```
******************************************************************
*                                                                *
* CP/M vers 2.2 Cold Start Loader.                               *
*                                                                *
* Written by Bobby Dale Gifford.                                 *
* 3/17/8Ø                                                        *
*                                                                *
* The following routines will boot CP/M from the Disk Jockey    *
* 2D (all revs and models), or from the Disk Jockey Hard         *
* disk controller.                                               *
*                                                                *
*        The  cold boot loader (sector 1,  track Ø) is  loaded   *
* into  the ram of the controller by the cold boot  routine  of  *
* the firmware. The first thing the boot does is to load into    *
* the controller ram, a version of the Disk Jockey 2D firmware.  *
* From then on, all calls to the firmware will actually be       *
* directed to the Disk Jockey Ram. The next process is to load   *
* in a boot routine which can load in all of CP/M. This is       *
* done by determining the size of the sectors on track 1, and    *
* using this information to load in the proper boot into 8ØH.    *
*                                                                *
* The following tables explain the order of sector loading for   *
* each of the different sector sizes. An entry of ------         *
* represents a wrap back around (negative DMA adjustment).       *
* An entry flagged with astricks represents a partial sector     *
* load.                                                          *
*                                                                *
* All sector sizes:                                              *
* Track Ø sector 1      e7ØØ                                     *
*       Ø        8      2cØØh                                    *
*       Ø       1Ø      2dØØh                                    *
*       Ø       12      2eØØh                                    *
*       Ø       14      2fØØh                                    *
*       Ø       16      3ØØØh                                    *
*       Ø       18      278Øh                                    *
*       Ø       2Ø      288Øh                                    *
*       Ø       22      298Øh                                    *
*       Ø       24      2a8Øh                                    *
*       Ø       26      2b8Øh                                    *
*       Ø        9      2c8Øh                                    *
*       Ø       11      2d8Øh                                    *
*       Ø       13      2e8Øh                                    *
*       Ø       15      2f8Øh                                    *
*       Ø       17      27ØØh                                    *
*       Ø       19      28ØØh                                    *
*       Ø       21      29ØØh                                    *
*       Ø       23      2aØØh                                    *
*       Ø       25      2bØØh                                    *
*                                                                *
* The following depend on the sector size, all sectors are from  *
* track 1.                                                       *
*                                                                *
* 256                   512                    1Ø24              *
* sec   address         sec     address        sec     address   *
* 1     2cØØh           1       2cØØh          1       2cØØh     *
* 3     2eØØh           3       3ØØØh          3       34ØØh     *
* 5     3ØØØh           5       34ØØh          5       3cØØh     *
* 7     32ØØh           7       38ØØh       ** 7       44ØØh     *
* 9     34ØØh           9       3cØØh          -------------     *
* 11    36ØØh           11      4ØØØh          2       3ØØØh     *
* 13    38ØØh           13      44ØØh          4       38ØØh     *
* 15    3aØØh           -------------          6       4ØØØh     *
* 17    3cØØh           2       2eØØh                            *
* 19    3eØØh           4       32ØØh                            *
* 21    4ØØØh           6       36ØØh                            *
```

ABOOT8.ASM
Provided with M26
1/13/82

MAC ABOOT8 $+S

1/16/82 changes shown

```
* 23    42ØØh              8      3aØØh                        *
* 25    44ØØh              1Ø     3eØØh                        *
* -----------              12     42ØØh                        *
* 2     2dØØh                                                  *
* 4     2fØØh                                                  *
* 6     31ØØh              Discus M1Ø, M2Ø, M26                *
* 8     33ØØh              3      e4ØØh (If 2D is present)     *
* 1Ø    35ØØh              4      e6ØØh (If 2D is present)     *
* 12    37ØØh           ** 2Ø     45ØØh                        *
* 14    39ØØh              5      27ØØh                        *
* 16    3bØØh              6      29ØØh                        *
* 18    3dØØh              7      2bØØh                        *
* 2Ø    3fØØh              8      2dØØh                        *
* 22    41ØØh              9      2fØØh                        *
* 24    43ØØh              1Ø     31ØØh                        *
* 26    45ØØh              11     33ØØh                        *
*                          12     35ØØh                        *
*                          13     37ØØh                        *
*                          14     39ØØh                        *
*                          15     3bØØh                        *
*                          16     3dØØh                        *
*                          17     3fØØh                        *
*                          18     41ØØh                        *
*                          19     43ØØh                        *
*                                                              *
****************************************************************

        title   '*** Cold Boot Loader for CP/M Ver. 2.2 ***'

msize   equ     ~~28~~ 56       ;Memory size of target CP/M      <-------- JJD 1/16/82
9000 bias    equ     (msize-2Ø)*1Ø24 ;Memory offset from 2Øk system
B700 ccp     equ     27ØØh+bias      ;Console command processor
CD00 bios    equ     ccp+16ØØh       ;CBIOS address
retries equ     1Ø              ;Maximum # of disk retries

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

first   equ     ~~1~~ 0         ;Ø = Floppies are A-D drives and    <-------- JJD 1/16/82
                                ;       Hard Disk are E-P
                                ;1 = Hard Disks are A-L drives and
                                ;       Floppies are M-P
maxhd   equ     1               ;Set to number of hard disks
maxflop equ     ~~4~~ 2         ;Set to number of floppies          <-------- ~~LEFT AT 4. LOOKS LIKE ALL IS TESTED~~
                                                                               IS ~~"ZERO" OR "NOT EQUAL TO ZERO"~~
****************************************************************
*                                                              *
* The following equates are for the Diskus Hard disk if wanted. *
*                                                              *
****************************************************************

        if      (maxhd ne Ø) and first  ;Want Hard Disk included ?
