CBIOS & .ASM

Provided with M26
1/13/82

MAC CBIOSS B+S

```
*****************************************************************
*                                                               *
* Cbios for CP/M Ver 2.2 for Disk Jockey 2D controller (all     *
* revs, and models A & B). Handles diskettes with sector sizes  *
* of 128 bytes single density, 256, 512, 1Ø24 bytes double      *
* density. There are conditional assemblies for Diskus Hard     *
* Disk Controller.                                              *
*                                                               *
* Written by Bobby Dale Gifford.                                *
* 12/3/8Ø                                                       *
*                                                               *
* Disk Map of sectors used by Cold Boot, Warm Boot, Firmware,   *
* and CP/M:                                                     *
*                                                               *
* trk Ø sec  1 = First sector of cold boot.            e7ØØh    *
*     Ø      2 = Cold boot 256.                          8Øh    *
*     Ø      3 = Cold boot 512.                          8Øh    *
*     Ø      4 = Cold boot 1Ø24.                         8Øh    *
*     Ø      5 = Warm boot 256.                          8Øh    *
*     Ø      6 = Warm boot 512.                          8Øh    *
*     Ø      7 = Warm boot 1Ø24.                         8Øh    *
*     Ø      8 = Cold/Warm boot.                       2cØØh    *
*     Ø      9 = Firmware.                             e4ØØh    *
*     Ø     1Ø = Firmware+8Øh.                         e48Øh    *
*     Ø     11 = Firmware+1ØØh                         e5ØØh    *
*     Ø     12 = Firmware+18Øh.                        e58Øh    *
*     Ø     13 = Firmware+2ØØh.                        e6ØØh    *
*     Ø     14 = Firmware+28Øh.                        e68Øh    *
*     Ø     15 = Firmware+3ØØh.                        e7ØØh    *
*     Ø     16 = Firmware+38Øh.                        e78Øh    *
*     Ø     17 = CCP.                                  27ØØh    *
*     Ø     1Ø = CCP+8Øh.                              278Øh    *
*     Ø     12 = CCP+1ØØh.                             28ØØh    *
*     Ø     14 = CCP+18Øh.                             288Øh    *
*     Ø     16 = CCP+2ØØh.                             29ØØh    *
*     Ø     18 = CCP+28Øh.                             298Øh    *
*     Ø     2Ø = CCP+3ØØh.                             2aØØh    *
*     Ø     22 = CCP+38Øh.                             2a8Øh    *
*     Ø     24 = CCP+4ØØh.                             2bØØh    *
*     Ø     26 = CCP+48Øh.                             2b8Øh    *
*     1        = Rest of CP/M.                    2cØØh-4fffh   *
*                                                               *
*****************************************************************

        title   '*** Cbios For CP/M Ver. 2.2 ***'

*****************************************************************
*                                                               *
* The following revision number is in reference to the CP/M    *
* 2.2 Cbios.                                                    *
*                                                               *
*****************************************************************

revnum  equ     28              ;Cbios revision number
cpmrev  equ     22              ;CP/M revision number

*****************************************************************
*                                                               *
* The following equates set up the relationship between the     *
* 2D floppies and the Hard Disk Controllers.                    *
*                                                               *
*****************************************************************

first   equ     ~~1~~ 0          ;Ø = Floppies are A,B,C,D drives and
                                ;       Hard Disk are E,F,G,H
```

← ✓ 1

```
                                        ;1 = Hard Disks are A,B,C,D drives and
                                        ;       Floppies are E,F,G,H
maxhd   equ     1                       ;Set to number of hard disks
maxflop equ     ~~1~~ 2                 ;Set to number of floppies          ✓

M26     equ     1                       ;Set only one of these variables
M2Ø     equ     Ø
M1Ø     equ     Ø

        if      m1Ø or m2Ø
sdelay  equ     Ø                       ;Software head settle delay (Ø = no, 1 = yes)
        else
sdelay  equ     1
        endif

mrev    equ     26*m26+2Ø*m2Ø+1Ø*m1Ø    ;Hard disk type
logdsk  equ     3*m26+3*m2Ø+2*m1Ø       ;Logical disks per drive
hdspt   equ     32*m26+21*m2Ø+21*m1Ø    ;Sectors per track

****************************************************************
*                                                              *
* The following equates selects the type of I/O to be included *
* with the Cbios.                                              *
*                                                              *
****************************************************************

iotype  equ     ~~1~~ 2                 ;Ø = No I/O, jmp to self configuration
                                        ;1 = Disk Jockey 2D I/O as console
                                        ;2 = Switchboard as console
                                        ;3 = SOL as console            Delete ✓
                                        ;4 = EXIDY as console
swbd    equ     1                       ;Ø = No switchboard printer implementations
                                        ;1 = Include Switchboard routines

        if      (iotype eq 2) or swbd
base    equ     Ø                       ;Base of the SWITCHBOARD
        endif

****************************************************************
*                                                              *
* The following equates relate the Thinker Toys 2D controller. *
* If the controller is non standard (ØEØØØH) only the ORIGIN   *
* equate need be changed. This version of the Cbios will work  *
* with 2D controller boards rev Ø, 1, 3, 3.1, 4, Model B.      *
*                                                              *
****************************************************************


        if      maxflop ne Ø    ;Include Discus 2D ?
origin  equ     ØEØØØH
djram   equ     origin+4ØØh     ;Disk Jockey 2D RAM address
djboot  equ     djram           ;Disk Jockey 2D initialization
djcin   equ     origin+3h       ;Disk Jockey 2D character input routine
djcout  equ     origin+6h       ;Disk Jockey 2D character output routine
djhome  equ     djram+9h        ;Disk Jockey 2D track zero seek
djtrk   equ     djram+Øch       ;Disk Jockey 2D track seek routine
djsec   equ     djram+Øfh       ;Disk Jockey 2D set sector routine
djdma   equ     djram+Ø12h      ;Disk Jockey 2D set DMA address
djread  equ     djram+15h       ;Disk Jockey 2D read routine
djwrite equ     djram+18h       ;Disk Jockey 2D write routine
djsel   equ     djram+1bh       ;Disk Jockey 2D select drive routine
djtstat equ     origin+21h      ;Disk Jockey 2D terminal status routine
djstat  equ     djram+27h       ;Disk Jockey 2D status routine
djerr   equ     djram+2ah       ;Disk Jockey 2D error, flash led
djden   equ     djram+2dh       ;Disk Jockey 2D set density routine
djside  equ     djram+3Øh       ;Disk Jockey 2D set side routine
```

```
dblsid  equ     8               ;Side bit from controller
        endif

*****************************************************************
*                                                               *
* The following equates are for the Diskus Hard disk wanted.    *
*                                                               *
*****************************************************************


        if      maxhd ne Ø      ;Want Hard Disk included ?
