LINE PROCESSOR PROTOCOL CODE
<Hopper> LPP.NLS

```
PAGE  ;initial constants
        nchar    EQU      80              ;characters per line
        nline    EQU      24              ;number of lines per screen
        prbase   EQU      0000h    ; 00000h or 5000h or ?
        cpbase   EQU      4F90h    ; 3F90h or 4f90h (with extra ram)
        dibbase  EQU      40h      ;hi ord addr. display memory
        lptp              EQU      46B           ;determines if scroll window ~~av~~
                 ;45B for no scroll window, 46B with: requires #2 prom    available
        dtim              EQU      4+40B         ;delay time
        pr0base  EQU      prbase+40h      ;code after interrupt vectors
        pr2base  EQU      prbase+0800h    ;page two of program
        pr3base  EQU      prbase+0C00h    ;page three of program
        xmax:    EQU      nchar-1         ;maximum x coordinate
        ymax:    EQU      nline-1         ;maximum y coordinate
        moncll   EQU      1Fh             ;US recalls monitor
        monloc   EQU      32              ;RST 1 address monitor start
        ;
        lpesc    EQU      1Bh             ;line processor escape char
        cooresc  EQU      36B      ;esc code for big coordinates
        bell              EQU      07h           ;bell code
        rubout   EQU      177B            ;rubout character
        bugchr   EQU      317B     ;rev. video D
        ;
        ;
        ;  Device names
        ;
        kbdmsk   EQU      177B            ;clear parity
        bttns    EQU      090h            ;mouse and keyset switches
        msmsk    EQU      340B            ;mouse button mask
        kstmsk   EQU      037B            ;keyset mask
        ;
        ad                EQU      093h          ;control address
        xtrig    EQU      001B
        ytrig    EQU      002B            ;call for y AD
        admsb    EQU      091h            ;count mid bits
        adlsb    EQU      092h            ;count lsb's
        adhsb    EQU      093h            ;count higest bits
        epcntl   EQU      0A0h            ;external processor contro
        epx               EQU      0A1h          ;ep transmitter
        txbzy    EQU      001B            ;transmitter ready bit
        parity   EQU      177B            ;clear parity bit
        msk07    EQU      200B
        nullline EQU      376B            ;null line if in first char
        ;
        ; symbols used in program from DM
        dmploc   EQU      800h     ;loc of Data Media proms
        tab0     EQU      dibbase*100h
        tab0b    EQU      dibbase ;high order add scrambler table
        tab0a    equ      00h      ;low order add scrambler table
        sdmem    EQU      2*nline+tab0     ;start of display memory
        dlsize   EQU      nchar+1 ;#bytes per display line
                 ;(including "clear" indicator)
        tab1     EQU      nline*dlsize+sdmem
        tsio     EQU      0A1h     ;transm start i/o
        rsio     EQU      0A5h     ;Receiver start i/o
        rdar     EQU      0A5h     ;recv. read
        psio     EQU      0A9h     ;cp out
        psrd     EQU      0A8h     ;cp status read (0: busy, 1: ready)
        ksio     EQU      0A0h     ;keyboard start i/o
        kswr     EQU      0AFh     ;keyb. status write (lights)
        ksrd1   EQU     0ACh    ;keyb. status read 1
        ksrd2   EQU     0AEh    ;keyb. status rd. 2 (lights)
        ksrd3   EQU     0AFh    ;pushbuttons
        kdar    equ     0ADh    ;keyb read
        matt    EQU     0C0h    ;display attribute mem.
        iorst   EQU     0FEh    ;i/o reset
        mskwr   EQU     0FDh    ;hardware interr. mask write
        dcux    EQU     0E8h    ;dma write cursor x
        dcuy    EQU     0E9h    ;dma write cursor y
        dmal    EQU     0EBh    ;dma memory address low
        dmah    EQU     0EAh    ;dma memory address high
        dsio    EQU     0EDh    ;dma start i/o
        dcio    EQU     0ECh    ;dma clear io
        dsrd    EQU     0E8h    ;dma status
        dsrdswitch      EQU     0E9h    ;dip swtch on dma card
        dbel    EQU     0EEh    ;ring bell
        dmode   EQU     0EFh    ;dma mode: 1=line/ 0=page
SPACE 2;  MODE BUTTONS and DMA board switch assignments
; Mode Buttons and DMA Board Switch assignments
        mdswcp  EQU     1B      ;cp (not graphics) mode pushbutton
        mdswtty EQU     2B      ;tty (no LP prot.) mode pushbutton
        mdswdisctrl     EQU     4B      ;screen_EP transparent mode switch
        mdswlp  EQU     10B     ;LP mode pushbutton
        dssatt  EQU     1       ;switch "1" for attr mem standout
        dsgrswkey       EQU     2       ;switch "2" for keyboard switch instead
        dsdisvkey       EQU     4       ;switch "3" for "disave" option      of foot
        dscppromkey     EQU     10B     ;switch "4" for cp prom exists        graphics
        dsrvv   EQU     20B     ;switch "5" for reverse video                   Switch
PAGE ;CONSTANTS
;-----------------------------------------------;
;                                               ;
;  CONSTANTS                                    ;
;                                               ;
;-----------------------------------------------;
;
        grswkey EQU     35B     ;^] for graphics switch key
        msknoad EQU     357B    ;turn off AD int. mask
        mskad   EQU     20B     ;turn on AD int. bit
        stndatbit       EQU     01B     ;bit in attr mem for standout
        nostndatbit     EQU     0B      ;to turn off attr mem bit
        mstak   EQU     2FD2h           ;top of stack
        r1loc   EQU     2FFDh
        r2loc   EQU     2FFAh
        r3loc   EQU     2FF7h
        r6loc   EQU     2FF4h   ;start of 3 jmp instrs for interrupts
        r4loc   EQU     2FF1h   ;RST 4 jmp, used by MON for program "BP"
        startxloc       EQU     2FEEh   ;option "4" system reset loc
        ;
        cpdvcore        EQU     141B    ;device code for dwl
        cpnline EQU     nline
        rqictime        EQU     4       ;# 8sec periods til rerequest
        cpnchar EQU     nchar
        cpnfunny        EQU     352B    ;funny number to verify CP (not ~~graphic~~
        cpmreopen       EQU     1       ;flag for cp reopen          graphics) has
        cproutmax       EQU     5       ;max# cp requests outstanding to EP  been
        cponlite        EQU     4B      ;lite indicates cp active            running
        cpofflite       EQU     373B    ;mask to turn off cp lite
        cmonlite        EQU     2B      ;lite for coordinate mode
        cmofflite       EQU     375B    ;mask to turn off coordinate mode lite
        svdlite EQU     10B     ;"save" display lite
        SPACE 2
;these next three values will have to be determined.
SPACE 2
        rate            EQU     42B             ;40B+(9600/baud rate)
        ;
        nmout   EQU     2C3h            ;ACC -> Display
        co              EQU     1E3h            ;C -> Display
        ;
        cr              EQU     0Dh             ;carrage return
        ;
PAGE;  CODE IN RAM (2F00h)
        ORG     2F00h
;DATA
;---------------------------------------------------------;
        msksdt: DS      1
        omsdta: DS      1
        okydta: DS      1
        cordmd: DS      1
        kboflg: DS      1
        procnt: DS      1
        proptr: DS      2
        proend: DS      2
        probuf: DS      26
        fstln:  DS      2
        secln:  DS      2
        scrtop: DS      1
        scrbot: DS      1
        rndcnt: DS      1
        mtf:            DS      1
        xflag:  DS      1
        xval:   DS      2       ;don't reorder
        xcur:   DS      1       ; ", note hi comes before low
        xcurlo: DS      1       ; "
        yval:   DS      2       ; "
        ycur:   DS      1       ; "
        ycurlo: DS      1       ; ", note hi comes before low
        curptr: DS      2       ;memory pointer corr. to xcur,ycur
                ;xval,yval contain lasst a/d read.  xcur,ycur are dejittered
;row,col corr. to xval,yval.
        curnty: DS      1       ;scratch for nstlnl
        ttyx:   DS      1       ;saved xpos while positioned
        ttyy:   DS      1       ;(don't reorder ttyx-ypos)
        xpos:   DS      1       ;cur. pos for writing chars
        ypos:   DS      1       ;(if tracking, is next tty pos.)
        posptr: DS      2
        massof  DS      1       ;master standout flag
                ;0 for normal, 1 for standout
        stndoutf:       DS      1       ;standout mode bit storage
                ;used only for bit in regular memory
                ;WHITE-ON-BLACK
                        ;0 when normal, 200B when standout
                ;BLACK-ON-WHITE:
                        ;200B when normal, 0 when standout
        swindt: DS      1
        swindb: DS      1
        char:   DS      1
        epbs0:  DS      2
        dwlbyret:       DS      2
        epcnt:  DS      1
        xbug:   DS      1       ;temp store for x bug pos
        epinptr:        DS      2
        epoutptr:       DS      2
        epbuf:  DS      32      ;ep in buffer
        epbend: DS      0
        epcldoff        EQU     24      ;turn off display at this many
        epcldon EQU     10      ;turn on display at this many
        epclerr EQU     32      ;error at this many
        ldrlsb: DS      1       ;loader lower half byte
        mar:    DS      2       ;loader memory address
        dwlret: DS      2       ;dwl return
        dwltrv: DS      2       ;dwl transfer vector
        ;dont' reorder tintflag, tintcnt
                tintflag:       DS      1
                tintcnt:        DS      1       ;cycles about ever 4 sec.s
        msk:    DS      1       ;int. mask
        disaveflag:     DS      1       ;copy display to "save" buffer
        ;don't reorder grtrackf, grcurf, and curchflag.  see chkgrcur.
