

```
*****
*
* CP/M vers 2.2 Cold Start Loader.
*
* Written by Bobby Dale Gifford.
* 3/17/80
*
```

```
* 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 0) 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 80H.
*
```

```
* 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 0	sector	1	address
0	8		2c00h
0	10		2d00h
0	12		2e00h
0	14		2f00h
0	16		3000h
0	18		2780h
0	20		2880h
0	22		2980h
0	24		2a80h
0	26		2b80h
0	9		2c80h
0	11		2d80h
0	13		2e80h
0	15		2f80h
0	17		2700h
0	19		2800h
0	21		2900h
0	23		2a00h
0	25		2b00h

```
* The following depend on the sector size, all sectors are from
* track 1.
*
```

256		512		1024	
sec	address	sec	address	sec	address
1	2c00h	1	2c00h	1	2c00h
3	2e00h	3	3000h	3	3400h
5	3000h	5	3400h	5	3c00h
7	3200h	7	3800h	** 7	4400h
9	3400h	9	3c00h	-----	
11	3600h	11	4000h	2	3000h
13	3800h	13	4400h	4	3800h
15	3a00h	-----		6	4000h
17	3c00h	2	2e00h		
19	3e00h	4	3200h		
21	4000h	6	3600h		

*ABOOT & .ASM
 Provided with M26
 1/13/82*

MAC ABOOT & \$+S

1/16/82 changes shown

```

* 23 4200h      8 3a00h      *
* 25 4400h     10 3e00h     *
* -----     12 4200h     *
* 2 2d00h      *
* 4 2f00h      *
* 6 3100h      Discus M10, M20, M26 *
* 8 3300h      3 e400h (If 2D is present) *
* 10 3500h     4 e600h (If 2D is present) *
* 12 3700h    ** 20 4500h *
* 14 3900h     5 2700h *
* 16 3b00h     6 2900h *
* 18 3d00h     7 2b00h *
* 20 3f00h     8 2d00h *
* 22 4100h     9 2f00h *
* 24 4300h    10 3100h *
* 26 4500h    11 3300h *
*          12 3500h *
*          13 3700h *
*          14 3900h *
*          15 3b00h *
*          16 3d00h *
*          17 3f00h *
*          18 4100h *
*          19 4300h *
*

```

```

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

```

```

msize equ 2856 ;Memory size of target CP/M
bias equ (msize-20)*1024 ;Memory offset from 20k system
ccp equ 2700h+bias ;Console command processor
bios equ ccp+1600h ;CBIOS address
retries equ 10 ;Maximum # of disk retries

```

← JJD 1/16/82

```

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

```

```

first equ 10 ;0 = Floppies are A-D drives and
; 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 2 ;Set to number of floppies

```

← JJD 1/16/82

```

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

```

← LEFT AT 4. LOOKS LIKE ALL IS TESTED
IS "ZERO" OR "NOT EQUAL TO ZERO"

```

if (maxhd ne 0) and first ;Want Hard Disk included ?
hdorg equ 50h ;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
hdcmd 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 ;Operation done bit of status

```

```

complt equ 4 ;Complete bit of status
tmout equ 8 ;Time out bit of status
wfault equ 10h ;Write fault bit of status
drvrdy equ 20h ;Drive ready bit of status
indx equ 40h ;Index bit of status
pstep equ 4 ;Step bit of function
nstep equ 0fbh ;Step bit mask of function
hdrln equ 4 ;Sector header length
secln equ 512 ;Sector data length
wenabl equ 0fh ;Write enable
wreset equ 0bh ;Write reset of function
scenbl equ 5 ;Controller control
dskclk equ 7 ;Disk clock for control
mdir equ 0f7h ;Direction mask for function
null equ 0fch ;Null command
idbuff equ 0 ;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 e700h in a standard
* configuration. It is responsible for reading most of track 0
* into memory on cold boots.
*
*****

```

```

if maxflop ne 0
origin equ 0E000H
djram equ origin+400h
djboot equ djram
endif

```

```

(2900) if (maxflop ne 0)
offsetb equ 900h-origin
else
offsetb equ 0
endif

```

```

if not first
putden equ origin+02dh ;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+0fh ;Disk Jockey 2D set sector routine
puttrk equ origin+0ch ;Disk Jockey 2D set track routine
puthom equ origin+9h ;Disk Jockey 2D track 0 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+700h-$ ;Offset to boot loader address
```