hdorg   equ     5Øh             ;Hard Disk Controller origin
hdstat  equ     hdorg           ;Hard Disk Status
hdcntl  equ     hdorg           ;Hard Disk Control
hddata  equ     hdorg+3         ;Hard Disk Data
hdfunc  equ     hdorg+2         ;Hard Disk Function
hdcmnd  equ     hdorg+1         ;Hard Disk Command
hdreslt equ     hdorg+1         ;Hard Disk Result
retry   equ     2               ;Retry bit of result
tkzero  equ     1               ;Track zero bit of status
opdone  equ     2               ;Operaction done bit of status
complt  equ     4               ;Complete bit of status
tmout   equ     8               ;Time out bit of status
wfault  equ     1Øh             ;Write fault bit of status
drvrdy  equ     2Øh             ;Drive ready bit of status
index   equ     4Øh             ;Index bit of status
pstep   equ     4               ;Step bit of function
nstep   equ     Øfbh            ;Step bit mask of function
hdrlen  equ     4               ;Sector header length
seclen  equ     512             ;Sector data length
wenabl  equ     Øfh             ;Write enable
wreset  equ     Øbh             ;Write reset of function
scenbl  equ     5               ;Controller control
dskclk  equ     7               ;Disk clock for control
mdir    equ     Øf7h            ;Direction mask for function
null    equ     Øfch            ;Null command
idbuff  equ     Ø               ;Initialize data command
isbuff  equ     8               ;Initialize header command
rsect   equ     1               ;Read sector command
wsect   equ     5               ;Write sector command
        endif

*****************************************************************
*                                                               *
* CP/M system equates. If reconfiguration of the CP/M system    *
* is being done, the changes can be made to the following       *
* equates.                                                      *
*                                                               *
*****************************************************************

     msize   equ     ~~28~~ 56          ;Memory size of target CP/M          <---- Size
9000 bias    equ     (msize-2Ø)*1Ø24 ;Memory offset from 2Øk system  9000
B700 ccp     equ     27ØØh+bias      ;Console command processor      B700
BF00 bdos    equ     ccp+8ØØh        ;BDOS address
C000 bios    equ     ccp+16ØØh       ;CBIOS address C000
5A00 offsetc equ     27ØØh-bios      ;Offset for sysgen 5A00
     cdisk   equ     4               ;Address of last logged disk
     buff    equ     8Øh             ;Default buffer address
     tpa     equ     1ØØh            ;Transient memory
     intioby equ     ~~Ø~~ 192           ;Initial IOBYTE                      <---- IOByte
     iobyte  equ     3               ;IOBYTE location
     wbot    equ     Ø               ;Warm boot jump address
     entry   equ     5               ;BDOS entry jump address

*****************************************************************
*                                                               *
* The following are internal Cbios equates. Most are misc.      *
* constants.                                                  *
*                                                             *
***************************************************************

retries equ     10              ;Max retries on disk i/o before error
acr     equ     0dh             ;A carriage return
alf     equ     0ah             ;A line feed
clear   equ     19h             ;clear screen for MSDV
***************************************************************
*                                                             *
* The jump table below must remain in the same order, the     *
* routines may be changed, but the function executed must be  *
* the same.                                                   *
*                                                             *
***************************************************************

        org     bios            ;CBIOS starting address

        jmp     cboot           ;Cold boot entry point
wboote  jmp     wboot           ;Warm boot entry point

        if      iotype ne 0
        jmp     const           ;Console status routine
        jmp     conin           ;Console input
cout    jmp     conout          ;Console output
        jmp     list            ;List device output
        jmp     punch           ;Punch device output
        jmp     reader          ;Reader device input
        ~~else~~
        ~~jmp     $~~               ;Console status routine
        ~~jmp     $~~               ;Console input
~~cout    jmp     $~~               ;Console output
        ~~jmp     $~~               ;List device output
        ~~jmp     $~~               ;Punch device output
        ~~jmp     $~~               ;Reader device input
        endif
        jmp     home            ;Home drive
        jmp     setdrv          ;Select disk
        jmp     settrk          ;Set track
        jmp     setsec          ;Set sector
        jmp     setdma          ;Set DMA address
        jmp     read            ;Read the disk
        jmp     write           ;Write the disk

        if      iotype ne 0
        jmp     listst          ;List device status
        ~~else~~
        ~~jmp     $~~               ;List device status
        endif
        jmp     sectran         ;Sector translation

        if      maxflop ne 0
djdrv   jmp     djsel           ;Hook for SINGLE.COM program
        else
        ~~jmp     donop~~
        ~~endif~~

***************************************************************
*                                                             *
* Terminal driver routines. Iobyte is initialized by the cold *
* boot routine, to modify, change the "intioby" equate. The   *
* I/O routines that follow all work exactly the same way. Using *
* iobyte, they obtain the address to jump to in order to execute*
* the desired function. There is a table with four entries for *
* each of the possible assignments for each device. To modify  *
```

aeTX equ 3 ; A ETX char
aack equ 6 ; a ack char

TERM DRIVERS
Delete entirely —

```
* the I/O routines for a different I/O configuration, just        *
* change the entries in the tables.                               *
*                                                                 *
*******************************************************************

        if      iotype eq 0                    ;Blank I/O
        ds      512
        else
        if      iotype eq 1
tinit   jmp     tstart                         ;Disk Jockey 2D I/O configuation
cstty   jmp     csdj
citty   jmp     djcin
cotty   jmp     djcout

clear   equ     1ah                            ;Clear screen char on ADM3 terminal
tstart  mvi     c,clear                        ;Initialize the terminal routine
        jmp     cout

csdj    call    djtstat
        mvi     a,0
        rnz
        dcr     a
        ret
        endif

        if      iotype eq 2
~~tinit   jmp     sbinit~~                         ;Switchboard as console configuration
cstty   jmp     swbdst
citty   jmp     swbdin
cotty   jmp     swbdout

~~clear   equ     1ah                            ;Clear screen char on ADM3 terminal~~
~~sbinit  mvi     c,clear~~                        ;Initialize the terminal routine
        ~~jmp     cout~~

swbdst  in      base+2
        ani     4
        xri     4
        mvi     a,0
        rz
        dcr     a
        ret

swbdin  in      base+2                         ;Get switchboard status
        ani     4
        jz      citty
        in      base
        ani     7fh                            ;Strip off parity
        ret

swbdout in      base+2                         ;Check status
        ani     8
        jz      cotty
        mov     a,c
        out     base
        ret
        endif

        if      iotype eq 3
tinit   jmp     solinit                        ;Sol I/O configuration
cstty   jmp     cssol
citty   jmp     solin
cotty   jmp     solout

solos   equ     0c000h
sinp    equ     solos+1fh
```

Get From old BIOS at D73A

GET TINIT

```
CLEAR eQU 19H    ; Clear Screen for MSDV
TINIT mvi c,clear ; get ready to clear screen
      in  7       ; get keyboard interlock byte
      ani 1       ; get bit 1 only
      adi intioby ; add it to intioby
      sta iobyte  ; initialize IOBITE
      jmp cout    ; clear screen
```

```
sout    equ     solos+19h
clear   equ     Øbh

solinit mvi     c,clear
        jmp     cout

cssol   lda     statchr
        ana     a
        mvi     a,Øffh
        rnz
        call    sinp
        jz      zzret
        sta     statchr
        mvi     a,Øffh
        ret
zzret   xra     a
        ret

solin   mvi     b,Ø
statchr equ     $-1
        xra     a
        sta     statchr
        mov     a,b
        ani     7fh
        rnz
        call    cssol
        jmp     solin

solout  mov     a,c
        mov     b,c
        cpi     acr
        sta     lastchr
        jnz     dosout
        mvi     a,Ø
lastchr equ     $-1
        cpi     acr
        rz
dosout  mov     a,c
        sta     lastchr
        jmp     sout
        endif

        if      iotype eq 4
tinit   jmp     exinit          ;Exidy I/O configuration
cstty   jmp     csexdy
citty   jmp     exin
cotty   jmp     exout

exidy   equ     Øe000h          ;Exidy monitor
exdyin  equ     exidy+9         ;Exidy input
exdyout equ     exidy+Øch       ;Exidy output
clear   equ     1ah

exinit  mvi     c,clear         ;Initialize the terminal routine
        jmp     cout

csexdy  lda     statchr
        ana     a
        mvi     a,Øffh
        rnz
        call    exdyin
        jz      zzret
        sta     statchr
        mvi     a,Øffh
        ret
zzret   xra     a
```

```
        ret

exin    mvi     b,Ø
statchr equ     $-1
        xra     a
        sta     statchr
        mov     a,b
        ani     7fh
        rnz
        call    csexdy
        jmp     exin

exout   mov     a,c
        jmp     exdyout
        endif

***************************************************************
*                                                             *
* The following equates set all these I/O devices to output   *
* in the selected method.                                     *
*                                                             *
***************************************************************

cocrt   equ     $               ;Output from crt
coucl   equ     $               ;Output from user console 1
coptp   equ     $               ;Output from paper tape punch
coup1   equ     $               ;Output from user punch 1
coup2   equ     $               ;Output from user punch 2
colpt   equ     $               ;Output from line printer

        if      swbd
coswbd  in      base+2
        ani     8Øh             ;Wait until ok to send
        jz      coswbd
        mov     a,c             ;output the character
        out     base+1
        ret
        else
        if      iotype eq 3
aout    equ     solos+Ø1ch
        mvi     a,2
        mov     b,c
        jmp     aout
        else
        if      iotype eq 4
exparrl equ     exidy+21h               ;Exidy parallel output
        jmp     exparrl
        else
        jmp     cotty           ;Default to console
        endif
        endif
        endif

***************************************************************
*                                                             *
* Custom I/O printer driver for Diablo printer with 12ØØ baud *
* ETX/ACK handshake.                                          *
*                                                             *
***************************************************************

        if      swbd
aetx    equ     3               ;A ETX char
aack    equ     6               ;A ACK char
coull   call    colpt           ;Output the character
        mvi     a,5Ø
count   equ     $-1
```

```
        dcr     a
        sta     count
        rnz
        mvi     a,50
        sta     count
        mvi     c,aetx
        call    colpt
pwait   call    ciptr
        cpi     aack
        jnz     pwait
        ret
        else
coull   equ     colpt           ;Otherwise default to printer
        endif

*****************************************************************
*                                                               *
* The following equates set the input from the devices to come  *
* from the selected method.                                     *
*                                                               *
*****************************************************************

ciucl   equ     $               ;Input from user console 1
cicrt   equ     $               ;Input from crt
ciurl   equ     $               ;Input from user reader 1
ciur2   equ     $               ;Input from user reader 2
ciptr   equ     $               ;Input from paper tape reader

        if      swbd
ciswbd  in      base+2          ;Input from paper tape reader
        ani     40h             ;Wait for character
        jz      ciswbd
        in      base+1
        ani     7fh             ;Strip off the parity
        ret
        else
        jmp     citty           ;Default to input from tty
        endif

*****************************************************************
*                                                               *
* The following equates cause the devices to get status in      *
* the selected way.                                             *
*                                                               *
*****************************************************************

csurl   equ     $               ;Status of user reader 1
csur2   equ     $               ;Status of user reader 2
csptr   equ     $               ;Status of paper tape reader
csucl   equ     $               ;Status of user console 1
cscrt   equ     $               ;Status from crt

        if      swbd
csswbd  in      base+2
        ani     40h             ;Strip of data ready bit
        xri     40h             ;Make correct polarity
        mvi     a,0
        rz
        dcr     a
        ret
        else
        jmp     cstty           ;Default to status from console
        endif

*****************************************************************
*                                                               *
* List device status routines.                                   *
*                                                                *
******************************************************************


        if      swbd
lslpt   in      base+2          ;All other devices wait
        ani     8Øh
        rz
        else
lslpt   equ     $
        endif
ready   mvi     a,Øffh
        ret

******************************************************************
*                                                                *
* const: get the status for the currently assigned console       *
*        device. The console device can be gotten from iobyte,   *
*        then a jump to the correct console status routine is    *
*        performed.                                              *
*                                                                *
******************************************************************

const   lxi     h,cstble        ;Beginning of jump table
        jmp     coninl          ;Select correct jump

******************************************************************
*                                                                *
* csreader: if the console is assigned to the reader then a      *
*           jump will be made here, where another jump will      *
*           occur to the correct reader status.                  *
*                                                                *
******************************************************************

csreadr lxi     h,csrtble       ;Beginning of reader status table
        jmp     readera

******************************************************************
*                                                                *
* conin: take the correct jump for the console input routine.    *
*        The jump is based on the two least significant bits of  *
*        iobyte.                                                 *
*                                                                *
******************************************************************

conin   call    flush           ;Flush the disk buffer
        lxi     h,citble        ;Beginning of character input table

*
* Entry at coninl will decode the two least significant bits
* of iobyte. This is used by conin,conout, and const.