        grcurf: DS      1       ;graphics cursor active
        grtrackf:       DS      1       ;graphcs xfer not in progress, may ~~trac~~
                                                                      track
        curchflag:      DS      1       ;cursor has changed
        dello:  DS      1       ;hysteresis const lo bits
        delhi:  DS      1       ;hysteresis const hi bits
        coorlo: DS      1       ;for big coordinates
        coorhi: DS      1
        scwdir: DS      2       ;double incr. or decr.
        scwinptr:       DS      2
        scwxpos:        DS      5       ;storage for parameters
                ;xleft, xright, ytop, ybot, nlines
        ;mouse/keyset state
                ;curmb-maxmouse grouped together for initialization
                curmb:  DS      1       ;current mouse button state
                mousesent:      DS      1       ;last mouse sent to ep
                charsnt:        DS      1       ;last (non-mouse) character's;m
                maxmouse:       DS      1       ;"or" of mouse buttons since la
                sxcur:  DS      1                                  mouse state
                sycur:  DS      1                                  last "o" state
ORG       cpbase ;after "save" display buffer
        cpcnt:  DS      1
        cpinptr:        DS      2
        cpoutptr:       DS      2
        cpepcnt:        DS      1       ;remaining chars expected from ep for c
        rqoutl: DS      1
        cpcksum:        DS      1
        cphcnt: DS      1
        cpdev:  DS      1
        cpignrf:        DS      1
        cpsqnum:        DS      1
        cpcntptr:       DS      2
        cpnopnf:        DS      1
        cpnopver:       DS      1
        cprptcnt:       DS      1
        cprptchar:      DS      1
        cpoutrv:        DS      2
        ;don't reorder next 3
                cprqcnt:        DS      1       ;# requests which
                        ;can be sent to EP (# of buffers available)
                cpcmtdcnt:      DS      1       ;# of buffers committed (reques
                rqoseqn:        DS      1       ;seq# corr. to cpoutptr   outstanding
PAGE;   INIT and RST locs
        ;org for system reset
```

```
                ORG     prbase
                jmp     inita   ;used for system reset when non-prom version
                        ;doesn't check mdswlp mode button.
                        ;when prbase is 0 (for proms), this isntruction is
                        ;overwritten by the code ORG'd at 0 below
        ;  RESTART 0
                ORG     0
                out     iorst
                in      ksrd3
                jmp     init
        ;  RESTART 1 (keyboard interrupt)
                ORG     8h
                jmp     r1loc
        ;  RESTART 2 (line receive interrupt)
                ORG     10h
                jmp     r2loc
        ;  RESTART 3 (line transmit interrupt)
                ORG     18h
                jmp     r3loc
        ;  RESTART 4
                ORG     20h
                jmp     r4loc
        ;  RESTART 6
                ORG     30h
                jmp     r6loc
;for debug startup
        ORG startxloc
        jmp     inita
PAGE;  CODE AT 040h (goes at 7040h when debugging)
        ;see other "ORG" after epenup
                ORG     pr0base
SPACE 2  ;  INITIALIZATION
;-----------------------------------------------------;
;                                                     ;
;  INITIALIZATION                                     ;
;                                                     ;
;-----------------------------------------------------;
;
        SPACE   2
        init:
                in      ksrd3
                ani     mdswlp  ;lp button
                jz      inita
                in      dsrdswitch      ;no, jump according to switch settings
                rlc
                rlc
                rlc
                ani     7B              ;switches "8", "7", "6"
                lxi     h,startvec
                jmp     eps99a
        inita:
                mvi     a,1
                out     dmode   ;set dma to line mode
                lxi     d,sdmem+1       ;init line table
                lxi     b,nchar+1       ;is one dummy byte before each line
                mvi     a,nline
                lxi     h,tab0  ;addr of line table
        pup2:
                mov     m,e
                inx     h
                mov     m,d
                inx     h
                xchg            ;add nchar+1 to DE
                dad     b
                xchg
                dcr     a
                jnz     pup2
                xra     a       ;put null at beg. of each line
                out     dmal    ;while a is zero
                out     matt;so funny display bits are 0
                lxi     sp,nstak
                mvi     a,tab0b
                out     dmah
                out     ksio
                out     rsio
                out     dsio
                mvi     a,10h
                out     mskwr
                sta     msk     ;current int. mask kept here
                mvi     a,xtrig ;start a/d
                out     ad
                call    cendstd ;get all standout flags set right
                call    vdclr
PAGE ; Continue Init
        initcont:
                xra     a               ;init epin state
                sta     epcnt
                sta     omsdta          ;so a zero won't be sent at start
                sta     xflag
                sta     disaveflag
                sta     rndcnt
                sta     cordmd
                sta     swindt
                sta     xpos
                sta     ypos
                sta     dello
                sta     grcurf
                lxi     h,curmb
                mov     m,a
                inx     h       ;mousesent
                mov     m,a
                inx     h       ;charsnt
                mov     m,a
                inx     h       ;maxmouse
                mov     m,a
                inr     a
                sta     grtrackf        ;this is 1 except when gr. xfer from ep
                sta     mtf             ;not positioned ("tracking")   in
                mvi     a,1B    ;this could be deleted of it stays "1"   Progress
                sta     delhi   ;hysteresis of alpha and gr. cursor
                call    nposxy
                mvi     a,ymax
                sta     swindb
                lxi     h,eps10
                shld    epbs0
                lxi     h,r6loc ;copy interrupt jump instructions
                lxi     d,ijtab
                mvi     c,12
        ij2:
                ldax    d
        mov     m,a
        inx     h
        inx     d
        dcr     c
        jnz     ij2
bufint:
        lxi     h,epbuf         ;epin ptrs
        shld    epoutptr
        shld    epinptr
        lxi     h,50B*100h+34B  ;system reset code
        shld    probuf
        lxi     h,probuf        ;ep output buffer
        shld    proptr
        inx     h
        inx     h
        shld    proend
        mvi     a,2
        sta     procnt
        call    cppromchk
        cnz     cpinit  ;cpin, out ptrs
        ei                      ;cpu enable interrupts
mloop:  lda     epcnt           ;see if anything in epin buffer
        ana     a               ;this loop waits for interrupts
        jz      mloop1
        call    epcod
        jmp     mloop
mloop1:
        lda     disaveflag
        ana     a
        cnz     disav
        lda     tintflag
        ana     a
        cnz     timsvc
        call    cppromchk       ;see if cp routines exist
        cnz     cpoutchk        ;yes, see if ready, etc.
mloop3:
        lda     procnt          ;see if anything waiting to go
        ana     a               ;(set flags)
        jz      mloop2          ;go out to ep
        in      epcntl
        ani     txbzy
        jnz     mloop2
        call    prout
        jmp     mloop
mloop2:
        call    sblankchk
        jmp     mloop
cppromchk:
        in      dsrdswitch
        ani     dscppromkey
        ret
ijtab:
        jmp     adint   ;intructions to be copied to scratch
        jmp     adint   ;memory
        jmp     rint
        jmp     kint
startvec:
        ;system reset startup locations as per last 3 switches on
        ;DMA board (only if LP mode button in)
        DW      inita
```

```
                DW      dmploc
                DW      3800h   ;assumes "switch" display not used
                DW      7000h   ;assues extra memory option
                DW      startxloc
                        ;assumes no debug monitor (which may use this ram loc.)
                DW      inita   ;spares
                DW      inita
                DW      inita
PAGE;  INTERRUPT SERVICE
;---------------------------------------------;
;                                             ;
;  INTERRUPT SERVICE                          ;
;                                             ;
;---------------------------------------------;
;
        SPACE 2
;3 interrupts are used: EP RECEIVE, KEYBOARD, and AD (CLOCK)
;they cause RST 1,2,6 .  These are currently routed through
;scratch locations starting at 2FFAh.  These locations are
;set by this program to jump to RINT, KINT, and ADINT.
        kint:
                push    psw     ;save regs
                push    h
                push    b
                push    d
                jmp     kodts0
        adint:
                push    psw     ;save regs
                push    h
                push    b
                push    d
                jmp     tintsvc
        rint:
                push    psw     ;save regs
                push    h
                push    b
                push    d
                call    eptst
        idone:
        iexit:
                pop     d       ;restore regs
                pop     b
                pop     h
                pop     psw
                ei
                ret             ;return
                SPACE   2
PAGE;  CALCULATE LINE POINTER
;---------------------------------------------;
;                                             ;
;  CALCULATE LINE POINTER                     ;
;                                             ;
;---------------------------------------------;
;
        SPACE 2
        ;lncnt converts a line number to a pointer into refresh memory by
;getting the appropriate entry from the line table

        lncnt:
                lxi     h,tab0
        lncntb:
                rlc
                add     l
                mov     l,a
                mov     a,m      ;low order byte
                inx     h
                mov     h,m     ;high byte
                mov     l,a
                dcx     h       ;position to dummy byte before printing bytes
                ret
SPACE 2 ;--SCROLL REFRESH MEMORY
;--------------------------------------------------;
;                                                  ;
;  SCROLL REFRESH MEMORY                           ;
;                                                  ;
;--------------------------------------------------;
;
        scroll:
                lda     swindt          ;top of window
                mov     b,a
                lda     swindb          ;bottom of window
                mov     c,a
                call    scren
                call    nposxy
                ret
        ;scren accepts top and bottom (line number) values in b,c.
