```
D>PIP LST:=CBIOS&.ASM[T8]
*************************************************************************
*                                                                       *
* Morrow Designs CBIOS for CP/M Version 2.2.                            *
*                                                                       *
* This CBIOS can be configured to run with the following devices.      *
* The disks may be configured to run with any or all of the disk        *
* systems.  The logical order of the disks can be set to any order.    *
*                                                                       *
* Disk systems:                                                         *
*       HDCA 1Ø, 2Ø and 26 megabyte hard disks.                         *
*       HDDMA 5, 1Ø, 16, megabyte hard disk systems.                    *
*       DJDMA floppy disk controller with 8 and 5 1/4 inch disks.       *
*       DJ 2D/B floppy disk controller with 8 inch disks.               *
*                                                                       *
* Console I/O:                                                          *
*       Disk Jockey 2D/B serial.                                        *
*       Disk Jockey DMA serial.                                         *
*       Multi I/O serial.                                               *
*       Decision I serial.                                              *
*                                                                       *
* Printer I/O:                                                          *
*       Multi I/O serial with handshaking.                              *
*       Multi I/O Diablo 162Ø simulator for the Hytype II.              *
*                                                                       *
* Note: Floppy systems diskette (drive A:) has to have 1Ø24 byte        *
*       sectors in order for the cold and warm boot loaders to          *
*       work.  Be sure to format all new system diskettes with          *
*       1Ø24 byte sectors.  The system diskette can be either           *
*       single or double sided.  The sector size on normal (non         *
*       A: drive) diskettes is not restricted.  Thus if you have        *
*       a diskette with software that is supposed to run on the         *
*       A: drive then you should mount the diskette in the B:           *
*       drive and then PIP it over to a 1Ø24 byte sector                *
*       system diskette.                                                *
*                                                                       *
* Written by Les Kent and Marc Kupper           3/4/82                  *
*                                                                       *
*  Date    Programmer   Description                                     *
*                                                                       *
**11 20 82 Marc         Public release of revision E.31                 *
* 11 19 82 Marc         Changed HDC3 equate to HDCA                     *
* 11 19 82 Marc         Changed blank IO routines from RET to JMP $     *
* 11 19 82 Marc         Converted BIOSLN to a byte value                *
* 11  9 82 Marc         Reduced bad map size to 1 for non MW systems    *
* 11  8 82 Marc         Deleted baud rate test from Multio drivers      *
* 11  4 82 Marc         Added initial IOBYTE to IOCONF                  *
* 11  3 82 Marc         Added the North Star character I/O system       *
* 11  2 82 Marc         Added character redirection code for the IOBYTE *
* 11  1 82 Marc         Changed serial i/o entry names to IOBYTE names  *
* 1Ø 18 82 Marc         Fixed SETHIGH for 2 sided DJDMA 8 inch disks   *
* 1Ø 18 82 Marc         Deleted the HyType drivers                      *
**1Ø  1 82 Marc         Public release of revision E.3                  *
*  9 29 82 Marc         4ØH now points to the HDDMA command channel     *
*  9 28 82 Marc         MW's now have 1Ø24 directory entries            *
*  9 28 82 Marc         Deleted the Centronics drivers                  *
*  9 27 82 Marc         Changed login message to look like a label      *
*  9 27 82 Marc         Changed the login messages to say M5, M1Ø, ...  *
*  9 27 82 Marc         Redefined the dparam table structure            *
*  9 22 82 Marc         Added a serial console for the Switchboard      *
*  9 22 82 Marc         Added initialization code for serial group 2    *
*  9 22 82 Marc         Added sector size byte to the HDCA DPB's        *
*  9 22 82 Marc         Added sector size parameter to DPBGEN           *
*  9  9 82 Marc         Fixed system length checks for 64K systems      *
```

1/16/83

ORIG M20/DJDMA
CBIOS

NEW DECISION

64K
Bios offset = 3700
Boot offset = 0B00

NOTE: changed ABOOT&.ASM AS FOLLOWS: MSIZE EQU 64. THAT'S ONLY CHG

PROC: DDT CPM64.COM
IABOOT&.HEX
R0B00
ICBIOS&N.HEX
R3700
↑C
SYSGEN (DEST = HD)

CBIOS&5.ASM: 6/10/84 ADDED SPOOL-Z-Q
LST:=PRN: SPOOL Z Q

```
 *  9  9 82 Marc         SETHIGH was botching 2 sided DPB pointers      *
 *  8 31 82 Marc         Changed TRACKS in HD driver to HDTRAK          *
 *  8 27 82 Marc         Added code/system length checker               *
 *  8 27 82 Marc         mwreset save/restores the track number         *
 *  8 26 82 Marc         mwreset now sets *step and *dir for CMI        *
 *  8 20 82 Marc         Added 'equ'ed handshaking to the serial LST:   *
 *  8 19 82 Marc         Removed clock switching code from HDCA driver  *
 *  8 18 82 Marc         Added handshake configuration code             *
 *  8 18 82 Marc         Added handshake configuration bytes            *
 *  8 18 82 Marc         Removed 'equ'ed handshaking from LST:          *
 *  8 12 82 Marc         Added configuration entries for a0 & d0        *
 *  8 11 82 Marc         Added the autostart command structure          *
 *  8 11 82 Marc         Redefined the configuration table              *
 *  8 11 82 Marc         Added DJDMA drive parameter table              *
 *  8  9 82 Marc         Added clock switching to HDCA code             *
 *  8  9 82 Marc         Added seek complete clearing in HDCA           *
 *  8  6 82 Marc         Added buffer disable on home                   *
 *  8  6 82 Marc         Fixed 8250 UART initialization sequence        *
 *  8  6 82 Marc         Strip parity on conout to clear up glitches    *
 *  8  6 82 Marc         Fixed the 8 inch dpb256ss DPB's EXM            *
 *  8  6 82 Marc         Increased the HD capacities slightly           *
 *  8  6 82 Marc         Deleted all non-supported MW drives            *
 *  8  6 82 Marc         Deleted call to flush in conout                *
 *  8  6 82 Marc         Moved printer back to port 3                   *
 *  7 28 82 Marc         Moved conin flush call to conout               *
