ; Set Rpt <Ov for IFL>
1307: A9 FF    >443            lda   #$FF
1309: 85 C4    >443            sta   Rp          ; Set non-zero.
               >443            eom
130B: 60       >444  :noRpt    rts
               >445
130C: A9 C6    >446  :flderr   lda   #"F"        ; Signal Field error
130E: 85 C1    >447            sta   ERR
1310: D8       >448            cld               ; Clear decimal mode.
1311: 60       >449            rts
```

```
1312: 84 D6    RTF      sty   inptr+1    ; 'inptr+1' = 0
1314: 84 D7             sty   t1         ; 't1' = 0
1316: 20 75 16          jsr   midNN      ; Extract NN (word count)
1319: 85 D5             sta   inptr      ; Save binary NN (1..100)
131B: A6 95             ldx   rB+1       ; Convert rB to MEM
131D: E0 9A             cpx   #$99+1     ;  address in 'ptr'.
131F: B0 51             bcs   :underr    ; Undigit error.
1321: A4 94             ldy   rB
1323: C0 4A             cpy   #$49+1
1325: B0 4E             bcs   :addrerr   ; Address error.
1327: BD E8 1A          lda   BCDLadrl,x
132A: 79 1C 1C          adc   BCDHadrl,y
132D: 85 D3             sta   ptr
132F: BD 82 1B          lda   BCDLadrh,x
1332: 79 66 1C          adc   BCDHadrh,y
1335: B0 3B             bcs   :underr    ; Carry out ==> undigit.
1337: 85 D4             sta   ptr+1      ; 'ptr' = dest MEM addr.
1339: A5 D5             lda   inptr      ; Binary NN
133B: 0A                asl              ; NN * 2 (2..200)
133C: 65 D5             adc   inptr      ; NN * 3 (3..300)
133E: 26 D6             rol   inptr+1    ; Capture high bit.
1340: 0A                asl
1341: 26 D6             rol   inptr+1    ; NN * 6 (6..600)
1343: AA                tax              ; Byte count lo
1344: A0 00             ldy   #0
1346: B1 D1    :movelp  lda   (memptr),y ; Move bytes upward.
1348: 91 D3             sta   (ptr),y
134A: CA                dex              ; Dec byte count lo
134B: F0 09             beq   :ckhi      ; If 0, chk hi byte.
134D: C8       :cont    iny
134E: D0 F6             bne   :movelp
1350: E6 D2             inc   memptr+1   ; Advance ptr pages
1352: E6 D4             inc   ptr+1
1354: D0 F0             bne   :movelp    ; (always)

1356: C6 D6    :ckhi    dec   inptr+1    ; Dec byte count hi
1358: 10 F3             bpl   :cont      ; Continue if >= 0.
135A: A5 D8             lda   NN         ; NN = 00 (100)?
135C: D0 02             bne   :lt100     ; -No, less than 100.
135E: E6 D7             inc   t1         ; -Yes, set 100.
1360: F8       :lt100   sed              ; / Decimal mode.
1361: 18                clc
1362: A5 95             lda   rB+1       ; rB = rB + NN
1364: 65 D8             adc   NN
1366: 85 95             sta   rB+1
1368: A5 94             lda   rB
136A: 65 D7             adc   t1         ; 1 if NN = 0, else 0.
136C: 85 94             sta   rB
136E: D8                cld              ; \ Back to binary.
136F: 4C 52 09          jmp   fetch

1372: 4C 49 0A :underr  jmp   UNDIGerr   ; Relay jump.
1375: 4C 3F 0A :addrerr jmp   ADDRerr    ; Relay jump.
```

```
1378: F8         IBB      sed                 ; / Decimal mode.
1379: 18                  clc
137A: A5 95               lda   rB+1          ; rB = rB + rC(4:4)
137C: 65 9A               adc   rC+VV
137E: 85 95               sta   rB+1
1380: A5 94               lda   rB
1382: 65 99               adc   rC+sL
1384: 85 94               sta   rB
1386: D8                  cld                 ; \ Back to binary.
1387: 90 58               bcc   BUN           ; No overflow ==> branch
1389: B0 66               bcs   ]fetch3       ; Overflow ==> continue

138B: F8         DBB      sed                 ; / Decimal mode.
138C: 38                  sec
138D: A5 95               lda   rB+1          ; rB = rB - rC(4:4)
138F: E5 9A               sbc   rC+VV
1391: 85 95               sta   rB+1
1393: A5 94               lda   rB
1395: E5 99               sbc   rC+sL
1397: 85 94               sta   rB
1399: D8                  cld                 ; \ Back to binary.
139A: B0 45               bcs   BUN           ; No underflow ==> branch
139C: 90 53               bcc   ]fetch3       ; Underflow. (always)
```

```
139E: A5 C3    >529  BOF      lda   Ov          ; Overflow indicator set?
13A0: D0 02    >530           bne   :ovflo      ; -Yes, clear it and branch.
13A2: F0 4D    >531           beq   ]fetch3     ; (always)
               >532
               >533  :ovflo   resi  Ov          ; Reset Overflow indicator
13A4: A9 00    >533           lda   #0
13A6: 85 C3    >533           sta   Ov          ; Zero indicator.
               >533           eom
13A8: 4C E1 13 >534           jmp   BUN         ;  and take the branch.
               >535
13AB: A5 C4    >536  BRP      lda   Rp          ; Repeat indicator set?
13AD: D0 32    >537           bne   BUN         ; -Yes, branch.
13AF: F0 40    >538           beq   ]fetch3     ; (always)
               >539
13B1: A5 9A    >540  BSA      lda   rC+VV       ; Get comparand digit
13B3: 29 0F    >541           and   #$0F
13B5: C5 9E    >542           cmp   rA+S        ; Equal to rA sign?
13B7: F0 28    >543           beq   BUN         ; -Yes, take branch.
13B9: D0 36    >544           bne   ]fetch3     ; (always)
               >545
13BB: A5 9A    >546  BCH      lda   rC+VV       ; BCH or BCL?
13BD: 29 01    >547           and   #$01
13BF: F0 06    >548           beq   :bch        ; -BCH.
13C1: A5 C2    >549           lda   COMP        ; -BCL.
13C3: 30 1C    >550           bmi   BUN         ; Branch if Lo
13C5: 10 2A    >551           bpl   ]fetch3     ; (always)
               >552
13C7: A5 C2    >553  :bch     lda   COMP
13C9: F0 26    >554           beq   ]fetch3     ; Equal.
13CB: 10 14    >555           bpl   BUN         ; Branch if Hi
13CD: 30 22    >556           bmi   ]fetch3     ; (always)
               >557
13CF: A5 9A    >558  BCE      lda   rC+VV       ; BCE or BCU?
13D1: 29 01    >559           and   #$01
13D3: F0 06    >560           beq   :bce        ; BCE.
13D5: A5 C2    >561           lda   COMP
13D7: D0 08    >562           bne   BUN         ; Branch if unequal.
13D9: F0 16    >563           beq   ]fetch3     ; (always)
               >564
13DB: A5 C2    >565  :bce     lda   COMP
13DD: F0 02    >566           beq   BUN         ; Branch if equal.
13DF: D0 10    >567           bne   ]fetch3     ; (always)
               >568
13E1: A5 9C    >569  BUN      lda   rC+ADDR     ; Set new P reg
13E3: 85 96    >570           sta   rP
13E5: A5 9D    >571           lda   rC+ADDR+1
13E7: 85 97    >572           sta   rP+1
13E9: A5 D1    >573           lda   memptr      ;  and instptr.
13EB: 85 CF    >574           sta   instptr
13ED: A5 D2    >575           lda   memptr+1
13EF: 85 D0    >576           sta   instptr+1
13F1: 4C 52 09 >577  ]fetch3  jmp   fetch
```

```
13F4: A2 A4    >579  BFR      ldx   #rR         ; X points to rR
13F6: D0 02    >580           bne   ]bfr
               >581
13F8: A2 9E    >582  BFA      ldx   #rA         ; X points to rA
13FA: A4 9A    >583  ]bfr     ldy   rC+VV       ; Y = 2-digit comparand
13FC: A5 99    >584           lda   rC+sL
13FE: 29 10    >585           and   #$10        ; s even or odd?
1400: F0 0E    >586           beq   :even       ; -Even, no digit swap.
1402: 98       >587           tya               ; -Odd, swap digits.
1403: C9 80    >588           cmp   #$80        ; Hi bit to C
1405: 2A       >589           rol               ;  and rotate 1 bit.
1406: C9 80    >590           cmp   #$80        ; Hi bit to C
1408: 2A       >591           rol               ;  and rotate 1 bit.
1409: C9 80    >592           cmp   #$80        ; Hi bit to C
140B: 2A       >593           rol               ;  and rotate 1 bit.
140C: C9 80    >594           cmp   #$80        ; Hi bit to C
140E: 2A       >595           rol               ;  and rotate 1 bit.
140F: A8       >596           tay
1410: 84 B5    >597  :even    sty   rD10+5      ; Expand comparand
1412: 84 B4    >598           sty   rD10+4      ;  to full width in rD10.
1414: 84 B3    >599           sty   rD10+3
1416: 84 B2    >600           sty   rD10+2
1418: 84 B1    >601           sty   rD10+1
141A: 98       >602           tya
141B: 29 0F    >603           and   #$0F        ; Mask off hi sign digit.
141D: 85 B0    >604           sta   rD10
141F: A5 D2    >605           lda   memptr+1    ; Push 'memptr' on stack.
1421: 48       >606           pha
1422: A5 D1    >607           lda   memptr
1424: 48       >608           pha
1425: A9 B0    >609           lda   #rD10       ; Point 'memptr' at rD10
1427: 85 D1    >610           sta   memptr
1429: A9 00    >611           lda   #0
142B: 85 D2    >612           sta   memptr+1
               >613
142D: A0 01    >614           ldy   #1          ; Partial field compare
142F: A9 B0    >615           lda   #BCSop      ; Unsigned compare
1431: 20 32 0F >616           jsr   compare
1434: AA       >617           tax               ; Save A
1435: 68       >618           pla               ; Pop 'memptr'
1436: 85 D1    >619           sta   memptr
1438: 68       >620           pla
1439: 85 D2    >621           sta   memptr+1
143B: A5 C1    >622           lda   ERR         ; Error detected?
143D: D0 05    >623           bne   :err        ; -Yes, report it.
143F: 8A       >624           txa               ; Recover COMP flags
1440: F0 9F    >625           beq   BUN         ; -Branch if equal.
1442: D0 7A    >626           bne   ]fetch2     ; -Else NOP. (always)
               >627
1444: 4C 4B 0A >628  :err     jmp   ]err
               >629
1447: A5 99    >630  BCS      lda   rC+sL       ; Get switch #
1449: 4A       >631           lsr
144A: 4A       >632           lsr
144B: 4A       >633           lsr
144C: 4A       >634           lsr
144D: AA       >635           tax
144E: B5 B6    >636           lda   CSW,x       ; Get switch state
1450: D0 8F    >637           bne   BUN         ; -True, take branch.
1452: F0 6A    >638           beq   ]fetch2     ; -False, no branch.
```