hdorg   equ     5Øh             ;Hard Disk Controller
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
tkz     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
indx    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
secln   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

****************************************************************
*                                                              *
* Cold Boot loader common to all sector sizes.                 *
* This sector is loaded into memory at e7ØØh in a standard     *
* configuration. It is responsible for reading most of track Ø *
* into memory on cold boots.                                   *
*                                                              *
****************************************************************


        if      maxflop ne Ø
origin  equ     ØEØØØH
djram   equ     origin+4ØØh
djboot  equ     djram
        endif

        if      (maxflop ne Ø)
2700 offsetb equ     9ØØh-origin
        else
offsetb equ     Ø
        endif

        if      not first
putden  equ     origin+Ø2dh     ;Set density routine on Disk Jockey 2D
putdma  equ     origin+12h      ;Disk Jockey 2D set DMA address routine
getstat equ     origin+27h      ;Disk status routine on Disk Jockey 2D
putsec  equ     origin+ØfH      ;Disk Jockey 2D set sector routine
puttrk  equ     origin+Øch      ;Disk Jockey 2D set track routine
puthom  equ     origin+9h       ;Disk Jockey 2D track Ø seek
doread  equ     origin+15h      ;Disk Jockey 2D read routine
boterr  equ     origin+2ah      ;Disk Jockey 2D flash error light routine

        org     origin          ;Disk Jockey 2D ram

diff    set     origin+7ØØh-$   ;Offset to boot loader address

        lxi     sp,stac+diff
firmlod mvi     a,6             ;Previous sector #
newsec  equ     $-1
        inr     a               ;Update sector #
        inr     a
        cpi     27              ;Test if all done
        jz      ccp+5ØØh
        jc      nowrap+diff     ;Test if wrap around
        sui     19
nowrap  sta     newsec+diff     ;Save the updated sector #
```

```
        mov     c,a
        call    putsec          ;Set up the sector
        lxi     h,ccp+4ØØh      ;Previous DMA address
newdma  equ     $-2
        lxi     d,1ØØh          ;Update DMA address
        dad     d
        mov     a,h
        cpi     (ccp+98Øh)/1ØØh
        jc      nowrp+diff
        jnz     wrp+diff
        mov     a,l
        cpi     (ccp+98Øh) mod 1ØØh
        jc      nowrp+diff
wrp     lxi     d,-98Øh
        dad     d
nowrp   shld    newdma+diff     ;Save the updated DMA address
        mov     b,h
        mov     c,l
        call    putdma          ;Set up the new DMA address
        lxi     b,retries*1ØØh+Ø;Maximum # of errors
fread   push    b
        call    puttrk          ;Set up the proper track
        call    doread          ;Read the sector
        pop     b
        jnc     firmlod+diff    ;Continue if no error
        dcr     b
        jnz     fread+diff      ;Keep trying if error
        jmp     boterr          ;To many errors, flash the light
        ds      8Øh-($ mod 8Øh)
stac    equ     $

******************************************************************
*                                                                *
* The following equates relate to 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.                 *
*                                                                *
******************************************************************

djcin   equ     djram+3h        ;Disk Jockey 2D character input routine
djcout  equ     djram+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
djdmast equ     djram+24h       ;Disk Jockey 2D dma status
djstat  equ     djram+27h       ;Disk Jockey 2D status routine
djerr   equ     djram+2ah       ;Disk Jockey 2D error, blink led
djden   equ     djram+2dh       ;Disk Jockey 2D set density routine
djtstat equ     djram+21h       ;Disk Jockey 2D terminal status routine
djside  equ     djram+3Øh       ;Disk Jockey 2D set side routine

******************************************************************
*                                                                *
* The following three sectors of code reside at 8ØH. There is  *
* one sector for each of the possible sector sizes (256,512,    *
* 1Ø24). Each sector is responsible for performing a Cold Boot   *
* for the specified sector size.                                 *
*                                                                *
******************************************************************

diff    set     8Øh-$
```

```
        lxi     sp,cstk256+diff ;Set up stack at end of this sector
        lxi     b,26*1ØØh+1     ;B = sector count, C = sector #
clod256 push    b               ;Save sector and count
        call    djsec           ;Set the next sector to read
        lxi     h,ccp+3ØØh      ;Get DMA address (self modifying)
cdma256 equ     $-2             ;Storage for previous DMA address
        lxi     d,2ØØh          ;Offset to new DMA address
        dad     d               ;Add in offset, HL = new DMA address
        shld    cdma256+diff    ;Save new DMA address
        mov     b,h             ;Put DMA address into BC
        mov     c,l
        call    djdma           ;Set the DMA address
        call    crd256+diff     ;Attempt a read
        pop     b               ;Recover sector number and count
                                ;       B = count, C = number
        dcr     b               ;Update sector count
        jz      bios            ;All done ?
        mvi     a,2             ;Sector update
        add     c               ;Add in the sector skew factor
        mov     c,a             ;Put new sector back into C
        cpi     27              ;Past the end of the track ?
        jc      clod256+diff    ;Take jump if not past end of track
        sui     25              ;Perform a negative sector adjustment
        mov     c,a             ;Put new sector in C
        lxi     h,ccp+4ØØh      ;Negative DMA adjustment
        shld    cdma256+diff    ;Save the new DMA address
        jmp     clod256+diff    ;Continue reading

****************************************************************
*                                                              *
* Crd256 does the actual read from the controller, the DMA     *
* address and sector # have already been set up.               *
*                                                              *
****************************************************************

crd256  lxi     b,retries*1ØØh+1 ;Maximum # of attempts
cr256   push    b               ;Save error count
        call    djtrk           ;Initialize the track
        call    djread          ;Attempt the read
        pop     b               ;Restore the error count
        rnc                     ;Return if no error
        dcr     b               ;Update error count
        jnz     cr256+diff      ;Try again if not to many errors
        jmp     djerr           ;Go and flash the light on controller

        ds      8Øh-($ mod 8Øh)
cstk256 equ     $

****************************************************************
*                                                              *
* The next loads CP/M from a 512 byte sector diskette.         *
*                                                              *
****************************************************************

diff    set     8Øh-$

        lxi     sp,cstk512+diff ;Set up stack at end of this sector
        lxi     b,13*1ØØh+1     ;B = sector count, C = sector #
clod512 push    b               ;Save sector and count
        call    djsec           ;Set the next sector to read
        lxi     h,ccp+1ØØh      ;Get DMA address (self modifying)
cdma512 equ     $-2             ;Storage for previous DMA address
        lxi     d,4ØØh          ;Offset to new DMA address
        dad     d               ;Add in offset, HL = new DMA address
        shld    cdma512+diff    ;Save new DMA address
        mov     b,h             ;Put DMA address into BC
        mov     c,l
        call    djdma           ;Set the DMA address
        call    crd512+diff     ;Attempt a read
        pop     b               ;Recover sector number and count
                                ;       B = count, C = number
        dcr     b               ;Update sector count
        jz      bios            ;All done ?
        mvi     a,2             ;Sector update
        add     c               ;Add in the sector skew factor
        mov     c,a             ;Put new sector back into C
        cpi     14              ;Past the end of the track ?
        jc      clod512+diff    ;Take jump if not past end of track
        sui     13              ;Perform a negative sector adjustment
        mov     c,a             ;Put new sector in C
        lxi     h,ccp+3ØØh      ;Negative DMA adjustment
        shld    cdma512+diff    ;Save the new DMA address
        jmp     clod512+diff    ;Continue reading

*****************************************************************
*                                                               *
* Crd512 does the actual read from the controller, the DMA      *
* address and sector # have already been set up.                *
*                                                               *
*****************************************************************

crd512  lxi     b,retries*1ØØh+1 ;Maximum # of attempts
cr512   push    b               ;Save error count
        call    djtrk           ;Initialize the track
        call    djread          ;Attempt the read
        pop     b               ;Restore the error count
        rnc                     ;Return if no error
        dcr     b               ;Update error count
        jnz     cr512+diff      ;Try again if not to many errors
        jmp     djerr           ;Go and flash the light on controller

        ds      8Øh-($ mod 8Øh)
cstk512 equ     $

*****************************************************************
*                                                               *
* The next sector loads CP/M from a 1Ø24 byte sector diskette.  *
*                                                               *
*****************************************************************

diff    set     8Øh-$

        lxi     sp,cstk124+diff ;Set up stack at end of this sector
        lxi     b,1*1ØØh+7      ;B = sector count, C = sector #
        call    clod124+diff    ;Load sector 5 into CCP
        lxi     h,ccp+1dØØh     ;Destination of move
        lxi     d,ccp+5ØØh      ;Source of move
        lxi     b,2ØØh
cmov124 ldax    d               ;Get a byte of source
        mov     m,a             ;Move it
        inx     h               ;Bump destination
        inx     d               ;Bump source
        dcr     c               ;All done with this page ?