*

coninl  lda     iobyte
        ral

*
* Entry at seldev will form an offset into the table pointed
* to by H&L and then pick up the address and jump there.
*

seldev  ani     6h              ;Strip off unwanted bits
        mvi     d,Ø             ;Form offset
        mov     e,a
        dad     d               ;Add offset
        mov     a,m             ;Pick up high byte
        inx     h
        mov     h,m             ;Pick up low byte
        mov     l,a             ;Form address
dopchl  pchl                    ;Go there !

*****************************************************************
*                                                               *
* conout: take the proper branch address based on the two least *
*         significant bits of iobyte.                           *
*                                                               *
*****************************************************************

conout  push    b               ;Save the character
        call    flush           ;Flush the disk buffer
        pop     b               ;Restore the character
        lxi     h,cotble        ;Beginning of the character out table
        jmp     coninl          ;Do the decode

*****************************************************************
*                                                               *
* reader: select the correct reader device for input. The       *
*         reader is selected from bits 2 and 3 of iobyte.       *
*                                                               *
*****************************************************************

reader  lxi     h,rtble         ;Beginning of reader input table

*
* Entry at readera will decode bits 2 & 3 of iobyte, used
* by csreader.
*

readera lda     iobyte

*
* Entry at readerl will shift the bits into position, used
* by list and punch.
*

readrl  rar
        jmp     seldev

*****************************************************************
*                                                               *
* punch: select the correct punch device. The selection comes   *
*        from bits 4&5 of iobyte.                               *
*                                                               *
*****************************************************************

punch   lxi     h,ptble         ;Beginning of punch table
        lda     iobyte

*
* Entry at pnchl rotates bits a little more in prep for
* seldev, used by list.
*

pnchl   rar
        rar
        jmp     readrl

*****************************************************************
*                                                               *
* list: select a list device based on bits 6&7 of iobyte        *
*                                                               *
*****************************************************************
```

```
list    lxi     h,ltble         ;Beginning of the list device routines
listl   lda     iobyte
        rar
        rar
        jmp     pnchl

****************************************************************
*                                                              *
* Listst: Get the status of the currently assigned list device *
*                                                              *
****************************************************************

listst  lxi     h,lstble        ;Beginning of the list device status
        jmp     listl

****************************************************************
*                                                              *
* If customizing I/O routines is being performed, the table    *
* below should be modified to reflect the changes. All I/O     *
* devices are decoded out of iobyte and the jump is taken from *
* the following tables.                                        *
*                                                              *
****************************************************************

*
* console input table
*

citble  dw      citty           ;Input from tty
        dw      cicrt           ;Input from crt
        dw      reader          ;Input from reader
        dw      ciucl           ;Input from user console 1

*
* console output table
*

cotble  dw      cotty           ;Output to tty
        dw      cocrt           ;Output to crt
        dw      list            ;Output to list device
        dw      coucl           ;Output to user console 1

*
* list device table
*

ltble   dw      cotty           ;Output to tty
        dw      cocrt           ;Output to crt
        dw      colpt           ;Output to line printer
        dw      coull           ;Output to user line printer 1

*
* punch device table
*

ptble   dw      cotty           ;Output to the tty
        dw      coptp           ;Output to paper tape punch
        dw      coupl           ;Output to user punch 1
        dw      coup2           ;Output to user punch 2

*
* reader device input table
*

rtble   dw      citty           ;Input from tty
```

```
        dw      ciptr           ;Input from paper tape reader
        dw      ciurl           ;Input from user reader 1
        dw      ciur2           ;Input from user reader 2

*
* console status table
*

cstble  dw      cstty           ;Status of tty
        dw      cscrt           ;Status from crt
        dw      csreadr         ;Status from reader
        dw      csucl           ;Status from user console 1

*
* status from reader device
*

csrtble dw      cstty           ;Status from tty
        dw      csptr           ;Status from paper tape reader
        dw      csurl           ;Status from user reader 1
        dw      csur2           ;Status of user reader 2

*
* Status from list device
*

lstble  dw      ready           ;Console always ready
        dw      ready           ;Get list status
        dw      lslpt
        dw      lslpt
        endif
```

END OF TERM DRIVERS

```
*****************************************************************
*                                                               *
* Gocpm is the entry point from cold boots, and warm boots. It  *
* initializes some of the locations in page Ø, and sets up the  *
* initial DMA address (8Øh).                                    *
*                                                               *
*****************************************************************

gocpm   lxi     h,buff          ;Set up initial DMA address
        call    setdma
        mvi     a,(jmp)         ;Initialize jump to warm boot
        sta     wbot
        sta     entry           ;Initialize jump to BDOS
        lxi     h,wboote        ;Address in warm boot jump
        shld    wbot+1
        lxi     h,bdos+6        ;Address in BDOS jump
        shld    entry+1
        xra     a               ;A <- Ø
        sta     bufsec          ;Disk Jockey buffer empty
        sta     bufwrtn         ;Set buffer not dirty flag
        lda     cdisk           ;Jump to CP/M with currently selected disk in C
        mov     c,a
        lda     cwflg
        ana     a
        lxi     d,coldbeg       ;Beginning of initial command
        mvi     a,coldend-coldbeg+1 ;Length of command
        jz      cldcmnd
        lxi     d,warmbeg
        mvi     a,warmend-warmbeg+1
cldcmnd lxi     h,ccp+8         ;Command buffer
        sta     ccp+7
        mov     b,a
        call    movlop
        lda     cwflg
        ana     a
        lda     autoflg
        jz      cldbot
        rar
cldbot  rar
        jc      ccp
        jmp     ccp+3           ;Enter CP/M

cwflg   db      Ø               ;Cold/warm boot flag

****************************************************************
*                                                              *
* The following byte determines if an initial command is to be *
* given to CP/M on warm or cold boots. The value of the byte is *
* used to give the command to CP/M:                            *
*                                                              *
* Ø = never give command.                                      *
* 1 = give command on cold boots only.                         *
* 2 = give the command on warm boots only.                     *
* 3 = give the command on warm and cold boots.                 *
*                                                              *
****************************************************************

autoflg db      ~~Ø~~ 1         ;Auto command feature

****************************************************************
*                                                              *
* If there is a command inserted here, it will be given if the *
* auto feature is enabled.                                     *
*       For Example:                                           *
*                                                              *
*       coldbeg db      'MBASIC MYPROG'                        *
*       coldend db      Ø                                      *
*                                                              *
* will execute microsoft basic, and mbasic will execute the    *
* "MYPROG" basic program.                                      *
*                                                              *
****************************************************************

coldbeg db      'SUBMIT STARTUP'        ;Cold boot command goes here
coldend db      Ø
warmbeg db      ''              ;Warm boot command goes here
warmend db      Ø

****************************************************************
*                                                              *
* Wboot loads in all of CP/M except the CBIOS, then initializes *
* system parameters as in cold boot. See the Cold Boot Loader  *
* listing for exactly what happens during warm and cold boots. *
*                                                              *
****************************************************************

wboot   lxi     sp,tpa          ;Set up stack pointer
        mvi     a,1
wflg    equ     $-1             ;Test if beginning or
        ana     a               ;       ending a warm boot
        mvi     a,1
        sta     wflg
        sta     cwflg           ;Set cold/warm boot flag
        jz      gocpm
        xra     a
        sta     wflg
        mov     c,a

        if      (maxhd ne Ø) and first  ;Supply Warm Boot from Hard Disk ?
        lxi     h,ccp-2ØØh      ;Initial DMA address
        push    h
        sta     head
        mvi     a,4
        push    psw             ;Save first sector
        call    hddrv           ;Select drive A
        mvi     c,Ø
        call    hdtrk           ;Home the drive
warmlod pop     psw             ;Restore sector
        pop     h               ;Restore DMA address
        inr     a
        sta     hdsectr
        cpi     16              ;Past BDOS ?
        jz      wboot           ;Yes, all done
        inr     h               ;Update DMA address
        inr     h
        shld    hdadd
        push    h
        push    psw
warmrd  lxi     b,retries*1ØØh+Ø ;Retry counter
wrmread push    b               ;Save the retry count
        call    hdread          ;Read the sector
        pop     b
        jnc     warmlod         ;Test for error
        dcr     b               ;Update the error count
        jnz     wrmread         ;Keep trying if not to many errors
        hlt                     ;Can't warm boot
        endif

        if      (maxflop ne Ø) and not first ;Supply Warm Boot from 2D ?