 *  7 27 82 Marc         Fixed double sided head settle time            *
 *  7 14 82 Marc         Optimized MWissue                              *
 *  7 14 82 Marc         Clean up login message for HD a bit            *
 *  6 30 82 Marc         Fixed MF multi density problems                *
 *  6 29 82 Marc         Added Olivetti HD561/1 HD561/2 drives          *
 *  6 28 82 Marc         Added a MW error reporter                      *
 *  6 18 82 Marc         Added nonstandard system mode flag             *
 *  6 17 82 Marc         Added a buffer error flag                      *
 *  6 17 82 Marc         Added save/restore of 50-52 to MW driver       *
 *  6 17 82 Marc         Fixed Centronics drivers                       *
 *  6  7 82 Marc         Fixed allocation map sizes                     *
 *  6  7 82 Marc         Fixed MW partitioning                          *
 *  6  7 82 Marc         Fixed HD partitioning (again)                  *
 *  5 13 82 Marc         Fixed illegal MAC labels                       *
 *  5 11 82 Marc         Fixed North Star drive configurations          *
 *  4 30 82 Marc         Fixed Quantum Q2040 tracks to 512              *
 *  4 29 82 Marc         Fixed ST412 step constant to 0                 *
 *  4 26 82 Marc         Added unallocated writing                      *
 *  4 22 82 Marc         Fixed HD partition overlap                     *
 *  4 20 82 Marc         Started testing and debugging of E.3           *
 *  4 19 82 Marc         Added 1 sector to HD warm boot loader          *
 *  4 19 82 Marc         Added mod. number to CBIOS rev. number         *
 *  4 19 82 Marc         Clean up login message 'if's                   *
 *  4 15 82 Marc         Fixed MCR Initialization for LST:              *
 *  4 15 82 Marc         Added Seagate ST412 drive                      *
 *  4  6 82 Marc         Moved serial LST: device to port 2             *
 *  4  1 82 Marc         Added common group select routines             *
 *  4  1 82 Marc         Fixed Diablo HyType II initialization          *
 *  4  1 82 Marc         Fixed LISTST for PROM driver                   *
 *  3 16 82 Marc         Added Tandon TM602 and TM603 drives            *
 *  3 16 82 Marc         Use 'part number' equates for MW drives        *
 *  3 15 82 Marc         Dropped hdrev and mwrev equates                *
 *  3 15 82 Marc         Seagate ST506 head settle is 0 ms.             *
 *  3 15 82 Marc         Added MiniScribe 1006 and 1012 drives          *
 * *3  1 82 Marc         Public release of revision E.2                 *
 *  2 -- 82 Marc         Pre-release testing and debugging              *
 *  2  1 82 Les + Marc   Initial coding of revision E                   *
 *                                                                      *
************************************************************************

        title   'CBIOS Revision E for CP/M Version 2.2 - March 4, 1982'
```

```
revnum  equ     53                  ;CBIOS revision number 5.x = E
cpmrev  equ     22                  ;CP/M revision number 2.2

************************************************************************
*                                                                      *
* The following flags set a 'non-standard' system mode and an          *
* assembly time debugger.                                              *
*                                                                      *
* If this CBIOS is used with the CP/M 2.2 system that is shipped on    *
* a Morrow Designs diskette then NOSTAND can be set to 1.  This        *
* will allow the CBIOS to use various data areas found inside of       *
* the CP/M 2.2 BDOS.  If the CBIOS is used with a different            *
* operating system then NOSTAND should be set to Ø.                    *
*                                                                      *
* The DEBUG flag merely causes various internal values and             *
* addresses to be printed during the assembly process.  This           *
* printing is forced via assembly errors and thus should not           *
* affect the resulting code in any way.                                *
*                                                                      *
************************************************************************

nostand equ     1                   ;Set to 1 for non-standard mode
debug   equ     Ø                   ;Set to 1 for debugging mode

************************************************************************
*                                                                      *
* The following is set to the memory size of the CP/M the CBIOS is     *
* being created for.                                                   *
*                                                                      *
************************************************************************

msize   equ     ~~48~~64            ;Memory size of target CP/M

biosln  equ     16h                 ;BIOS length.  Also in ABOOT&.ASM

************************************************************************
*                                                                      *
* The following equates set up the disk systems to be included         *
* along with the types of drives and the logical order of the          *
* drives.                                                              *
*                                                                      *
************************************************************************

maxhd   equ     1                   ;Set to number of HDCA hard disk drives
maxmw   equ     Ø                   ;Set to number of HDDMA hard disks
maxfd   equ     Ø                   ;Set to number of 2D/B floppies
maxdm   equ     2  1                ;Set to number of DJ DMA floppies 8 inch
maxmf   equ     2  0                ;Set to number of DJ DMA floppies 5 1/4 inch

hdorder equ     1                   ;Set the order of logical drives ELSE Ø if
mworder equ     Ø                   ; not included.
fdorder equ     Ø
dmorder equ     2
mforder equ     3 0

                                    ;HDCA controller disk drives. Set only one
ml0f    equ     Ø                   ;Fujitsu M23Ø1B
m2Ø     equ     1                   ;Fujitsu M23Ø2B
m26     equ     Ø                   ;Shugart SA4ØØØ
mlØm    equ     Ø                   ;Memorex

                                    ;HDDMA controller disk drives. Set only one
mwquiet equ     Ø                   ;Set for no names printed on login
st5Ø6   equ     Ø                   ;Seagate ST-5Ø6
st412   equ     Ø                   ;Seagate ST-412
cm5619  equ     Ø                   ;CMI CM-5619
```

```
wmdrive equ     Ø                 ;Device to warm boot from.  This is the
                                  ; CP/M logical drive number.