        jnz     cmov124+diff
        dcr     b
        jnz     cmov124+diff
        lxi     h,ccp-3ØØh      ;Initial DMA address
        shld    cdma124+diff
        lxi     b,6*1ØØh+1      ;B = sector count, C = sector #
        call    clod124+diff
        jmp     bios
```

```
clodl24 push    b               ;Save sector and count
        call    djsec           ;Set the next sector to read
        lxi     h,ccp-3ØØh      ;Get DMA address (self modifying)
cdmal24 equ     $-2             ;Storage for previous DMA address
        lxi     d,8ØØh          ;Offset to new DMA address
        dad     d               ;Add in offset, HL = new DMA address
        shld    cdmal24+diff    ;Save new DMA address
        mov     b,h             ;Put DMA address into BC
        mov     c,l
        call    djdma           ;Set the DMA address
        call    crdl24+diff     ;Attempt a read
        pop     b               ;Recover sector number and count
                                ;       B = count, C = number
        dcr     b               ;Update sector count
        rz                      ;All done ?
        mvi     a,2             ;Sector update
        add     c               ;Add in the sector skew factor
        mov     c,a             ;Put new sector back into C
        cpi     7               ;Past the end of the track ?
        jc      clodl24+diff    ;Take jump if not past end of track
        sui     5               ;Perform a negative sector adjustment
        mov     c,a             ;Put new sector in C
        lxi     h,ccp+1ØØh      ;Negative DMA adjustment
        shld    cdmal24+diff    ;Save the new DMA address
        jmp     clodl24+diff    ;Continue reading

****************************************************************
*                                                              *
* Rdl24 does the actual read from the controller, the DMA      *
* address and sector # have already been set up.               *
*                                                              *
****************************************************************

crdl24  lxi     b,retries*1ØØh+1 ;Maximum # of attempts
crl24   push    b               ;Save error count
        call    djtrk           ;Initialize the track
        call    djread          ;Attempt the read
        pop     b               ;Restore the error count
        rnc                     ;Return if no error
        dcr     b               ;Update error count
        jnz     crl24+diff      ;Try again if not to many errors
        jmp     djerr           ;Go and flash the light on controller

        ds      8Øh-($ mod 8Øh)
cstkl24 equ     $

****************************************************************
*                                                              *
* The next three sectors of code also reside at 8ØH. There is  *
* one sector for each of the possible sector sizes (256,512,   *
* 1Ø24). Each sector is responsible for performing a WARM Boot *
* for the specified sector size.                               *
*                                                              *
* The following table shows how sectors are read in, skewing   *
* of the sectors is necessary because sequential sectors can   *
* not be read without waiting one complete revolution between  *
* each one. Entries of ---- represent a wrap around (negative  *
* DMA adjustment). An entry flagged with ** represents only a  *
* partial load from that sector.                               *
*                                                              *
* 256                   512                     1Ø24           *
* sec   address         sec     address         sec    address *
* 1     31ØØh           1       31ØØh           1      31ØØh   *
* 3     33ØØh           3       35ØØh           3      39ØØh   *
* 5     35ØØh           5       39ØØh        ** 5      41ØØh   *
* 7     37ØØh           7       3dØØh           -------------  *
* 9     3900h          ** 9         4100h          2        3500h    *
* 11    3b00h             -------------            4        3d00h    *
* 13    3d00h             2         3300h                            *
* 15    3f00h             4         3700h                            *
* 17    4100h             6         3b00h                            *
* -----------             8         3f00h                            *
* 2     3200h                                                        *
* 4     3400h                                                        *
* 6     3600h                                                        *
* 8     3800h                                                        *
* 10    3a00h                                                        *
* 12    3c00h                                                        *
* 14    3e00h                                                        *
* 16    4000h                                                        *
*                                                                    *
**********************************************************************

diff    set     80h-$

        lxi     sp,wstk256+diff ;Set up stack at end of this sector
        lxi     b,17*100h+1     ;B = sector count, C = sector #
wlod256 push    b               ;Save sector and count
        call    djsec           ;Set the next sector to read
        lxi     h,ccp+300h      ;Get DMA address (self modifying)
wdma256 equ     $-2             ;Storage for previous DMA address
        lxi     d,200h          ;Offset to new DMA address
        dad     d               ;Add in offset, HL = new DMA address
        shld    wdma256+diff    ;Save new DMA address .
        mov     b,h             ;Put DMA address into BC
        mov     c,l
        call    djdma           ;Set the DMA address
        call    wrd256+diff     ;Attempt a read
        pop     b               ;Recover sector number and count
                                ;       B = count, C = number
        dcr     b               ;Update the sector count
        jz      bios+3          ;All done ?
        mvi     a,2             ;Sector update
        add     c               ;Add in the sector skew factor
        mov     c,a             ;Put new sector back into C
        cpi     19              ;Past the end of the track ?
        jc      wlod256+diff    ;Take jump if not past end of track
        sui     17              ;Perform a negative sector adjustment
        mov     c,a             ;Put new sector in C
        lxi     h,ccp+400h      ;Negative DMA adjustment
        shld    wdma256+diff    ;Save the new DMA address
        jmp     wlod256+diff    ;Continue reading

**********************************************************************
*                                                                    *
* Wrd256 does the actual read from the controller, the DMA           *
* address and sector # have already been set up.                     *
*                                                                    *
**********************************************************************

wrd256  lxi     b,retries*100h+1 ;Maximum # of attempts
wr256   push    b               ;Save error count
        call    djtrk           ;Initialize the track
        call    djread          ;Attempt the read
        pop     b               ;Restore the error count
        rnc                     ;Return if no error
        dcr     b               ;Update error count
        jnz     wr256+diff      ;Try again if not to many errors
        jmp     djerr           ;Go and flash the light on controller

        ds      80h-($ mod 80h)
wstk256 equ     $
```

```
*****************************************************************
*                                                               *
* Disk Jockey 2D CP/M from a 512 byte sector diskette.          *
*                                                               *
*****************************************************************

diff    set     8Øh-$

        lxi     sp,wstk512+diff ;Set up stack at end of this sector
        lxi     b,1*1ØØh+9      ;B = sector count, C = sector #
        call    wlod512+diff    ;Load sector 9 into CCP
        lxi     h,ccp+15ØØh     ;Destination of move
        lxi     d,ccp+5ØØh      ;Source of move
        mvi     c,Ø
mov512  ldax    d               ;Get a byte of source
        mov     m,a             ;Move it
        inx     h               ;Bump destination
        inx     d               ;Bump source
        dcr     c               ;All done with this page ?