;It moves the corresponding line pointer entries (rotates them
;up one).  The top entry is moded to the
;bottem and is blanked (null inserted as first char).
        scren:
                call    setrot  ;inits hl, de, a for rotate
                ;rotate up, de has (b) entry, hl
                ;points to (c) entry
                inx     h       ;point to second byte of last entry
        scren1:
                mov     c,d
                mov     d,m     ;do second byte
                mov     m,c
                dcx     h
                mov     b,e
                mov     e,m     ;first byte (low order)
                mov     m,b
                cmp     l
                rz              ;if done
                dcx     h
                jmp     scren1
        setrot:
                ;called for both rotate up and down to set a, hl, and de
                ;returns: de has (b) entry, hl points to (c) entry
                mov     a,b     ;pick up (b) enry
                call    lncnt   ;h points to dummy byte
                mvi     m,nulline       ;null this row
                inx     h       ;actual start of display line
                xchg
                mvi     h,tab0b
                mov     a,c
                rlc             ;times 2
                ;(adi  tab0a)
                mov     l,a
                mov     a,b
                rlc                     ;loop term check now in place
                ret
        insbc:
                ;insert at line number (c), rotating down through line (b)
                call    setrot
                inr     a               ;for term test
        insbc2:
                mov     b,e
                mov     e,m             ;first (low order) byte
                mov     m,b
                inx     h
                mov     c,d
                mov     d,m             ;second (high) byte
                mov     m,c
                cmp     1
                rz                      ;if done
                inx     h
                jmp     insbc2
PAGE;  CLEAR SCREEN
;-------------------------------------------------;
;                                                 ;
;  CLEAR SCREEN                              ;
;                                                 ;
;-------------------------------------------------;
;
        SPACE 2
        ;vdclr sets display memory to spaces.  It is called during
;initialization, by the clear screen command of nls, and by
;hardware reset.
        SPACE 2
        vdclr:
                mvi     a,nulline
                mvi     c,nline
                lxi     d,nchar+1
                lxi     h,sdmem
        vdcl2:
                mov     m,a
                dad     d               ;add nline to hl
                dcr     c
                jnz     vdcl2
                ret
SPACE 2 ;--CLEAR LINE
;-------------------------------------------------;
;                                                 ;
;  CLEAR LINE                             ;
;                                                 ;
;-------------------------------------------------;
;
        SPACE 2
;clrln expects register b to contain number of
;characters to be cleared andhl to contain pointer to first locatin.
SPACE 2
        clrln:
                lda     stndoutf
                ori     40B             ;blanks
        clrln1: mov     m,a
                inx     h
                dcr     b
                jnz     clrln1
                ret
```

```
PAGE;  VIDEO DISPLAY OUTPUT
;-----------------------------------------------------;
;                                                     ;
;  VIDEO DISPLAY OUTPUT                               ;
;                                                     ;
;-----------------------------------------------------;
;
        ;vdout accepts a character in register C to output to the refresh
;memory.  It checks mtf to see if it should hand crlf and do
;scrolling.  Char goes to xpos,ypos.
        vdout:  lda     mtf             ;are we positioned?
                ana     a       ;set flags
                mov     a,c             ;get char
                jz      vdout2          ;yes, ignore cr,lf
                ani     177B
                cpi     0Dh     ;CR?
                jz      vdcr    ;yes
                cpi     0Ah     ;LF?
                jz      vdlf    ;yes
        vdout2: cpi     08h     ;^h backspace
                jz      vdbs
                cpi     bell
                jz      vdbell
                cpi     40B     ;is it a control char
                jp      vdout3  ;no
                in      ksrd3   ;check mode button
                ani     mdswdisctrl     ;for transparent
                mov     a,c     ;in case jump to vdout3
                jnz     vdout3  ;yes, don't convert to space
                mvi     a,40B   ;normal: convert to space
                SPACE 2
                ;otherwise, simply put char out to xpos,ypos
                SPACE 2
        vdout3:
                lxi     h,stndoutf      ;or in standout bit (if set)
                ora     m
                lhld    posptr
                mov     m,a             ;put out ot refresh memory
                inx     h
                shld    posptr
                lxi     h,xpos          ;update pointers
                inr     m
                mov     a,m     ;check for wrap
                cpi     xmax+1
                rc              ;ok, no wrap
        ;If CR, set xpos_0
        vdcr:
                xra     a               ;just go to left margin
                sta     xpos
                call    nposxy
                ret
        vdbs:
                lda     xpos
                ana     a       ;set flags
                rz
                dcr     a
                sta     xpos
                call    nposxy
                mvi     b,1
                call    clrln
                ret
        vdbell:
                out     dbel
                ret
        ;
        ;If LF, scroll if ypos is at botton of
;tty window, else bump ypos
        SPACE 2
        vdlf:
                lda     swindo
                mov     c,a
                lda     ypos
                cmp     c
                jz      vdlf1
                inr     a       ;not at bottom, bump.
                sta     ypos
                call    nposxy
                ret
        vdlf1:
                call    scroll
                ret
        SPACE 2
        sblankchk:      ;blank first nulled line (if ay)
                lda     massof  ;see if standout on
                ana     a
                rnz             ;if so, don't try this
                mvi     c,nline
                lxi     d,nchar+1
                lxi     h,sdmem ;positioned to first line
        sblnk2:
                mov     a,m     ;is line nulled
                cpi     nulline
                jz      sblnk3  ;yes, go clear it
                dad     d
                dcr     c
                jnz     sblnk2
                ret
        sblnk3:
                mvi     b,nchar+1
                call    clrln   ;blank rest of line
                ret
PAGE;  A/D CONVERTER DRIVER
;---------------------------------------------------;
;                                                   ;
;  A/D CONVERTER DRIVER                             ;
;                                                   ;
;---------------------------------------------------;
;
        ;  Called to update x or y value
        ;  Calling param:
                ; xflag=0: the x coordinate was last requested and
                        ; will be read this time.
                        ; Results will be put in xval.
                ; xflag=-1: same eXcept y instead of x
        convrt:
                lxi     h,yval  ;result is y
                mvi     e,xtrig
                lda     xflag   ;0 = x, -1 = y
                rlc
                jc      cnvtr
        lxi     h,xval  ;result is x
        mvi     e,ytrig
;
cnvtr:  cma
        sta     xflag   ;next one next time
        in      admsb
        ana     a
        ral
        mov     c,a
        in      adhsb   ;highest bits
        ral
        ani     1Fh
        sui     10B
        jp      cnvrtq
        xra     a
        mvi     c,0
        cnvrtq:
        mov     b,a
        mov     a,e
        out     ad
        mvi     d,4
        call    gnbits  ;rot bc left 4
        mov     a,c
        sub     m
        mov     c,a
        inx     h
        mov     a,b
        sbb     m
        mov     b,a
        jc      cnvrts  ;going down, us new val
        push    h       ;not going down, get hysteresis
        lxi     h,dello
        mov     a,c
        sub     m
        mov     c,a
        inx     h
        mov     a,b
        sbb     m
        pop     h
        rm              ;not del above oldval, no change
        mov     b,a
        jnz     cnvrts  ;pos. change, >del
        mov     a,c     ;may be zero
        ana     a
        rz              ;exactly del above oldval, no change
cnvrts:
        mvi     a,1     ;are changing, set flag
        sta     curchflag
        mov     a,c     ;add bc to oldval
        dcx     h       ;low bits of ov
        add     m
        mov     m,a
        mov     c,a     ;also keep in c
        inx     h
        mov     a,b
        adc     m
        mov     m,a
cnvrtx:
        inx     h       ;hi "cur"
cnvrty:
```

```
                mov     m,a
                mov     b,a     ;for graphics cursor
                inx     h
                mov     m,c
                call    cppromchk       ;returns true if prom exists
                rz              ;not there
                jmp     cnvgraphics
        gnbits:
                push    d
                mvi     e,0
        gnb2:
                mov     a,c
                ral
                mov     c,a
                mov     a,b
                ral
                mov     b,a
                mov     a,e
                ral
                mov     e,a
                dcr     d
                jnz     gnb2
                pop     d
                ret
                SPACE 2
PAGE;  CURSOR ROUTINES
;------------------------------------------------;
;                                                ;
;  CURSOR ROUTINES                               ;
;                                                ;
;------------------------------------------------;
;
        ncurxy:
                call    cppromchk       ;is cp prom there?
                jnz     chkgrcur        ;prom there, check further
        ncurb:
                call    gycur   ;a-n cursor
                out     dcuy
                call    gxcur
                out     dcux
                ret
        gycur:  ;return valid y cursor pos in A
                ;preserves b,c
                lda     ycur    ;preserves h
                ana     a       ;turn off cary
                rar             ;divide by 2
                cma             ;subtract from ymax
                adi     ymax+1
                rc              ;if result ok
                xra     a       ;no, (ymax-ycur/2 < 0)
                ret
        gxcur:  ;return valid x cursor pos in A
                ;preserves b,c
                lda     xcur    ;limit at xmax
                cpi     xmax+1
                rc
                mvi     a,xmax
                ret
                SPACE 1
        nposxy:
                ;set posptr, with check for nulled line
                lda     ypos
        nposax: ;entry if using a instead of ypos
                call    lncnt
                mov     a,m
                cpi     nulline
                jnz     npos2
                push    h
                lda     massof  ;reset reverse video if set
                ana     a
                push    psw
                cnz     cendstd
                mvi     b,nchar+1
                call    clrln
                pop     psw     ;old massof
                cnz     cstdout ;reset reverse video
                pop     h
        npos2:
                lda     xpos
                mov     c,a
                mvi     b,00h
                dad     b
                inx     h       ;move past dummy byte
                shld    posptr
                ret
PAGE;  A/D SERVICE ROUTINE
;-------------------------------------------------;
;                                                 ;
;  A/D SERVICE ROUTINE                            ;
;                                                 ;
;-------------------------------------------------;
;
        adsvc:
                call    convrt
                call    ncurxy
                ret
                SPACE 2
        tintsvc: ;service syc interrupt, just set flag
                lxi     h,tintflag
                inr     m
                lda     msk     ;turn off int.