```
1454: A5 9A    >640  SOR      lda   rC+VV      ; SOR / SOH / IOM?
1456: 29 0F    >641           and   #$0F
1458: C9 02    >642           cmp   #2         ; IOM?
145A: F0 05    >643           beq   :iom       ; -Yes.
145C: 85 CE    >644           sta   OvHlt      ; -No, set Ovflo mode.
145E: 4C 52 09 >645  :fetch   jmp   fetch
               >646
1461: A5 CE    >647  :iom     lda   OvHlt
1463: F0 F9    >648           beq   :fetch     ; No branch if SOR mode.
1465: 4C E1 13 >649           jmp   BUN        ; Branch if SOH mode.
               >650
1468: A5 9A    >651  STA      lda   rC+VV      ; STA, STR, STB?
146A: 29 0F    >652           and   #$0F       ; Isolate reg variant.
146C: A2 A4    >653           ldx   #rR
146E: C9 01    >654           cmp   #1         ; STR?
1470: F0 14    >655           beq   :store     ; -Yes.
1472: A2 90    >656           ldx   #rBx
1474: C9 02    >657           cmp   #2         ; STB?
1476: F0 0E    >658           beq   :store     ; -Yes.
1478: A2 9E    >659           ldx   #rA        ; STA
147A: A5 9A    >660           lda   rC+VV      ; STD (STore Display point)
147C: C9 20    >661           cmp   #$20       ;  variant?
147E: D0 06    >662           bne   :store     ; -No, ordinary store.
1480: A5 9F    >663           lda   rA+sL      ; -Yes, set the DISplay
1482: 09 01    >664           ora   #$01       ;   continue bit.
1484: 85 9F    >665           sta   rA+sL
1486: A5 9A    >666  :store   lda   rC+VV      ; Partial field :store?
1488: 29 10    >667           and   #$10
148A: D0 0F    >668           bne   :stfield   ; -Yes, do it.
148C: 8E 92 14 >669           stx   :stloop+1  ; -No, full word store.
148F: A0 05    >670           ldy   #5
1491: B9 00 00 >671  :stloop  lda   0*0,y      ; Store the register.
1494: 91 D1    >672           sta   (memptr),y
1496: 88       >673           dey
1497: 10 F8    >674           bpl   :stloop
1499: 30 23    >675           bmi   ]fetch2    ; (always)
               >676
149B: 8E AC 14 >677  :stfield stx   :evendig+1 ; Save register
149E: 8E C2 14 >678           stx   :odddig+1  ;  address...
14A1: 20 54 16 >679           jsr   splitsL    ; Split sL: A = s and X = L
14A4: 18       >680           clc
14A5: 69 01    >681           adc   #1         ; A = s + 1
14A7: 4A       >682           lsr              ; A = (s+1)/2, C = even dig
14A8: A8       >683           tay              ; Y = byte offset
14A9: 90 16    >684           bcc   :odddig    ; -Start digit is odd.
14AB: B9 00 00 >685  :evendig lda   0*0,y      ; -Start digit is even.
14AE: CA       >686           dex              ; Both even & odd digits?
14AF: D0 1D    >687           bne   :byte      ; -Yes, move full byte.
14B1: E8       >688           inx              ; -No, restore dig counter.
14B2: 29 0F    >689           and   #$0F       ; Isolate even digit
14B4: 85 D7    >690           sta   t1         ;  and save it.
14B6: B1 D1    >691           lda   (memptr),y ; Get MEM byte,
14B8: 29 F0    >692           and   #$F0       ;  clear target digit,
14BA: 05 D7    >693           ora   t1         ;   OR in new digit,
14BC: 91 D1    >694           sta   (memptr),y ;    and put it back.
14BE: 4C 52 09 >695  ]fetch2  jmp   fetch      ; All done.
               >696
14C1: B9 00 00 >697  :odddig  lda   0*0,y      ; Start digit is odd.
14C4: 29 F0    >698           and   #$F0       ; Isolate reg digit
14C6: 85 D7    >699           sta   t1         ;  and save it.
14C8: B1 D1    >700           lda   (memptr),y ; Get MEM byte,
14CA: 29 0F    >701           and   #$0F       ;  clear target digit,
14CC: 05 D7    >702           ora   t1         ;   OR in new digit,
14CE: 91 D1    >703  :byte    sta   (memptr),y ;    and put it back.
14D0: 88       >704           dey              ; Move byte index.
14D1: 30 05    >705           bmi   :flderr    ; -Err if field too long.
14D3: CA       >706           dex              ; More digits?
```

```
14D4: D0 D5    >707           bne   :evendig   ; -Yes, continue.
14D6: F0 E6    >708           beq   ]fetch2    ; -No, finished. (always)
               >709
14D8: 4C 3B 0A >710  :flderr  jmp   FIELDerr   ; Report field error.
```

```
14DB: A0 05    LDR      ldy   #5          ; MEM(ADDR) ==> rR
14DD: B1 D1    :ldr     lda   (memptr),y
14DF: 99 A4 00          sta   rR,y
14E2: 88                dey
14E3: 10 F8             bpl   :ldr
14E5: 30 41             bmi   ]fetch1     ; (always)

14E7: A5 9A    LDB      lda   rC+VV       ; LDB, LBC
14E9: A0 05             ldy   #5
14EB: 29 01             and   #$01
14ED: D0 0C             bne   :lbc        ; Load rB Complement
14EF: B1 D1    :ldb     lda   (memptr),y
14F1: 85 95             sta   rB+1
14F3: 88                dey
14F4: B1 D1             lda   (memptr),y
14F6: 85 94             sta   rB
14F8: 4C 52 09          jmp   fetch       ; -Yes, done.

14FB: F8       :lbc     sed               ; / Decimal mode
14FC: 38                sec               ;  for 10's complement.
14FD: A9 00    :ldbc    lda   #0
14FF: F1 D1             sbc   (memptr),y
1501: 85 95             sta   rB+1
1503: 88                dey
1504: A9 00             lda   #0
1506: F1 D1             sbc   (memptr),y
1508: 85 94             sta   rB
150A: D8                cld               ; \ -Yes, back to binary.
150B: 90 1B             bcc   ]fetch1     ; (always)

150D: A5 9A    LSA      lda   rC+VV       ; Load Sign A
150F: 29 0F             and   #$0F        ; Isolate new sign digit
1511: 85 9E             sta   rA+S        ;  and put into rA.
1513: 4C 52 09          jmp   fetch
```

```
1516: A0 05    >747  STP      ldy   #5          ; rP + 1 ==> MEM(0:4)
1518: F8       >748           sed               ; / Decimal mode
1519: 18       >749           clc
151A: A5 97    >750           lda   rP+1
151C: 69 01    >751           adc   #1
151E: 91 D1    >752           sta   (memptr),y
1520: 88       >753           dey
1521: A5 96    >754           lda   rP
1523: 69 00    >755           adc   #0
1525: 91 D1    >756           sta   (memptr),y
1527: D8       >757           cld               ; \ Back to binary
1528: 4C 52 09 >758  ]fetch1  jmp   fetch       ; -Yes, done.
               >759
152B: A5 9A    >760  CLA      lda   rC+VV       ; CLA/R/B
152D: 4A       >761           lsr               ; 1-bit to C
152E: 85 D7    >762           sta   t1          ; Save mask
1530: 90 05    >763           bcc   :notA       ; rA not included.
1532: A2 9E    >764           ldx   #rA
1534: 20 68 16 >765           jsr   clear       ; Clear rA.
1537: 46 D7    >766  :notA    lsr   t1          ; 2-bit to C
1539: 90 05    >767           bcc   :notR       ; rR not included.
153B: A2 A4    >768           ldx   #rR
153D: 20 68 16 >769           jsr   clear       ; Clear rR.
1540: 46 D7    >770  :notR    lsr   t1          ; 4-bit to C.
1542: 90 05    >771           bcc   :fetch      ; rB not included.
1544: A2 90    >772           ldx   #rBx
1546: 20 68 16 >773           jsr   clear       ; Clear rB.
1549: 4C 52 09 >774  :fetch   jmp   fetch
               >775
154C: A9 00    >776  CLL      lda   #0          ; CLear Location
154E: A0 05    >777           ldy   #5
1550: 91 D1    >778  :cllloop sta   (memptr),y
1552: 88       >779           dey
1553: 10 FB    >780           bpl   :cllloop
1555: 30 D1    >781           bmi   ]fetch1     ; (always)
```

```
1557: A5 9D     SRA      lda   rC+ADDR+1  ; SRA, SRT, SRS nn
1559: 29 1F              and   #$1F       ; Isolate count 0..19
155B: C9 10              cmp   #$10       ; Greater than 9?
155D: 90 02              bcc   :nocor     ; -No, don't correct.
155F: E9 06              sbc   #6         ; -Yes, cnvrt to binary.
1561: 0A        :nocor   asl              ; Multiply digit shift
1562: 0A                 asl              ;  count by 4 (bits/digit).
1563: A8                 tay              ; Y = bit shift count.
1564: A5 9A              lda   rC+VV      ; SRA, SRT, SRS
1566: 29 0F              and   #$0F
1568: C9 01              cmp   #1         ; SRT?
156A: D0 08              bne   :notsrt    ; -No.
156C: A6 9E              ldx   rA+S       ; -Yes, SRT. Set rR sign
156E: 86 A4              stx   rR+S       ;   to rA sign, then
1570: A2 0D              ldx   #<srT      ;    shift both A and R.
1572: D0 08              bne   :setsh     ; Go shift. (always)

1574: A2 00     :notsrt  ldx   #<srAS
1576: C9 02              cmp   #2         ; SRS?
1578: F0 02              beq   :setsh     ; -Yes, shift right A & Sign
157A: A2 02              ldx   #<srA      ; SRA
157C: 8E 84 15  :setsh   stx   :shiftr+1  ; Set shift subroutine.
157F: 98                 tya              ; Is shift count = 0?
1580: F0 07              beq   :fetch     ; -Yes, done.
1582: 18        :nxbit   clc              ; Shift in zeros.
1583: 20 02 16  :shiftr  jsr   srA        ; (or srT or srAS)
1586: 88                 dey              ; Count exhausted?
1587: D0 F9              bne   :nxbit     ; -No, keep shifting.
1589: 4C 52 09  :fetch   jmp   fetch      ; -Yes, done.
```

```
158C: A5 9D    >813  SLA      lda   rC+ADDR+1  ; SLA, SLT, SLS nn
158E: 29 1F    >814           and   #$1F       ; Isolate count 0..19
1590: C9 10    >815           cmp   #$10       ; Greater than 9?
1592: 90 02    >816           bcc   :nocor     ; -No, don't correct.
1594: E9 06    >817           sbc   #6         ; -Yes, cnvrt to binary.
1596: AA       >818  :nocor   tax              ; X = shift count.
1597: A5 9A    >819           lda   rC+VV      ; SLA, SLT, SLS?
1599: 29 0F    >820           and   #$0F
159B: C9 01    >821           cmp   #1         ; SLT?
159D: F0 19    >822           beq   :slt       ; -Yes, shift left AR
159F: E0 00    >823           cpx   #0         ; -No, check count.
15A1: F0 12    >824           beq   :fetch     ; Done if count = 0.
15A3: C9 02    >825           cmp   #2         ; SLS?
15A5: F0 3C    >826           beq   :sls       ; -Yes, shift left A + Sign
15A7: A0 04    >827  :sla     ldy   #4         ; SLA. Shift 4 bits/digit.
15A9: A5 9F    >828  :nxbita  lda   rA+1       ; To rotate rA,
15AB: 2A       >829           rol              ;  preset C to high bit.
15AC: 20 3B 16 >830           jsr   slA        ; Rotate A left 1 bit.
15AF: 88       >831           dey              ; More bits?
15B0: D0 F7    >832           bne   :nxbita    ; -Yes.
15B2: CA       >833           dex              ; More digits?
15B3: D0 F2    >834           bne   :sla       ; -Yes.
15B5: 4C 52 09 >835  :fetch   jmp   fetch
               >836
15B8: A5 A4    >837  :slt     lda   rR+S       ; Copy rR Sign
15BA: 85 9E    >838           sta   rA+S       ;  to rA Sign.
15BC: 8A       >839           txa              ; Is count = 0?
15BD: F0 F6    >840           beq   :fetch     ; -Yes, done.
15BF: E0 0A    >841           cpx   #10        ; -No, count >= 10?
15C1: 90 10    >842           bcc   :nxdig     ; -No, do general case.
15C3: 86 D7    >843           stx   t1         ; -Yes, special case SLT >= 10.
15C5: 20 46 16 >844           jsr   exchAR     ; Exchange A and R magnitudes
15C8: A5 D7    >845           lda   t1         ; Recover count.
15CA: 38       >846           sec
15CB: E9 0A    >847           sbc   #10        ; Is count = 10?
15CD: F0 E6    >848           beq   :fetch     ; -Yes, done.
15CF: AA       >849           tax              ; -No, keep shifting.
15D0: A5 9F    >850           lda   rA+1       ; Hi magnitude digit.
15D2: 2A       >851           rol              ; High bit to C
15D3: A0 04    >852  :nxdig   ldy   #4         ; 4 bits/digit
15D5: A5 9F    >853  :nxbitt  lda   rA+1       ; To rotate rA, rR
15D7: 2A       >854           rol              ;  preset C to high bit.
15D8: 20 31 16 >855           jsr   slT        ; Rotate AR left 1 bit.
15DB: 88       >856           dey              ; More bits?
15DC: D0 F7    >857           bne   :nxbitt    ; -Yes.
15DE: CA       >858           dex              ; More digits?
15DF: D0 F2    >859           bne   :nxdig     ; -Yes.
15E1: F0 D2    >860           beq   :fetch     ; (always)
               >861
15E3: A0 04    >862  :sls     ldy   #4         ; SLS. 4 bits/digit
15E5: A5 9E    >863  :nxbit   lda   rA+S       ; Use sign digit
15E7: 29 0F    >864           and   #$0F       ;  and mask it.
15E9: C9 08    >865           cmp   #8         ; Hi bit of sign to C
15EB: 20 3B 16 >866           jsr   slA        ; Rotate A left 1 bit
15EE: A5 9E    >867           lda   rA+S       ;  then rotate sign.
15F0: 2A       >868           rol
15F1: 29 0F    >869           and   #$0F       ; Mask again
15F3: 85 9E    >870           sta   rA+S       ;  and put it back.
15F5: 88       >871           dey              ; More bits?
15F6: D0 ED    >872           bne   :nxbit     ; -Yes.
15F8: CA       >873           dex              ; More digits?
15F9: D0 E8    >874           bne   :sls       ; -Yes.
15FB: F0 B8    >875           beq   :fetch     ; (always)
```