        jnz     mov512+diff
        lxi     h,ccp+3ØØh      ;Initial DMA address
        shld    wdma512+diff
        lxi     b,8*1ØØh+2      ;B = sector count, C = sector #
        call    wlod512+diff
        jmp     bios+3
wlod512 push    b               ;Save sector and count
        call    djsec           ;Set the next sector to read
        lxi     h,ccp+1ØØh      ;Get DMA address (self modifying)
wdma512 equ     $-2             ;Storage for previous DMA address
        lxi     d,4ØØh          ;Offset to new DMA address
        dad     d               ;Add in offset, HL = new DMA address
        shld    wdma512+diff    ;Save new DMA address
        mov     b,h             ;Put DMA address into BC
        mov     c,l
        call    djdma           ;Set the DMA address
        call    wrd512+diff     ;Attempt a read
        pop     b               ;Recover sector number and count
                                ;       B = count, C = number
        dcr     b               ;Update sector count
        rz                      ;All done ?
        mvi     a,2             ;Sector update
        add     c               ;Add in the sector skew factor
        mov     c,a             ;Put new sector back into C
        cpi     1Ø              ;Past the end of the track ?
        jc      wlod512+diff    ;Take jump if not past end of track
        sui     9               ;Perform a negative sector adjustment
        mov     c,a             ;Put new sector in C
        lxi     h,ccp+1ØØh      ;Negative DMA adjustment
        shld    wdma512+diff    ;Save the new DMA address
        jmp     wlod512+diff    ;Continue reading

*****************************************************************
*                                                               *
* Wrd512 does the actual read from the controller, the DMA      *
* address and sector # have already been set up.                *
*                                                               *
*****************************************************************

wrd512  lxi     b,retries*1ØØh+1 ;Maximum # of attempts
wr512   push    b               ;Save error count
        call    djtrk           ;Initialize the track
        call    djread          ;Attempt the read
        pop     b               ;Restore the error count
        rnc                     ;Return if no error
        dcr     b               ;Update error count
        jnz     wr512+diff      ;Try again if not to many errors
        jmp     djerr           ;Go and flash the light on controller

        ds      8Øh-($ mod 8Øh)
wstk512 equ     $

****************************************************************
*                                                              *
* The next sector loads CP/M from a 1Ø24 byte sector diskette. *
*                                                              *
****************************************************************

diff    set     8Øh-$

        lxi     sp,wstk124+diff ;Set up stack at end of this sector
        lxi     b,1*1ØØh+5      ;B = sector count, C = sector #
        call    wlod124+diff    ;Load sector 6 into CCP
        lxi     h,ccp+15ØØh     ;Destination of move
        lxi     d,ccp+5ØØh      ;Source of move
        mvi     c,Ø
mov124  ldax    d               ;Get a byte of source
        mov     m,a             ;Move it
        inx     h               ;Bump destination
        inx     d               ;Bump source
        dcr     c               ;All done with this page ?
        jnz     mov124+diff
        lxi     h,ccp+1ØØh      ;Initial DMA address
        shld    wdma124+diff
        lxi     b,4*1ØØh+2      ;B = sector count, C = sector #
        call    wlod124+diff
        jmp     bios+3
wlod124 push    b               ;Save sector and count
        call    djsec           ;Set the next sector to read
        lxi     h,ccp-3ØØh      ;Get DMA address (self modifying)
wdma124 equ     $-2             ;Storage for previous DMA address
        lxi     d,8ØØh          ;Offset to new DMA address
        dad     d               ;Add in offset, HL = new DMA address
        shld    wdma124+diff    ;Save new DMA address
        mov     b,h             ;Put DMA address into BC
        mov     c,l
        call    djdma           ;Set the DMA address
        call    wrd124+diff     ;Attempt a read
        pop     b               ;Recover sector number and count
                                ;       B = count, C = number
        dcr     b               ;Update sector count
        rz                      ;All done ?
        mvi     a,2             ;Sector update
        add     c               ;Add in the sector skew factor
        mov     c,a             ;Put new sector back into C
        cpi     6               ;Past the end of the track ?
        jc      wlod124+diff    ;Take jump if not past end of track
        sui     5               ;Perform a negative sector adjustment
        mov     c,a             ;Put new sector in C
        lxi     h,ccp-3ØØh      ;Negative DMA adjustment
        shld    wdma124+diff    ;Save the new DMA address
        jmp     wlod124+diff    ;Continue reading

****************************************************************
*                                                              *
* Wr124 does the actual read from the controller, the DMA      *
* address and sector # have already been set up.               *
*                                                              *
****************************************************************

wrd124  lxi     b,retries*1ØØh+1 ;Maximum # of attempts
wr124   push    b               ;Save error count
```

```
        call    djtrk           ;Initialize the track
        call    djread          ;Attempt the read
        pop     b               ;Restore the error count
        rnc                     ;Return if no error
        dcr     b               ;Update error count
        jnz     wrl24+diff      ;Try again if not to many errors
        jmp     djerr           ;Go and flash the light on controller

        ds      8Øh-($ mod 8Øh)
wstkl24 equ     $

****************************************************************
*                                                              *
* The next sector of code resides at CCP+5ØØh. It's task is to *
* move the firmware code into the Disk Jockey Ram, then        *
* loading a sector into 8ØH which will load the rest of CP/M.  *
* The sector loaded at 8ØH is dependent on the sector size     *
* of the diskette being booted from.                           *
*                                                              *
****************************************************************


diff    set     ccp+5ØØh-$      ;Used to relocate this sector of code

        jmp     docold+diff     ;Jump to cold boot portion
        jmp     dowarm+diff     ;Jump to warm boot portion

docold  lxi     sp,stk+diff     ;Set up initial stack at end of this sector
        lxi     h,djram         ;Destination pointer
        lxi     d,stk+diff      ;Source pointer
        lxi     b,3ØØh          ;Length of transfer
mloop   ldax    d               ;Get one byte of source
        mov     m,a             ;Put at destination
        inx     h               ;Bump destination
        inx     d               ;Bump source
        dcx     b               ;Update count of bytes to move
        mov     a,b             ;Test if all done
        ora     c
        jnz     mloop+diff      ;Continue moving New Firmware
        call    djboot          ;Initialize the new firmware
        mvi     a,1
        sta     botbias+diff    ;Set up for proper sector select

dowarm  lxi     sp,stk+diff
        mvi     c,1             ;Set the density to double
        call    djden
        mvi     c,1             ;Set up to read sector 1 on track 1
        mov     a,c
        sta     trknum+diff     ;Set track
        call    djsec           ;Set sector
        lxi     b,stk+diff      ;Set the DMA address
        call    djdma
        call    reed+diff       ;Read the sector into memory at
                                ;       end of this sector
        call    djstat          ;Determine the sector size
        ani     Øch             ;Strip off unwanted bits
        rar                     ;Form the desired sector for Cold Boot
        rar                     ;       based on the length of the
        adi     4               ;       sectors on this diskette
botbias equ     $-1
        mov     c,a             ;Prepare to read the Cold Boot
        call    djsec           ;Set up the sector
        xra     a               ;Track Ø
        sta     trknum+diff
        lxi     b,8Øh           ;Cold Boot loads at 8ØH
        push    b               ;Used as jump address to Cold Boot--
        call    djdma           ;                                  |
        mvi     c,Ø             ;Density on track Ø is single      |
        call    djden           ;                                  |
        call    reed+diff       ;Read in the Cold Boot             |
        mvi     c,1             ;Set the density back to double    |
        jmp     djden           ;                                  |
                                ;Go to the Cold Boot <--------------

*****************************************************************
*                                                               *
* Reed does the actual read from the controller, the DMA        *
* address and sector # have already been set up.                *
*                                                               *
*****************************************************************

reed    lxi     b,retries*1ØØh+Ø ;Maximum # of attempts
trknum  equ     $-2             ;Storage for track number
reedl   push    b               ;Save error count
        call    djtrk           ;Initialize the track
        call    djread          ;Attempt the read
        pop     b               ;Restore the error count
        rnc                     ;Return if no error
        dcr     b               ;Update error count
        jnz     reedl+diff      ;Try again if not to many errors
        jmp     djerr           ;Go and flash the light on controller

        ds      8Øh-($ mod 8Øh)
stk     equ     $
        else

*****************************************************************
*                                                               *
* Cold Boot loader for Discus M1Ø, M2Ø, M26.                    *
*                                                               *
*****************************************************************

        if      maxflop ne Ø
        org     origin          ;Org program at Floppy origin
        else
        org     1ØØh            ;  or else at 1ØØh
        endif
diff    set     1ØØh-$

boothd  lxi     sp,cstkhd+diff  ;Set up stack at end of this sector
        lxi     b,1*1ØØh+2Ø     ;B = sector count, C = sector #
        call    clodhd+diff     ;Load sector 2Ø into CCP
        lxi     h,ccp+1eØØh     ;Destination of move
        lxi     d,ccp           ;Source of move
        mvi     c,Ø
cmovhd  ldax    d               ;Get a byte of source
        mov     m,a             ;Move it
        inx     h               ;Bump destination
        inx     d               ;Bump source
        dcr     c               ;All done with this page ?