                ani     msknoad ;until ad serviced (at timsvc)
                out     mskwr
                sta     msk
                jmp     idone
                SPACE 2
        timsvc: ;non-interrupt sevice of ad, mouse, keyset
                call    adsvc
                call    mousrd
                call    kysamp
                xra     a       ;clear tintflag
                lxi     h,tintflag
                mov     m,a
                lda     msk     ;turn on int.
                ori     mskad
                out     mskwr
                sta     msk
                inx     h       ;to tintcnt
                inr     m       ;just cycles round and round
                rnz
                call    cpproachk       ;each 256 ad services
                cnz     rqtochk
                ret
PAGE;  MOUSE AND KEYSET ROUTINES
;--------------------------------------------------;
;                                                  ;
;  MOUSE AND KEYSET ROUTINES                       ;
;                                                  ;
;--------------------------------------------------;
;
        mousrd: in      bttns
                sta     msksdt          ;save character
                ani     msmsk           ;read mouse & keyset
                mov     c,a
                lxi     h,omsdta
                cmp     m
                rz              ;hasn't changed
                mov     m,a
                rlc                     ;put in lower byte
                rlc
                rlc
                ori     240B            ;sign bit, adds 200b
                call    nkybuf          ;put in buffer
                ret
                SPACE   2
PAGE;  KEYSET INPUT
;--------------------------------------------------;
;                                                  ;
;  KEYSET INPUT                                    ;
;                                                  ;
;--------------------------------------------------;
;
        kysamp:
                lda     msksdt          ;get button info
                ani     kstmsk          ;extract keyset
                lxi     h,okydta
                cmp     m
                rz                      ;no change
                ana     a
                jnz     kysor           ;not done, or in these bits
                mov     a,m             ;done, get the char from okydta
                mvi     m,0             ;clear okydta
                adi     200B
                call    nkybuf          ;put in buffer
                ret
        kysor:  ;or new reading with old
                ora     m
                mov     m,a                     ;save
                ret                             ;exit
                SPACE   2
PAGE;  PUT KEYBOARD CHARACTER IN EP OUTPUT BUFFER
;  PUT KEYBOARD CHARACTER IN EP OUTPUT BUFFER
        ;keyboard in [1,177B], mouse in [240B,247B], keyset in [200B,237B]
        nkybuf:
                mov     c,a
                cpi     240B    ;see if from mouse
                jnc     kbfms   ;yes
                lda     curmb   ;not from mouse, check cur mouse
                mov     b,a     ;and keep in b
                cpi     5B      ;see if marker or viewspecs
        jc      nkybdems        ;no, make sure not in mouse sequence
        cpi     7       ;yes, upper case vs?
        jnz     nkybua  ;no
        dcr     a       ;yes
        nkybua:
        lxi     h,mousesent     ;need to send mouse?
        cmp     m
        jz      kbchrsnd        ;no, already done
        call    kbmsend ;send as current mouse
        jmp     kbchrsnd

nkybdems:       ;might need to send zero mouse
        lda     mousesent
        ani     7       ;ep has it non-zero?
        mvi     a,0     ;so won't clear flag
        cnz     kbmcsend        ;send zero mouse (if was non-zero)
kbchrsnd:       ;mouse state is taken care of
        ;send char in c
        mov     a,b     ;mouse state
        sta     charsnt ;a char is being sent in this state
        cpi     4
        jc      kbcha   ;is <4
        sui     5       ;was 4,5,6, or 7
kbcha:
        mov     d,a     ;keep in d
        inr     d       ;now in [0,4]
        cpi     3       ;control char?
        mov     a,c
        jz      kbchctrl
        ana     a       ;see if from keyset, non-letter
        jm      kbchkyss        ;yes
kbchckys:
        cpi     101B
        jc      nkybsing        ;non-letter, no translate
        ani     337B    ;put in [100B,137B]
        cpi     133B
        jnc     nkybsing        ;non-letter, no translate
        mov     a,d     ;is letter
        cpi     0       ;case 2?
        jz      kbchtwo
        cpi     3       ;upper case?
        mov     a,c
        jnz     nkybasing
        ani     337B    ;upper case it
nkybasing:
        mov     c,a
nkybsing:
        call    coorchk ;maybe send coordinates
nkybsb: ;entry for mouse button chars
        in      ksrd3
        ani     mdswlp  ;reserved keys for screen switch, graphics?
        mov     a,c
        jnz     kbncrd  ;no
        cpi     34B     ;yes, save screen?
        cz      kbfdisave       ;yes
        cpi     36B     ;switch screen?
        cz      kbfdswitch      ;yes (never returns with 36B in A)
        in      ksrd2   ;see if switched screen
        ani     10B
        cnz     kbfdswitch      ;yes, switch back
        mov     a,c     ;get back char
        cpi     grswkey ;switch graphics cursor?
        jnz     kbncrd  ;no
        call    cppromchk       ;see if cp prom there
        cnz     kbfgrsw ;yes, note: a is smashed
kbncrd: ;rquest calls here, requires "a" preserved
        lhld    proend  ;in case called here
        mov     m,a
        inx     h
        shld    proend
        lxi     h,procnt
        inr     m
        ret

kbchctrl:       ;make control char
        ani     37B
        jmp     nkybasing

kbchkyss:       ;keyset char
        sui     40B     ;now in [140B,177B]
        mov     c,a
        cpi     173B    ;see if special
        jc      kbchckys        ;process as ordinary char
        mov     a,d
        rlc
        rlc
        add     d       ;5 times mouse state
        add     c       ;plus value of char (in [173B,177B])
        lxi     h,kystab-173B
        mov     c,a
        mvi     b,0
        dad     b
        mov     a,m
        jmp     nkybasing

kbchtwo:        ;case two mouse
        mov     a,c
        ani     37B     ;will always trans into
        mov     c,a     ;a case two char
        mvi     b,0
        lxi     h,case2tab
        dad     b
        mov     a,m
        jmp     nkybasing

kbfms:  ;mouse char
        ani     7B
        sta     curmb
        lxi     h,maxmouse
        jnz     kbfmsx  ;non-zero change
        mov     c,m     ;now zero, get mouse char
        mov     m,a     ;and clear
        lxi     h,charsnt       ;see if a char was sent
        mov     a,m     ;while buttons were down
        cpi     2       ;but not just CA button
        mvi     m,0
        jnc     kbfmsy  ;yes, char was sent
        ;no, send mouse char
        mov     a,c
        cpi     7       ;ignore a 7
        rz
        mvi     b,0     ;look up in table
        lxi     h,mousechrtab
        dad     b
        mov     c,m
        cpi     6       ;CA, CD, or ^B?
        cc      coorca  ;yes
        jmp     nkybsb

kbfmsx: ;non-zero change
        ora     m       ;just or in to maxmouse
        mov     m,a
        call    setcoors        ;save current coors for later
        ret

kbfmsy:
        lda     mousesent       ;char was sent, what about mouse
        ana     a
        rz              ;no, it wasn't sent
        xra     a       ;yes it was, send zero mouse
        jmp     kbmsend

coorchk:        ;preserves b,c
        ;only called here for keyboard chars
        mov     a,c
        cpi     4B
        jz      coorcb
        cpi     2B
        jz      coorcb
        cpi     30B
        rnz
coorcb:
        call    setcoors
coorca: ;called here for mouse chars CA, CD, ^B
        ;preserves b,c
        push    b
        lhld    proend
        mvi     m,34B
        inx     h
        lda     grcurf
        ana     a
        jnz     kbgcoor
        mvi     m,42B
        mvi     d,4     ;basic # chars, adjusted by bbstore
        jmp     kbms2

kbmcsend:       ;set coors and send a
        ;preserves b,c
        push    psw
        call    setcoors
        pop     psw
kbmsend:        ;from mouse, has high order bit on
        ;preserves b,c
        sta     mousesent
        adi     100B
        push    b
        mov     c,a             ;sav char
        lhld    proend
        mvi     m,34B           ;put out 34b
        inx     h               ;increment pointer
        lda     grcurf  ;which cursor
        ana     a
        jnz     kbgcur  ;graphics cursor
        mvi     a,43B           ;put out 43b
        mov     m,a
        inx     h               ;increment pointer
        mvi     d,5     ;basic #chars, adjusted by bbstore
        mov     m,c             ;control  or mouse char
kbms2:
        inx     h
        lda     sxcur           ;gxcur, gycur preserve hl
        call    bbstore ;put a in buffer, maybe as two chars
        lda     sycur
        mov     c,a
        mvi     a,ymax  ;invert and add 40B
        sub     c
        call    bbstore ;put a in buffer, maybe as two chars
kben2:  ;entry from graphics cursor handling
        pop     b       ;restore b,c
        lda     cordmd  ;no mouse or coors unless in cordmd
        ana     a       ;wastes cycles, but saves program
        rz
        shld    proend          ;store it
        lxi     h,procnt        ;get address of procnt
        mov     a,m             ;move contents to a
        add     d       ;add count
        mov     m,a             ;store again in procnt
        ret
bbstore:                ;put a in buffer as coordinate, possibly big
        ;expects a count in d, will bump by 2 if big coord
        cpi     94      ;see if big coord. nesc.
        jc      bbsta   ;no
        inr     d
        inr     d       ;adjust count in d
        push    psw
        mvi     m,cooresc       ;escape char first
        inx     h
        rlc
        rlc             ;two top bits next
        ani     3B
        adi     40B
        mov     m,a
        inx     h
        pop     psw     ;calling param back
        ani     77B     ;bottom 6 bits
        bbsta:
        adi     40B
        mov     m,a
        inx     h
        ret
setcoors:               ;save current cursor pos.