```
               *********************************************************
               *                                                       *
               *            Utility Shifting Subroutines               *
               *                                                       *
               *********************************************************

                        align 256
15FD: 00 00 00          ds    *-1/256*256+256-*
                        eom
               ]keep    equ   */256      ; Keep here to 'kend' on one page.

1600: 66 9E    srAS     ror   rA         ; rA & sign right 1 bit
1602: 66 9F    srA      ror   rA+1       ; Sign not included
1604: 66 A0    srAM     ror   rA+2       ; FP mantissa
1606: 66 A1             ror   rA+3
1608: 66 A2             ror   rA+4
160A: 66 A3             ror   rA+5
160C: 60                rts

160D: 66 9F    srT      ror   rA+1       ; |rA| & |rR| right 1 bit
160F: 20 04 16 srAMR    jsr   srAM       ; Shift rA Mantissa & |rR|
1612: 66 A5    srR      ror   rR+1       ; Shift |rR|
1614: 66 A6             ror   rR+2
1616: 66 A7             ror   rR+3
1618: 66 A8             ror   rR+4
161A: 66 A9             ror   rR+5
161C: 60                rts

161D: A2 0A    srT2     ldx   #10        ; |rA| & |rR| right
161F: B5 9E    :shloop  lda   rA,x       ;  2 digits (1 byte).
1621: E0 05             cpx   #5         ; About to store in rR+S?
1623: D0 04             bne   :cont      ; -No, continue.
1625: 85 A5             sta   rR+1       ; -Yes, skip rR sign.
1627: F0 02             beq   :next      ;   and on to next byte.
1629: 95 9F    :cont    sta   rA+1,x
162B: CA       :next    dex
162C: D0 F1             bne   :shloop    ; Exclude rA sign.
162E: 86 9F             stx   rA+1       ; Shift in zeros.
1630: 60                rts

1631: 26 A9    slT      rol   rR+5       ; Rotate |rR| & |rA| left
1633: 26 A8             rol   rR+4       ;  one bit.
1635: 26 A7             rol   rR+3
1637: 26 A6             rol   rR+2
1639: 26 A5             rol   rR+1       ; Fall into slA.

163B: 26 A3    slA      rol   rA+5       ; Rotate |rA| left 1 bit
163D: 26 A2             rol   rA+4
163F: 26 A1             rol   rA+3
1641: 26 A0             rol   rA+2
1643: 26 9F             rol   rA+1
1645: 60                rts

1646: A2 05    exchAR   ldx   #5         ; Exchange |rA| and |rR|
1648: B5 9E    :exch    lda   rA,x       ; (equivalent to SLT 10)
164A: B4 A4             ldy   rR,x
164C: 95 A4             sta   rR,x
164E: 94 9E             sty   rA,x
1650: CA                dex
1651: D0 F5             bne   :exch
1653: 60                rts

               ]kend    equ   *-1/256    ; Warn if page crossing
                        err   ]kend-]keep ; between ]keep and ]kend.
```

```
                **********************************************************
                *                                                        *
                *         Split sL field into A = s and X = L            *
                *                                                        *
                **********************************************************

1654: A5 99     splitsL  lda   rC+sL      ; Get field specifier
1656: 29 0F              and   #$0F       ; L = digit count
1658: D0 02              bne   :notz
165A: A9 0A              lda   #10        ; "0" ==> 10
165C: AA        :notz    tax              ; X = digit count (L)
165D: A5 99              lda   rC+sL
165F: 4A                 lsr              ; Isolate field start s
1660: 4A                 lsr
1661: 4A                 lsr
1662: 4A                 lsr
1663: D0 02              bne   :ret
1665: A9 0A              lda   #10        ; "0" ==> 10
1667: 60        :ret     rts              ; A = start digit (s)

                **********************************************************
                *                                                        *
                *                   Clear Register                       *
                *                                                        *
                * At entry: X = Register address                         *
                * At exit:  A = 0, X = $FF                               *
                *                                                        *
                **********************************************************

1668: 8E 70 16  clear    stx   :clrloop+1 ; Save reg address
166B: A2 05              ldx   #5
166D: A9 00              lda   #0
166F: 95 00     :clrloop sta   0*0,x      ; Clear the register.
1671: CA                 dex
1672: 10 FB              bpl   :clrloop
1674: 60                 rts

                **********************************************************
                *                                                        *
                *           Extract NN from 3:2 field of rC              *
                *                                                        *
                * Returns: NN in BCD in 'NN' and Y, in binary in A,      *
                *          X unchanged.                                  *
                *                                                        *
                **********************************************************

1675: A5 99     midNN    lda   rC+sL      ; Extract NN from xN Nx.
1677: 0A                 asl              ; Return binary NN in A.
1678: 0A                 asl
1679: 0A                 asl
167A: 0A                 asl
167B: 85 D8              sta   NN         ; N0
167D: A5 9A              lda   rC+VV      ; Nx (low digit)
167F: 4A                 lsr
1680: 4A                 lsr
1681: 4A                 lsr
1682: 4A                 lsr              ; 0N
1683: 05 D8              ora   NN
1685: 85 D8              sta   NN         ; 'NN' = BCD NN
1687: A8                 tay
1688: B9 8C 16           lda   bcd2bin,y  ; A = binary NN.
168B: 60                 rts
```

```
                >1003 * Map 2-digit BCD 00..99 ==> Binary 100..99
                >1004
168C: 64 01 02 >1005 bcd2bin  db    100,1,2,3,4,5,6,7,8,9 ; BCD 00 ==> 100.
1696: 00 00 00 >1006          ds    6
169C: 0A 0B 0C >1007          db    10,11,12,13,14,15,16,17,18,19
16A6: 00 00 00 >1008          ds    6
16AC: 14 15 16 >1009          db    20,21,22,23,24,25,26,27,28,29
16B6: 00 00 00 >1010          ds    6
16BC: 1E 1F 20 >1011          db    30,31,32,33,34,35,36,37,38,39
16C6: 00 00 00 >1012          ds    6
16CC: 28 29 2A >1013          db    40,41,42,43,44,45,46,47,48,49
16D6: 00 00 00 >1014          ds    6
16DC: 32 33 34 >1015          db    50,51,52,53,54,55,56,57,58,59
16E6: 00 00 00 >1016          ds    6
16EC: 3C 3D 3E >1017          db    60,61,62,63,64,65,66,67,68,69
16F6: 00 00 00 >1018          ds    6
16FC: 46 47 48 >1019          db    70,71,72,73,74,75,76,77,78,79
1706: 00 00 00 >1020          ds    6
170C: 50 51 52 >1021          db    80,81,82,83,84,85,86,87,88,89
1716: 00 00 00 >1022          ds    6
171C: 5A 5B 5C >1023          db    90,91,92,93,94,95,96,97,98,99
                >1024
                >1025 * $00..$89 B220 character code to ASCII
                >1026
                >1027 b220asc  equ   *           ; B220 code to ASCII
1726: A0       >1028          db    $A0         ; $00 = Blank
1727: 00       >1029          ds    1           ; $01 skip
1728: 00       >1030          db    $00         ; $02 = Ignore
1729: AE A9    >1031          asc   ".)"        ; $03..$04
172B: 00 00 00 >1032          ds    11          ; $05..$0F skip
1736: A8       >1033          asc   "("         ; $10
1737: 00 00    >1034          ds    2           ; $11..$12 skip
1739: AB AA    >1035          asc   "+*"        ; $13..$14
173B: 8C       >1036          db    $8C         ; $15 = Eject
173C: 8D       >1037          db    $8D         ; $16 = CR
173D: 00 00 00 >1038          ds    3+6         ; $17..$1F skip
1746: AD AF    >1039          asc   "-/"        ; $20..$21
1748: 00       >1040          ds    1           ; $22 skip
1749: AC       >1041          asc   ","         ; $23
174A: A5       >1042          asc   "%"         ; $24 (For SNAP CR translation)
174B: 00       >1043          ds    1           ; $25 skip
174C: 89       >1044          db    $89         ; $26 = TAB
174D: A4       >1045          asc   "$"         ; $27
174E: 00 00 00 >1046          ds    2+6+2       ; $28..$31 skip
1758: BF BD A7 >1047          asc   "?='"       ; $32..$34
175B: 00 00 00 >1048          ds    5+6+1       ; $35..$40 skip
1767: C1 C2 C3 >1049          asc   "ABCDEFGHI" ; $41..$49
1770: 00 00 00 >1050          ds    6+1         ; $4A..$50 skip
1777: CA CB CC >1051          asc   "JKLMNOPQR" ; $51..$59
1780: 00 00 00 >1052          ds    6+2         ; $5A..$61 skip
1788: D3 D4 D5 >1053          asc   "STUVWXYZ"  ; $62..$69
1790: 00 00 00 >1054          ds    6+16        ; $6A..$7F skip
17A6: B0 B1 B2 >1055          asc   "0123456789" ; $80..$89
```

```
                        put   B220MT
               **********************************************************
               *                                                        *
               *                Mag Tape Instructions                   *
               *                                                        *
               **********************************************************

               blkcnt   equ   line2       ; Block count
               MxRflg   equ   line2+1     ; Flag for MxR op
               compsL   equ   line4       ; sL for compare
               compwd   equ   line4+1     ; Number of comparison word.
               ctlblk   equ   line4+1     ; 'Found ctl block' flag
               ltflag   equ   line8       ; Search found < block.
               mtcptr   equ   line8       ; ptr to preface of mtc block
               keyflg   equ   line8       ; >0 ==> processing key word
               wrdcnt   equ   line8+1     ; Binary word count.
               ctlflg   equ   linev+1     ; Read ctl blocks as normal

17B0: 88       MTS      dey               ; Y = $FF.
17B1: 84 DC             sty   ctlflg      ; Set 'stop on EOT block' flag.
17B3: A2 04             ldx   #MTUclass   ; Mag Tape class
17B5: 20 61 08          jsr   M_iosel     ; Select device.
17B8: 20 91 08          jsr   M_setlan    ; Set tape lane (0/1).
17BB: A5 9A             lda   rC+VV       ; Decode variant digit.
17BD: 29 04             and   #$04
17BF: D0 66             bne   :done       ; MLS = 4,5,6,7.
17C1: A5 9A             lda   rC+VV
17C3: 29 08             and   #$08
17C5: F0 06             beq   :mtsmfs     ; MTS/MFS = 0,1,2,3
17C7: 20 9D 08          jsr   M_resetd    ; MRW/MDA = 8,9.
17CA: 4C 27 18          jmp   :done