        jnz     cmovhd+diff
        lxi     h,ccp-2ØØh      ;Initial DMA address
        shld    cdmahd+diff
        lxi     b,15*1ØØh+5     ;B = sector count, C = sector #
        call    clodhd+diff

        if      maxflop ne Ø
        mvi     c,17            ;Check if Discus 2D is present
        mvi     a,(jmp)         ;Should be 17 jumps in the jump table
        lxi     h,origin
clop    cmp     m
        jnz     bios            ;Not 17 jumps, don't read in the ramware
        inx     h               ;Skip over the jump instruction
        inx     h
        inx     h
        dcr     c               ;Update jump counter
        jnz     clop+diff       ;Continue checking
        lxi     h,djram-2ØØh    ;Load in the Disk Jockey 2D ramware
        shld    cdmahd+diff
        lxi     b,2*1ØØh+3
        call    clodhd+diff
        endif
        jmp     bios            ;Go to CP/M
clodhd  push    b               ;Save sector and count
        mov     a,c
        sta     hdsec+diff
        lxi     h,ccp-2ØØh      ;Get DMA address (self modifying)
cdmahd  equ     $-2             ;Storage for previous DMA address
        lxi     d,2ØØh          ;Offset to new DMA address
        dad     d               ;Add in offset, HL = new DMA address
        shld    cdmahd+diff     ;Save new DMA address
        call    crdhd+diff      ;Attempt a read
        pop     b               ;Recover sector number and count
                                ;       B = count, C = number
        dcr     b               ;Update sector count
        rz                      ;All done ?
        inr     c
        jmp     clodhd+diff     ;Continue reading

*****************************************************************
*                                                               *
* Rdhd does the actual read from the controller, the DMA        *
* address and sector # have already been set up.                *
*                                                               *
*****************************************************************

crdhd   lxi     b,retries*1ØØh+1 ;Maximum # of attempts
crhd    push    b               ;Save error count
        call    hdread+diff     ;Attempt the read
        pop     b               ;Restore the error count
        rnc                     ;Return if no error
        dcr     b               ;Update error count
        jnz     crhd+diff       ;Try again if not to many errors
        jmp     $               ;Dynamic error halt

hdread  call    hdprep+diff     ;Prepare the sector header image
        rc                      ;Error exit
        mvi     a,rsect         ;Read sector command
        out     hdcmnd
        call    process+diff    ;Process the read
        rc                      ;Error exit
        xra     a               ;Pointer to data buffer
        out     hdcmnd
        mvi     b,secln/4       ;Number of bytes to read
        lhld    cdmahd+diff     ;Get destination of data
        in      hddata          ;Two dummy data bytes
        in      hddata
rtloop  in      hddata          ;Move four bytes
        mov     m,a             ;Byte one
        inx     h
        in      hddata          ;Byte two
        mov     m,a
        inx     h
        in      hddata          ;Byte three
        mov     m,a
        inx     h
        in      hddata          ;Byte four
        mov     m,a
        inx     h
        dcr     b               ;Update byte count
        jnz     rtloop+diff
        ret

process in      hdstat          ;Wait for command to finish
        mov     b,a
        ani     opdone
        jz      process+diff
        mvi     a,dskclk        ;Turn on Disk Clock
        out     hdcntl
        in      hdstat
        ani     tmout           ;Timed out ?
        stc
        rnz
        in      hdreslt
        ani     retry           ;Any retries ?
        stc
        rnz
        xra     a               ;No error exit
        ret

hdprep  in      hdstat          ;Is Drive ready ?