```

lxi sp,stack+diff
firmlod mvi a,6 ;Previous sector #
newsec equ $-1
inr a ;Update sector #
inr a
cpi 27 ;Test if all done
jz ccp+500h
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
newdma  lxi      h,ccp+400h ;Previous DMA address
equ     $-2
lxi     d,100h         ;Update DMA address
dad     d
mov     a,h
cpi     (ccp+980h)/100h
jc      nowrp+diff
jnz     wrp+diff
mov     a,1
cpi     (ccp+980h) mod 100h
jc      nowrp+diff
wrp     lxi     d,-980h
dad     d
nowrp   shld    newdma+diff ;Save the updated DMA address
mov     b,h
mov     c,1
call    putdma         ;Set up the new DMA address
lxi     b,retries*100h+0;Maximum # of errors
fread   push    b
call    puttrk         ;Set up the proper track
call    doread         ;Read the sector
pop     b
jnc     firmlod+diff   ;Continue if no error
dcr     b
jnz     fread+diff    ;Keep trying if error
jmp     boterr         ;To many errors, flash the light
ds      80h-($ mod 30h)
stac   equ     $

```

```

*****
*
* The following equates relate to the Thinker Toys 2D controller*
* If the controller is non standard (0E000H) only the ORIGIN *
* equate need be changed. This version of the Cbios will work *
* with 2D controller boards rev 0, 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+0ch     ;Disk Jockey 2D track seek routine
djsec   equ     djram+0fh     ;Disk Jockey 2D set sector routine
djdma   equ     djram+012h    ;Disk Jockey 2D set DMA address
djread  equ     djram+15h     ;Disk Jockey 2D read routine
djwrite equ     djram+18h     ;Disk Jockey 2D write routine
djssel  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+30h     ;Disk Jockey 2D set side routine

```

```

*****
*
* The following three sectors of code reside at 80H. There is *
* one sector for each of the possible sector sizes (256,512, *
* 1024). Each sector is responsible for performing a Cold Boot *
* for the specified sector size. *
*
*****

```

```

diff   set     80h-$

```

```

    lxi    sp,cstk256+diff ;Set up stack at end of this sector
    lxi    b,26*100h+1    ;B = sector count, C = sector #
clod256  push  b          ;Save sector and count
        call  djsec       ;Set the next sector to read
    lxi    h,ccp+300h     ;Get DMA address (self modifying)
cdma256  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   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+400h   ;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*100h+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    80h-($ mod 80h)
cstk256 equ   $

```

```

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

```

```

diff    set    80h-$

    lxi    sp,cstk512+diff ;Set up stack at end of this sector
    lxi    b,13*100h+1    ;B = sector count, C = sector #
clod512  push  b          ;Save sector and count
        call  djsec       ;Set the next sector to read
    lxi    h,ccp+100h     ;Get DMA address (self modifying)
cdma512  equ   $-2        ;Storage for previous DMA address
    lxi    d,400h        ;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+300h   ;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*100h+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    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      30h-($ mod 30h)
cstk512 equ  $

```

```

*****
*
* The next sector loads CP/M from a 1024 byte sector diskette.
*
*****

```

```

diff    set     30h-$

        lxi    sp,cstk124+diff ;Set up stack at end of this sector
        lxi    b,1*100h+7     ;B = sector count, C = sector #
        call   clod124+diff   ;Load sector 5 into CCP
        lxi    h,ccp+1d00h    ;Destination of move
        lxi    d,ccp+500h     ;Source of move
        lxi    b,200h

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-300h      ;Initial DMA address
        shld  cdma124+diff
        lxi   b,6*100h+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-300h ;Get DMA address (self modifying)
cdmal24 equ $-2 ;Storage for previous DMA address
        lxi d,800h ;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+100h ;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*100h+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 30h-($ mod 30h)
cstkl24 equ $

```