        ;preserves b,c
        call    gxcur
        sta     sxcur
        call    gycur
        sta     sycur
        ret
kbfdisave:              ;copy di buffer to save buffer
        in      dsrdswitch
        ani     dsdisvkey
```

```
                rz
                mvi     a,1     ;set flag and return
                sta     disaveflag
                inx     sp      ;return to callers caller
                inx     sp
                ret
        kbfdswitch:     ;switch display images
                in      dsrdswitch
                ani     dsdisvkey
                rz
                in      ksrd2   ;get lights
                xri     svdlite ;using "insert delete" light
                out     kswr
                ani     svdlite
                lxi     h,tab1
                jnz     kbfds2  ;if we are switching to the "save" buffer
                lxi     h,tab0  ;no, switching to regular buffer
        kbfds2:
                mov     a,h
                out     dmah    ;switch
                mov     a,l
                out     dmal
                inx     sp      ;return to caller's caller
                inx     sp
                ret
        disav:  ;may want to make this do it in parts
                call    cppromchk
                cnz     cpopnchk
                lxi     h,tab0
                lxi     d,tab1
                lxi     b,tab1-tab0+256
        disloop:
                mov     a,m
                stax    d
                inx     h
                inx     d
                dcr     c
                jnz     disloop
                dcr     b
                jnz     disloop
                lxi     h,disaveflag
                mov     m,c
                lxi     h,tab1  ;fix up "save" line table
                lxi     b,tab1-tab0     ;delta in bc
                mvi     e,24
        disl2:
                mov     a,m
                add     c
                mov     m,a
                inx     h
                mov     a,m
                adc     b
                mov     m,a
                inx     h
                dcr     e
                jnz     disl2
                ret
PAGE;  KEYBOARD INPUT
;-------------------------------------------------;
;                                                 ;
;  KEYBOARD INPUT                                      ;
;                                                      ;
;------------------------------------------------------;
;
       kbdts0:
               in      kdar     ;get char
               ani     kbdmsk
               out     ksio     ;call for next char
               call    nkybuf
               jmp     idone
PAGE;  OUTPUTS PROCESSOR BUFFER TO EP
;------------------------------------------------------;
;                                                      ;
;  OUTPUTS PROCESSOR BUFFER TO EP                      ;
;                                                      ;
;------------------------------------------------------;
;
       prout:  lhld    proptr   ; processor buffer
               mov     a,m              ;get char
               out     epx              ;prout called when txrdy
               inx     h                ;increment pointer
               shld    proptr           ;store new pointer
               lxi     h,procnt         ;get buffer count
               di                       ;so this gets done right
               dcr     m                ;decrement by 1
               jnz     proute                   ;exit
               lxi     h,probuf         ;buff empty, reset ptrs
               shld    proptr           ;procnt already 0
               shld    proend
       proute:
               ei                       ;touchy stuff done
               ret
PAGE;  EXTERNAL PROCESSOR COMMAND ROUTINES (except CP and GRAPHICS)
;------------------------------------------------------;
;                                                      ;
;  EXTERNAL PROCESSOR COMMAND ROUTINES                 ;
;                                                      ;
;------------------------------------------------------;
;
       trackoff:
               lxi     h,mtf
               mov     a,m
               mvi     m,0      ;clear mtf
               rrc              ;see if it was set
               rnc              ;no,just return
               lhld    xpos     ;yes, save xpos,ypos in ttyx,ttyy
               shld    ttyx
               ret

       poscur:
               call    trackoff
               call    epssnxt
       poscr1: sui     20h               ;subtract 40(8)
               mvi     c,Xmax
               call    bndchk   ;make sure in [0,xmax]
               sta     xpos                     ;x coordinate
               call    epssnxt
       poscr2: mvi     a,ymax+40B
               sub     c        ;change so 0 is at top os screen
               mvi     c,ymax
```

```
        call    bndchk  ;make sure in [0,ymax]
        sta     ypos                    ;y coordinate
        call    nposxy  ;update posptr
        jmp     eps14
        SPACE 2
specty:
        call    epssnxt
spcty1: mvi     a,ymax+40B
        sub     c               ;ymax-(c-40B)
        mvi     c,ymax
        call    bndchk  ;make sure in [0,ymax]
        sta     swindt          ;top of window line #
        call    epssnxt
spcty2: mvi     a,ymax+40B
        sub     c               ;ymax-(c-40B)
        mvi     c,ymax
        call    bndchk  ;make sure in [0,ymax]
        sta     swindb          ;line # botton of window
        sta     ttyy
        xra     a               ;zero ttyx
        sta     ttyx
        lda     mtf             ;are we positioned?
        rrc
        jnc     eps14           ;yes, dont' fix xpos,ypos
        jmp     epenup          ;no, copy ttyx,ttyy to xpos,ypos
        SPACE 2
bndchk: ;assure a in [0,c]
        jnc     bch2    ;assumses flags set
        xra     a
bch2:
        cmp     c
        rc
        mov     a,c
        ret
        SPACE 2
stdout:
        call    cstdout
        jmp     eps14
        SPACE 2
cstdout:                ;**must preserve h
        mvi     a,1     ;set master standout flag
        sta     massof
        in      dsrdswitch      ;check for reverse video
        ani     dsrvv   ;using copy printer button temporarily
        jnz     cestd3  ;do the reverse
cstd3:
        in      dsrdswitch      ;which hardware option?
        ani     dssatt  ;standout bit in attr mem.
        mvi     a,stndatbit     ;only used for bit in attr mem.
        jnz     cestd4  ;yes
        mvi     a,200B  ;no, set flag
        sta     stndoutf        ;could save a byte by jumping from here
        ret
endstd:
        call    cendstd
        jmp     eps14
        SPACE 2
cendstd:
        xra     a       ;clear master standout flag
        sta     massof
        in      dsrdswitch      ;check for reverse video
        ani     dsrvv   ;on dma board switch
        jnz     cstd3   ;do reverse
cestd3:
        in      dsrdswitch      ;which hardware option?
        ani     dssatt  ;standout bit in attr mem.
        jz      cestd2  ;no
        mvi     a,nostndatbit
cestd4:
        out     matt    ;set hardware (next writes will use)
cestd2:
        xra     a       ;clear flag
        sta     stndoutf
        ret
nstln:
        mvi     b,ymax  ;set up for insbc
        lda     ypos
        mov     c,a
        call    insbc   ;move down fro ypos to ymax
        call    nposxy  ;not sure if this nescessary
        jmp     eps14
        SPACE 2
clrscr: call vdclr
        jmp     eps14
        SPACE 2
inter:
        lhld    proend
        mvi     m,34B
        inx     h
        mvi     m,46B
        inx     h
        mvi     m,xmax+40B
        inx     h
        mvi     m,ymax+40B
        inx     h
        mvi     m,lptp
        inx     h
        mvi     m,dtim
        inx     h
        push    h
        in      adlsb
        rar
        rar
        rar
        rar
        ani     17B     ;4 bits of speed info
        mov     l,a
        mvi     h,0
        lxi     d,rattab
        dad     d
        mov     a,m
        pop     h
        mov     m,a
        inx     h
        shld    proend
        lxi     h,procnt
        mvi     a,07h
        add     m
        mov     m,a
        jmp     eps14
```

```
                SPACE 2
        ncrdmd: xra     a
                sta     cordmd
                in      ksrd2   ;turn off light
                ani     cmofflite       ;bit 1, led 6
                out     kswr
                jmp     eps14
                SPACE 2
        crdmd:  mvi     a,01h
                sta     cordmd
                in      ksrd2   ;turn on light
                ori     cmonlite        ;bit 1, led 6
                out     kswr
                jmp     eps14
                SPACE 2
        blkch:
                call    epssnxt
        blkch1: sui     40B
                mov     b,a
                lda     xpos
                add     b
                sta     xpos
                lhld    posptr
                call    clrln
                shld    posptr
                jmp     eps14
        SPACE 2
;this code is very incomplete.  i need to reset lpcnt somewhere
;and is reset the right place to jump off to?
        lpcod:
                lxi     h,rndcnt
                inr     m
                mov     a,m
                cpi     10
                jz      reset
                ret
SPACE 1; PSHBUG
;PSHBUG
        ;don't bother to stack bugs, rely on NLS to rewrite them
                ;change NLS8 to avoid leaving bugmarks on spaces
        pshbug:
                call    epssnxt
        pshb1:
                sui     40B
                sta     xbug
                call    epssnxt
        pshb2:
                mvi     a,ymax+40B
                sub     c
                lhld    xpos    ;and ypos
                push    h       ;save them
                sta     ypos
                lda     xbug
                sta     xpos
                call    nposxy  ;sets h
                call    cstdout ;turn on standout (preservs h)
                mov     c,m     ;get current char
                call    vdout   ;put it out (but with standout)
                call    cendstd
                pop     h       ;restore xpos,ypos
                shld    xpos
                call    nposxy
                jmp     epend   ;resume tracking
                SPACE 2
        ;do nothing (see pshbug)
        popbug:
                jmp     eps14
                SPACE 2
        reset:
                call    vdclr
                lxi     h,swindt
                mvi     m,0     ;set tty window full
                inx     h
                mvi     m,ymax
                jmp     epend   ;resume tracking
                SPACE 2
        dltln:  lda     ypos            ;which line
                mov     b,a             ;delete it
                mvi     c,ymax
                call    scren
                xra     a
                sta     xpos            ;left of screen
                call    nposxy
                jmp     eps14
                SPACE 2
PAGE;  EXTERNAL PROCESSOR INPUT HANDLER
;--------------------------------------------------;
;                                                  ;
;  EXTERNAL PROCESSOR INPUT HANDLER                ;
;                                                  ;
;--------------------------------------------------;
;
        SPACE 2
;eptst is called thru the interrupt handler "intr:".  It
;reads a character and exits if it doesn't get one.