        ani     drvrdy
        stc
        rnz
        mvi     a,isbuff        ;Initialize pointer to header buffer
        out     hdcmnd
        mvi     a,null
        out     hdfunc          ;Select drive A
        xra     a
        out     hddata          ;Form head byte
        out     hddata          ;Form track byte
        mvi     a,Ø             ;Form sector byte
hdsec   equ     $-1
        out     hddata
        mvi     a,8Øh           ;Form Key
        out     hddata
        mvi     a,dskclk        ;Turn on Disk clock
        out     hdcntl
        mvi     a,wenabl        ;Write enable on
        out     hdcntl
        ret

        org     boothd+2ØØh-2

cstkhd  equ     $
        dw      boothd+diff
        ds      2ØØh
        endif

****************************************************************
*                                                              *
* Disk Jockey 2D firmware revision 3.1 and 4.Ø                 *
* By George Morrow                                             *
*                                                              *
* The following firmware is loaded into memory and then moved  *
* into the controller ram.                                     *
*                                                              *
****************************************************************

        if      maxflop ne Ø
rom     equ     origin
ram     equ     origin+4ØØh

IO      EQU     ROM+3f8h
```

```
UDATA     EQU     IO
DCMD      EQU     IO+1
DSTAT     EQU     DCMD
DREG      EQU     IO+2
USTAT     EQU     DREG
CMDREG    EQU     IO+4
CSTAT     EQU     CMDREG
TRKREG    EQU     IO+5
SECREG    EQU     IO+6
DATREG    EQU     IO+7
*
*
RCMD      EQU     2ØØQ
WCMD      EQU     24ØQ
HEAD      EQU     4
LOAD      EQU     2ØQ
DENSTY    EQU     1
ULOAD     EQU     3ØQ
RSTBIT    EQU     4
ACCESS    EQU     2
READY     EQU     4ØQ
INDEX     EQU     2ØQ
RACMD     EQU     3Ø4Q
CLRCMD    EQU     32ØQ
SVCMD     EQU     35Q
SKCMD     EQU     3ØQ
HCMD      EQU     11Q
ISTAT     EQU     4
OSTAT     EQU     1ØQ
DSIDE     EQU     1ØQ
TZERO     EQU     4
MDINT     EQU     3
LIGHT     EQU     36Q
NOLITE    EQU     76Q
*
*
DBOOT     JMP     BOOT
TERMIN    JMP     origin+3
TRMOUT    JMP     origin+6
TKZERO    JMP     HOME
TRKSET    JMP     SEEK
SETSEC    JMP     SECSET
SETDMA    JMP     DMA
DREAD     JMP     READ
DWRITE    JMP     WRITE
SELDRV    JMP     DRIVE
TPANIC    JMP     origin+leh
TSTAT     JMP     origin+21h
DMAST     JMP     DMSTAT
STATUS    JMP     DISKST
DSKERR    JMP     ROM+52Q
SETDEN    JMP     DENFIX
SETSID    JMP     SIDEFX
*
*
BOOT      mov     a,c
          sta     disk
          sta     bdisk
          mvi     a,7fh
dsrt      rlc
          dcr     c
          jp      dsrt
          sta     drvsel
          sta     bdrvsel
          if      first
          call    fixio
```

```
        endif
        lda     io-4            ;Test for Model A or B
        cpi     (ret)
        jz      modela
        lxi     d,origin
        lxi     h,ram
        mvi     c,boot-dboot    ;Copy prom jump table into ram
        if      first
        call    modelm
        mvi     a,7fh
        call    sdsel
        mvi     a,clrcmd
        sta     cmdreg
        lda     bdrvsel
        ori     6Øh
        ani     7fh
        call    sdsel
        mvi     a,9h
        call    scbits
        mvi     a,(jz)
        sta     indx1
        sta     indx3
        mvi     a,(jnz)
        sta     indx2
        call    measur
        xchg
        shld    btimer
        lxi     d,btble
        lxi     h,origin+7e3h
        mvi     c,23
        endif
modelm  ldax    d
        mov     m,a
        inx     d
        inx     h
        dcr     c
        jnz     modelm
        ret

btble   db      Ø
btimer  dw      18ØØh,Ø
        db      8,Ø
bdrvsel db      7eh
bdisk   db      Ø,8,Ø,9,Øffh,9,Øffh,9,Øffh,9,Øffh,9,Ø,1,Ø

fixio   LHLD    ROM+7           ;find the 2nd
        INX     H               ;   byte of input routine
        LXI     D,4             ;offset
        MOV     A,M             ;get addr of USTAT
        LXI     H,SDSEL+1       ;I/O routines
        MOV     M,A             ;store USTAT addr
        DAD     D               ;increment mem addr
        MOV     M,A             ;store USTAT addr
        DAD     D               ;increment mem addr
        XRI     3               ;switch the addr
        MOV     M,A             ;store DSTAT addr
        DAD     D               ;increment mem addr
        MOV     M,A             ;store DATAT addr
        DAD     D               ;increment mem addr
        MOV     M,A             ;store DSTAT addr
        ret

modela  equ     $
        if      not first
        mvi     c,Ø             ;Copy last page of ram
        call    modelm
        endif
        call    fixio
        mvi     a,3
        call    scbits
        MVI     A,CLRCMD        ;1791 reset
        STA     CMDREG          ;   command
        lda     drvsel          ;initialize 1791
        call    sdsel           ;    control bits
        call    measur
        xchg
        SHLD    TIMER
        RET
*
*
DISKST
        LDA     SECREG          ;get current
        MOV     B,A             ;   sector no in B
        LDA     TRKREG          ;get current
        MOV     C,A             ;   track no in C
        LDA     DCREG           ;get current
        CMA                     ;   density in
        ANI     1               ;   the msb
        RRC                     ;position
        MOV     D,A             ;save in D
        LDA     SIDE            ;put the
        RLC                     ;   side
        RLC                     ;   select
        RLC                     ;   flag
        ADD     D               ;   in bit
        MOV     D,A             ;   position 6
        LDA     SECLEN          ;put the
        RLC                     ;   sector length
        RLC                     ;   code P bits
        ADD     D               ;   2 & 3
        MOV     D,A
        LDA     CDISK           ;put the current
        ADD     D               ;   disk no in bits
        RET                     ;   Ø & 1
*
*
DMSTAT
        PUSH    H               ;save the H-L pair
        LHLD    DMAADR          ;H-L pain
        MOV     B,H             ;move the DMA
        MOV     C,L             ;addr to B-C
        POP     H               ;recover H-L
        RET
*
*
DRIVE
        MVI     A,374Q          ;test for the
        ADD     C               ;   new drive number
        MVI     A,2ØQ           ;less than 4
        RC
        MOV     A,C             ;store the new
        STA     DISK            ;drive in DISK
        RET
*
*
DMA
        LXI     H,8-ROM         ;test the
        DAD     B               ;   DMA address
        JNC     DMASET          ;   for conflict
        LXI     H,-RAM          ;   with the I/O
        DAD     B               ;   on the DJ/2D
        JC      DMASET          ;controller
        STC
        MVI     A,2ØQ
        RET
DMASET
        MOV     H,B             ;get the DMA addr
        MOV     L,C             ;to the H-L par
        SHLD    DMAADR          ;store
        XRA     A               ;clear the error
        RET                     ;   flag and return
*
*
HOME
        call    hdload          ;load the head
        rc                      ;not ready error
        call    hentry          ;move the head
        PUSH    PSW             ;save the flags
        SBB     A               ;update the
        STA     TRACK           ; track
        sta     trkreg          ; registers
        xra     a               ;set the not
        sta     tzflag          ; verified flag
        JMP     LEAVE+1         ;unload the head
HENTRY
        XRA     A               ;update
        STA     HDFLAG          ;   flags
        LXI     H,Ø             ;time out constant
        MVI     A,HCMD          ;do the home
        CALL    CENTRY          ;   command
        ANI     TZERO           ;track zero bit
        RNZ
        STC                     ;error flag
        RET
*
*
SECSET
        XRA     A               ;test for
        ORA     C               ;   sector zero
        STC                     ;error flag
        RZ
        MOV     A,C             ;test for
        CPI     27              ;   sector
        CMC                     ;too large
        RC
        STA     SECTOR          ;save
        RET
*
*
SEEK
        MOV     A,C             ;test for
        CPI     77              ;   track
        CMC                     ;   too large
        RC
        STA     TRACK           ;save
        RET
*
*
issue
        sta     ecount+1        ;update count
        call    measur          ;find the index
        mvi     c,1             ;start w/sector 1
isloop
        mov     a,c             ;initialize the
        sta     secreg          ; sector register
        lda     sector          ;test for
        cmp     c               ; target sector
        rz
        mvi     a,rcmd          ;do a fake
        call    comand          ; read command
        jc      pleave          ;abort on error
        inr     c               ;increment sector no.