        call    djdrv           ;Select drive A
        mvi     c,Ø             ;Select single density
        call    djden
        mvi     c,Ø             ;Select side Ø
        call    djside
        mvi     a,15            ;Initialize the sector to read
        sta     newsec
        lxi     h,ccp-1ØØh      ;And the DMA address
        shld    newdma
        call    warmlod         ;Read in CP/M
        lxi     b,ccp+5ØØh      ;Load address for rest of warm boot
        call    djdma
        mvi     c,8
        call    djsec
        call    warmrd
        jmp     ccp+5Ø3h

warmlod mvi     a,15            ;Previous sector
newsec  equ     $-1
        inr     a               ;Update the previous sector
        inr     a
        cpi     27              ;Was it the last ?
        jc      nowrap
        sui     9               ;Yes
        cpi     19
        rz
        lhld    newdma
        lxi     d,-48Øh
        dad     d
        shld    newdma
nowrap  sta     newsec          ;Save the new sector to read
        mov     c,a
        call    djsec
        lxi     h,ccp-1ØØh      ;Get the previous DMA address
newdma  equ     $-2
        lxi     d,1ØØh          ;Update the DMA address
        dad     d
        shld    newdma          ;Save the DMA address
        mov     b,h
        mov     c,l
        call    djdma           ;Set the DMA address
        call    warmrd
        jmp     warmlod

warmrd  lxi     b,retries*1ØØh+Ø;Maximum # of errors
wrmread push    b
        call    djtrk           ;Set the track
        call    djread          ;Read the sector
        pop     b
        rnc                     ;Continue if successful
        dcr     b
        jnz     wrmread         ;Keep trying
        jmp     djerr
        endif

*****************************************************************
*                                                               *
* Setsec just saves the desired sector to seek to until an      *
* actual read or write is attempted.                            *
*                                                               *
*****************************************************************

setsec  mov     h,b
        mov     l,c
        shld    cpmsec
donop   ret

*****************************************************************
*                                                               *
* Setdma saves the DMA address for the data transfer.           *
*                                                               *
*****************************************************************

setdma  mov     h,b             ;hl <- bc
        mov     l,c
        shld    cpmdma          ;CP/M dma address
        ret

*****************************************************************
*                                                               *
* Home is translated into a seek to track zero.                 *
*                                                               *
*****************************************************************

home    mvi     c,Ø             ;Track to seek to

*****************************************************************
*                                                               *
* Settrk saves the track # to seek to. Nothing is done at this  *
* point, everything is deffered until a read or write.          *
*                                                               *
*****************************************************************

settrk  mov     a,c             ;A <- track #
        sta     cpmtrk          ;CP/M track #
        ret

*****************************************************************
*                                                               *
* Sectran translates a logical sector # into a physical sector  *
* #.                                                            *
*                                                               *
*****************************************************************
```

```
        if      (maxhd ne Ø) and (maxflop ne Ø) ;Both types ?
sectran lda     cpmdrv          ;Get the Drive Number

        if      first
        cpi     maxhd*logdsk    ;Over the # of hard disks ?
        jc      tranhd
        else
        cpi     maxflop         ;Over the # of floppies ?
        jnc     tranhd
        endif
        endif

        if      (maxhd eq Ø) or (maxflop eq Ø) ;Just one type ?

sectran equ     $
        endif

        if      maxflop ne Ø    ;Floppy translation
tranfp  inx     b
        push    d               ;Save table address
        push    b               ;Save sector #
        call    getdpb          ;Get DPB address into HL
        mov     a,m             ;Get # of CP/M sectors/track
        ora     a               ;Clear cary
        rar                     ;Divide by two
        sub     c
        push    psw             ;Save adjusted sector
        jm      sidetwo
sidea   pop     psw             ;Discard adjusted sector
        pop     b               ;Restore sector requested
        pop     d               ;Restor address of xlt table
sideone xchg                    ;hl <- &(translation table)
        dad     b               ;bc = offset into table
        mov     l,m             ;hl <- physical sector
        mvi     h,Ø
        ret

sidetwo lxi     b,15            ;Offset to side bit
        dad     b
        mov     a,m
        ani     8               ;Test for double sided
        jz      sidea           ;Media is only single sided
        pop     psw             ;Retrieve adjusted sector
        pop     b
        cma                     ;Make sector request positive
        inr     a
        mov     c,a             ;Make new sector the requested sector
        pop     d
        call    sideone
        mvi     a,8Øh           ;Side two bit
        ora     h               ;       and sector
        mov     h,a
        ret
        endif

        if      maxhd ne Ø      ;Hard Disk translation routine
tranhd  mov     h,b
        mov     l,c
        inx     h
        ret
        endif

*****************************************************************
*                                                               *
* Setdry selects the next drive to be used in read/write        *
* operations. If the drive has never been selected before, a    *
* parameter table is created which correctly describes the      *
* diskette currently in the drive. Diskettes can be of four     *
* different sector sizes:                                       *
*       1) 128 bytes single density.                            *
*       2) 256 bytes double density.                            *
*       3) 512 bytes double density.                            *
*       4) 1Ø24 bytes double density.                           *
*                                                               *
*****************************************************************

setdrv  mov     a,c             ;Save the drive #
        sta     cpmdrv
        cpi     maxflop+(maxhd*logdsk) ;Check for a valid drive #
        jnc     zret            ;Illegal drive #
        mov     a,e             ;Test if drive ever logged in before
        ani     1
        jnz     setdrv1         ;Bit Ø of E = Ø -> Never selected before

        if      (maxhd ne Ø) and (maxflop ne Ø) ;Both types ?
        lda     cpmdrv          ;Get the Drive Number

        if      first
        cpi     maxhd*logdsk    ;Over the # of hard disks ?
        jc      drvhd
        sui     maxhd*logdsk
        else
        cpi     maxflop         ;Over the # of floppies ?
        jnc     subfp
        endif
        endif

        if      (maxflop ne Ø) and first
        mov     c,a             ;Save drive #
        mvi     a,Ø             ;Have the floppies been accessed yet ?
flopflg equ     $-1
        ana     a
        jnz     flopok
        mvi     b,17            ;Floppies havn't been accessed
        lxi     h,djboot        ;Check if 2D controller is installed
        mvi     a,(jmp)
clopp   cmp     m
        jnz     zret
        dcr     b
        jnz     clopp
        call    djboot          ;Initialize the controller
        mvi     a,1             ;Save 2D initialized flag
        sta     flopflg
        endif
        if      maxflop ne Ø
flopok  lxi     h,1             ;Select sector 1 of track 1
        shld    truesec
        mvi     a,1
        sta     cpmtrk
        call    fill            ;Flush buffer and refill
        jc      zret            ;Test for error return
        call    djstat          ;Get status on current drive
        ani     Øch             ;Strip off unwanted bits
        push    psw             ;Used to select a DPB
        rar
        lxi     h,xlts          ;Table of XLT addresses
        mov     e,a
        mvi     d,Ø
        dad     d
        push    h               ;Save pointer to proper XLT
        call    getdph          ;Get DPH pointer into DE
        xchg                    ;
        pop     d
        mvi     b,2             ;Number of bytes to move
        call    movlop          ;Move the address of XLT
        lxi     d,8             ;Offset to DPB pointer
        dad     d               ;HL <- &DPH.DPB
        push    h
        lhld    origin+7        ;Get address of DJ terminal out routine
        inx     h               ;Bump to look at address of
                                ;       uart status location
        mov     a,m
        xri     3               ;Adjust for proper rev DJ
        mov     l,a
        mvi     h,(origin+3ØØh)/1ØØh
        mov     a,m
        ani     dblsid          ;Check double sided bit
        lxi     d,dpb128s       ;Base for single sided DPB's
        jnz     sideok
        lxi     d,dpb128d       ;Base of double sided DPB's
sideok  xchg                    ;HL <- DBP base, DE <- &DPH.DPB
        pop     d               ;Restore DE (pointer into DPH)
        pop     psw             ;Offset to correct DPB
        ral
        ral
        mov     c,a
        mvi     b,Ø
        dad     b
        xchg                    ;Put DPB address in DPH
        mov     m,e
        inx     h
        mov     m,d
        endif

        if      (maxhd ne Ø) and (maxflop ne Ø)
        jmp     setdrvl         ;Skip over the Hard Disk select
        if      not first
subfp   sui     maxflop         ;Adjust the drive #
        endif
        endif

        if      maxhd ne Ø
drvhd   call    divlog          ;Divide by logical disks per drive
        mov     a,c
        sta     hddisk
        call    drvptr
        mov     a,m
        inr     a
        jnz     setdrvl
        ori     null            ;Select drive
        out     hdfunc
        mvi     a,scenbl        ;Enable the controller
        out     hdcntl
        mvi     c,239           ;Wait approx 2 minutes for Disk to ready
        lxi     h,Ø
tdelay  dcx     h
        mov     a,h
        ora     l
        cz      dcrc
        rz
        in      hdstat          ;Test if ready yet
        ani     drvrdy
        jnz     tdelay

        if      sdelay
        lxi     h,Ø             ;Time one revolution of the drive
        mvi     c,index
        in      hdstat
        ana     c
        mov     b,a             ;Save current index level in B
indx1   in      hdstat
        ana     c
        cmp     b               ;Loop util index level changes
        jz      indx1
indx2   inx     h
        in      hdstat          ;Start counting until index returns to
        ana     c               ;       previous state
        cmp     b
        jnz     indx2
        if      m1Ø
        dad     h
        endif
        shld    settle          ;Save the Count for timeout delay
        endif
        call    hdhome
        endif

setdrv1 call    getdpb          ;Get address of DPB in HL
        lxi     b,15            ;Offset to sector size
        dad     b
        mov     a,m             ;Get sector size
        ani     7h
        sta     secsiz
        mov     a,m
        rar
        rar
        rar
        rar
        ani     Øfh
        sta     secpsec
        xchg                    ;HL <- DPH
        ret

zret    lxi     h,Ø             ;Seldrv error exit
        ret

        if      maxhd ne Ø
dcrc    dcr     c               ;Conditional decrement C routine
        ret

divlog  mvi     c,Ø
divlogx sui     logdsk
        rc
        inr     c
        jmp     divlogx
        endif

*****************************************************************
*                                                               *
* Getdpb returns HL pointing to the DPB of the currently        *
* selected drive, DE pointing to DPH.                           *
*                                                               *
*****************************************************************

getdpb  lda     cpmdrv
        mov     l,a             ;Form offset
        mvi     h,Ø
        dad     h
        dad     h
        dad     h
        dad     h
        lxi     d,dpbase        ;Base of DPH's
        dad     d
        push    h               ;Save address of DPH
        lxi     d,1Ø            ;Offset to DPB
        dad     d
        mov     a,m             ;Get low byte of DPB address
        inx     h
        mov     h,m             ;Get low byte of DPB
        mov     l,a
        pop     d
        ret

****************************************************************
*                                                              *
* Xlts is a table of address that point to each of the xlt     *
* tables for each sector size.                                 *
*                                                              *
****************************************************************

        if      maxflop ne Ø
xlts    dw      xlt128          ;Xlt for 128 byte sectors
        dw      xlt256          ;Xlt for 256 byte sectors
        dw      xlt512          ;Xlt for 512 byte sectors
        dw      xlt124          ;Xlt for 1Ø24 byte sectors
        endif

****************************************************************
*                                                              *
* Write routine moves data from memory into the buffer. If the *
* desired CP/M sector is not contained in the disk buffer, the *
* buffer is first flushed to the disk if it has ever been      *
* written into, then a read is performed into the buffer to get *
* the desired sector. Once the correct sector is in memory, the *
* buffer written indicator is set, so the buffer will be       *
* flushed, then the data is transferred into the buffer.       *
*                                                              *
****************************************************************

write   mov     a,c             ;Save write command type
        sta     writtyp
        mvi     a,1             ;Set write command
        db      (mvi) or (b*8)  ;This "mvi b" instruction  causes
                                ;       the following "xra a" to
                                ;       be skipped over.