        if      maxmw ne Ø        ;Only HDDMA drives use the bad map
badsiz  equ     32                ;Number of badmap entries
        else
badsiz  equ     1                 ;Leave one entry as filler
        endif

*************************************************************************
*                                                                       *
* Since most hard disk drives hold more than 8 megabytes we             *
* partition the drive.  We partition our drives using two different     *
* formulas.                                                             *
*                                                                       *
* One is the so called 'standard partitioning' where we try to          *
* create as many 8 megabyte partitions as possible plus a small         *
* partition to take up the slack on the end of the drive.               *
*                                                                       *
* Another way the drives are partitioned is the so called 'even         *
* partition' formula.  This means that the drive is split into          *
* equale sized partitions with the only restriction being that no       *
* partition be over 8 megabytes in length.                              *
*                                                                       *
* All hard disk drives shipped from Morrow Designs are partitioned      *
* using the standard partition formula.  If the user wishes to          *
* implement even partitioning then he/she must set HDPART or MWPART     *
* to the number of partitions desired.                                  *
*                                                                       *
*************************************************************************

hdpart  equ     Ø                 ;Set to number of non standard partitions
mwpart  equ     Ø                 ;Set to number of non standard partitions

*************************************************************************
*                                                                       *
* The following equates define the console and printer environments.    *
*                                                                       *
*************************************************************************

*************************************************************************
*                                                                       *
* Define the console driver to be used.                                 *
*                                                                       *
* CONTYP is:    Ø       Nothing, used for patching to PROM's.           *
*               1       Provide for 128 bytes of patch space.           *
*               2       Multi I/O or Decision I driver.                 *
*               3       2D/B driver.                                    *
*               4       DJDMA serial port                               *
*               5       Switchboard serial port                         *
*               6       North Star motherboard (2 serial + 1 parallel)  *
*                                                                       *
* Set CBAUD to the divisor latch value for the console.  For an         *
* explanation of the values look at the DEFCON table.                   *
*                                                                       *
*************************************************************************

contyp  equ     2
cbaud   equ     12 6 NOT IN FIRST TRY

*************************************************************************
*                                                                       *
* Define the printer driver to be used.                                 *
*                                                                       *
* LSTTYP is:    Ø       Nothing, used for patching to PROM's.           *
```

← need autoboot ~~TV 910~~ INIT

```
*               1       Provide for 128 bytes of patch space.              *
*               2       Multio serial, no protocol.                        *
*               3       Multio serial, Clear To Send protocol.             *
*               4       Multio serial, Data Set Ready protocol.            *
*               5       Multio serial, Xon/Xoff protocol.                  *
*                                                                          *
* Note: The Decision board is functionally identical to the Multi         *
*       I/O board for serial printer I/O.  Selections 2 to 5 will         *
*       work on the Wunderbuss i/o board.  To use drivers 6 or 7          *
*       the MULTR3 equate will have to be set.                            *
*                                                                          *
* Set pbaud to the divisor latch value for the printer.  For an           *
* explanation of the values see the deflst table.                         *
*                                                                          *
****************************************************************************

lsttyp  equ     3
lbaud   equ     96

****************************************************************************
*                                                                          *
* The next equate determines if you have a Multi I/O Rev 3 or a           *
* Decision I mother board for parallel i/o.  If are not using             *
* either of these boards then you need not worry about this equate.       *
* If you are using a Multi I/O rev. other than 3.x or 4.x then you        *
* should set MULTR3 to Ø.                                                  *
*                                                                          *
****************************************************************************

multr3  equ     Ø               ;Ø = Decision, 1 = Multi I/O rev. 3 or 4

congrp  equ     1               ;Cosole port (1 = p1, 2 = p2, 3 = p3)

lstgrp  equ     3               ;Printer port (1 = p1, 2 = p2, 3 = p3)

****************************************************************************
*                                                                          *
* The following equates are internal to the CBIOS.                        *
*                                                                          *
****************************************************************************

m1Ø     equ     m1Øf or m1Øm

        if      hdpart ne Ø             ;Use non standard partitions
hdlog   equ     hdpart
        else
hdlog   equ     m1Ø*2+m2Ø*3+m26*3       ;Logical disks per drive for HDCA
        endif

        if      mwpart ne Ø             ;Use non standard partitions
mwlog   equ     mwpart
        else
mwlog   set     st5Ø6+st412*2++cm5619*2 ;Logical disks per drive for HDDMA
        endif

hdca    equ     m26 or m2Ø or m1Ø               ;HDCA controller
fujitsu equ     m2Ø  or m1Øf
hdspt   equ     32*m26+21*m2Ø+21*m1Ø            ;Sectors per track

hdma    set     st5Ø6 or st412 or cm5619        ;HD DMA controller
mwspt   equ     9                               ;Sectors per track

maxlog  equ     (maxhd*hdlog)+(maxmw*mwlog)+maxfd+maxdm+maxmf

****************************************************************************
*                                                                          *
* CP/M system equates.                                                 *
*                                                                      *
************************************************************************

ccpln   equ     800h
bdosln  equ     0e00h

size    equ     (msize*1024)
ccp     equ     size-(biosln*100h+ccpln+bdosln)
bdos    equ     ccp+ccpln
bios    equ     ccp+ccpln+bdosln

offsetc equ     2100h-bios      ;Offset for sysgen

        if      debug
dbgtmp  set     offsetc         ;DDT offset     ! <debug>
dbgtmp  set     ccp             ;CCP address    ! <debug>
dbgtmp  set     bdos            ;BDOS address   ! <debug>