        jmp     isloop

comndp
        sta     cmdreg          ;do the command
        mov     c,b             ;initialize block count
        lxi     d,datreg        ;data register
        lhld    dmaadr          ;transfer address
        ret

write
        CALL    PREP            ;prepare for write
        jc      leave           ;abort operation
wrentry
        mvi     a,wcmd          ;start a write
        call    comndp          ; sector operation
wrloop
        mov     a,m             ;load 1st byte of data
        inx     h               ;advance pointer
        stax    d               ;write 1st byte of data
        mov     a,m             ;load 2nd byte of data
        inx     h               ;advance pointer
        stax    d               ;write 2nd byte of data
        mov     a,m             ;load 3rd byte of data
        inx     h               ;advance pointer
        stax    d               ;write 3rd byte of data
        dcr     c               ;reduce block count
        mov     a,m             ;load 4th byte of data
        inx     h               ;advance pointer
        stax    d               ;write 4th byte of data
        jnz     wrloop          ;write next 4 bytes
        lxi     h,wrentry       ;return entry address
        jmp     cbusy

read
        call    prep            ;prepare for read
        jc      leave           ;abort operation
rdentry
        mvi     a,rcmd          ;start a read
        call    comndp          ; sector operation
rdloop
        ldax    d               ;read 1st byte
        mov     m,a             ;store 1st byte
        inx     h               ;advance pointer
        ldax    d               ;read 2nd byte
        mov     m,a             ;store 2nd byte
        inx     h               ;advance pointer
        ldax    d               ;read 3rd byte
        mov     m,a             ;store 3rd byte
        inx     h               ;advance pointer
        dcr     c               ;reduce block count
        ldax    d               ;read 4th byte
        mov     m,a             ;store 4th byte
        inx     h               ;advance pointer
        jnz     rdloop          ;read next 4 bytes
        lxi     h,rdentry       ;return entry address
CBUSY
        push    h               ;save return address
        lxi     h,cstat         ;wait for the 1791
        call    busy            ; to finish command
        ani     137Q            ;error bit mask
        jz      leave-1         ; test
        cpi     1Øh             ;premature interrupt
        jnz     pleave          ;other error type
        lda     ecount          ;decrement error
        dcr     a               ; count number 1
        jm      stest           ;hard interrupt error
        sta     ecount          ;update count
        ret                     ;do operation over
stest
        lda     ecount+1        ;decrement error
        dcr     a               ; count number 2
        jp      issue           ;issue a command
        mvi     a,1Øh           ;irrecoverable error
pleave
        stc                     ;error flag
        pop     h               ;adjust the stack
LEAVE
        PUSH    PSW             ;save the flags
        LDA     DCREG           ;1791 control bits
        XRI     LOAD            ;toggle the
        CALL    SCBITS          ;   head load bits
        POP     PSW             ;recover the flags
        RET
*
*
PREP
        CALL    HDLOAD          ;load the head
        RC                      ;disk not ready?
        LDA     TRKREG          ;get the old trk
        INR     A               ;test for head
        CZ      HENTRY          ; not calibrated
        rc                      ;seek error?
        LXI     H,TRKREG        ;present trk
        LDA     TRACK           ;the new track
        CMP     M               ;test for head motion
        INX     H               ;advance to the
        INX     H               ;   data register
        MOV     M,A             ;save the new trk
        MOV     A,C             ;turn off data
        CALL    SCBITS          ;   access control bit
        JZ      TVERFY          ;test for seej
        XRA     A               ;force a read
        STA     HDFLAG          ;   header operation
        CALL    LDSTAT          ;get the
        ANI     DSIDE           ;   double
        RAR                     ;
        RAR                     ;   flag
        RAR                     ;   to do 3 ms
        ADI     SKCMD           ;   step operation
        LXI     H,Ø             ;do a seek
        CALL    CENTRY          ; command
        JC      SERROR          ;seek error?
TVERFY
        LDA     HDFLAG          ;get the force
        ORA     A               ; verify track flag
        JNZ     CHKSEC          ;no seek & head OK
        MVI     B,2             ;verify retry no
SLOOP
        MVI     A,SVCMD         ;do a verify
        CALL    COMAND          ; command
        ANI     231Q            ;error bit mask
        mov     d,a             ;save
        JZ      RDHDR           ;no error
        LDA     DCREG           ;1791 control reg
        XRI     DENSTY          ;flip the density bit
        STA     DCREG           ;update
        XRI     ACCESS
        CALL    SCBITS          ;change density
        DCR     B               ;dec retry count
        JNZ     SLOOP           ; and try again
        mov     a,d             ;restore error bits
SERROR
        stc                     ;error flag
        push    psw             ;save the status
        call    hentry          ;seek to track Ø
        pop     psw             ;recover errors
        ret
RDHDR
        MVI     B,12Q           ;number of retrys
RHLOOP
        LXI     D,DATREG        ;data register
        LXI     H,TRACK+1       ;storage area
        MVI     A,RACMD         ;do the read
        STA     CMDREG          ;   header command
RHL1
        LDAX    D               ;get a data byte
        MOV     M,A             ;store in memory
        INR     L               ;inc mem pointer
        JNZ     RHL1            ;test for more data
        LXI     H,CSTAT         ;wait for 1791
        CALL    BUSY            ;to finish cmd
        ORA     A               ;test for errors
        JZ      CHKSEC          ;transfer OK?
        DCR     B               ;dec retry count
        JNZ     RHLOOP          ;test for
        JMP     SERROR          ; hard error
CHKSEC
        LDA     SECLEN          ;get the sector
        MOV     C,A             ; size and setup
        MVI     B,Ø             ; the offset
        LXI     H,STABLE        ;sec size tbl
        DAD     B               ;add the offset
        LDA     SECTOR          ;get the sector
        MOV     B,A             ;save in B
        ADD     M               ;compare w/table entry
        MVI     A,2ØQ           ;error flag
        RC                      ;error return
        MOV     A,B             ;save the sector
        STA     SECREG          ; in sector reg
        mvi     a,4Øq           ;128 byte sector
        lxi     h,5Ø5h          ;initialize
        shld    ecount          ; error counts
SZLOOP
        DCR     C               ;reduce size count
        MOV     b,a             ;sector size to b
        rm                      ;return on minus
        ral                     ;double the count
        ora     a               ;clear the carry
        JMP     SZLOOP
*
*
SIDEFX
        MOV     A,C             ;get the side bit
        ANI     1               ;trim excess bits
        RAL                     ;move the bit
        RAL                     ;   to the side
        RAL                     ;   select bit
        RAL                     ;   position
        STA     SIDE            ;save
        RET
TOEND
*
*
        DS      3ØØh-TOEND+DBOOT-66Q
        DS      25Q
*
*
STACK
*
*
STABLE
        DB      345Q
        DB      345Q
        DB      36ØQ
        DB      367Q
*
*
ecount  dw      Ø               ;error count cells
TIMER   DW      3ØØØh           ;head load time
DMAADR  DW      2ØØQ            ;dma address
HDFLAG  DB      Ø               ;read header flag
DRVSEL  DB      376Q            ;drive select constant
DISK    DB      Ø               ;new drive
CDISK   DB      1ØQ             ;current drive
TZFLAG  DB      Ø               ;track zero indicator
DØPRAM  DB      3               ;drive Ø parameters
DØTRK   DB      377Q            ;drive Ø track no
D1PRAM  DB      3               ;drive 1 parameters
D1TRK   DB      377Q            ;drive 1 track no
D2PRAM  DB      3               ;drive 2 parameters
D2TRK   DB      377Q            ;drive 2 track no
D3PRAM  DB      3               ;drive 3 parameters
D3TRK   DB      377Q            ;drive 3 track no
DCREG   DB      3               ;current parameters
SIDE    DB      Ø               ;new side select
SECTOR  DB      3               ;new sector
TRACK   DB      Ø               ;new track
TRKNO   DB      Ø               ;disk
SIDENO  DB      Ø               ;   sector
SECTNO  DB      Ø               ;   header
SECLEN  DB      Ø               ;   data
CRCLO   DB      Ø               ;   buffer
CRCHI   DB      Ø
*
*
HDLOAD
        LXI     H,DISK
        MOV     C,M             ;new disk no to C
        INX     H
        MOV     E,M             ;current disk to E
        MOV     M,C             ;update current disk
        INX     H               ;addr of disk table
        MOV     A,E             ;test for
        CMP     C               ;   disk change
        MOV     A,M             ;head load flag
        MVI     M,HEAD          ;update head load
        INX     H               ;addr of disk table
        JZ      HDCHK           ;no disk change?