****************************************************************
*                                                              *
* Read routine to buffer data from the disk. If the sector     *
* requested from CP/M is in the buffer, then the data is simply *
* transferred from the buffer to the desired dma address. If   *
* the buffer does not contain the desired sector, the buffer is *
* flushed to the disk if it has ever been written into, then   *
* filled with the sector from the disk that contains the       *
* desired CP/M sector.                                         *
*                                                              *
****************************************************************

read    xra     a               ;Set the command type to read
        sta     rdwr            ;Save command type

****************************************************************
*                                                              *
* Redwrt calculates the physical sector on the disk that       *
* contains the desired CP/M sector, then checks if it is the   *
* sector currently in the buffer. If no match is made, the     *
* buffer is flushed if necessary and the correct sector read   *
* from the disk.                                               *
*                                                              *
```

```
*****************************************************************

redwrt  mvi     b,Ø             ;The Ø is modified to contain the log2
secsiz  equ     $-1             ;       of the physical sector size/128
                                ;       on the currently selected disk.
        lhld    cpmsec          ;Get the desired CP/M sector #
        mov     a,h
        ani     8Øh             ;Save only the side bit
        mov     c,a             ;Remember the side
        mov     a,h
        ani     7fh             ;Forget the side bit
        mov     h,a
        dcx     h               ;Temporary adjustment
divloop dcr     b               ;Update repeat count
        jz      divdone
        ora     a
        mov     a,h
        rar
        mov     h,a
        mov     a,l
        rar                     ;Divide the CP/M sector # by the size
                                ;       of the physical sectors
        mov     l,a
        jmp     divloop         ;
divdone inx     h
        mov     a,h
        ora     c               ;Restore the side bit
        mov     h,a
        shld    truesec         ;Save the physical sector number
        lxi     h,cpmdrv        ;Pointer to desired drive,track, and sector
        lxi     d,bufdrv        ;Pointer to buffer drive,track, and sector
        mvi     b,5             ;Count loop
dtslop  dcr     b               ;Test if done with compare
        jz      move            ;Yes, match. Go move the data
        ldax    d               ;Get a byte to compare
        cmp     m               ;Test for match
        inx     h               ;Bump pointers to next data item
        inx     d
        jz      dtslop          ;Match, continue testing

*****************************************************************
*                                                               *
* Drive, track, and sector don't match, flush the buffer if     *
* necessary and then refill.                                    *
*                                                               *
*****************************************************************

        call    fill            ;Fill the buffer with correct physical sector
        rc                      ;No good, return with error indication

*****************************************************************
*                                                               *
* Move has been modified to cause either a transfer into or out *
* the buffer.                                                   *
*                                                               *
*****************************************************************

move    lda     cpmsec          ;Get the CP/M sector to transfer
        dcr     a               ;Adjust to proper sector in buffer
        ani     Ø               ;Strip off high ordered bits
secpsec equ     $-1             ;The Ø is modified to represent the # of
                                ;       CP/M sectors per physical sectors
        mov     l,a             ;Put into HL
        mvi     h,Ø
        dad     h               ;Form offset into buffer
        dad     h
        dad     h
        dad     h
        dad     h
        dad     h
        dad     h
        lxi     d,buffer        ;Beginning address of buffer
        dad     d               ;Form beginning address of sector to transfer
        xchg                    ;DE = address in buffer
        lxi     h,Ø             ;Get DMA address, the Ø is modified to
                                ;       contain the DMA address
cpmdma  equ     $-2
        mvi     a,Ø             ;The zero gets modified to contain
                                ;       a zero if a read, or a 1 if write
rdwr    equ     $-1
        ana     a               ;Test which kind of operation
        jnz     into            ;Transfer data into the buffer
outof   call    mover
        xra     a
        ret

into    xchg                    ;
        call    mover           ;Move the data, HL = destination
                                ;       DE = source
        mvi     a,1
        sta     bufwrtn         ;Set buffer written into flag
        mvi     a,Ø             ;Check for directory write
writtyp equ     $-1
        dcr     a
        mvi     a,Ø
        sta     writtyp         ;Set no directory write
        rnz                     ;No error exit

****************************************************************
*                                                              *
* Flush writes the contents of the buffer out to the disk if   *
* it has ever been written into.                               *
*                                                              *
****************************************************************

flush   mvi     a,Ø             ;The Ø is modified to reflect if
                                ;       the buffer has been written into
bufwrtn equ     $-1
        ana     a               ;Test if written into
        rz                      ;Not written, all done

        if      (maxhd ne Ø) and (maxflop ne Ø)
        lxi     h,djwrite       ;Write operation for Disk Jockey
        lxi     d,hdwrite       ;Write operation for Hard Disk
        call    decide
        else
        if      maxhd ne Ø
        lxi     h,hdwrite
        endif
        if      maxflop ne Ø
        lxi     h,djwrite
        endif
        endif

****************************************************************
*                                                              *
* Prep prepares to read/write the disk. Retries are attempted. *
* Upon entry, H&L must contain the read or write operation     *
* address.                                                     *
*                                                              *
****************************************************************
```

```
prep     xra     a               ;Reset buffer written flag
         sta     bufwrtn
         shld    retryop         ;Set up the read/write operation
         mvi     b,retries       ;Maximum number of retries to attempt
retrylp  push    b               ;Save the retry count
         lda     bufdrv          ;Get drive number involved in the operation

         if      (maxhd ne 0) and (maxflop ne 0)
         if      first
         cpi     maxhd*logdsk
         jc      noadjst
         sui     maxhd*logdsk
         else
         cpi     maxflop
         jc      noadjst
         sui     maxflop
         endif

noadjst  mov     c,a
         lxi     h,djdrv         ;Select drive
         lxi     d,hddrv
         call    decidgo
         else
         mov     c,a
         if      maxhd ne 0
         call    hddrv
         endif
         if      maxflop ne 0
         call    djdrv           ;Select the drive
         endif
         endif

         lda     buftrk
         ana     a               ;Test for track zero
         mov     c,a
         push    b

         if      (maxhd ne 0) and (maxflop ne 0)
         lxi     h,djhome
         lxi     d,hdhome
         cz      decidgo
         else
         if      maxhd ne 0
         cz      hdhome
         endif
         if      maxflop ne 0
         cz      djhome          ;Home the drive if track 0
         endif
         endif

         pop     b               ;Restore track #

         if      (maxhd ne 0) and (maxflop ne 0)
         lxi     h,djtrk
         lxi     d,hdtrk
         call    decidgo
         else
         if      maxhd ne 0
         call    hdtrk
         endif
         if      maxflop ne 0
         call    djtrk           ;Seek to proper track
         endif
         endif

         lhld    bufsec
```

```
        mov     a,h             ;Get sector involved in operation
        rlc                     ;Bit Ø of A equals side #
        ani     1               ;Strip off unnecessary bits
        mov     c,a             ;C <- side #

        if      (maxhd ne Ø) and (maxflop ne Ø)
        lxi     h,djside
        lxi     d,hdside
        call    decidgo
        else
        if      maxhd ne Ø
        call    hdside
        endif
        if      maxflop ne Ø
        call    djside          ;Select the side
        endif
        endif

        lhld    bufsec
        mov     a,h
        ani     7fh             ;Strip off side bit