17CD: 85 E3    :mtsmfs  sta   ltflag      ; Clear '<' flag.
17CF: A5 98             lda   rC+S        ; MTS or MFS?
17D1: 29 04             and   #$04
17D3: F0 02             beq   :setsL      ; MTS "field" = 00
17D5: A5 94             lda   rB          ; MFS field = rB:82
17D7: 85 E1    :setsL   sta   compsL      ; Save sL for compare.
17D9: 20 79 08 :nxblk   jsr   M_getwrd    ; Read next word.
17DC: A5 AA             lda   rD+S        ; Isolate sign flag.
17DE: 29 F0             and   #$F0
17E0: C9 B0             cmp   #PREF       ; Block preface word?
17E2: D0 49             bne   ]IOerr3     ; -No, I/O error.
17E4: A5 AB             lda   rD+sL       ; -Yes, save preface
17E6: 85 E4             sta   wrdcnt      ;   word count.
17E8: 20 79 08          jsr   M_getwrd    ; rD = block key word.
17EB: A5 E4             lda   wrdcnt      ; Recover word count.
17ED: 25 DC             and   ctlflg      ; Mask with 'stop on  EOT'.
17EF: C9 01             cmp   #1          ; Is it an EOT block?
17F1: F0 29             beq   :finish     ; -Yes, finish.
17F3: A5 E1             lda   compsL      ; -No, MFS field = rB:82
17F5: 85 99             sta   rC+sL       ;  and fake it in rC.
17F7: A2 AA             ldx   #rD         ; Compare rD w/ search key.
17F9: A0 01             ldy   #1          ; Partial field
17FB: A9 B0             lda   #BCSop      ; Unsigned compare.
17FD: 20 32 0F          jsr   compare     ; Do the compare.
1800: A8                tay               ; A state (1,0,-1) to flags.
1801: F0 19             beq   :finish     ; Comparand = key.
1803: 10 0B             bpl   :grtr       ; Comparand > key.
1805: 85 E3             sta   ltflag      ; Comparand < key
1807: 20 A9 08          jsr   M_nxtblk    ; Advance to next block
180A: 88                dey               ; Y = $FF.
180B: 84 DC             sty   ctlflg      ; $FF = 'stop on EOT block'.
180D: 4C D9 17          jmp   :nxblk      ;  and continue search.

1810: A5 E3    :grtr    lda   ltflag      ; Have we seen < block?
1812: D0 08             bne   :finish     ; -Yes, this is the hit.
```

```
1814: 20 B5 08          jsr   M_prvblk    ; -No, back up 1 block
1817: 84 DC             sty   ctlflg      ; 0 = 'no stop on EOT block'.
1819: 4C D9 17          jmp   :nxblk      ;  and continue search.

181C: 38       :finish  sec               ; Back ptr up 2 words
181D: A5 D3             lda   ptr         ;  to preface of current block.
181F: E9 0C             sbc   #6*2
1821: 85 D3             sta   ptr
1823: B0 02             bcs   :done       ; No borrow.
1825: C6 D4             dec   ptr+1
1827: 20 6D 08 :done    jsr   M_iodsel    ; De-select device.
182A: 4C 52 09          jmp   fetch

182D: 4C 43 0A ]IOerr3  jmp   IOerr

1830: A5 9A    MTC      lda   rC+VV       ; Isolate word count.
1832: 29 0F             and   #$0F
1834: D0 02             bne   :nonzero    ; Word count of zero
1836: A9 0A             lda   #10         ;  means tenth word.
1838: 85 E2    :nonzero sta   compwd      ; Save word count.
183A: A5 98             lda   rC+S        ; MTC or MFC?
183C: 29 04             and   #$04
183E: F0 02             beq   :setsL      ; MTC "field" = 00
1840: A5 94             lda   rB          ; MFC field = rB:82
1842: 85 E1    :setsL   sta   compsL      ; Save sL for compare.
1844: A2 04             ldx   #MTUclass   ; Mag Tape class
1846: 20 61 08          jsr   M_iosel     ; Select device.
1849: 20 91 08          jsr   M_setlan    ; Set tape lane (0/1).
184C: A5 D3    :nxblk   lda   ptr         ; Save ptr to preface.
184E: 85 E3             sta   mtcptr
1850: A5 D4             lda   ptr+1
1852: 85 E4             sta   mtcptr+1
1854: 20 79 08          jsr   M_getwrd    ; Read preface word.
1857: A5 AA             lda   rD+S        ; Isolate sign flag.
1859: 29 F0             and   #$F0
185B: C9 B0             cmp   #PREF       ; Block preface word?
185D: D0 CE             bne   ]IOerr3     ; -No, I/O error.
185F: A5 AB             lda   rD+sL       ; Get block word count.
1861: C9 01             cmp   #1          ; Is it an EOT block?
1863: F0 39             beq   :finish     ; -Yes, finish.
1865: A0 00             ldy   #0          ; -No.
1867: B1 D3             lda   (ptr),y     ; Get next word's sign.
1869: C9 07             cmp   #07         ; Is this a control block?
186B: F0 31             beq   :finish     ; -Yes, regard as hit.
186D: C6 E2    :complp  dec   compwd      ; -No. Is comparand next word?
186F: F0 0D             beq   :comp       ; -Yes, compare.
1871: 18       :wrdlp   clc               ; -No, inc ptr to next word.
1872: A5 D3             lda   ptr
1874: 69 06             adc   #6
1876: 85 D3             sta   ptr
1878: 90 F3             bcc   :complp
187A: E6 D4             inc   ptr+1
187C: D0 EF             bne   :complp     ; (always)

187E: 20 79 08 :comp    jsr   M_getwrd    ; rD = comparand.
1881: A5 E3             lda   mtcptr      ; Restore ptr to
1883: 85 D3             sta   ptr         ;  block preface.
1885: A5 E4             lda   mtcptr+1
1887: 85 D4             sta   ptr+1
1889: A5 E1             lda   compsL      ; Get saved sL
188B: 85 99             sta   rC+sL       ;  and fake it in rC.
188D: A2 AA             ldx   #rD         ; Compare rD w/ scan key.
188F: A0 01             ldy   #1          ; Partial field
1891: A9 B0             lda   #BCSop      ; Unsigned compare.
1893: 20 32 0F          jsr   compare     ; Do the compare.
1896: F0 0E             beq   :done       ; -Block key = scan key.
1898: 20 A9 08          jsr   M_nxtblk    ; -Unequal, Adv to nxt block.
```

```
189B: 4C 4C 18           jmp   :nxblk     ;  and continue scan.

189E: A5 E3    :finish  lda   mtcptr     ; Restore ptr to
18A0: 85 D3             sta   ptr        ;  ctl block preface.
18A2: A5 E4             lda   mtcptr+1
18A4: 85 D4             sta   ptr+1
18A6: 20 6D 08 :done    jsr   M_iodsel   ; Deselect device.
18A9: 4C 52 09          jmp   fetch

18AC: C8       MRR      iny              ; Set MRR flag.
18AD: 84 E0    MRD      sty   MxRflg     ; 1 = MRR, 0 = MRD.
18AF: A5 9A             lda   rC+VV      ; Check variant digit.
18B1: 29 08             and   #$08       ; Isolate and save
18B3: 85 DB             sta   Bmodflg    ;  B-modificatiion flag.
18B5: A5 9A             lda   rC+VV
18B7: 29 01             and   #$01       ; Isolate and save
18B9: 85 DC             sta   ctlflg     ;  ctl blocks normal flag.
18BB: A5 99             lda   rC+sL
18BD: 29 0F             and   #$0F       ; Isolate and save
18BF: D0 02             bne   :stblkct   ;  block count.
18C1: A9 0A             lda   #10        ; Count = 0 ==> 10.
18C3: 85 DF    :stblkct sta   blkcnt
18C5: A2 04             ldx   #MTUclass  ; Mag Tape class.
18C7: 20 61 08          jsr   M_iosel    ; Select device.
18CA: 20 79 08 :blklp   jsr   M_getwrd   ; Preface word to rD.
18CD: A5 AA             lda   rD+S       ; Preface sign byte.
18CF: 29 F0             and   #$F0
18D1: C9 B0             cmp   #PREF      ; Is it flagged as preface?
18D3: D0 64             bne   :ioerr     ; -No, error!
18D5: A9 00             lda   #0         ; -Yes, proceed.
18D7: 85 E2             sta   ctlblk     ; Clear 'found ctl block'
18D9: A4 AB             ldy   rD+sL      ; Block word count (BCD)
18DB: 84 D8             sty   NN         ; Save it.
18DD: B9 8C 16          lda   bcd2bin,y  ; Convert it to binary
18E0: 85 E4             sta   wrdcnt     ;  and save it.
18E2: 85 E3             sta   keyflg     ; First data word is key word.
18E4: A5 E0             lda   MxRflg     ; MRR?
18E6: F0 09             beq   :ckeot     ; -No, don't store preface.
18E8: A5 AA             lda   rD+S       ; -Yes, clear the PREF flag
18EA: 29 0F             and   #$0F       ;  before storing.
18EC: 85 AA             sta   rD+S
18EE: 20 9C 19          jsr   strDinc    ; Store preface word for MRR.
18F1: A5 E4    :ckeot   lda   wrdcnt     ; Length = 1 ==> EOT.
18F3: C9 01             cmp   #1         ; End-Of-Tape block?
18F5: F0 45             beq   ]eot       ; -Yes, handle it.
18F7: 20 79 08 :wrdlp   jsr   M_getwrd   ; Get next data word.
18FA: A5 AA             lda   rD+S       ; Should this word
18FC: 25 DB             and   Bmodflg    ;  be B-modified?
18FE: F0 03             beq   :noBmod    ; -No.
1900: 20 B9 0B          jsr   BmodrD     ; -Yes, modify it.
1903: A5 DC    :noBmod  lda   ctlflg     ; Read ctl blocks?
1905: D0 16             bne   :store     ; -Yes, store it.
1907: A5 E2             lda   ctlblk     ; -No. Are we in
1909: C9 07             cmp   #$07       ;   a control block?
190B: D0 06             bne   :notctl    ; -No, continue.
190D: A5 E4             lda   wrdcnt     ; -Yes. Is this the final
190F: C9 01             cmp   #1         ;   (control) word)?
1911: F0 2C             beq   :ctlblk    ; -Yes, handle it.
1913: A5 E3    :notctl  lda   keyflg     ; -No, is this the key word?
1915: F0 06             beq   :store     ; -No, store it.
1917: A5 AA             lda   rD+S       ; -Yes, is this
1919: 29 0F             and   #$0F       ;   a control block?
191B: 85 E2             sta   ctlblk     ; Sign = 7 if control block.
191D: 20 9C 19 :store   jsr   strDinc    ; -No, store rD and advance.
1920: A9 00             lda   #0         ; Reset key word
1922: 85 E3             sta   keyflg     ;  (1st word) flag.
1924: C6 E4             dec   wrdcnt     ; More words in block?
```

```
1926: D0 CF              bne   :wrdlp     ; -Yes, continue.
1928: A5 D8              lda   NN         ; Full 100-word block?
192A: F0 03              beq   :noskip    ; -Yes, nothing to skip.
192C: 20 A9 08           jsr   M_nxtblk   ; -No, skip remaining words.
192F: C6 DF    :noskip   dec   blkcnt     ; More blocks?
1931: D0 97              bne   :blklp     ; -Yes, read next block.
1933: 20 6D 08           jsr   M_iodsel   ; -No, deselect device.
1936: 4C 52 09           jmp   fetch

1939: 4C 43 0A :ioerr    jmp   IOerr

193C: 20 79 08 ]eot      jsr   M_getwrd   ; rD = EOT control word.
193F: 20 4F 19 :ctlblk   jsr   doctlblk   ; Do control block.
1942: A5 D8              lda   NN         ; Full 100-word block?
1944: F0 03              beq   :nskip     ; -Yes, nothing to skip.
1946: 20 A9 08           jsr   M_nxtblk   ; -No, skip remaining words.
1949: 20 6D 08 :nskip    jsr   M_iodsel   ; Deselect device
194C: 4C 34 09           jmp   newP       ;  and branch to bbbb.