```

*****
*
* The next three sectors of code also reside at 80H. There is
* one sector for each of the possible sector sizes (256,512,
* 1024). 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	1024	
* sec	address	sec	address
* 1	3100h	1	3100h
* 3	3300h	3	3500h
* 5	3500h	5	3900h
* 7	3700h	7	3d00h
		** 5	4100h

```

* 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    80h-$

        lxi    sp,wstk512+diff ;Set up stack at end of this sector
        lxi    b,1*100h+9      ;B = sector count, C = sector #
        call   wld512+diff     ;Load sector 9 into CCP
        lxi    h,ccp+1500h     ;Destination of move
        lxi    d,ccp+500h      ;Source of move
        mvi    c,0

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+300h       ;Initial DMA address
        shld  wdma512+diff
        lxi   b,8*100h+2       ;B = sector count, C = sector #
        call  wld512+diff
        jmp   bios+3

wld512  push  b                ;Save sector and count
        call  djsec            ;Set the next sector to read
        lxi   h,ccp+100h       ;Get DMA address (self modifying)
wdma512 equ   $-2              ;Storage for previous DMA address
        lxi   d,400h           ;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,1
        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    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   10               ;Past the end of the track ?
        jc    wld512+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+100h       ;Negative DMA adjustment
        shld  wdma512+diff     ;Save the new DMA address
        jmp   wld512+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*100h+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 80h-($ mod 80h)
wstk512 equ $
```

```
*****
*
* The next sector loads CP/M from a 1024 byte sector diskette.
*
*****
```

```
diff set 80h-$

    lxi sp,wstk124+diff ;Set up stack at end of this sector
    lxi b,1*100h+5 ;B = sector count, C = sector #
    call wlod124+diff ;Load sector 6 into CCP
    lxi h,ccp+1500h ;Destination of move
    lxi d,ccp+500h ;Source of move
    mvi c,0
movl24 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 movl24+diff
    lxi h,ccp+100h ;Initial DMA address
    shld wdma124+diff
    lxi b,4*100h+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-300h ;Get DMA address (self modifying)
wdma124 equ $-2 ;Storage for previous DMA address
    lxi d,800h ;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-300h ;Negative DMA adjustment
    shld wdma124+diff ;Save the new DMA address
    jmp wlod124+diff ;Continue reading
```

```
*****
*
* Wrl24 does the actual read from the controller, the DMA
* address and sector # have already been set up.
*
*****
```

```
wrd124 lxi b,retries*100h+1 ;Maximum # of attempts
wrl24 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 80h-($ mod 80h)
wstk124 equ $

```

```

*****
*
* The next sector of code resides at CCP+5000h. It's task is to
* move the firmware code into the Disk Jockey Ram, then
* loading a sector into 80H which will load the rest of CP/M.
* The sector loaded at 80H is dependent on the sector size
* of the diskette being booted from.
*
*****

```

```

diff set ccp+500h-$ ;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,300h ;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 0ch ;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 0
sta trknum+diff
lxi b,80h ;Cold Boot loads at 80H
push b ;Used as jump address to Cold Boot--

```

```

call    djdma                ;
mvi     c,0                  ;Density on track 0 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*100h+0 ;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   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

```

```

stk     ds     80h-($ mod 80h)
        equ   $
        else

```

```

*****
*
* Cold Boot loader for Discus M10, M20, M26.
*
*****

```

```

        if     maxflop ne 0
        org   origin          ;Org program at Floppy origin
        else
        org   100h            ; or else at 100h
        endif
diff    set    100h-$

boothd  lxi     sp,cstkhd+diff  ;Set up stack at end of this sector
        lxi   b,1*100h+20      ;B = sector count, C = sector #
        call  clodhd+diff       ;Load sector 20 into CCP
        lxi   h,ccp+1e00h      ;Destination of move
        lxi   d,ccp            ;Source of move
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        ;Initial DMA address
        lxi  h,ccp-200h         ;Initial DMA address
        shld cdmahd+diff
        lxi  b,15*100h+5       ;B = sector count, C = sector #
        call clodhd+diff

```

```

        if     maxflop ne 0
        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-200h ;Load in the Disk Jockey 2D ramware
shld    cdmahd+diff
lxi      b,2*100h+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-200h ;Get DMA address (self modifying)
cdmahd  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    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*100h+1 ;Maximum # of attempts
crhd    push    b      ;Save error count
call    hhread+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