        mov     b,a             ;C <- sector #
        mov     c,l

        if      (maxhd ne Ø) and (maxflop ne Ø)
        lxi     h,djsec
        lxi     d,hdsec
        call    decidgo
        else
        if      maxhd ne Ø
        call    hdsec
        endif
        if      maxflop ne Ø
        call    djsec           ;Select the side
        endif
        endif

        lxi     b,buffer        ;Set the DMA address

        if      (maxhd ne Ø) and (maxflop ne Ø)
        lxi     h,djdma
        lxi     d,hddma
        call    decidgo
        else
        if      maxhd ne Ø
        call    hddma
        endif
        if      maxflop ne Ø
        call    djdma           ;Select the side
        endif
        endif

        call    Ø               ;Get operation address
retryop equ     $-2
        pop     b               ;Restore the retry counter
        mvi     a,Ø             ;No error exit status
        rnc                     ;Return no error
        dcr     b               ;Update the retry counter
        stc                     ;Assume retry count expired
        mvi     a,Øffh          ;Error return
        rz
        mov     a,b
        cpi     retries/2
        jnz     retrylp         ;Try again

        push    b
        if      (maxhd ne Ø) and (maxflop ne Ø)
        lxi     h,djhome
        lxi     d,hdhome
        call    decidgo
        else
        if      maxhd ne Ø
        call    hdhome
        endif
        if      maxflop ne Ø
        call    djhome          ;Home the drive if track Ø
        endif
        endif

        pop     b
        jmp     retrylp

*****************************************************************
*                                                               *
* Fill fills the buffer with a new sector from the disk.       *
*                                                               *
*****************************************************************

fill    call    flush           ;Flush buffer first
        rc                      ;Check for error
        lxi     d,cpmdrv        ;Update the drive, track, and sector
        lxi     h,bufdrv
        mvi     b,4             ;Number of bytes to move
        call    movlop          ;Copy the data

        lda     rdwr
        ana     a
        jz      fread
        lda     writtyp
        dcr     a
        dcr     a
        rz
        call    getdpb
        lxi     d,15
        dad     d
        mov     a,m
        ani     3
        dcr     a
        rz

fread   equ     $
        if      (maxhd ne Ø) and (maxflop ne Ø)
        lxi     h,djread
        lxi     d,hdread
        call    decide
        else
        if      maxhd ne Ø
        lxi     h,hdread
        endif
        if      maxflop ne Ø
        lxi     h,djread        ;Select the side
        endif
        endif
        jmp     prep            ;Select drive, track, and sector.
                                ;       Then read the buffer

*****************************************************************
*                                                               *
* Mover moves 128 bytes of data. Source pointer in DE, Dest    *
* pointer in HL.                                                *
*                                                               *
*****************************************************************
```

```
mover   mvi     b,128                   ;Length of transfer
movlop  ldax    d                       ;Get a bte of source
        mov     m,a                     ;Move it
        inx     d                       ;Bump pointers
        inx     h
        dcr     b                       ;Update counter
        jnz     movlop                  ;Continue moving until done
        ret

*****************************************************************
*                                                               *
* Routines to decide which controller to use.                   *
*                                                               *
*****************************************************************

        if      (maxhd ne Ø) and (maxflop ne Ø)
decidgo call    decide  ;which controller ?
        pchl
        endif

        if      (maxhd ne Ø) and (maxflop ne Ø)
decide  lda     bufdrv                  ;Get proper routine into H&L, based
        if      first                   ;   on currently selected drive
        cpi     maxhd*logdsk
        rnc
        else
        cpi     maxflop
        rc
        endif
        xchg
        ret
        endif

*****************************************************************
*                                                               *
* The following is the equivalent of the lowest level drivers   *
* for the Hard Disk.                                            *
*                                                               *
*****************************************************************

        if      maxhd ne Ø
hddrv   mov     a,c                     ;Select Hard Disk drive
        call    divlog                  ;Get the physical drive #
        mov     a,c
        sta     hddisk                  ;Select the drive
        ori     null
        out     hdfunc
        mvi     a,wenabl
        out     hdcntl
        ret

hdhome  call    drvptr
        mvi     m,Ø                     ;Set track to zero

        if      sdelay
stepo   in      hdstat                  ;Test status
        ani     tkzero                  ;At track zero ?
        jz      delay
        mvi     a,1
        stc
        call    accok                   ;Take one step out
        jmp     stepo

        else
```

New

```
        in      hdstat
        ani     tkzero
        rz
        xra     a
        jmp     accok
        endif

        if      sdelay
delay   lxi     h,Ø             ;Get delay
settle  equ     $-2
deloop  dcx     h               ;Wait 2Øms
        mov     a,h
        ora     l
        inx     h
        dcx     h
        jnz     deloop
        ret
        endif

hdtrk   call    drvptr          ;Get pointer to current track
        mov     e,m             ;Get current track
        mov     m,c             ;Update the track
        mov     a,e             ;Need to seek at all ?
        sub     c
        rz
        cmc                     ;Get carry into direction
        jc      hdtrk2
        cma
        inr     a
        if      not sdelay
hdtrk2  jmp     accok
        else
hdtrk2  call    accok
        jmp     delay
        endif

accok   mov     b,a             ;Prep for build
        call    build
sloop   ani     nstep           ;Get step pulse low
        out     hdfunc          ;Output low step line
        ori     pstep           ;Set step line high
        out     hdfunc          ;Output high step line
        dcr     b               ;Update repeat count
        jnz     sloop           ;Keep going the required # of tracks
        jmp     wsdone

hddma   mov     h,b             ;Save the DMA address
        mov     l,c
        shld    hdadd
hdside  equ     $
        ret

wsdone  in      hdstat          ;Wait for seek complete to finish
        ani     complt
        jz      wsdone
        ret

        if      m26
hdsec   mvi     a,Ø1fh          ;For compatibility with cbios rev 2.3, 2.4
        ana     c
        cz      getspt
        sta     hdsectr
        mvi     a,ØeØh
        ana     c
        rlc
        rlc
        rlc
        sta     head
getspt  mvi     a,hdspt
        ret

        else

hdsec   mov     a,c
        call    divspt
        adi     hdspt
        ana     a
        cz      getspt
        sta     hdsectr
        mov     a,c
        sta     head
getspt  mvi     a,hdspt
        dcr     c
        ret

divspt  mvi     c,Ø
divsptx sui     hdspt
        rc
        inr     c
        jmp     divsptx
        endif

hdread  call    hdprep
        rc
        xra     a
        out     hdcmnd
        cma
        out     hddata
        out     hddata
        mvi     a,rsect         ;Read sector command
        out     hdcmnd
        call    process
        rc
        xra     a
        out     hdcmnd
        mvi     b,seclen/4
        lxi     h,Ø
hdadd   equ     $-2
        in      hddata
        in      hddata
rtloop  in      hddata          ;Move four bytes
        mov     m,a
        inx     h
        in      hddata
        mov     m,a
        inx     h
        in      hddata
        mov     m,a
        inx     h
        in      hddata
        mov     m,a
        inx     h
        dcr     b
        jnz     rtloop
        ret

hdwrite call    hdprep          ;Prepare header
        rc
        xra     a
        out     hdcmnd
        lhld    hdadd
        mvi     b,seclen/4
```

```
wtloop  mov     a,m                     ;Move 4 bytes
        out     hddata
        inx     h
        mov     a,m
        out     hddata
        inx     h
        mov     a,m
        out     hddata
        inx     h
        mov     a,m
        out     hddata
        inx     h
        dcr     b
        jnz     wtloop
        mvi     a,wsect                 ;Issue write sector command
        out     hdcmnd
        call    process
        rc
        mvi     a,wfault
        ana     b
        stc
        rz
        xra     a
        ret

process in      hdstat                  ;Wait for command to finish
        mov     b,a
        ani     opdone
        jz      process
        mvi     a,dskclk
        out     hdcntl
        in      hdstat
        ani     tmout                   ;Timed out ?
        stc
        rnz
        in      hdreslt
        ani     retry                   ;Any retries ?