194F: A6 AD    doctlblk ldx   rD+OP      ; Process ctl word in rD.
1951: A4 AC              ldy   rD+VV      ; High 2 digits of aaaa
1953: C0 4A              cpy   #$49+1     ; ADDR error?
1955: B0 30              bcs   :addrerr   ; -Yes, error!
1957: BD E8 1A           lda   BCDLadrl,x ; -No, compute 'memptr'
195A: 79 1C 1C           adc   BCDHadrl,y
195D: 85 D1              sta   memptr     ; Low byte of mem address.
195F: BD 82 1B           lda   BCDLadrh,x
1962: 79 66 1C           adc   BCDHadrh,y
1965: B0 23              bcs   :undiger   ; Carry out ==> undigit(s)
1967: 85 D2              sta   memptr+1   ; High byte of 'memptr'
1969: A0 05              ldy   #ADDR+1    ; (memptr):04 = rP.
196B: A5 97              lda   rP+1
196D: 91 D1              sta   (memptr),y
196F: 88                 dey
1970: A5 96              lda   rP
1972: 91 D1              sta   (memptr),y
1974: 88                 dey              ; (memptr):64 = rC:04.
1975: A5 9D              lda   rC+ADDR+1
1977: 91 D1              sta   (memptr),y
1979: 88                 dey
197A: A5 9C              lda   rC+ADDR
197C: 91 D1              sta   (memptr),y
197E: A5 AE              lda   rD+ADDR    ; Put bbbb into rP.
1980: 85 96              sta   rP
1982: A5 AF              lda   rD+ADDR+1
1984: 85 97              sta   rP+1
1986: 60                 rts

1987: 4C 3F 0A :addrerr  jmp   ADDRerr
198A: 4C 49 0A :undiger  jmp   UNDIGerr

198D: A2 AA    resetran ldx   #rD        ; Set rD = 0 00 0000 0001
198F: 20 68 16           jsr   clear      ;  to simulate Reset-Transfer.
1992: A9 01              lda   #1
1994: 85 AF              sta   rD+5
1996: 20 4F 19           jsr   doctlblk   ; Store rC 0:4 & rP at
1999: 4C 34 09           jmp   newP       ;  0000 and branch to 0001.

199C: 20 CF 0B strDinc   jsr   storerD    ; (memptr) = rD, inc memptr.
199F: F8                 sed              ; / Increment rC:04 (BCD).
19A0: 18                 clc
19A1: A5 9D              lda   rC+ADDR+1
19A3: 69 01              adc   #1
19A5: 85 9D              sta   rC+ADDR+1
19A7: A5 9C              lda   rC+ADDR
19A9: 69 00              adc   #0
19AB: 85 9C              sta   rC+ADDR
```

```
19AD: D8                cld              ; \
19AE: 60                rts

19AF: C8       MIR      iny              ;
19B0: 84 E0    MIW      sty   MxRflg     ; 1 = MIR, 0 = MIW.
19B2: A5 99             lda   rC+sL
19B4: 29 0F             and   #$0F       ; Isolate the
19B6: D0 02             bne   :stblkct   ;  block count.
19B8: A9 0A             lda   #10        ; Count = 0 ==> 10.
19BA: 85 DF    :stblkct sta   blkcnt     ; Save block count.
19BC: A5 9A             lda   rC+VV      ; Word count (BCD)
19BE: 85 D8             sta   NN         ; Save word count.
19C0: A2 04             ldx   #MTUclass  ; Mag Tape class
19C2: 20 61 08          jsr   M_iosel    ; Select device.
19C5: C9 EE             cmp   #EMPTY     ; Is buffer empty?
19C7: D0 03             bne   :ckEOF     ; -No, are we at EOF?
19C9: 20 C1 08          jsr   M_readbf   ; -Yes, fill buffer.
19CC: C9 EF    :ckEOF   cmp   #EOF       ; Are we at EOF?
19CE: D0 27             bne   :ioerr     ; -No, I/O error!
19D0: A5 E0             lda   MxRflg     ; -Yes, MIR or MIW?
19D2: D0 26             bne   :mir       ; -MIR, skip making preface.
19D4: A2 B0             ldx   #rD10      ; -MIW, build preface
19D6: 20 68 16          jsr   clear      ;   word in rD10.
19D9: A5 D8             lda   NN         ; Set word count
19DB: 85 B1             sta   rD10+sL    ;  in 22 field
19DD: A9 B0             lda   #PREF      ;   and preface flag
19DF: 85 B0             sta   rD10+S     ;    in sign.
19E1: A5 E0    :blklp   lda   MxRflg     ; MIR or MIW?
19E3: D0 15             bne   :mir       ; -MIR.
19E5: A2 05             ldx   #5         ; -MIW, copy rD10 to rD.
19E7: B5 B0    :copylp  lda   rD10,x
19E9: 95 AA             sta   rD,x
19EB: CA                dex
19EC: 10 F9             bpl   :copylp
19EE: A6 D8             ldx   NN         ; Restore MIW
19F0: BD 8C 16          lda   bcd2bin,x  ;  binary
19F3: 85 E4             sta   wrdcnt     ;   word count.
19F5: D0 13             bne   :putpref   ; (always)

19F7: 4C 43 0A :ioerr   jmp   IOerr

19FA: 20 22 0C :mir     jsr   loadrD     ; Load preface from mem.
19FD: A5 AA             lda   rD+S       ;  Set 'preface' flag
19FF: 09 B0             ora   #PREF      ;   in sign byte,
1A01: 85 AA             sta   rD+S
1A03: A6 AB             ldx   rD+sL      ;    get the word count,
1A05: BD 8C 16          lda   bcd2bin,x  ;     convert to binary,
1A08: 85 E4             sta   wrdcnt     ;      and save it.
1A0A: 20 85 08 :putpref jsr   M_putwrd   ; Put preface word.
1A0D: 20 22 0C :wrdlp   jsr   loadrD     ; Data word to rD
1A10: 20 85 08          jsr   M_putwrd   ;  and put it.
1A13: C6 E4             dec   wrdcnt     ; More words in block?
1A15: D0 F6             bne   :wrdlp     ; -Yes, continue.
1A17: A5 D8             lda   NN         ; Full 100-word block?
1A19: F0 03             beq   :nskip     ; -Yes, nothing to skip.
1A1B: 20 A9 08          jsr   M_nxtblk   ; -No, skip remaining words.
1A1E: C6 DF    :nskip   dec   blkcnt     ; More blocks?
1A20: D0 BF             bne   :blklp     ; -Yes, continue.
1A22: A9 EF             lda   #EOF       ; -No, set EOF.
1A24: A0 00             ldy   #0
1A26: 8D 04 C0          sta   WRITMAIN
1A29: 91 D3             sta   (ptr),y
1A2B: 8D 05 C0          sta   WRITAUX
1A2E: 20 6D 08          jsr   M_iodsel   ; Deselect device.
1A31: 4C 52 09          jmp   fetch
```

```
1A34: C8       >334  MOR      iny
1A35: 84 E0    >335  MOW      sty   MxRflg     ; 1 = MOR, 0 = MOW.
1A37: A5 99    >336           lda   rC+sL
1A39: 29 0F    >337           and   #$0F       ; Isolate the
1A3B: D0 02    >338           bne   :stblkct   ;  block count.
1A3D: A9 0A    >339           lda   #10        ; Count = 0 ==> 10.
1A3F: 85 DF    >340  :stblkct sta   blkcnt     ; Save block count.
1A41: A5 9A    >341           lda   rC+VV      ; MOW word count (BCD)
1A43: 85 D8    >342           sta   NN         ; Save MOW word count.
1A45: A2 04    >343           ldx   #MTUclass  ; Mag Tape class
1A47: 20 61 08 >344           jsr   M_iosel    ; Select device.
1A4A: C9 EF    >345  :blklp   cmp   #EOF       ; Are we at end-of-file?
1A4C: F0 46    >346           beq   :ioerr     ; -Yes, I/O error!
1A4E: 20 79 08 >347           jsr   M_getwrd   ; -No, read preface.
1A51: A5 AA    >348           lda   rD+S       ; Preface flag/sign byte
1A53: 29 F0    >349           and   #$F0       ; Isolate flag.
1A55: C9 B0    >350           cmp   #PREF      ; Is this a preface?
1A57: D0 3B    >351           bne   :ioerr     ; -No, block sync error!
1A59: A5 E0    >352           lda   MxRflg     ; -Yes. MOR or MOW?
1A5B: F0 09    >353           beq   :mow       ; -MOW (use OP's NN)
1A5D: A0 01    >354           ldy   #sL        ; -MOR, compare with
1A5F: B1 D1    >355           lda   (memptr),y ;   memory preface.
1A61: 85 D8    >356           sta   NN         ; Save in NN.
1A63: 20 D9 0B >357           jsr   incmem     ; Advance memptr to data.
1A66: A5 D8    >358  :mow     lda   NN         ; Compare NN
1A68: C5 AB    >359           cmp   rD+sL      ;  with preface.
1A6A: F0 07    >360           beq   :ok        ; Length matches preface.
1A6C: C9 01    >361           cmp   #1         ; Mismatch!  Is it EOT?
1A6E: D0 24    >362           bne   :ioerr     ; -No, preface mismatch!
1A70: 4C 3C 19 >363           jmp   ]eot       ; -Yes, handle EOT record.
               >364
1A73: A8       >365  :ok      tay
1A74: B9 8C 16 >366           lda   bcd2bin,y  ; Convert NN to
1A77: 85 E4    >367           sta   wrdcnt     ;  binary word count.
1A79: 20 22 0C >368  :wrdlp   jsr   loadrD     ; Data word to rD
1A7C: 20 85 08 >369           jsr   M_putwrd   ;  and put it to file.
1A7F: C6 E4    >370           dec   wrdcnt     ; More words in block?
1A81: D0 F6    >371           bne   :wrdlp     ; -Yes, continue.
1A83: A5 D8    >372           lda   NN         ; Full 100-word block?
1A85: F0 03    >373           beq   :noskip    ; -Yes, don't skip rest.
1A87: 20 A9 08 >374           jsr   M_nxtblk   ; -No, skip to next block.
1A8A: C6 DF    >375  :noskip  dec   blkcnt     ; More blocks?
1A8C: D0 BC    >376           bne   :blklp     ; -Yes, continue.
1A8E: 20 6D 08 >377           jsr   M_iodsel   ; Deselect device.
1A91: 4C 52 09 >378           jmp   fetch
               >379
1A94: 4C 43 0A >380  :ioerr   jmp   IOerr
```