dbgtmp  set     bios            ;CBIOS address  ! <debug>
        endif

cdisk   equ     4               ;Address of last logged disk
buff    equ     80h             ;Default buffer address
tpa     equ     100h            ;Transient memory
iobyte  equ     3               ;IOBYTE location
wbot    equ     0               ;Warm boot jump address
entry   equ     5               ;BDOS entry jump address

        if      nostand ne 0
cblock  equ     bios-19h        ;Current actual block# * blkmsk
                                ;Used for unallocated writting
        endif

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

retries equ     10              ;Max retries on disk i/o before error
clear   equ     'Z'-64          ;Clear screen on an ADM 3

anul    equ     0               ;Null
aetx    equ     'C'-64          ;ETX character
aack    equ     'F'-64          ;ACK character
abel    equ     'G'-64          ;Bell
abs     equ     'H'-64          ;Back Space
aht     equ     'I'-64          ;Horizontal tab
alf     equ     'J'-64          ;Line feed
avt     equ     'K'-64          ;Vertical tab
aff     equ     'L'-64          ;Form Feed
acr     equ     'M'-64          ;Carriage return
xon     equ     'Q'-64          ;Xon character
xoff    equ     'S'-64          ;Xoff character
aesc    equ     1bh             ;Escape character
ars     equ     1eh             ;RS character
aus     equ     1fh             ;US character
asp     equ     ' '             ;Space
adel    equ     7fh             ;Delete

************************************************************************
*                                                                      *
* The following are the macros used in generating the DPH, DPB and     *
* allocation tables.                                                   *
*                                                                      *
************************************************************************
```

```
dpbgen  macro   nam,log,dspt,dbsh,dblm,dexm,ddsm,ddrm,dalØ,dall,dcks,doff,ssiz
dpb&nam&log     equ     $
        dw      dspt
        db      dbsh
        db      dblm
        db      dexm
        dw      ddsm
        dw      ddrm
        db      dalØ
        db      dall
        dw      dcks
        dw      doff
        db      ssiz
        endm

dphgen  macro   nam,log,dpbl,dpb2
dph&nam&log     equ     $
        dw      Ø
        dw      Ø,Ø,Ø
        dw      dirbuf
        dw      &dpbl&dpb2
        dw      csv&nam&log
        dw      alv&nam&log
        endm

alloc   macro   nam,log,al,cs
csv&nam&log:    ds      cs
alv&nam&log:    ds      al
        endm

**************************************************************************
*                                                                        *
* The following marco is used in generating the logical order of the    *
* CP/M drives.                                                           *
*                                                                        *
**************************************************************************

order   macro   num
        if      num eq hdorder
        dw      hddst
        endif

        if      num eq mworder
        dw      mwdst
        endif

        if      num eq fdorder
        dw      fddst
        endif

        if      num eq dmorder
        dw      dmdst
        endif

        if      num eq mforder
        dw      mfdst
        endif
        endm

**************************************************************************
*                                                                        *
* The folloing are offset numbers of Device Specification Tables.        *
*                                                                        *
**************************************************************************
```

```
d$wboot equ     0       ;Warm boot
d$stran equu    1       ;Sector translation
```

```
d$wboot equ     0       ;Warm boot
d$stran equ     1       ;Sector translation
d$sel1  equ     2       ;Drive select, Return DPH
d$sel2  equ     3       ;Drive select
d$home  equ     4       ;Home drive
d$strk  equ     5       ;Set track
d$ssec  equ     6       ;Set sector
d$sdma  equ     7       ;Set DMA address
d$read  equ     8       ;Read a physical sector
d$write equ     9       ;Write a physical sector
d$bad   equ     10      ;Return pointer to bad sector info

*************************************************************************
*                                                                       *
* The jump table below must remain in the same order, the routines      *
* may be changed, but the function executed must be the same.           *
*                                                                       *
*************************************************************************

        org     bios            ;Cbios starting address

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

        if      contyp ne 0
const:  jmp     conist          ;Console status routine
cin:    jmp     conin           ;Console input
cout:   jmp     costrp          ;Console output
        else
const:  jmp     $               ;Console status routine PROM pointer
cin:    jmp     $               ;Console input PROM pointer
cout:   jmp     $               ;Console output PROM pointer
        endif

        if      (lsttyp ne 0) or (contyp eq 6)
pout:   jmp     lstout          ;List device output
        else
pout:   jmp     cout            ;List device output
        endif

        if      contyp eq 6     ;North Star drivers have punch entry points
        jmp     punout          ;Punch device output
        else
        jmp     cout            ;Use console I/O
        endif

        if      contyp eq 6     ;North Star drivers have reader entry points
        jmp     rdrin           ;Reader device input
        else
        jmp     cin             ;Use console I/O
        endif

        jmp     home            ;Home drive
        jmp     setdrv          ;Select disk
        jmp     settrk          ;Set track
        jmp     setsec          ;Set sector
        jmp     setdma          ;Set DMA address
        jmp     read            ;Read the disk
        jmp     write           ;Write the disk

        if      lsttyp ne 0
        jmp     lstost          ;List device status
        else
        jmp     donop           ;List device status
        endif
        jmp     sectran         ;Sector translation

;
;       The following jumps are extended BIOS calls defined by Morrow Designs
;

        if      maxfd ne Ø
        jmp     fdsel           ;Hookup for SINGLE.COM program
        else
        jmp     donop
        endif

        jmp     Ø               ;End of the jump table

**************************************************************************
*                                                                        *
* Drive configuration table.                                             *
*                                                                        *
**************************************************************************

drconf: db      Ø               ;Revision Ø structure
        db      32              ;32 bytes long now

**************************************************************************
*                                                                        *
* The following is the table of pointers to the Device                   *
* Specification Tables.  The order of this table defines the             *
* logical order of the CP/M drives.                                      *
*                                                                        *
**************************************************************************

dsttab: equ     $

dn      set     1
        rept    16
        order   %dn
dn      set     dn+1
        endm

**************************************************************************
*                                                                        *