        stc
        rnz
        xra     a
        ret

hdprep  in      hdstat
        ani     drvrdy
        stc
        rnz
        mvi     a,isbuff                ;Initialize pointer
        out     hdcmnd
        call    build
        ori     Øch
        out     hdfunc
        lda     head
        out     hddata                  ;Form head byte
        call    drvptr
        mov     a,m                     ;Form track byte
        out     hddata
        ana     a
        mvi     b,8Øh
        jz      zkey
        mvi     b,Ø
zkey    mvi     a,Ø                     ;Form sector byte
hdsectr equ     $-1
        out     hddata
        mov     a,b
        out     hddata
        mvi     a,dskclk
        out     hdcntl
        mvi     a,wenabl
        out     hdcntl
        xra     a
        ret

drvptr  lhld    hddisk
        xchg
        mvi     d,Ø
        lxi     h,drives
        dad     d
        ret

build   mvi     a,Ø
head    equ     $-1
        ral
        ral
        ral
        ral
        ori     Ø
hddisk  equ     $-1
        xri     ØfØh
        ret

drives  equ     $
        rept    maxhd
        db      Øffh
        endm
        endif
```

End now

```
*******************************************************************
*                                                                 *
* Xlt tables (sector skew tables) for CP/M 2.Ø. These tables      *
* define the sector translation that occurs when mapping CP/M     *
* sectors to physical sectors on the disk. There is one skew      *
* table for each of the possible sector sizes. Currently the      *
* tables are located on track Ø sectors 6 and 8. They are         *
* loaded into memory in the Cbios ram by the cold boot routine.   *
*                                                                 *
*******************************************************************

        if      maxflop ne Ø
xlt128  db      Ø
        db      1,7,13,19,25
        db      5,11,17,23
        db      3,9,15,21
        db      2,8,14,2Ø,26
        db      6,12,18,24
        db      4,1Ø,16,22

xlt256  db      Ø
        db      1,2,19,2Ø,37,38
        db      3,4,21,22,39,4Ø
        db      5,6,23,24,41,42
        db      7,8,25,26,43,44
        db      9,1Ø,27,28,45,46
        db      11,12,29,3Ø,47,48
        db      13,14,31,32,49,5Ø
        db      15,16,33,34,51,52
        db      17,18,35,36

xlt512  db      Ø
        db      1,2,3,4,17,18,19,2Ø
        db      33,34,35,36,49,5Ø,51,52
        db      5,6,7,8,21,22,23,24
        db      37,38,39,4Ø,53,54,55,56
        db      9,1Ø,11,12,25,26,27,28
        db      41,42,43,44,57,58,59,6Ø
        db      13,14,15,16,29,3Ø,31,32
        db      45,46,47,48

xltl24  db      Ø
        db      1,2,3,4,5,6,7,8
        db      25,26,27,28,29,3Ø,31,32
        db      49,5Ø,51,52,53,54,55,56
        db      9,1Ø,11,12,13,14,15,16
        db      33,34,35,36,37,38,39,4Ø
        db      57,58,59,6Ø,61,62,63,64
        db      17,18,19,2Ø,21,22,23,24
        db      41,42,43,44,45,46,47,48

****************************************************************
*                                                              *
* Each of the following tables describes a diskette with the   *
* specified characteristics.                                   *
*                                                              *
****************************************************************


****************************************************************
*                                                              *
* The following DPB defines a  diskette for 128 byte sectors,  *
* single density, and single sided.                            *
*                                                              *
****************************************************************

dpb128s dw      26              ;CP/M sectors/track
        db      3               ;BSH
        db      7               ;BLM
        db      Ø               ;EXM
        dw      242             ;DSM
        dw      63              ;DRM
        db      ØcØh            ;ALØ
        db      Ø               ;AL1
        dw      16              ;CKS
        dw      2               ;OFF
        db      1h              ;16*((#cpm sectors/physical sector) -1) +
                                ;log2(#bytes per sector/128) + 1 +
                                ;8 if double sided.

****************************************************************
*                                                              *
* The following DPB defines a diskette for 256 byte sectors,   *
* double density, and single sided.                            *
*                                                              *
****************************************************************

dpb256s dw      52              ;CP/M sectors/track
        db      4               ;BSH
        db      15              ;BLM
        db      Ø               ;EXM
        dw      242             ;DSM
        dw      127             ;DRM
        db      ØcØh            ;ALØ
        db      Ø               ;AL1
        dw      32              ;CKS
        dw      2               ;OFF
        db      12h             ;16*((#cpm sectors/physical sector) -1) +
                                ;log2(#bytes per sector/128) + 1 +
                                ;8 if double sided.

****************************************************************
```

```
*                                                                *
* The following DPB defines a diskette as 512 byte sectors,     *
* double density, and single sided.                              *
*                                                                *
******************************************************************

dpb512s dw      6Ø              ;CP/M sectors/track
        db      4               ;BSH
        db      15              ;BLM
        db      Ø               ;EXM
        dw      28Ø             ;DSM
        dw      127             ;DRM
        db      ØcØh            ;ALØ
        db      Ø               ;AL1
        dw      32              ;CKS
        dw      2               ;OFF
        db      33h             ;16*((#cpm sectors/physical sector) -1) +
                                ;log2(#bytes per sector/128) + 1 +
                                ;8 if double sided.

******************************************************************
*                                                                *
* The following DPB defines a diskette as 1Ø24 byte sectors,    *
* double density, and single sided.                              *
*                                                                *
******************************************************************

dp1Ø24s dw      64              ;CP/M sectors/track
        db      4               ;BSH
        db      15              ;BLM
        db      Ø               ;EXM
        dw      299             ;DSM
        dw      127             ;DRM
        db      ØcØh            ;ALØ
        db      Ø               ;AL1
        dw      32              ;CKS
        dw      2               ;OFF
        db      74h             ;16*((#cpm sectors/physical sector) -1) +
                                ;log2(#bytes per sector/128) + 1 +
                                ;8 if double sided.

******************************************************************
*                                                                *
* The following DPB defines a diskette for 128 byte sectors,    *
* single density, and double sided.                              *
*                                                                *
******************************************************************

dpb128d dw      52              ;CP/M sectors/track
        db      4               ;BSH
        db      15              ;BLM
        db      1               ;EXM
        dw      242             ;DSM
        dw      127             ;DRM
        db      ØcØh            ;ALØ
        db      Ø               ;AL1
        dw      32              ;CKS
        dw      2               ;OFF
        db      9h

******************************************************************
*                                                                *
* The following DPB defines a diskette as 256 byte sectors,     *
* double density, and double sided.                              *
*                                                                *
******************************************************************
```

```
dpb256d dw      104             ;CP/M sectors/track
        db      4               ;BSH
        db      15              ;BLM
        db      0               ;EXM
        dw      486             ;DSM
        dw      255             ;DRM
        db      0f0h            ;AL0
        db      0               ;AL1
        dw      64              ;CKS
        dw      2               ;OFF
        db      1ah

****************************************************************
*                                                              *
* The following DPB defines a diskette as 512 byte sectors,    *
* double density, and double sided.                            *
*                                                              *
****************************************************************

dpb512d dw      120             ;CP/M sectors/track
        db      4               ;BSH
        db      15              ;BLM
        db      0               ;EXM
        dw      561             ;DSM
        dw      255             ;DRM
        db      0f0h            ;AL0
        db      0               ;AL1
        dw      64              ;CKS
        dw      2               ;OFF
        db      3bh

****************************************************************
*                                                              *
* The following DPB defines a diskette as 1024 byte sectors,   *
* double density, and double sided.                            *
*                                                              *
****************************************************************

dp1024d dw      128             ;CP/M sectors/track
        db      4               ;BSH
        db      15              ;BLM
        db      0               ;EXM
        dw      599             ;DSM
        dw      255             ;DRM
        db      0f0h            ;AL0
        db      0               ;AL1
        dw      64              ;CKS
        dw      2               ;OFF
        db      7ch
        endif

****************************************************************
*                                                              *
* The following DPB defines a 10 Megabyte Hard disk, with 512  *
* byte sectors.                                                *
*                                                              *
****************************************************************

        if      maxhd ne 0
        if      m26 ne 0
dpbhd1  dw      1024            ;CP/M sectors/track
        db      5               ;BSH
        db      31              ;BLM
        db      1               ;EXM
        dw      1973            ;DSM
        dw      511             ;DRM
        db      0ffh            ;AL0
        db      0ffh            ;AL1
        dw      0               ;CKS
        dw      1               ;OFF
        db      33h             ;16*((#cpm sectors/physical sector) -1) +
                                ;log2(#bytes per sector/128) + 1 +
                                ;8 if double sided.
dpbhd2  dw      1024            ;CP/M sectors/track
        db      5               ;BSH
        db      31              ;BLM
        db      1               ;EXM
        dw      1973            ;DSM
        dw      511             ;DRM
        db      0ffh            ;AL0
        db      0ffh            ;AL1
        dw      0               ;CKS
        dw      64              ;OFF
        db      33h             ;16*((#cpm sectors/physical sector) -1) +
                                ;log2(#bytes per sector/128) + 1 +
                                ;8 if double sided.