```

```

hread   call    hhread+diff ;Prepare the sector header image
rc      ;Error exit
mvi     a,rsect ;Read sector command
out     hrcmd
call    process+diff ;Process the read
rc      ;Error exit
xra     a      ;Pointer to data buffer
out     hrcmd
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      hdcmdnd
mvi      a,null
out      hdfunc          ;Select drive A
xra      a
out      hddata          ;Form head byte
out      hddata          ;Form track byte
mvi      a,0             ;Form sector byte
hdsec    equ      $-1
out      hddata
mvi      a,80h           ;Form Key
out      hddata
mvi      a,dskclk        ;Turn on Disk clock
out      hdcntl
mvi      a,wenabl        ;Write enable on
out      hdcntl
ret

org      boothd+200h-2

cstkhd   equ      $
dw       boothd+diff
ds       200h
endif

```

```

*****
*
* Disk Jockey 2D firmware revision 3.1 and 4.0
* By George Morrow
*
* The following firmware is loaded into memory and then moved
* into the controller ram.
*
*****

```

```

rom      if      maxflop ne 0
ram      equ      origin
          equ      origin+400h

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	200Q
WCMD	EQU	240Q
HEAD	EQU	4
LOAD	EQU	20Q
DENSTY	EQU	1
ULOAD	EQU	30Q
RSTBIT	EQU	4
ACCESS	EQU	2
READY	EQU	40Q
INDEX	EQU	20Q
RACMD	EQU	304Q
CLRCMD	EQU	320Q
SVCMD	EQU	35Q
SKCMD	EQU	30Q
HCMD	EQU	11Q
ISTAT	EQU	4
OSTAT	EQU	10Q
DSIDE	EQU	10Q
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+1eh
TSTAT	JMP	origin+21h
DMAST	JMP	DMSTAT
STATUS	JMP	DISKST
DSKERR	JMP	ROM+52Q
SETDEN	JMP	DENFIX
SETSID	JMP	SIDEX
*		
*		
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      60h
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      0
btimer  dw      1800h,0
        db      3,0
bdrvsel db      7eh
bdisk   db      0,8,0,9,0ffh,9,0ffh,9,0ffh,9,0ffh,9,0,1,0

fixio   LHL D     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,0          ;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     ;  0 & 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,20Q        ;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,200
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
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,0          ;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
incr 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 1370 ;error bit mask
jz leave-1 ;test
cpi 10h ;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,10h      ;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,0        ;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 0
          pop      psw      ;recover errors
          ret

RDHDR    MVI      B,120   ;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,0      ; 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,200    ;error flag
          RC       ;error return
          MOV     A,B      ;save the sector
          STA     SECREG   ; in sector reg
          mvi    a,40q     ;128 byte sector
          lxi    h,505h    ;initialize
          shld   ecoun    ; 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

TOEND   *
          *
          DS     300h-TOEND+DBOOT-660

```

DS 25Q

*
*
STACK
*
*

STABLE

DB 345Q
DB 345Q
DB 360Q
DB 367Q

*
*

ecount dw 0 ;error count cells
TIMER DW 3000h ;head load time
DMAADR DW 200Q ;dma address
HDFLAG DB 0 ;read header flag
DRVSEL DB 376Q ;drive select constant
DISK DB 0 ;new drive
CDISK DB 10Q ;current drive
TZFLAG DB 0 ;track zero indicator
D0PRAM DB 3 ;drive 0 parameters
D0TRK DB 377Q ;drive 0 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 0 ;new side select
SECTOR DB 3 ;new sector
TRACK DB 0 ;new track
TRKNO DB 0 ;disk
SIDENO DB 0 ; sector
SECTNO DB 0 ; header
SECLN DB 0 ; data
CRCLO DB 0 ; buffer
CRCHI DB 0

*
*

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,0 ;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
DSROT	MVI	A,1770	;disk select bits
	RLC		;rotate to
	DCR	C	; select the
	JP	DSROT	; proper drive
	STA	DRVSEL	;save
HDCHK	XRA	A	;force head load
	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 = 0
	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	; 40 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,2000	;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

```

NBUSY  MOV      M,A          ;to the 1791
      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,0          ;initialize count
      CALL   LOADS        ;status port
      MVI    C,INDEX      ;index bit flag

INDXHI MOV     A,M          ;wait for
      ANA    C           ;   index
      JNZ   INDXHI        ;   pulse low

indx1  JNZ   INDXHI
INDXLO

indx2  MOV     A,M          ;wait for
      ANA    C           ;   index
      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
      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,0         ;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

```