* I/O configuration table.                                               *
*                                                                        *
* At this CBIOS revision 11 bytes are defined for this table.            *
* Several extensive changes are planned for the table.  Future           *
* revision of the IOCONF table will have independant entries for         *
* three serial ports and will be used by several character drivers.      *
* Also the IOBYTE will be implemented for all the character              *
* drivers.  I might even write an external program to edit this         *
* table.                                                                 *
*                                                                        *
* The first two bytes show the I/O configuration that the CBIOS was      *
* assembled with.  These bytes are used by external software to          *
* determine the configuration options that are available.                *
*                                                                        *
* The next byte is the initial IOBYTE value.  This value is written      *
* to location 3 on cold boots.  See the CP/M 2 alternation guide         *
* for a description of the IOBYTE.                                       *
*                                                                        *
* The next byte is to make sure that the group select byte on the        *
* Mult I/O or Decsion I stays consistant throughout the Cbios.           *
* Only the group bits themselves (bits Ø and 1) should be changed        *
* as you output to the group port.  If you modify one of the other       *
* bits (such as driver-enable) then you should modify the same bit       *
* in this byte.  For example:                                            *
*                                                                        *
*                               ;Select console group                   *
*       lda     group           ;Get group byte                         *
*       ori     congrp          ;Select the console port                *
*       out     grpsel          ;Select the group                       *
*                                                                       *
*                               ;Modify a bit in the group byte         *
*       lda     group           ;Get group byte                         *
*       ori     bank            ;Set the bank bit                       *
*       sta     group           ;Save new group setting                 *
*       ori     group2          ;Select second serial port              *
*       out     grpsel          ;Select the desired group               *
*                                                                       *
* Note: You should not set the group bits themselves in the             *
*       group byte.                                                     *
*                                                                       *
*                                                                       *
* The following two words define the default baud rates for the        *
* console and the list devices.  These words are provided so that      *
* the user can easily modify them and that they will also be used      *
* in the future by Morrow Designs software.                             *
*                                                                       *
* The following is a list of possible baud rates and the decimal       *
* value needed for the defcon or deflst words.                          *
*                                                                       *
* Baud rate     defcon/deflst   Baud rate      defcon/deflst            *
*        50     2304                 2000      58                       *
*        75     1536                 2400      48                       *
*       110     1047                 3600      32                       *
*       134.5    857                 4800      24                       *
*       150      768                 7200      16                       *
*       300      384                 9600      12                       *
*       600      192                19200       6                       *
*      1200       96                38400       3                       *
*      1800       64                56000       2                       *
*                                                                       *
*                                                                       *
* The next two bytes are ued to configure the hardware handshaking     *
* protocall used by the serial list drivers with the Multio or          *
* Wunderbuss I/O boards.  The first of these two bytes is a mask.       *
* This mask is ANDed with the 8250's MODEM Status Register to strip     *
* out the desired handshake lines.  Next the result of the ANDing       *
* is XORed with the second of the two bytes.  This XORing allows        *
* the handshake lines to be inverted.  Common byte values are           *
* shown below.                                                          *
*                                                                       *
* cts   equ     10h             ;Clear To Send status mask              *
*                                                                       *
*       db      cts             ;Morrow Designs 'Clear To Send'         *
*       db      0                                                       *
*                                                                       *
*       db      cts             ;Inverted Clear To Send                 *
*       db      cts                                                     *
*                                                                       *
*       db      0               ;No handshaking                         *
*       db      0ffh                                                    *
*                                                                       *
*                                                                       *
* The last byte in the revision one structure is the last character    *
* that was recieved from the printer.  This byte is used to             *
* implement Xon/Xoff software handshaking.  This handshaking            *
* protocol should not bother printers that have not implemented         *
* Xon/Xoff protocol so this driver is enabled all the time.             *
*                                                                       *
*************************************************************************

ioconf: db      2               ;Revision 2 structure
```

```
        db      11              ;11 bytes long now
        db      contyp          ;Console device driver number
        db      lsttyp          ;List device drive number

iobyt   equ     $               ;The initial IOBYTE is kept here
        db      ØØ$ØØ$ØØ$ØØb    ;All devices go to CON:          ← DB  11$00$00$00b

group:  db      Ø               ;Group byte
defcon: dw      cbaud           ;Console baud rate divisor value
deflst: dw      lbaud           ;Printer baud rate divisor value

        if      (lsttyp ne 3) and (lsttyp ne 4) ;Xon/Xoff protocol
lstand: db      Ø               ;Serial list handshake mask
lstxor: db      Øffh            ;Serial list inversion flag
        endif

        if      lsttyp eq 3     ;Clear To Send protocol
lstand: db      cts             ;Serial list handshake mask
lstxor: db      Ø               ;Serial list inversion flag
        endif

        if      lsttyp eq 4     ;Data Set Ready protocol
lstand: db      dsr             ;Serial list handshake mask
lstxor: db      Ø               ;Serial list inversion flag
        endif

lastch: db      xon             ;Last character recieved from the printer

***********************************************************************
*                                                                     *
* The following table are drive parameters for drives connected to    *