        PUSH    H               ;save table address
        MVI     D,Ø             ;set up the
        MOV     B,D             ;   offset address
        DAD     D               ;get the current
        DAD     D               ;   disk parameters
        LDA     DCREG           ;save the
        MOV     M,A             ;density info
        INX     H               ;current track
        LXI     D,TRKREG
        LDAX    D               ;get current trk
        MOV     M,A             ;save
        POP     H               ;recover tbl addr
        DAD     B               ;add the
        DAD     B               ;   offset
        MOV     A,M             ;get control bits
        STA     DCREG           ;update DCREG
        INX     H               ;get the old
        MOV     A,M             ;track number
        STAX    D               ;and update 1791
        MVI     A,177Q          ;disk select bits
DSROT
        RLC                     ;rotate to
        DCR     C               ;   select the
        JP      DSROT           ;   proper drive
        STA     DRVSEL          ;save
        XRA     A               ;force head load
HDCHK
        CALL    LOADS           ;test for
        ANA     M               ;   head loaded
        STA     HDFLAG          ;save the head
        PUSH    PSW             ;   loaded status
        LDA     DRVSEL          ;get current drive
        MOV     C,A             ;save
        LDA     SIDE            ;get current side
        CMA                     ;and merge
        ANA     C               ;   with drive select
        CALL    SDSEL           ;select drive & side
        LDA     DCREG           ;1791 control bits
        MOV     C,A             ;save
        LDA     TRACK           ;get the new trk
        SUI     1               ;force single
        SBB     A               ;   density
        DCR     A               ;   if track = Ø
        CMA                     ;compliment
        ORA     C               ;merge w/control bits
        MOV     M,A             ;set 1791 control
        XRI     ACCESS          ;toggel access bit
        MOV     C,A             ;save PREP routine
        POP     PSW             ;head load status
        JNZ     RDYCHK          ;conditionally
        PUSH    H               ;   wait for head
        LHLD    TIMER           ;   load time out
TLOOP
        DCX     H               ;count down
        MOV     A,H             ;   4Ø ms for
        ORA     L               ;   head load
        JNZ     TLOOP           ;   time out
        POP     H               ;disk status addr
RDYCHK
        MOV     A,M             ;test for
        ANI     READY           ;   disk ready
        RZ
UNLOAD
        LDA     DCREG           ;force a
        ORI     ULOAD           ;   head
        MOV     M,A             ;   unload
        MVI     A,2ØØQ          ;set disk
        STC                     ;   not ready
        RET                     ;   error flag
*
*
COMAND
        LHLD    TIMER           ;get index count
        DAD     H               ;   and multiply
        DAD     H               ;   by four
CENTRY
        XCHG                    ;save in D-E pair
        LXI     H,CSTAT         ;issue command
        MOV     M,A             ;to the 1791
NBUSY
        MOV     A,M             ;wait
        RAR                     ;   for the
        JNC     NBUSY           ;   busy flag
BUSY
        MOV     A,M             ;test for
        RAR                     ;   device busy
        MOV     A,M             ;restore status
        RNC                     ;return if not busy
        DCX     D               ;test for
        MOV     A,D             ;   two disk
        ORA     E               ;   revolutions
        JNZ     BUSY            ;47 machine cycles
        mov     e,m             ;save error code
        PUSH    H               ;save cmd address
        INX     H               ;track register
        MOV     D,M             ;save present track
        xthl                    ;recover cmd reg.
        push    d               ;save status
        xchg                    ;adjust registers
        call    loads           ;get control reg
        LDA     DCREG           ;1791 control bits
        xri     RSTBIT          ;reset the 1791
        mov     m,a             ;   controller to
        xri     rstbit          ;   clear fault
        xchg                    ;adjust registers
        stax    d               ;start controller
        MVI     M,CLRCMD        ;force an interrupt
        pop     d               ;recover status
        POP     H               ;recover track reg
        mov     m,d             ;restore track
        mov     a,e             ;restore error code
        STC                     ;   error flag
        RET
*
*
MEASUR
        LXI     D,Ø             ;initialize count
        CALL    LOADS           ;status port
        MVI     C,INDEX         ;index bit flag
INDXHI
        MOV     A,M             ;wait for
        ANA     C               ;   index
indx1   JNZ     INDXHI          ;   pulse low
INDXLO
        MOV     A,M             ;wait for
        ANA     C               ;   index
indx2   JZ      INDXLO          ;   pulse high
INDXCT
        INX     D               ;advance count
        XTHL                    ;four
        XTHL                    ;   dummy
        XTHL                    ;   instructions
        XTHL                    ;   for delay
        MOV     A,M             ;wait
        ANA     C               ;   for next
indx3   JNZ     INDXCT          ;   low index
        RET                     ;98 machine cycles
*
*
DENFIX
        MOV     A,C             ;trim excess
        ANI     1               ;   bits,
        CMA                     ;   compliment
        MOV     B,A             ;   B and save
        LXI     H,DISK          ;new disk
        MOV     E,M             ;get disk no
        MVI     D,Ø             ;offset addr
        INX     H               ;current disk
        MOV     A,M             ;move to  ACC
        XRA     E               ;compare w/new
        PUSH    PSW             ;save status
        INX     H               ;disk table
        INX     H               ;   address
        DAD     D               ;add the
        DAD     D               ;offset
        MOV     A,M             ;get parameters
        ORI     1               ;mask off density
        ANA     B               ;set new density
        MOV     M,A             ;update
        POP     PSW             ;check for nd=cd
        RNZ                     ;new disk not old
        MOV     A,M             ;update CDISK
        STA     DCREG           ;   also
        RET
*
*
SDSEL
        STA     DREG            ;drive select reg
        RET
LUSTAT
        LDA     USTAT           ;UART status reg
        RET
SCBITS
        STA     DCMD            ;1791 control reg
        RET
LDSTAT
        LDA     DSTAT           ;drive status reg
        RET
LOADS
        LXI     H,DSTAT         ;drive status reg
        RET

        endif
        end
```