SPACE 2
        eptst:
                in      rdar
                ana     a               ;ignore nulls
                out     rsio    ;start receiver for next char
        eptst2:         ;non IO entry point
                ani     177B
                rz
                cpi     rubout  ;don't pass padding
                rz
                lhld    epinptr
                mov     m,a             ;put char in buffer
                inx     h               ;bump in ptr
                mov     a,l
                cpi     epbend
                jnz     eptsb
                lxi     h,epbuf
        eptsb:
                shld    epinptr
                lxi     h,epcnt
                inr     m
                mov     a,m
                cpi     epcldoff
                jnz     eptsc
                out     dcio
        eptsc:
                cpi     epclerr
                rnz
                rst     4
                ret
        ;epcod processes one char from the epin buffer.  If it's in
;command mode it immediately goes off to the correct command handler.
;On the "first round" however it checks for lpesc (the code for
;a command) and rubouts (ignores them and returns for next character).
        epcod:
                lxi     h,epcnt         ;bump down cnt
                dcr     m
                mov     a,m
                cpi     epcldon
                jnz     epc2
                out     dsio
        epc2:
                lhld    epoutptr
                mov     c,m
                inx     h               ;bump out ptr
                mov     a,l             ;wrap?
                cpi     epbend
                jnz     epc1            ;not last char in buff
                lxi     h,epbuf         ;last char, init ptrs
        epc1:
                shld    epoutptr
                lxi     h,rndcnt                ;a command loop?
                mov     a,m
                ana     a
                jnz     secrnd                  ;yes
                mov     a,c                     ;no
                cpi     lpesc
                jz      eps12
        epin0:  lhld    epbs0
                pchl                            ;jump off to correct routine
                SPACE 2
        secrnd: mov     a,c
                jmp     epin0
                SPACE   2
                ;this simply transmits a character
        epsvcont:
                call    epssnxt
        eps10:
                mov     c,a     ;transmit char
        eps11:
                call    vdout
        eps14:
                xra     a               ; and reset mode
                sta     rndcnt
                jmp     epsvcont                ;reset state switch

        epssnxt:        ;co-routine for next char
                pop     h
                shld    epbs0
                ret                     ;next

        eps12:
                in      ksrd3   ;check mode buttons
                ani     mdswtty ;for screen_EP transparent
                jnz     eps11   ;transp., handle like regular char.
        inr     m       ;assumes hl points to rndcnt
        call    epssnxt
eps20:  mov     a,c
        cpi     lpesc           ;2nd lpesc char?
        jz      lpcod           ;yes
        mov     a,c             ;get char again
        sui     40B
        jm      eps100  ;is < 40B
        cpi     26B
        jm      eps97   ;in [40B-65B]
        sui     40B
        jm      eps100  ;in [66B-77B], not assigned
        cpi     5
        jp      eps100  ;is > 104B
        call    epsdwl  ;is for loader
                ;command codes 100B-104B are used by tboot to load the
        jmp     eps14   ;returns here after several chars
                                                                Program

epsdwl: ;do a loader sequence for code in a
        pop     h       ;set return loc
        shld    dwlret
        call    edentset        ;preserves a
eps98:  lxi     h,memtab ;in[100B, 105B]
eps99:
eps99a:
        add     a               ;double index
        mov     c,a
        mvi     b,0
        dad     b               ;add to hl
        mov     e,m             ;get address there
        inx     h
        mov     d,m
        xchg
        pchl                    ;go off to it
        ;
eps97:  lxi     h,fcntab
        jmp     eps99
        ;
eps100:(error code here)
        di              ;no interrupts
        rst     4       ;break to mon. for now
        SPACE   2
edentset:       ;set epps0 to edwlent
        ;preserves a if "call"ed
        call    epssnxt
edwlent:        ;dwl dispatch
        lhld    dwltrv
        pchl

dwlnxt: ;coroutine call for next char
        pop     h
        shld    dwltrv
        ret

eps21:  ;set mem address reg (mar)
        call    dwlbyte
        sta     mar
        call    dwlbyte
        sta     mar+1
        mvi     c,53B
        call    vdout
        ;jmp    dwltrm
dwltrm:
        lhld    dwlret
        pchl

eps22:  ;memory load turn on
        xra     a       ;clear flag
        sta     rndcnt  ;so we can escape from loading
eps22a:
        call    dwlbyte
        lhld    mar
        mov     m,a
        inx     h
        shld    mar
        jmp     eps22a  ;loops, depends on esc to break out

eps24:  jmp     dwltrm  ;starting address ignored now
        ;
eps26:  ;might use eps24, don't know yet
        lhld    mar     ;start program at mar
        di
        pchl

epsbyte:
        call    edentset
dwlbyte:                ;get 8 bit byte from two chars
        pop     h
        shld    dwlbyret
        call    dwlnxt
eps30:  ani     0Fh     ;lsb to temp (ldrlsb)
        sta     ldrlsb
        call    dwlnxt
eps40:  ani     0Fh     ;msb V lsb to a
        rlc             ;rotate to msb
        rlc
        rlc
        rlc
        lxi     h,ldrlsb        ;or in lsb
        ora     m
        lhld    dwlbyret        ;post byte state
        pchl            ;jump there
        ;
cpdwl:  ;dwl called from cp
        cpi     lpesc
        jz      cpdwla
        lhld    dwltrv
        pchl
                ;note somewhat dangerous situation: cpdwl must be
                ;called first with an esc or might wander into boonies

cpdwla:
        call    dwlnxt  ;get type code
        sui     100B
        jmp     eps98
epend:
        lxi     h,mtf
        mov     a,m
        rrc
        mvi     m,1
```

```
                jc      eps14
        epenup:
                lhld    ttyx
                shld    xpos
                call    nposxy
                jmp     eps14
                SPACE 2
PAGE;  EXTERNAL PROCESSOR TABLES
;-----------------------------------------------------;
;                                                     ;
;  EXTERNAL PROCESSOR TABLES                          ;
;                                                     ;
;-----------------------------------------------------;
;
        memtab: DW      eps21
                DW      eps22
                DW      eps14
                DW      eps24
                DW      eps26
                ;
        fcntab: DW      poscur
                DW      specty
                DW      epend
                DW      blkch
                DW      dltln
                DW      nstln
                DW      pshbug
                DW      popbug
                DW      clrscr
                DW      reset
                DW      tptstr
                DW      tptopn
                DW      tptcls
                DW      inter
                DW      stdout
                DW      endstd
                DW      ncrdmd
                DW      crdmd
                DW      curres
                DW      tptnstr ;53B
                DW      tptnopn ;54B
                DW      scwindow        ;65B
        rattab: ;by decreasing baud rate
                DB      41B,41B,042B,042B,043B,044B,045B,046B
                DB      50B,60B,100B,140B,140B,140B,140B,140B
SPACE 1; ORG for 2nd prom, scroll window (normal 0800h, debug x800h)
        ;stuff in this prom should not be called if lptp is 45B instead of 46B
        ;**** ORG  pr2base
;Scroll Window protocal
        scwindow:
                call    trackoff
                lxi     h,scwxpos-1     ;set up for getting 5 parameters
                shld    scwinptr
        scwlm:
                lxi     h,scwlp
                jmp     epcoor
        scwlp:
                lhld    scwinptr        ;store a coordinate
                inx     h
                mov     m,a
        shld    scwinptr
        mov     a,l
        cpi     scwxpos+4
        jnz     scwlm   ;not done yet
        mov     a,m     ;+ or - displacement
        mov     b,a     ;could get it from c
        ana     a       ;set flags
        jz      epend   ;nothing to do
        lxi     d,0FEFFh        ;double register decrementer
        jm      scwlq   ;moving down
        cma     ;moving up, make negative count
        inr     a
        lxi     d,101h  ;double register incrementer
        dcx     h       ;point to ybot
        scwlq:
        mov     c,a
        dcx     h       ;ybot for down, ytop for up
        mvi     a,ymax  ;change to internal
        sub     m
        xchg            ;put away inc/dec
        shld    scwdir
        mov     e,a     ;dest line#
        add     b
        mov     d,a     ;src line#
        lxi     h,scwxpos
        mov     a,m     ;left margin
        sta     xpos
        cma
        inr     a       ;-xleft
        inx     h       ;xright
        add     m       ;xright-xleft
        inr     a       ;width
        mov     b,a
        inx     h       ;top line, external format (ytop>ybot)
        mov     a,m
        inx     h       ;ybot
        sub     m       ;ytop-ybot
        inr     a       ;height
        add     c       ;number to preserve and move
        jnc     epend   ;**might want to remove this when things are ~~de~~ debugged
        push    b       ;for use when clearing vacated lines
        mov     c,a     ;in place for moving
        jz      scwd    ;now move, just do clearing
scwa:
        push    d       ;source and dest line nums
        push    b       ;move line ctr and width
        mov     a,d