* the DJDMA floppy disk controller.  There is one entry for each of   *
* the the eight drive that the controller can address.  The first     *
* four entries are for the 8 inch drives and the last four are for    *
* the 5 1/4 inch drives.  Users with fast stepping 8 inch drives      *
* (SA85Ø/1) or slow 5 1/4 inch drives (SA4ØØ) should adjust this      *
* table for optimal device performace.                                *
*                                                                     *
* Each table entry contains four fixed length fields.  The fields     *
* are defined as follows:                                             *
*                                                                     *
*       tracks  This byte contains the number of tracks on the        *
*               drive.  Most 8 inch drives have 77 tracks and         *
*               most 5 1/4 inch drives have 35 or 4Ø tracks.          *
*                                                                     *
*       config  This a a flag byte that indicates as to whether       *
*               or not this drive has been configured.  Set to        *
*               Ø to force reconfiguration.                           *
*                                                                     *
*       step    This word contains the stepping rate constant.        *
*               The DJDMA's delay routines tick 34.1 times per        *
*               millisecond.  Thus the step constant would be the     *
*               drive manufactors recomended stepping delay times     *
*               34.1.  Example.  Shugart SA 85Ø's step at 3           *
*               milliseond intervals.  The step constant would be     *
*               3 * 43.1 or 1Ø2.                                      *
*                                                                     *
*       rfu     The next two words are reserved for future use.       *
*               They must be zero.                                    *
*                                                                     *
*       settle  This word is similar to the previously defined        *
*               step word.  This specifies the head settle timing     *
*               after the heads have been stepped.  Example,          *
*               Shugart's SA 85Ø head settle time is 15               *
*               milliseconds.  The settle constant would be 15 *      *
*               34.1 or 512.                                          *
*                                                                       *
* An assembler macro (dconf) has been provided to assist in             *
* generating the dparam table.  This macros parameters are the          *
* number of tracks, the step rate in milliseconds, and the head         *
* settle time in milliseconds.  For example:                            *
*                                                                       *
*                               ;Shugart SA 850                         *
*       dconf   77, 3, 15       ;77 tracks, 3 ms step, 15 ms settle     *
*                                                                       *
*                               ;Shugart SA 400                         *
*       dconf   35, 40, 10      ;35 tracks, 40 ms step, 10 ms settle    *
*                                                                       *
* Note: Caution should be used when defining the drive parameters.      *
* Incorrect definations may damage the floppy disk drive.  Morrow       *
* Designs takes no responsibility for damage that occures through       *
* the misuse of this macro.                                             *
*                                                                       *
*************************************************************************

        if      (maxdm ne 0) or (maxmf ne 0)    ;DJDMA present?

dconf   macro   tracks, step, settle
        db      tracks                  ;Number of tracks
        db      0                       ;Reset the calibrated flag
        dw      step*341/10             ;Step time
        dw      0                       ;Reserved for future use, must be zero
        dw      0                       ;Reserved for future use, must be zero
        dw      settle*341/10           ;Head settle time
        endm

dmarap: db      0, 10*8                 ;Revision 0, length 80 bytes

dparam: equ     $                       ;Drive parameter table

*************************************************************************
*                                                                       *
* Define 8 inch drive parameters                                        *
* Use SA800 parameters: 77 tracks, 8 ms step, 8 ms settle               *
*                                                                       *
*************************************************************************

        dconf   77, 8, 8                ;Drive 0
        dconf   77, 8, 8                ;Drive 1
        dconf   77, 8, 8                ;Drive 2
        dconf   77, 8, 8                ;Drive 3

*************************************************************************
*                                                                       *
* Define 5 1/4 inch drive parameters                                    *
* Use Tandon parameters: 40 tracks, 5 ms step, 15 ms settle             *
*                                                                       *
*************************************************************************

        dconf   40, 5, 15               ;Drive 0
        dconf   40, 5, 15               ;Drive 1
        dconf   40, 5, 15               ;Drive 2
        dconf   40, 5, 15               ;Drive 3

        endif

*************************************************************************
*                                                                       *
* Console driver routines.                                              *
*                                                                       *
* Routine used depends on the value of CONTYP.  Possible CONTYP         *
* values are listed as follows:                                         *
*                                                                       *
* CONTYP is:    Ø       Nothing, used for patching to PROM's            *
*               1       Provide for 128 bytes of patch space            *
*               2       Multi I/O or Decision I driver                  *
*               3       2D/B driver                                     *
*               4       DJDMA serial port                               *
*               5       Switchboard serial port                         *
*               6       North Star motherboard (2 serial + 1 parallel)  *
*                                                                       *
*************************************************************************

*************************************************************************
*                                                                       *
* This routine is an experiment to reduce missed and garbled            *
* characters on console output.                                         *
*                                                                       *
*************************************************************************

        if      contyp ne Ø

costrp: mov     a,c             ;Strip parity on conout
        ani     7fh
        mov     c,a
        jmp     conout

        endif

*****************************************************************
*                                                               *
* The folowing equates will define the Decision I mother        *
* board I/O or the Multi I/O environments if needed.            *
*                                                               *
*****************************************************************

multio  equ     (contyp eq 2) or (lsttyp ge 2)  ;Multi I/O board used?