dpbhd3  dw      1024            ;CP/M sectors/track
        db      5               ;BSH
        db      31              ;BLM
        db      1               ;EXM
        dw      1973            ;DSM
        dw      511             ;DRM
        db      0ffh            ;AL0
        db      0ffh            ;AL1
        dw      0               ;CKS
        dw      127             ;OFF
        db      33h             ;16*((#cpm sectors/physical sector) -1) +
                                ;log2(#bytes per sector/128) + 1 +
                                ;8 if double sided.
        endif
        if      m10 ne 0
dpbhd1  dw      336             ;CP/M sectors/track
        db      5               ;BSH
        db      31              ;BLM
        db      1               ;EXM
        dw      1269            ;DSM
        dw      511             ;DRM
        db      0ffh            ;AL0
        db      0ffh            ;AL1
        dw      0               ;CKS
        dw      1               ;OFF
        db      33h             ;16*((#cpm sectors/physical sector) -1) +
                                ;log2(#bytes per sector/128) + 1 +
                                ;8 if double sided.
dpbhd2  dw      336             ;CP/M sectors/track
        db      5               ;BSH
        db      31              ;BLM
        db      1               ;EXM
        dw      1280            ;DSM
        dw      511             ;DRM
        db      0ffh            ;AL0
        db      0ffh            ;AL1
        dw      0               ;CKS
        dw      122             ;OFF
        db      33h             ;16*((#cpm sectors/physical sector) -1) +
                                ;log2(#bytes per sector/128) + 1 +
                                ;8 if double sided.
        endif
        if      m20 ne 0
dpbhd1  dw      672             ;CP/M sectors/track
```

```
        db      5               ;BSH
        db      31              ;BLM
        db      1               ;EXM
        dw      2Ø15            ;DSM
        dw      511             ;DRM
        db      Øffh            ;ALØ
        db      Øffh            ;AL1
        dw      Ø               ;CKS
        dw      1               ;OFF
        db      33h             ;16*((#cpm sectors/physical sector) -1) +
                                ;log2(#bytes per sector/128) + 1 +
                                ;8 if double sided.
dpbhd2  dw      672             ;CP/M sectors/track
        db      5               ;BSH
        db      31              ;BLM
        db      1               ;EXM
        dw      2Ø15            ;DSM
        dw      511             ;DRM
        db      Øffh            ;ALØ
        db      Øffh            ;AL1
        dw      Ø               ;CKS
        dw      98              ;OFF
        db      33h             ;16*((#cpm sectors/physical sector) -1) +
                                ;log2(#bytes per sector/128) + 1 +
                                ;8 if double sided.

dpbhd3  dw      672             ;CP/M sectors/track
        db      5               ;BSH
        db      31              ;BLM
        db      1               ;EXM
        dw      1Ø28            ;DSM
        dw      511             ;DRM
        db      Øffh            ;ALØ
        db      Øffh            ;AL1
        dw      Ø               ;CKS
        dw      195             ;OFF
        db      33h             ;16*((#cpm sectors/physical sector) -1) +
                                ;log2(#bytes per sector/128) + 1 +
                                ;8 if double sided.
        endif
        endif

*************************************************************
*                                                           *
* CP/M disk parameter headers, unitialized.                 *
*                                                           *
*************************************************************

header  macro   nd,dpb
        dw      Ø               ;Translation table filled in later
        dw      Ø,Ø,Ø           ;Scratch
        dw      dirbuf          ;Directory buffer
        dw      dpb             ;DPB filled in later
        dw      csv&nd          ;Directory check vector
        dw      alv&nd          ;Allocation vector
        endm

dpbase  equ     $
dn      set     Ø
        if      first
        rept    maxhd           ;Generate Hard Disk DPH's followed
        header  %dn,dpbhd1      ;        by Floppy DPH's
dn      set     dn+1
        header  %dn,dpbhd2
dn      set     dn+1
        if      (m26 ne Ø) or (m2Ø ne Ø)
        header  %dn,dpbhd3
dn      set     dn+1
        endif
        endm
        rept    maxflop
        header  %dn,Ø
dn      set     dn+1
        endm
        else
        rept    maxflop         ;Generate Floppy DPH's followed by
        header  %dn,Ø           ;      HArd Disk DPH's
dn      set     dn+1
        endm
        rept    maxhd
        header  %dn,dpbhd1
dn      set     dn+1
        header  %dn,dpbhd2
dn      set     dn+1
        if      (m26 ne Ø) or (m2Ø ne Ø)
        header  %dn,dpbhd3
dn      set     dn+1
        endif
        endm
        endif

*****************************************************************
*                                                               *
* Cbios ram locations that don't need initialization.           *
*                                                               *
*****************************************************************

cpmsec  dw      Ø               ;CP/M sector #
cpmdrv  db      Ø               ;CP/M drive #
cpmtrk  db      Ø               ;CP/M track #
truesec dw      Ø               ;Disk Jockey sector that contains CP/M sector
bufdrv  db      Ø               ;Drive that buffer belongs to
buftrk  db      Ø               ;Track that buffer belongs to
bufsec  dw      Ø               ;Sector that buffer belongs to
buffer  equ     $

*****************************************************************
*                                                               *
* Signon message output during cold boot.                       *
*                                                               *
*****************************************************************

hexnum  macro   num
        if      (num/16) > 9
        db      (num/16 and Øfh) + 'A' - 1Ø
        else
        db      (num/16 and Øfh) + 'Ø'
        endif
        if      (num and Øfh) > 9
        db      (num and Øfh) + 'A' - 1Ø
        else
        db      (num and Øfh) + 'Ø'
        endif
        endm

prompt  db      acr,alf,alf
        db      'Morrow Designs '
        db      'Ø'+msize/1Ø            ;CP/M memory size
        db      'Ø'+(msize mod 1Ø)
        db      'K CP/M '               ;CP/M version number
        db      cpmrev/1Ø+'Ø'
        db      '.'
        db      (cpmrev mod 1Ø)+'Ø'
        db      ', Cbios rev '
        db      revnum/1Ø+'Ø','.'       ;Cbios revision number
        db      revnum mod 1Ø+'Ø'
        if      maxhd ne Ø
        db      '.'
        db      mrev/1Ø+'Ø'
        db      mrev mod 1Ø+'Ø'
        if      (m1Ø or m2Ø) and sdelay
        db      'M'
        endif
        if      (m1Ø or m2Ø) and not sdelay
        db      'F'
        endif
        endif
        db      acr,alf
        db      'For '

        if      maxflop ne Ø
        db      'a Disk Jockey 2D @ '
        hexnum  %(origin/256)
        db      'ØØH '
        endif

        if      (maxhd ne Ø) and (maxflop ne Ø)
        db      'and '
        endif

        if      maxhd ne Ø
        if      maxhd eq 1
        db      'an '
        endif
        if      maxhd eq 2
        db      'two '
        endif
        if      maxhd eq 3
        db      'three '
        endif
        if      maxhd eq 4
        db      'four '
        endif
        if      mrev eq 1Ø
        db      'M1Ø '
        endif
        if      mrev eq 2Ø
        db      'M2Ø '
        endif
        if      mrev eq 26
        db      'M26 '
        endif
        db      'hard disk'
        if      maxhd ne 1
        db      's'
        endif
        db      ' @ '
        hexnum  %hdorg
        db      'H.'
        endif
        db      acr,alf

        if      iotype ne 1
        if      iotype eq Ø
        db      'No '
        endif
        if      iotype eq 2
        db      'Switchboard '
        endif
        if      iotype eq 3
        db      'Sol '
        endif
        if      iotype eq 4
        db      'Exidy Sorcerer '
        endif
        db      ' configured as console.'
        db      acr,alf
        endif
        db      Ø

*****************************************************************
*                                                               *
* Utility routine to output the message pointed at by H&L,      *
* terminated with a null.                                       *
*                                                               *
*****************************************************************

message mov     a,m             ;Get a character of the message
        inx     h               ;Bump text pointer
        ana     a               ;Test for end
        rz                      ;Return if done
        push    h               ;Save pointer to text
        mov     c,a             ;Output character in C
        call    cout            ;Output the character
        pop     h               ;Restore the pointer
        jmp     message         ;Continue until null reached

*****************************************************************
*                                                               *
* Cboot is the cold boot loader. All of CP/M has been loaded in *
* when control is passed here.                                  *
*                                                               *
*****************************************************************

cboot   lxi     sp,tpa          ;Set up stack

        if      iotype ne Ø
        mvi     a,intioby
        sta     iobyte
        call    tinit           ;Initialize the terminal
        endif

        lxi     h,prompt        ;Prep for sending signon message
        call    message         ;Send the prompt
        xra     a               ;Select disk A
        sta     cpmdrv
        sta     cdisk

        if      (maxflop ne Ø) and first
        sta     flopflg
        endif
        lxi     h,bios+3
        shld    bios+1
        jmp     gocpm

        ds      512-($-buffer)  ;Maximum size buffer for 512 byte sectors

        if      maxflop ne Ø
        ds      512             ;Additional space for floppies 1k sectors
        endif

        if      (maxflop ne Ø) or (maxhd ne Ø)
dirbuf  ds      128             ;Directory buffer
        endif
```

```
alloc   macro   nd,al,cs
alv&nd  ds      al
csv&nd  ds      cs
        endm

dn      set     0

        if      not first
        rept    maxflop
        alloc   %dn,75,64
dn      set     dn+1
        endm
        rept    maxhd
        if      m26 ne 0
        alloc   %dn,247,0
dn      set     dn+1
        alloc   %dn,247,0
dn      set     dn+1
        alloc   %dn,247,0
dn      set     dn+1
        endif
        if      m10 ne 0
        alloc   %dn,159,0
dn      set     dn+1
        alloc   %dn,161,0
dn      set     dn+1
        endif
        if      m20 ne 0
        alloc   %dn,252,0
dn      set     dn+1
        alloc   %dn,252,0
dn      set     dn+1
        alloc   %dn,129,0
dn      set     dn+1
        endif
        endm

        else

        rept    maxhd
        if      m26 ne 0
        alloc   %dn,247,0
dn      set     dn+1
        alloc   %dn,247,0
dn      set     dn+1
        alloc   %dn,247,0
dn      set     dn+1
        endif
        if      m10 ne 0
        alloc   %dn,159,0
dn      set     dn+1
        alloc   %dn,161,0
dn      set     dn+1
        endif
        if      m20 ne 0
        alloc   %dn,252,0
dn      set     dn+1
        alloc   %dn,252,0
dn      set     dn+1
        alloc   %dn,129,0
dn      set     dn+1
        endif
        endm
        rept    maxflop
        alloc   %dn,75,64
        endm
        endif
        end
```