```
1A97: A5 99    >382  MPF      lda   rC+sL      ; Get block count.
1A99: 29 0F    >383           and   #$0F
1A9B: D0 02    >384           bne   :setblk
1A9D: A9 0A    >385           lda   #10        ; '0' ==> 10.
1A9F: 85 DF    >386  :setblk  sta   blkcnt     ; Save block count.
1AA1: A2 04    >387           ldx   #MTUclass  ; Mag Tape class
1AA3: 20 61 08 >388           jsr   M_iosel    ; Select the device.
1AA6: A8       >389           tay              ; Save next flag byte.
1AA7: A5 9A    >390           lda   rC+VV      ; MPF, MPB, oe MPE?
1AA9: 29 0F    >391           and   #$0F       ; Isolate variant digit.
1AAB: C9 01    >392           cmp   #1
1AAD: F0 11    >393           beq   :mpb       ; Mag tape Position Backward.
1AAF: C9 02    >394           cmp   #2
1AB1: F0 16    >395           beq   :mpe       ; Mag tape Position at End.
1AB3: 20 A9 08 >396  :mpf     jsr   M_nxtblk   ; MPF, advance to next block.
1AB6: C6 DF    >397           dec   blkcnt     ; More blocks to skip?
1AB8: D0 F9    >398           bne   :mpf       ; -Yes, keep going.
1ABA: 20 6D 08 >399  :done    jsr   M_iodsel   ; -No, deselect the device.
1ABD: 4C 52 09 >400           jmp   fetch
               >401
1AC0: 20 B5 08 >402  :mpb     jsr   M_prvblk   ; Position to previous block.
1AC3: C6 DF    >403           dec   blkcnt     ; More blocks to skip?
1AC5: D0 F9    >404           bne   :mpb       ; -Yes, continue.
1AC7: F0 F1    >405           beq   :done      ; -No, done. (always)
               >406
1AC9: 98       >407  :mpe     tya              ; Recover next flag byte.
1ACA: C9 EF    >408  :mpelp   cmp   #EOF       ; At End-Of-File?
1ACC: F0 EC    >409           beq   :done      ; -Yes, done!
1ACE: 20 A9 08 >410           jsr   M_nxtblk   ; -No, adv to next block
1AD1: 4C CA 1A >411           jmp   :mpelp     ;       and check for EOF.
               >412
1AD4: A5 9A    >413  MIB      lda   rC+VV      ; MIB or MIE
1AD6: 29 0F    >414           and   #$0F       ; Isolate variant digit.
1AD8: C9 01    >415           cmp   #1         ; Is it MIE?
1ADA: D0 03    >416           bne   :mib       ; -No, it's an MIB.
1ADC: 4C 52 09 >417  :nop     jmp   fetch      ; -Yes, MIE = NOP.
1ADF: A5 99    >418  :mib     lda   rC+sL
1AE1: 29 E0    >419           and   #$E0       ; Is unit = 0 or 1?
1AE3: D0 F7    >420           bne   :nop       ; -No, so it's a NOP.
1AE5: 4C E1 13 >421           jmp   BUN        ; -Yes, so it's a BUN.
```

```
                  81            put   B220BCDTBL
                 >1     **********************************************************
                 >2     *                                                        *
                 >3     *      4-digit BCD to binary word address tables         *
                 >4     *                                                        *
                 >5     **********************************************************
                 >6
                 >7     BCDLadrl equ   *          ; BCD lo 2 dig --> addr lo byte
1AE8: D0 D6 DC >8               db    208,214,220,226,232,238,244,250,0,6
1AF2: 00 00 00 >9               db    0,0,0,0,0,0
1AF8: 0C 12 18 >10              db    12,18,24,30,36,42,48,54,60,66
1B02: 00 00 00 >11              db    0,0,0,0,0,0
1B08: 48 4E 54 >12              db    72,78,84,90,96,102,108,114,120,126
1B12: 00 00 00 >13              db    0,0,0,0,0,0
1B18: 84 8A 90 >14              db    132,138,144,150,156,162,168,174,180,186
1B22: 00 00 00 >15              db    0,0,0,0,0,0
1B28: C0 C6 CC >16              db    192,198,204,210,216,222,228,234,240,246
1B32: 00 00 00 >17              db    0,0,0,0,0,0
1B38: FC 02 08 >18              db    252,2,8,14,20,26,32,38,44,50
1B42: 00 00 00 >19              db    0,0,0,0,0,0
1B48: 38 3E 44 >20              db    56,62,68,74,80,86,92,98,104,110
1B52: 00 00 00 >21              db    0,0,0,0,0,0
1B58: 74 7A 80 >22              db    116,122,128,134,140,146,152,158,164,170
1B62: 00 00 00 >23              db    0,0,0,0,0,0
1B68: B0 B6 BC >24              db    176,182,188,194,200,206,212,218,224,230
1B72: 00 00 00 >25              db    0,0,0,0,0,0
1B78: EC F2 F8 >26              db    236,242,248,254,4,10,16,22,28,34
                 >27
                 >28    BCDLadrh equ   *          ; BCD lo 2 dig --> addr hi byte
1B82: 4A 4A 4A >29              db    74,74,74,74,74,74,74,74,75,75
1B8C: FF FF FF >30              db    255,255,255,255,255,255
1B92: 4B 4B 4B >31              db    75,75,75,75,75,75,75,75,75,75
1B9C: FF FF FF >32              db    255,255,255,255,255,255
1BA2: 4B 4B 4B >33              db    75,75,75,75,75,75,75,75,75,75
1BAC: FF FF FF >34              db    255,255,255,255,255,255
1BB2: 4B 4B 4B >35              db    75,75,75,75,75,75,75,75,75,75
1BBC: FF FF FF >36              db    255,255,255,255,255,255
1BC2: 4B 4B 4B >37              db    75,75,75,75,75,75,75,75,75,75
1BCC: FF FF FF >38              db    255,255,255,255,255,255
1BD2: 4B 4C 4C >39              db    75,76,76,76,76,76,76,76,76,76
1BDC: FF FF FF >40              db    255,255,255,255,255,255
1BE2: 4C 4C 4C >41              db    76,76,76,76,76,76,76,76,76,76
1BEC: FF FF FF >42              db    255,255,255,255,255,255
1BF2: 4C 4C 4C >43              db    76,76,76,76,76,76,76,76,76,76
1BFC: FF FF FF >44              db    255,255,255,255,255,255
1C02: 4C 4C 4C >45              db    76,76,76,76,76,76,76,76,76,76
1C0C: FF FF FF >46              db    255,255,255,255,255,255
1C12: 4C 4C 4C >47              db    76,76,76,76,77,77,77,77,77,77
                 >48
                 >49    BCDHadrl equ   *          ; BCD Hi 2 dig --> bin lo byte
1C1C: 00 58 B0 >50              db    0,88,176,8,96,184,16,104,192,24
1C26: 00 00 00 >51              db    0,0,0,0,0,0
1C2C: 70 C8 20 >52              db    112,200,32,120,208,40,128,216,48,136
1C36: 00 00 00 >53              db    0,0,0,0,0,0
1C3C: E0 38 90 >54              db    224,56,144,232,64,152,240,72,160,248
1C46: 00 00 00 >55              db    0,0,0,0,0,0
1C4C: 50 A8 00 >56              db    80,168,0,88,176,8,96,184,16,104
1C56: 00 00 00 >57              db    0,0,0,0,0,0
1C5C: C0 18 70 >58              db    192,24,112,200,32,120,208,40,128,216
                 >59
                 >60    BCDHadrh equ   *          ; BCD Hi 2 dig --> bin Hi byte
1C66: 00 02 04 >61              db    0,2,4,7,9,11,14,16,18,21
1C70: FF FF FF >62              db    255,255,255,255,255,255
1C76: 17 19 1C >63              db    23,25,28,30,32,35,37,39,42,44
1C80: FF FF FF >64              db    255,255,255,255,255,255
1C86: 2E 31 33 >65              db    46,49,51,53,56,58,60,63,65,67
1C90: FF FF FF >66              db    255,255,255,255,255,255
```

```
                              ===== Page 128 =====

1C96: 46 48 4B >67            db    70,72,75,77,79,82,84,86,89,91
1CA0: FF FF FF >68            db    255,255,255,255,255,255
1CA6: 5D 60 62 >69            db    93,96,98,100,103,105,107,110,112,114
               >70
               >71   simend   equ   *-1         ; End of B220SIM code
               >72            err   simend/MEM ; Can't encroach on MEM area.
               >73
                82
                83            org               ; Reestablish code offset
                84   AUXend   equ   *           ; End of Aux code
                85            err   */$9600     ; Total code limit.
                86   freemain equ   ptrdr0bf-MAINend ; Free space in main mem.
                87   freeaux  equ   MEM-AUXend        ; Free space in aux mem.


--End assembly, 11400 bytes, Errors: 0


Symbol table - alphabetical order:

   ADCYop  =$79        ADCZop  =$65        ADD     =$0D1C      ADDR    =$04
   ADDRerr =$0A3F      ADDRerrR=$0928      ADL     =$0D8C      ALTCHAR =$C00F
   AR1     =$0700      AR2     =$0680      AR4     =$0600      AR8     =$0580
   ARbord  =$10E1      ARmid   =$1107      ARv     =$0428      AUXcode =$2100
   AUXend  =$3488      AUXrts  =$0921      Aattr   =$0FA7      Acol    =$05
   Ain     =$0C33      Alab    =$0583      Aparm   =$182D    ? B220SIM =$0800
   B220col =$0C        B220end =$CE        B220msg =$10CC      B220strt=$90
   BASCALC =$FBC1      BASL    =$28        BCDHadrh=$1C66      BCDHadrl=$1C1C
   BCDLadrh=$1B82      BCDLadrl=$1AE8      BCE     =$13CF      BCH     =$13BB
   BCS     =$1447      BCSop   =$B0        BEEP    =$FBDD      BFA     =$13F8
   BFR     =$13F4      BITZop  =$24        BNEop   =$D0        BOF     =$139E
   BPC1    =$0728      BPC2    =$06A8      BPC4    =$0628      BPC8    =$05A8
   BPCbord =$112D      BPCmid  =$1153      BPCv    =$0450      BPLop   =$10
   BRP     =$13AB      BSA     =$13B1      BSSTATE =$BE42      BUN     =$13E1
   Battr   =$0FD7      Bcol    =$05        Bin     =$0C37      Blab    =$05AB
   Bmodflg =$DB        BmodrD  =$0BB9      Bparm   =$1835      CAA     =$0D15
   CAD     =$0CC6      CASSOUT =$C020      CFA     =$0F12      CH      =$24
   CLA     =$152B      CLCop   =$18        CLL     =$154C      CMPIop  =$C9
   COMP    =$C2        COMPcol =$19        COUT    =$FDED      CROUT   =$FD8E
   CSU     =$0CB1      CSW     =$B6        Cattr   =$0FF7      Ccol    =$15
   Cin     =$0C3B      Clab    =$05BB      DBB     =$138B      DFL     =$1256
   DIV     =$0E4D      DLB     =$1266      DOSCMD  =$BE03      DOSCON  =$03D0
   EMPTY   =$EE        EOB     =$EB        EOF     =$EF        ERR     =$C1
   ERRcol  =$15        ERRlab  =$0567      EXP     =$01        EXT     =$0EEA
   Eparm   =$1831      FAD     =$0FD0      FDV     =$118D      FIELDerr=$0A3B
   FMU     =$10F2      FSU     =$10DD      GBASH   =$27        GBASL   =$26
?  HGR2    =$F3D8      HGRinit =$1A4D      HIRES   =$C056      HLT     =$0B10
   HMASK   =$30        HNDX    =$E5        HOME    =$FC58      HPAG    =$E6
   HPOSN   =$F411      Help1   =$11A0    ? Help2   =$11C5    ? Help3   =$11EB
?  Help4   =$120E      IBB     =$1378      IFL     =$1210      IN      =$0200
   INDshow =$12DD      IOerr   =$0A43      KAD     =$0A40      KBD     =$C000
   KBSTROBE=$C010      LDAIop  =$A9        LDB     =$14E7      LDR     =$14DB
   LSA     =$150D      MAINend =$3880      MANT    =$02        MEM     =$4AD0
   MIB     =$1AD4      MIR     =$19AF      MIW     =$19B0      MIXED   =$C052
   MOR     =$1A34      MOW     =$1A35      MPF     =$1A97      MRD     =$18AD
   MRR     =$18AC      MTC     =$1830      MTS     =$17B0      MTUclass=$04
   MUL     =$0DC4      M_COUT  =$0909      M_PRBL2 =$0915      M_ckspo =$08CD
   M_disp  =$0855      M_getwrd=$0879      M_iodsel=$086D      M_iosel =$0861
   M_keyin =$0843      M_lpread=$08F1      M_nxtblk=$08A9      M_plot  =$08E5
   M_prvblk=$08B5      M_putwrd=$0885      M_readbf=$08C1      M_resetd=$089D
   M_setlan=$0891      M_stop  =$084C      M_trace =$08D9      M_xdrawc=$08FD
   MxRflg  =$E0        NN      =$D8        NOP     =$0B10      NOPop   =$EA
   OFF     =$00        OFLcol  =$1F        OFLerr  =$0A37      ON      =$01
   OP      =$03        OPerr   =$0A33      Ov      =$C3        OvHlt   =$CE
   PAGE2   =$C054      PB0     =$C061      PB1     =$C062      PDL     =$C064
   PDebx   =$17F1      PDfae   =$17B9      PRB     =$0B57      PRBL2   =$F94A
   PRBYTE  =$FDDA      PRD     =$0B13      PREF    =$B0        PRHEX   =$FDE3
   PRI     =$0BE5      PRNTAX  =$F941      PTPclass=$02        PTRIG   =$C070
   PTRclass=$00        PWI     =$0C2F      PWR     =$0BE8      Pattr   =$0FE7
   Pcol    =$0D        Pin     =$0C3F      Plab    =$05B3      READAUX =$C003
   READMAIN=$C002   ?  RESTART =$0803      RND     =$0EC8      RPTcol  =$22
   RTF     =$1312      RUN     =$C0        RUNcol  =$11        Rattr   =$0FBF
   Rcol    =$17        Rin     =$0C43      Rlab    =$0595      Rp      =$C4
   S       =$00        SBCYop  =$F9        SBCZop  =$E5        SECop   =$38
   SLA     =$158C      SOR     =$1454      SPKR    =$C030      SPO     =$0C32
   SRA     =$1557      STA     =$1468      STAT    =$1179      STATlin =$0550
   STP     =$1516      SUB     =$0DAE      SW1col  =$06        SWlab   =$0553
   TEXT    =$C050      UNDIGerR=$092B      UNDIGerr=$0A49      VIEWend =$3880
   VIEWhlp1=$0A50      VIEWhlp2=$0AD0      VIEWhlp3=$0B50      VIEWhlp4=$0BD0
   VV      =$02        WNDTOP  =$22        WRITAUX =$C005      WRITMAIN=$C004
   X_IOerr =$0821      X_cont  =$0818      X_fetch =$0806      X_incP  =$0833
   X_newP  =$080F      X_resetr=$082A   V  ]IOerr1 =$1556   V  ]IOerr2 =$1776
V  ]IOerr3 =$182D   V? ]Ov     =$1307   V? ]X_sound=$0999   V? ]adc    =$12BE
V  ]add    =$0D2E   V  ]analyze=$0A91   V? ]bfr    =$13FA   V  ]ckstop =$0A3C
V? ]clc    =$12AF   V? ]cmp    =$12C0   V? ]contin =$09B1   V? ]dfl    =$127B
V? ]done   =$19F4   V  ]eot    =$193C   V  ]err    =$0A4B   V  ]errpt  =$1263
V  ]fad    =$0FDE   V  ]fetch1 =$1528   V  ]fetch2 =$14BE   V? ]fetch3 =$13F1
V  ]fetch4 =$1253   V? ]finish =$0F77   MV ]hd     =$05     V  ]incptr6=$1534
V  ]kbdloop=$0E3F   V  ]keep   =$16     V  ]kend   =$16     V  ]loadrA =$0CD6
V? ]nop    =$12E8   V? ]prd    =$0B6B   V  ]resptr =$1667   V? ]restore=$09AD
V? ]rts    =$15A4   V  ]stop   =$0A40   V? ]sub    =$12EB   V  ]waitkey=$0A82
   advoff  =$164F   MD align   =$8000   MD auxjmp  =$8000   MD auxjsr  =$8000
   b220asc =$1726      b220plot=$1B40      back2sim=$09B1      backoff =$162F
   bcd2bin =$168C      bcdxy   =$DF        beepget =$0C6B      bfclasch=$137F
   bfdirty =$1386      bfend   =$1379      bffn    =$1381      bflane  =$1385
   bfoff   =$1382      bfptr   =$1377      bfscrn  =$137D      bfsiz   =$137B
   bfstart =$1375      bfunitch=$1380      blanklin=$109A      blkcnt  =$DF
   blksize =$025E      bload   =$181F      bsave   =$1826      changed =$E1
   charset =$0D7A      ckpref  =$15E3      ckspo   =$169C      classdbx=$13E1
   clear   =$1668      clearAR =$1182      common  =$0800      compare =$0F32
   compsL  =$E1        compwd  =$E2        crtkey  =$E5        ctlblk  =$E2
   ctlflg  =$DC        cursor  =$57        db      =$1375      dbsz    =$12
   dbx     =$D9        decblk  =$161D      delete  =$FF        delta   =$1D92
   disARmid=$10A2      disBPCbo=$10B0      disBPCmi=$10BE      disiocfg=$0DD8
   disopts =$0F7F      dispA   =$1256      dispB   =$1264      dispC   =$1272
   dispP   =$126B      dispR   =$125D      dispSTAT=$1279      dispcnt =$64
   dispctr =$DA        dispdig =$1344      disphelp=$1233      display =$1244
   disppanl=$1011      dispreg =$1306      divide  =$0E53      dnarrow =$8A
   doctlblk=$194F      doread  =$1779      dowrite =$1734      ediocfg =$0DCC
   emptydb =$16D2      endcomm =$0928      escape  =$9B        exchAR  =$1646
   execute =$0979      fetch   =$0952   MD fkey    =$8000      fktbl   =$1E80
   flushall=$1710      flushbuf=$1723      fnamecol=$0C        fnames  =$1400
   fnlen   =$19        fnx     =$DB        fnxdbx  =$13E7      fnxfn   =$13EF
?  freeaux =$1648   ?  freemain=$02CC      getdig  =$0C6E      getwrd  =$1520
   hgrx    =$E1        hgry    =$E2        histx   =$1F00      histy   =$2000
   hx      =$1D90      incP    =$0A17      incblk  =$15D1      incmem  =$0BD9
   init    =$0928      initstk =$09CA      inptr   =$D5        instctr =$C7
   instptr =$CF        intabl  =$0C33      invalid =$37        inverse =$0EFD
   iocfgstr=$0C85      iocfgtt =$0B        iodsel  =$14C8      iosel   =$14E5
   kbhelp1 =$1CF0      kbhelp2 =$1D18      kbhelp3 =$1D40      kbhelp4 =$1D68
   kbmode  =$CD        kbmodeon=$1958      kbserve =$192F   MD key     =$8000
   keyflg  =$E3        keyin   =$0A21      keyinR  =$092E      keytbl  =$1E00
   lc      =$00        line    =$E0        line1   =$DD        line2   =$DF
   line4   =$E1        line8   =$E3        linev   =$DB        loadrD  =$0C22
   lpen    =$CC        lpread  =$1C70      ltarrow =$88        ltflag  =$E3
MD mainjmp =$8000   MD mainjsr =$8000      memb    =$7530      memptr  =$D1
   midNN   =$1675      mt0bf   =$64B4      mt1bf   =$7C62      mtbfsz  =$17AC
   mtcptr  =$E3        multiply=$0DCA      ndb     =$06        newP    =$0934
   newcset =$0DBD      newp    =$C5        newsnd  =$0DC4      noAD    =$8000
   nokey   =$AF        nxtblk  =$15A5      off     =$A0        on      =$AA
   operr   =$8A33      optabh  =$0AB6      optabl  =$0A5C      optline1=$12
   optlines=$0D6D   ?  pdoscmd =$1867      pdosxeq =$186C      prind   =$190C
   print1  =$18F2      print2  =$1906      print2bl=$1927      print2c =$18FB
   print6  =$1916      printbl =$192A      printsgn=$18E9      prtrace =$1884
```

```
                              ===== Page 130 =====

   prvblk  =$15F9      ptbfsz  =$0258      ptpch0bf=$6000      ptpch1bf=$625A
   ptr     =$D3        ptrdr0bf=$3B4C      ptrdr1bf=$3DA6   MD putat   =$8000
   putbyte =$183D      putpdcmd=$1855      putwdhx =$1839      putwrd  =$1559
   px      =$1D94      py      =$1D95      rA      =$9E        rB      =$94
   rBx     =$90        rC      =$98        rD      =$AA        rD10    =$B0
   rP      =$96        rR      =$A4        readbuf =$159A      reset   =$0994
   resetdb =$16C4      resetdbs=$16B3      resetran=$198D   MD resi    =$8000
   restart =$09AA      rtmargin=$04        sL      =$01        savex   =$DE
   scale   =$1D91      scx     =$DD        scy     =$DE        selBASL =$E3
   selch   =$DF        selected=$DC        selsave =$DD     MD seti    =$8000
   setlan  =$1672      setptr  =$150F      shleft1 =$0C47      showhelp=$1976
   signtbl =$0F02      simend  =$1CAF      skipincP=$C6        slA     =$163B
   slT     =$1631      sndport =$0DA2      splitsL =$1654      srA     =$1602
   srAM    =$1604      srAMR   =$160F      srAS    =$1600   ?  srR     =$1612
   srT     =$160D      srT2    =$161D      stopR   =$0931      storerD =$0BCF
   strDinc =$199C      swaphelp=$1CB1      swapzp  =$09BA      t1      =$D7
   tabs    =$0CAC      togcset =$0F0A      togsound=$0F4D      traceflg=$CA
   uc      =$08        uparrow =$8B        viewkey =$198C      viewmode=$CB
   viewoff =$197C      viewon  =$1964      wrdcnt  =$E4        xbit    =$3600
   xbyte   =$3500      xdrawcur=$1BF2      xeqflg  =$DC        xl      =$1D98
   xlyl    =$1D00      xmap    =$2100      xx      =$1D96      xyinit  =$1A93
   y       =$DD        ybaseh  =$37C0      ybasel  =$3700      yl      =$1D9A
   ymap    =$2B00      yy      =$1D97      zeroff  =$DD        zpend   =$E6
   zpsave  =$09CB


Symbol table - numerical order:

   OFF     =$00        S       =$00        lc      =$00        PTRclass=$00
   ON      =$01        sL      =$01        EXP     =$01        VV      =$02
   MANT    =$02        PTPclass=$02        OP      =$03        ADDR    =$04
   rtmargin=$04        MTUclass=$04        Acol    =$05        Bcol    =$05
MV ]hd     =$05        ndb     =$06        SW1col  =$06        uc      =$08
   iocfgtt =$0B        fnamecol=$0C        B220col =$0C        Pcol    =$0D
   BPLop   =$10        RUNcol  =$11        optline1=$12        dbsz    =$12
   Ccol    =$15        ERRcol  =$15     V  ]keep   =$16     V  ]kend   =$16
   Rcol    =$17        CLCop   =$18        COMPcol =$19        fnlen   =$19
   OFLcol  =$1F        WNDTOP  =$22        RPTcol  =$22        BITZop  =$24
   CH      =$24        GBASL   =$26        GBASH   =$27        BASL    =$28
   HMASK   =$30        invalid =$37        SECop   =$38        cursor  =$57
   dispcnt =$64        ADCZop  =$65        ADCYop  =$79        ltarrow =$88
   dnarrow =$8A        uparrow =$8B        B220strt=$90        rBx     =$90
   rB      =$94        rP      =$96        rC      =$98        escape  =$9B
   rA      =$9E        off     =$A0        rR      =$A4        LDAIop  =$A9
   rD      =$AA        on      =$AA        nokey   =$AF        BCSop   =$B0
   PREF    =$B0        rD10    =$B0        CSW     =$B6        RUN     =$C0
   ERR     =$C1        COMP    =$C2        Ov      =$C3        Rp      =$C4
   newp    =$C5        skipincP=$C6        instctr =$C7        CMPIop  =$C9
   traceflg=$CA        viewmode=$CB        lpen    =$CC        kbmode  =$CD
   B220end =$CE        OvHlt   =$CE        instptr =$CF        BNEop   =$D0
   memptr  =$D1        ptr     =$D3        inptr   =$D5        t1      =$D7
   NN      =$D8        dbx     =$D9        dispctr =$DA        linev   =$DB
   fnx     =$DB        Bmodflg =$DB        selected=$DC        xeqflg  =$DC
   ctlflg  =$DC        line1   =$DD        selsave =$DD        zeroff  =$DD
   y       =$DD        scx     =$DD        savex   =$DE        scy     =$DE
   line2   =$DF        selch   =$DF        bcdxy   =$DF        blkcnt  =$DF
   line    =$E0        MxRflg  =$E0        line4   =$E1        changed =$E1
   hgrx    =$E1        compsL  =$E1        hgry    =$E2        compwd  =$E2
   ctlblk  =$E2        line8   =$E3        selBASL =$E3        ltflag  =$E3
   mtcptr  =$E3        keyflg  =$E3        wrdcnt  =$E4        SBCZop  =$E5
   crtkey  =$E5        HNDX    =$E5        zpend   =$E6        HPAG    =$E6
   NOPop   =$EA        EOB     =$EB        EMPTY   =$EE        EOF     =$EF
   SBCYop  =$F9        delete  =$FF        IN      =$0200      ptbfsz  =$0258
   blksize =$025E   ?  freemain=$02CC      DOSCON  =$03D0      ARv     =$0428
   BPCv    =$0450      STATlin =$0550      SWlab   =$0553      ERRlab  =$0567
   AR8     =$0580      Alab    =$0583      Rlab    =$0595      BPC8    =$05A8
   Blab    =$05AB      Plab    =$05B3      Clab    =$05BB      AR4     =$0600
```