        if      multio          ;Define Multi I/O environment
mbase   equ     48h             ;Base address of Multi I/O or Decision I
grpsel  equ     mbase+7         ;Group select port
dll     equ     mbase           ;Divisor (lsb)
dlm     equ     mbase+1         ;Divisor (msb)
ier     equ     mbase+1         ;Interupt enable register
clk     equ     mbase+2         ;WB14 printer select port
lcr     equ     mbase+3         ;Line control register
mcr     equ     mbase+4
lsr     equ     mbase+5         ;Line status register
msr     equ     mbase+6
rbr     equ     mbase           ;Read data buffer
thr     equ     mbase           ;Tranmitter data buffer
dlab    equ     8Øh             ;Divisor latch access bit
thre    equ     2Øh             ;Status line THRE bit
cts     equ     1Øh             ;Clear to send
dsr     equ     2Øh             ;Data set ready
dr      equ     1               ;Line status DR bit
wlsØ    equ     1               ;Word length select bit Ø
wls1    equ     2               ;Word length select bit 1 for 8 bit word
stb     equ     4               ;Stop bit count - 2 stop bits

; Define multi I/O ports addresses for group zero

gzero   equ     Ø
daisyØ  equ     mbase           ;Daisy input ports
daisy1  equ     mbase+1
sensesw equ     mbase+1         ;Sense switches

        if      multr3 eq Ø     ;Daisy output ports are different
```

```
daisiØ  equ     mbase           ;   for Decision I and Multi I/O.
daisi1  equ     mbase+1         ;These two are the Decision I ports
        else
daisiØ  equ     mbase+1         ;   and these are the Multi I/O's.
daisi1  equ     mbase
        endif

; Define daisy Ø status input bits

ribbon  equ     Ø1h             ;End of ribbon
paper   equ     Ø2h             ;Paper out
cover   equ     Ø4h             ;Cover open
pfrdy   equ     Ø8h             ;Paper feed ready
crrdy   equ     1Øh             ;Carriage ready
pwrdy   equ     2Øh             ;Print wheel ready
check   equ     4Øh             ;Printer check (error)
ready   equ     8Øh             ;Printer ready

; Define daisy Ø status input bits for Diablo HyType II driver

crstrd  equ     1Ø2Øh           ;Carriage ready
pfstrd  equ     81Øh            ;Paper feed ready
pwstrd  equ     2Ø4Øh           ;Print wheel ready

; Define daisy Ø output bits

d9      equ     Ø1h             ;Data bit  9
d1Ø     equ     Ø2h             ;Data bit 1Ø
d11     equ     Ø4h             ;Data bit 11
d12     equ     Ø8h             ;Data bit 12

pfstb   equ     1Øh             ;Paper feed strobe
crstb   equ     2Øh             ;Carriage strobe
pwstb   equ     4Øh             ;Print wheel strobe
rest    equ     8Øh             ;Printer restore (Ribbon lift on Multi I/O)

; Define clock select bits

rlift   equ     4Øh             ;Ribbon lift
pselect equ     8Øh             ;Select (Not used by Diablo)

; Define Modem Control Register bits

dtrenb  equ     1               ;DTR enable
rtsenb  equ     2               ;RTS enable

; Define group select bits

sØ      equ     Ø1h             ;Group number (Ø-3)
s1      equ     Ø2h
smask   equ     Ø3h
bank    equ     Ø4h
enint   equ     Ø8h
restor  equ     1Øh             ;Printer restore on Multi I/O
denable equ     2Øh             ;Driver enable on Multi I/O

; Define special constants for the HyTyp II driver

cperi   equ     1Ø              ;Default to 1Ø characters per inch
lperi   equ     6               ;Default lines per inch
hinc    equ     12Ø             ;Horizontal increments per inch
vinc    equ     48              ;Vertical increments per inch
numtabs equ     16Ø             ;Number of horizontal tabs
maxchrs equ     1Ø24            ;Maximum number of printer characters to queue
maxrgt  equ     1584            ;Maximum carriage position
dfrmln  equ     11Ø             ;Forms length times 1Ø
```

```
autolf  equ     Ø               ;Default to noIAuto line feed

        endif

*****************************************************************
*                                                               *
* CONTYP: 2     Multi I/O or Decision I console driver          *
*                                                               *
*****************************************************************

        if      contyp eq 2

*************************************************************************
*                                                                       *
* This driver on cold boot will inspect bits 1-3 of the sense          *
* switches.  If the value found is in the range Ø-6 then the           *
* console baud rate will be taken from the rate table.  Otherwise      *
* the baud rate will be set from the DEFCON word which is found        *
* just below the regular Cbios jump table.  The standard divisor       *
* table is given below.                                                 *
*                                                                       *
* Sense switch: 123  (Ø = off, 1 = on)                                  *
*               ØØØ = 11Ø                                               *
*               ØØ1 = 3ØØ                                               *
*               Ø1Ø = 12ØØ                                              *
*               Ø11 = 24ØØ                                              *
*               1ØØ = 48ØØ                                              *
*               1Ø1 = 96ØØ                                              *
*               11Ø = 192ØØ                                             *
*            defcon = 96ØØ                                              *
*                                                                       *
* Note: If you are using a Multio then the switches will not be        *
*       available so the baud rate will be taken from DEFCON.          *
*                                                                       *
*************************************************************************

*************************************************************************
*                                                                       *
* Due to its length, the TTYSET routine driver is below the CBOOT      *
* CBOOT routine.                                                        *
*                                                                       *
*************************************************************************

*****************************************************************
*                                                               *
* Read a character from the serial port.                        *
*                                                               *
*****************************************************************

conin:  call    selcon          ;Select console

coninl: in      lsr             ;Read status register
        ani     dr              ;Wait till character ready
        jz      coninl
        in      rbr             ;Read character
        ani     7fh             ;Strip parity
        ret

*****************************************************************
*                                                               *
* Output a character to serial port.                            *
*                                                               *
*****************************************************************

conout: call    selcon          ;Select console
```

```
conoutl:in      lsr             ;Read status
        ani     thre            ;Wait till transmitter buffer empty
        jz      conoutl
        mov     a,c             ;Character is in (c)
        out     thr             ;Output to transmitter buffer
        ret

*****************************************************************
*                                                               *
* Return serial port status.  Returns zero if character is not  *
* ready to be read.  Else returns 255 if ready.                 *
*                                                               *
*****************************************************************

conist: call    selcon          ;Select console

        in      lsr             ;Read status register
        ani     dr
        rz                      ;No charactter ready
        mvi     a,Øffh          ;Character ready
        ret

        endif                   ;Multi I/O or Decision I

*****************************************************************
*                                                               *
* CONTYP: 3     2DB console driver                              *