        call    nposax  ;get source ptr to hl
        mov     a,e
        xchg            ;now to de
        call    nposax
        pop     b       ;nposax might smash b
        push    b
scwb:
        ldax    d       ;move one line
        mov     m,a
        inx     h
        inx     d
        dcr     b
        jnz     scwb
```

```
                pop     b       ;one line done
                pop     d
                lhld    scwdir  ;incr or decr
                dad     d       ;src and dest line ptrs
                xchg
                dcr     c
                jnz     scwa
        scwd:   ;clear vacated lines
                pop     b       ;b=width, c=#vacated lines
        scwe:
                push    b
                mov     a,e
                call    nposax
                pop     b
                push    b       ;nposax smashes b,c
                call    clrln
                pop     b
                lhld    scwdir
                dad     d
                xchg
                inr     c
                jnz     scwe
                jmp     epend   ;all done
;big coordinate receive routines
        epcoor: ;set for cordinate collection
                shld    dwlbyret
                call    epssnxt
        epcoor0:        ;coordinate collection protocal
                cpi     cooresc ;check for big coord
                jnz     epconorm
                call    epssnxt
        epcoor1:        ;first 6 bits of big coord
                sui     40B
                rrc
                rrc
                mov     c,a
                ani     17B     ;hi 4 of top 6
                sta     coorhi
                mov     a,c
                ani     300B    ;lo 2 of top 6
                sta     coorlo
                call    epssnxt
        epcoor2:        ;second 6 bits of big coord
                sui     40B
                lxi     h,coorlo
                ora     m
                mov     c,a
                lhld    dwlbyret
                pchl
                SPACE 2
        epconorm:
                sui     40B
                mov     c,a
                lhld    dwlbyret
                pchl
;tables for new mouse/keyset handling
        case2tab:       DB      " !""#$%&'()@+-*/^^0123456789="
        kystab:
                        DB      "[]_",33B,15B   ;ms=4, case 2
                        DB      ",.;? " ;ms=0, case 0
                        DB      ",.;? " ;ms=1, case 3
                        DB      "<>:\",11B      ;ms=2, case 1
        mousechrtab:    DB      0, 4, 30B, 2, 1, 33B, 27B
SPACE 1; ORG for 3rd prom, CP and GRAPHICS (normal 0C00h, debug 7C00h )
        ORG  pr3base
;GRAPHICS, CP COMMANDS from EP
        tptcls:
                xra     a       ;clear block requests
                sta     cpnopver
                inr     a
        tptcl2:
                sta     grtrackf
                jmp     eps14
        tptopn:
                xra     a       ;clear flag
                jmp     tptcl2
        tptstr:
                call    epssnxt
        tpts2:  ;second char
                xra     a
                sta     cpignrf
                call    tpnxtln
                mvi     m,-100
                jmp     tptn3b
        tpnxtln:
                lxi     h,cpsio ;make sure xfer vector correct
                shld    cpoutrv
                lxi     h,cprqcnt       ;get next available buffer
                dcr     m               ;adjusting count
                inx     h
                lda     rqoutl
                add     m
                inr     m
                cpi     cpnline ;check for wrap
                jc      tpts2b
                sui     cpnline
                tpts2b:
                call    cplncnt
                ret
        tptnoreo:       ;attempt reopen
                lda     cpnopver        ;verify it was open
                sui     cpnfunny
                jnz     tptno3b ;no, do reguar open
                inr     a       ;ok, reopen it
                sta     cpnopnf
                jmp     eps14
        tptnopn:        ;open, new format
                call    epssnxt
        tptno2: ;device code
                sta     cpdev
                lxi     h,cpsio
                cpi     cpdvcore        ;loader?
                jnz     tptno2a ;no
                lxi     h,cpdwl ;yes
                tptno2a:
                shld    cpoutrv
                call    epssnxt
        tptno3:         ;mode code
                ani     cpmreopen       ;reopen?
                jnz     tptnoreo        ;yes
        tptns3b:
        mvi     a,1
        sta     cpnopnf
        mvi     a,cpnfunny      ;funny#
        sta     cpnopver
        mvi     a,cproutmax     ;# requests safe to make
        sta     cprqcnt
        xra     a
        sta     rqoutl
        sta     cpcmtdcnt
        mvi     a,40B
        sta     rqoseqn
        jmp     eps14
tptnstr:        ;new format cp record
        call    epsbyte
tptns1: ;checsum in a
        cma     ;form negative
        inr     a
        sta     cpcksum
        xra     a
        sta     cpignrf ;clear ingore flag
        sta     rndcnt
        call    epssnxt
tptns2: ;device #
        call    cksumadd        ;preserves a
        lxi     h,cpdev
        cmp     m
        jz      tptn2c  ;device num. ok
        lxi     h,cpignrf
        inr     m       ;set flag to ignore record
tptn2c:
        lxi     h,cprqcnt
        inr     m               ;ok to make another request
        call    epssnxt
tptns3: ;seq #
        call    cksumadd        ;preserves a
        sta     cpsqnum
        call    lnsq
        ana     a
        jm      tptn3c
        call    cplncnt ;seq. num. ok
        mov     a,m
        ana     a       ;make sure we are waiting for this one
        jp      tptn3c  ;no
tptn3b:
        shld    cpcntptr        ;used at tptns5
        inx     h
        shld    cpinptr ;put chars here
tptn3d:
        call    epssnxt
tptns4: ;char. cnt.
        call    cksumadd
        cpi     cpnchar+41B     ;too big?
        jnc     tptns4d ;yes
        sui     40B
        jp      tptns4c ;is ok
        tptns4d:
        mvi     a,cpnchar
        tptns4c:
        sta     cpepcnt
        sta     cphcnt  ;hold here for use by tptns5
        jz      tptns4b ;in case have 0 cnt
        call    epssnxt
tptns5: ;character collector
        call    cksumadd        ;preserves a
        lxi     h,cpignrf       ;see if ignoring
        inr     m
        dcr     m
        cz      cpbin   ;ok
        lxi     h,cpepcnt       ;dec. count
        dcr     m
        rnz     ;not done yet
        lda     cpignrf
        ana     a       ;see if ignoring
        jnz     eps14   ;yes, just clear dispatcher
tptn5b:
        lda     cphcnt
        lhld    cpcntptr
        mov     m,a     ;mark received
        lda     cpnopnf ;doing checksum?
        ana     a
        jz      eps14
        lda     cpcksum
        ani     177B
        jz      eps14   ;checksum ok, go cear dispatcher
tptn5c:
        lda     cpsqnum ;incorrect checsum, re-request
        call    rquest
        jmp     eps14

tptn3c:
        lxi     h,cpignrf       ;unexpected seq #
        inr     m
        jmp     tptn3d
tptns4b:
        lxi     h,cpnopnf
        mov     a,m
        cpi     1
        jnz     tptn5b
        lda     cpcksum
        ani     177B
        jnz     tptn5c  ;error, re-request
        inr     m       ;no more requests (but maybe re-requests)
        lda     cpsqnum ;adjust cpcmtdcnt
        call    lnsq
        mov     a,c
        inr     a
        inr     a
        sta     cpcmtdcnt
        jmp     tptn5b
cksumadd:       ;add a to cpcksum, preserve a
        push    psw
        lxi     h,cpcksum
        add     m
        mov     m,a
        pop     psw
        ret
rqtochk:        ;check requests for timeout, every 4 sec. (approx.)
        lda     cpnopnf
        ana     a
```

```
                rz
                lxi     h,rqosegn
                mov     d,m
                lxi     h,cpcmtdcnt
                mov     e,m
        rqota:
                mov     a,d
                call    seqnbnd
                mov     d,a
                push    d
                call    lnsq
                call    cplncnt
                pop     d
                push    d
                inr     m
                cp      rqchk
                pop     d
                inr     d
                dcr     e
                jnz     rqota
                ret
        cpbin:  ;put a in cp buf
                lhld    cpinptr
                mov     m,a     ;put away char
                inx     h
                shld    cpinptr
                ret
        cpopnchk:       ;return to caller's caller if cp open
                lda     grtrackf
                dcr     a
                lxi     h,cpnopnf
                ora     m
                lxi     h,cpcmtdcnt
                ora     m
                rz
                inx     sp      ;something's happening
                inx     sp
                ret
;graphics cursor
        gfbits: ;move (h)*2 to bc, get d bits to a
                mov     b,m
                inx     h
                mov     a,m
                ral             ;times 2
                mov     c,a
                mov     a,b
                ral
                mov     b,a     ;times 2 done
                jnc     gnbits  ;no overflow
                lxi     b,0FFFFh        ;overflow, set to max
                jmp     gnbits
        cnvgraphics:    ;cont. of convrt  code for graphics
                lda     grcurf  ;also clears cy, used below
                ana     a
                rz              ;a-n cursor, done
                mov     a,b     ;gr. cursor
                rar             ;mult by 1.5, cy was 0 from above
                mov     b,a
                mov     a,c
                rar
        add     c
        mov     m,a
        mov     a,b     ;rotated high bits
        dcx     h       ;points to high part
        adc     m       ;high bits
        mov     m,a
        ret
kbfgrsw:
        in      dsrdswitch
        ani     dsgrswkey       ;graphics cursor switch on keyboard?