```
                              ===== Page 131 =====

   BPC4    =$0628      AR2     =$0680      BPC2    =$06A8      AR1     =$0700
   BPC1    =$0728      common  =$0800   ?  B220SIM =$0800   ?  RESTART =$0803
   X_fetch =$0806      X_newP  =$080F      X_cont  =$0818      X_IOerr =$0821
   X_resetr=$082A      X_incP  =$0833      M_keyin =$0843      M_stop  =$084C
   M_disp  =$0855      M_iosel =$0861      M_iodsel=$086D      M_getwrd=$0879
   M_putwrd=$0885      M_setlan=$0891      M_resetd=$089D      M_nxtblk=$08A9
   M_prvblk=$08B5      M_readbf=$08C1      M_ckspo =$08CD      M_trace =$08D9
   M_plot  =$08E5      M_lpread=$08F1      M_xdrawc=$08FD      M_COUT  =$0909
   M_PRBL2 =$0915      AUXrts  =$0921      endcomm =$0928      init    =$0928
   ADDRerrR=$0928      UNDIGerR=$092B      keyinR  =$092E      stopR   =$0931
   newP    =$0934      fetch   =$0952      execute =$0979      reset   =$0994
V? ]X_sound=$0999      restart =$09AA   V? ]restore=$09AD      back2sim=$09B1
V? ]contin =$09B1      swapzp  =$09BA      initstk =$09CA      zpsave  =$09CB
   incP    =$0A17      keyin   =$0A21      OPerr   =$0A33      OFLerr  =$0A37
   FIELDerr=$0A3B   V  ]ckstop =$0A3C      ADDRerr =$0A3F   V  ]stop   =$0A40
   KAD     =$0A40      IOerr   =$0A43      UNDIGerr=$0A49   V  ]err    =$0A4B
   VIEWhlp1=$0A50      optabl  =$0A5C   V  ]waitkey=$0A82   V  ]analyze=$0A91
   optabh  =$0AB6      VIEWhlp2=$0AD0      HLT     =$0B10      NOP     =$0B10
   PRD     =$0B13      VIEWhlp3=$0B50      PRB     =$0B57   V? ]prd    =$0B6B
   BmodrD  =$0BB9      storerD =$0BCF      VIEWhlp4=$0BD0      incmem  =$0BD9
   PRI     =$0BE5      PWR     =$0BE8      loadrD  =$0C22      PWI     =$0C2F
   SPO     =$0C32      intabl  =$0C33      Ain     =$0C33      Bin     =$0C37
   Cin     =$0C3B      Pin     =$0C3F      Rin     =$0C43      shleft1 =$0C47
   beepget =$0C6B      getdig  =$0C6E      iocfgstr=$0C85      tabs    =$0CAC
   CSU     =$0CB1      CAD     =$0CC6   V  ]loadrA =$0CD6      CAA     =$0D15
   ADD     =$0D1C   V  ]add    =$0D2E      optlines=$0D6D      charset =$0D7A
   ADL     =$0D8C      sndport =$0DA2      SUB     =$0DAE      newcset =$0DBD
   newsnd  =$0DC4      MUL     =$0DC4      multiply=$0DCA      ediocfg =$0DCC
   disiocfg=$0DD8   V  ]kbdloop=$0E3F      DIV     =$0E4D      divide  =$0E53
   RND     =$0EC8      EXT     =$0EEA      inverse =$0EFD      signtbl =$0F02
   togcset =$0F0A      CFA     =$0F12      compare =$0F32      togsound=$0F4D
V? ]finish =$0F77      disopts =$0F7F      Aattr   =$0FA7      Rattr   =$0FBF
   FAD     =$0FD0      Battr   =$0FD7   V  ]fad    =$0FDE      Pattr   =$0FE7
   Cattr   =$0FF7      disppanl=$1011      blanklin=$109A      disARmid=$10A2
   disBPCbo=$10B0      disBPCmi=$10BE      B220msg =$10CC      FSU     =$10DD
   ARbord  =$10E1      FMU     =$10F2      ARmid   =$1107      BPCbord =$112D
   BPCmid  =$1153      STAT    =$1179      clearAR =$1182      FDV     =$118D
   Help1   =$11A0   ?  Help2   =$11C5   ?  Help3   =$11EB   ?  Help4   =$120E
   IFL     =$1210      disphelp=$1233      display =$1244   V  ]fetch4 =$1253
   dispA   =$1256      DFL     =$1256      dispR   =$125D   V  ]errpt  =$1263
   dispB   =$1264      DLB     =$1266      dispP   =$126B      dispC   =$1272
   dispSTAT=$1279   V? ]dfl    =$127B   V? ]clc    =$12AF   V? ]adc    =$12BE
V? ]cmp    =$12C0      INDshow =$12DD   V? ]nop    =$12E8   V? ]sub    =$12EB
   dispreg =$1306   V? ]Ov     =$1307      RTF     =$1312      dispdig =$1344
   db      =$1375      bfstart =$1375      bfptr   =$1377      IBB     =$1378
   bfend   =$1379      bfsiz   =$137B      bfscrn  =$137D      bfclasch=$137F
   bfunitch=$1380      bffn    =$1381      bfoff   =$1382      bflane  =$1385
   bfdirty =$1386      DBB     =$138B      BOF     =$139E      BRP     =$13AB
   BSA     =$13B1      BCH     =$13BB      BCE     =$13CF      classdbx=$13E1
   BUN     =$13E1      fnxdbx  =$13E7      fnxfn   =$13EF   V? ]fetch3 =$13F1
   BFR     =$13F4      BFA     =$13F8   V? ]bfr    =$13FA      fnames  =$1400
   BCS     =$1447      SOR     =$1454      STA     =$1468   V  ]fetch2 =$14BE
   iodsel  =$14C8      LDR     =$14DB      iosel   =$14E5      LDB     =$14E7
   LSA     =$150D      setptr  =$150F      STP     =$1516      getwrd  =$1520
V  ]fetch1 =$1528      CLA     =$152B   V  ]incptr6=$1534      CLL     =$154C
V  ]IOerr1 =$1556      SRA     =$1557      putwrd  =$1559      SLA     =$158C
   readbuf =$159A   V? ]rts    =$15A4      nxtblk  =$15A5      incblk  =$15D1
   ckpref  =$15E3      prvblk  =$15F9      srAS    =$1600      srA     =$1602
   srAM    =$1604      srT     =$160D      srAMR   =$160F   ?  srR     =$1612
   decblk  =$161D      srT2    =$161D      backoff =$162F      slT     =$1631
   slA     =$163B      exchAR  =$1646   ?  freeaux =$1648      advoff  =$164F
   splitsL =$1654   V  ]resptr =$1667      clear   =$1668      setlan  =$1672
   midNN   =$1675      bcd2bin =$168C      ckspo   =$169C      resetdbs=$16B3
   resetdb =$16C4      emptydb =$16D2      flushall=$1710      flushbuf=$1723
   b220asc =$1726      dowrite =$1734   V  ]IOerr2 =$1776      doread  =$1779
   mtbfsz  =$17AC      MTS     =$17B0      PDfae   =$17B9      PDebx   =$17F1
   bload   =$181F      bsave   =$1826      Aparm   =$182D   V  ]IOerr3 =$182D
   MTC     =$1830      Eparm   =$1831      Bparm   =$1835      putwdhx =$1839
   putbyte =$183D      putpdcmd=$1855   ?  pdoscmd =$1867      pdosxeq =$186C
   prtrace =$1884      MRR     =$18AC      MRD     =$18AD      printsgn=$18E9
   print1  =$18F2      print2c =$18FB      print2  =$1906      prind   =$190C
   print6  =$1916      print2bl=$1927      printbl =$192A      kbserve =$192F
V  ]eot    =$193C      doctlblk=$194F      kbmodeon=$1958      viewon  =$1964
   showhelp=$1976      viewoff =$197C      viewkey =$198C      resetran=$198D
   strDinc =$199C      MIR     =$19AF      MIW     =$19B0   V? ]done   =$19F4
   MOR     =$1A34      MOW     =$1A35      HGRinit =$1A4D      xyinit  =$1A93
   MPF     =$1A97      MIB     =$1AD4      BCDLadrl=$1AE8      b220plot=$1B40
   BCDLadrh=$1B82      xdrawcur=$1BF2      BCDHadrl=$1C1C      BCDHadrh=$1C66
   lpread  =$1C70      simend  =$1CAF      swaphelp=$1CB1      kbhelp1 =$1CF0
   xlyl    =$1D00      kbhelp2 =$1D18      kbhelp3 =$1D40      kbhelp4 =$1D68
   hx      =$1D90      scale   =$1D91      delta   =$1D92      px      =$1D94
   py      =$1D95      xx      =$1D96      yy      =$1D97      xl      =$1D98
   yl      =$1D9A      keytbl  =$1E00      fktbl   =$1E80      histx   =$1F00
   histy   =$2000      xmap    =$2100      AUXcode =$2100      ymap    =$2B00
   AUXend  =$3488      xbyte   =$3500      xbit    =$3600      ybasel  =$3700
   ybaseh  =$37C0      VIEWend =$3880      MAINend =$3880      ptrdr0bf=$3B4C
   ptrdr1bf=$3DA6      MEM     =$4AD0      ptpch0bf=$6000      ptpch1bf=$625A
   mt0bf   =$64B4      memb    =$7530      mt1bf   =$7C62      noAD    =$8000
   operr   =$8A33   MD fkey    =$8000   MD key     =$8000   MD putat   =$8000
MD align   =$8000   MD resi    =$8000   MD seti    =$8000   MD mainjsr =$8000
MD mainjmp =$8000   MD auxjsr  =$8000   MD auxjmp  =$8000      DOSCMD  =$BE03
   BSSTATE =$BE42      KBD     =$C000      READMAIN=$C002      READAUX =$C003
   WRITMAIN=$C004      WRITAUX =$C005      ALTCHAR =$C00F      KBSTROBE=$C010
   CASSOUT =$C020      SPKR    =$C030      TEXT    =$C050      MIXED   =$C052
   PAGE2   =$C054      HIRES   =$C056      PB0     =$C061      PB1     =$C062
   PDL     =$C064      PTRIG   =$C070   ?  HGR2    =$F3D8      HPOSN   =$F411
   PRNTAX  =$F941      PRBL2   =$F94A      BASCALC =$FBC1      BEEP    =$FBDD
   HOME    =$FC58      CROUT   =$FD8E      PRBYTE  =$FDDA      PRHEX   =$FDE3
   COUT    =$FDED
```