        rz              ;no, great as normal key
        lxi     h,grcurf        ;ok, change flag
        mvi     a,1     ;being a little stingy with the bits
        xra     m       ;just in case we need them
        mov     m,a
        inx     sp      ;return to caller's caller
        inx     sp
        ret
kbgcoor:        ;send coordinates, graphics format
        mvi     m,44B   ;graphics format
        mvi     d,6     ;char count
        jmp     kbgcub
kbgcur: ;do graphics cursor for ep
        mvi     m,45B   ;identify as graphics format
        inx     h
        mov     m,c     ;char code
        mvi     d,7     ;char count
kbgcub:
        inx     h
        push    d
        lxi     d,xcur  ;prepare to do x
        call    st26    ;puts out two six bit chunks
        lxi     d,ycur
        call    st26    ;do y chunks
        pop     d
        jmp     kben2   ;update pointers and count
st26:   ;move cursor at (d) to (h) in two six bit pieces (+40B)
        push    h
        xchg
        mvi     d,4 ;bits in first chunk
        call    gfbits ;loads bits to bc, gets first 6 in a
        pop     h       ;get back dest. pointer
        adi     60B
        mov     m,a
        inx     h
        mvi     d,6     ;bits in second
        call    gnbits  ;get next six bits to a
        adi     40B
        mov     m,a
        inx     h
        ret
chkgrcur:
        lxi     h,grcurf        ;see if gr. cursor active
        mov     a,m
        ana     a
        jz      ncurb   ;no, a-n cursor
        inx     h       ;see if we should send gr. cur to tek.
        ana     m
        inx     h
        ana     m
```

```
                rz      ;no, one of them is zero
                lda     cpcmtdcnt       ;how is the buffer
                cpi     18
                rp      ;too full
                xra     a       ;ok, do it
                mov     m,a     ;clear curchflag
                call    tpnxtln ;get a buffer
                mvi     m,6     ;six chars for a cursor position
                inx     h       ;to first char of buffer
                shld    cpinptr ;pointer for cpbin
                mvi     a,35B   ;set graph mode
                call    cpbin
                lxi     h,ycur
                mvi     c,140B  ;base code for lo x
                call    st25
                lxi     h,xcur
                mvi     c,100B
                call    st25
                mvi     a,37B   ;turn off graph mode
                call    cpbin
                ret
        st25:   ;store 2 5-bit parts in cp buffer
                push    b       ;c has lo x or lo y base
                mvi     d,5
                call    gfbits  ;load (h) to bc, get first 5 bits to a
                adi     40B     ;hix or hi y base
                call    cpbin   ;put in buffer (only uses a,hl)
                call    gnbits
                pop     b       ;get c back (lo x or lo y base)
                add     c
                call    cpbin
                ret
;cp, graphics output
        cpoutchk:       ;out to cp if char waiting and port ready
                lda     cpnopnf
                cpi     1
                jnz     cpck3   ;no new req. allowed
                lxi     h,cprqcnt       ;if cp needs
                mov     a,m     ;request sent, do it if appropriate
                ana     a
                jz      cpck3   ;none pending
        cpck4:
                lda     cpcmtdcnt       ;see if space there fo a req.
                cpi     cpnline-1
                jp      cpck3
                lda     procnt  ;see if buffer ok
                ana     a
                jnz     cpck3   ;no
                lda     cordmd  ;make sure in coordinate mode
                ana     a
                jz      cpck3
                dcr     m       ;pt to cprqcnt
                inx     h       ;pt to cpcmtdcnt
                lda     rqoseqn
                add     m
                inr     m
                call    seqnbnd
                call    rquest
        cpck3:
                lxi     h,cpcnt
        mov     a,m
        ana     a
        jz      cpck5           ;nothing wating for cp
        in      psrd    ;cp status
        ana     a
        rz              ;not ready
        in      ksrd3   ;check CP off button
        ani     mdswcp
        cnz     cpholdchk
        lda     cprptcnt
        dcr     a
        jp      cpck12
        dcr     m       ;ok, do it
        lhld    cpoutptr
        mov     c,m     ;char to go
        inx     h       ;bump pointer
        shld    cpoutptr
        jnz     cpck3b  ;more to do after this
        push    psw
        in      ksrd2   ;end of a buffer, turn off cp lite
        ani     cpofflite
        out     kswr
        pop     psw
        cpck3b:
        inr     a       ;cprptcnt
        jm      cpck11
        mov     a,c     char
        ana     a
        cpi     26B     ;^V
        jz      cpck8
        jmp     cpouch  ;put it out

cpck12:
        sta     cprptcnt
        lda     cprptchar
        jmp     cpouch  ;put it out

cpck11:
        inr     a
        jz      cpck9
        sta     cprptcnt
        mov     a,c
        cpi     'A      ;only one defined now
        jz      cpck10  ;yes, doing rubouts
        cpi     'B      ;sequence for "esc"?
        jz      cpck10b ;yes
        call    cpouch  no, not special
        xra     a
        sta     cprptcnt
        ret

cpck10:
        mvi     a,177B  ;rubout
        sta     cprptchar
        ret

cpck10b:
        xra     a
        sta     cprptcnt
        mvi     a,33B
        jmp     cpouch

cpck9:
        mov     a,c
        sui     40B
        sta     cprptcnt
        ret

cpck8:  ;found ^V
        mvi     a,-2
        sta     cprptcnt
        ret

cpouch: ;char in a
        lhld    cpoutrv
        pchl

cpsio:  ;output to cp port
        out     psio
        ret
cpholdchk:      ;hold if CP (by returning to caller's caller)
        lda     cpnopnf
        ana     a
        jnz     cphckb
        lda     cpnopver
        cpi     cpnfunny        ;funny#
        rnz             ;no, must be graphics, dont hold
        cphckb:
        inx     sp      ;return to caller's caller
        inx     sp
        ret
cpck5:  ;see if next buffer rcvd
        lda     cpcmtdcnt
        ana     a
        rz              ;no requests outstanding
        lda     rqoutl  ;see if next buff rcvd
        call    cplncnt ;get to the buffer
        mov     a,m     ;if <0, not received yet
        ana     a
        rm      ;not in yet
        sta     cpcnt   ;ok, put ch. cout here
        inx     h       ;to first char of buffer
        shld    cpoutptr        ;put char pointer here
        lxi     h,cpcmtdcnt
        jz      cpck14  ;may be "EOF" record
        in      ksrd2   ;turn on cp lite
        ori     cponlite
        out     kswr
        cpck15:
        dcr     m       ;dec. # requests outstanding
        inx     h       ;rqosegn
        mov     a,m
        inr     a
        call    segnbnd ;check for wrap
        mov     m,a
        lxi     h,rqoutl        ;could maybe inx?
        inr     m
        mov     a,m
        sui     cpnline ;check for wraparound
        jc      cpck3   ;no
        mov     m,a
        jmp     cpck3
cpck14:
        lda     cpnopnf ;new type?
        ana     a
        jz      cpck15  ;no
        xra     a       ;clear flags
        mov     m,a     ;cpcmdtcnt
        sta     cpnopnf
        sta     cpnopver
        mvi     c,40B   ;0 count for rqstn
        lda     rqoseqn
        call    rqstn   ;request 0 length record with same # as "EOF"
                ;tells ep we finished successfully
        ret
cpinit:
        xra     a
        sta     cpnopnf
        lda     cpnopver
        cpi     cpnfunny
        rz      ;a reopen is possible
        lxi     d,tab1+49
        lxi     b,cpnchar+1
        mvi     a,cpnline
        lxi     h,tab1
        pupx2:
        mov     m,e
        inx     h
        mov     m,d
        inx     h
        xchg
        dad     b
        xchg
        dcr     a
        jnz     pupx2
        mvi     a,40B
        sta     rqoseqn
        xra     a
        sta     cprqcnt
        sta     cpcmtdcnt
        sta     cpcnt
        sta     cprptcnt
        sta     rqoutl
        ret
cpincnt:
        lxi     h,tab1
        jmp     lncntb
seqnbnd:        ;make sure seq# in [40B,175B]
        ;in practice, must be called with a in [40B,335B]
        cpi     176B
        rc
        sui     176B-40B        ;char was >=176B
        ret
lnsq:   ;convert seq# to buffer (line) number
        lxi     h,rqoseqn
        sub     m       ;delta seqn
        jnc     lnsqb   ;no wrap involved
        adi     176B-40B
lnsqb:
        mov     c,a
```

```
                lda     cpcmtdcnt
                sub     c
                mvi     a,377B  ;in case delta too big
                rc              ;was too big
                lda     rqoutl  ;it's ok
                add     c
                cpi     cpnline ;MOD cpnline
                rc
                sui     cpnline
                ret
        rqchk:          ;called only from rqota
                dcr     m       ;undo previous inr
                rp              ;don't touch positive ones
                lda     procnt  ;see if room in ep outbuf
                cpi     22
                rp              ;no
                lda     cordmd  ;make sure in coordinate mode
                ana     a
                rz
                mov     a,d     ;seqn #
        rquest: ;request to ep, seq# in a
                mvi     c,cpnchar+40B
        rqstn:
                push    b       ;save count
                push    psw
                mvi     a,34B   ;lpesc
                di              ;make sure no keyboard chars get in
                call    kbncrd
                mvi     a,47B   ;request code
                call    kbncrd
                lda     cpdev   ;device received with "open"
                call    kbncrd
                pop     psw
                call    kbncrd  ;must preserve a
                call    lnsq    ;get line#
                call    cplncnt ;get cnt ptr to h
                mvi     m,-rqictime
                pop     b       ;get count back in c
                mov     a,c
                call    kbncrd
                ei
                ret
END
```