*                                                               *
*****************************************************************

        if      contyp eq 3

conout: jmp     fdcout          ;Console output

conin:  jmp     fdcin           ;Console input

conist: call    fdtstat         ;Console status
        mvi     a,Øffh
        rz
        inr     a
        ret

        endif                   ;2DB

*****************************************************************
*                                                               *
* CONTYP: 4     DJDMA console driver                            *
*                                                               *
*****************************************************************

        if      contyp eq 4
conout: lxi     h,dmchan
        mvi     m,serout        ;Command for serial output
        inx     h
        mov     m,c
        jmp     docmd

conin:  lxi     h,serin+1       ;Serial input status
        xra     a
ci2:    cmp     m               ;Wait till 4Øh deposited at 3fH
        jz      ci2
        mov     m,a             ;Clear status
        dcx     h               ;Point to input data
        mvi     a,7fh           ;For masking out parity
        ana     m
        ret
```

```
conist: lda     serin+1         ;Pick up serial input status
        ora     a
        rz                      ;If zero then no character ready
        mvi     a,Øffh          ;Set character ready
        ret
        endif

*************************************************************************
*                                                                       *
* CONTYP: 5     Switchboard as serial console                           *
*                                                                       *
*************************************************************************

        if      contyp eq 5

swbase  equ     Ø                       ;Base of the SWITCHBOARD

conist: in      swbase+2                ;Get the first ports status
        ani     4                       ;Mask the data ready bits
        rz                              ;Return console not ready
        mvi     a,Øffh
ttyset: ret                             ;NULL terminal initialization

conin:  in      swbase+2                ;Get switchboard status
        ani     4                       ;Test for data ready
        jz      conin
        in      swbase                  ;Get a character
        ani     7fh                     ;Strip off parity
        ret

conout: in      swbase+2                ;Check status
        ani     8                       ;Wait till output buffer empty
        jz      conout
        mov     a,c                     ;Write a character
        out     swbase
        ret

        endif

*************************************************************************
*                                                                       *
* Multio/Wunderbuss group select routines                               *
*                                                                       *
*************************************************************************

        if (contyp eq 2) or (lsttyp ge 2)       ;Need group select routines?

selgØ:  lda     group           ;Select group zero
        out     grpsel
        ret

selcon: lda     group           ;Select console group
        ori     congrp
        out     grpsel
        ret

selrdr: lda     group           ;Select reader/punch group
        ori     5-lstgrp        ;Use 'other' serial port
        out     grpsel
        ret

sellst: lda     group           ;Select printer group
        ori     lstgrp
        out     grpsel
        ret
        endif

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

autost: db      Ø               ;Revision Ø structure
        db      1ØØh - (low $)  ;The rest of the page is used for this stuff

autoflg:db      Ø               ;Auto command feature enable flag

coldmes:dw      coldcm          ;Pointer to the cold start command
warmes: dw      warmcm          ;Pointer to the warm start command

*****************************************************************
*                                                               *
* If there is a command inserted here, it will be passed to the *
* CCP if the auto feature is enabled.  For Example:             *
*                                                               *
*       coldcm: db      coldend-coldcm                          *
*               db      'MBASIC MYPROG'                         *
*       coldend equ     $                                       *
*                                                               *
* will execute Microsoft BASIC, and MBASIC will execute the     *
* "MYPROG" BASIC program.  Note: The command line must be in    *
* upper case for most commands.                                 *
*                                                               *
*****************************************************************

coldcm: db      coldend-coldcm          ;Length of cold boot command
        db      ''                      ;Cold boot command goes here
coldend equ     $

warmcm: db      warmend-warmcm          ;Length of warm boot command
        db      ''                      ;Warm boot command goes here
warmend equ     $

*************************************************************************
*                                                                       *
* At the first page boundry following the CBIOS we have a series of     *
* pointers that point to various internal tables. At the start of       *
* each of these tables we have a revision byte and a length byte.       *
* The revision byte is the current revision number for that             *
* particular structure and the length byte is the length of that        *
* structure.  This length does not include the revision byte nor        *
* the length byte itself.                                               *
*                                                                       *
*       Revision        Description                                     *
*       E.Ø             1 and 2 defined                                 *
*       E.3             This table is moved to a page boundry           *
*       E.3             Ø, 3 and 4 defined                                      *
*                                                                       *
* The pointers defined so far are as follows:                           *
*                                                                       *
* Ø)    High byte is the page number of the CBIOS.  Low byte is         *
*       the CBIOS revision number.  Used to determine pointer           *
*       structure.                                                     *
*                                                                      *
* 1)    This points to the drive configuration table.                  *
*                                                                      *
* 2)    This points to the I/O configuration bytes for the serial      *
*       drivers.  Eg, the console, printer, reader, and punch          *
*       devices.                                                       *
*                                                                      *
* 3)    This points to the drive parameter table for DJDMA floppy      *
*       disk drives.  If no DJDMA is present then this pointer is      *
*       null (Ø).                                                      *
*                                                                      *
* 4)    This points to the autostart command structures.  Used to      *
*       automatically invoke a command on cold or warm boot            *
*                                                                      *
* 5)    This will be a null (Ø) pointer.  It marks the end of the      *
*       table.                                                         *
*                                                                      *
************************************************************************

        if      $ gt bios+256   ;Test for code overlap
        'Fatal error, pointer table placement.'
        else
        ds      bios+256-$      ;Start at a page boundry
        endif

        db      high ($-1)      ;CBIOS page number
        db      revnum          ;Cbios revision number
        dw      drconf          ;Drive configuration table pointer
        dw      ioconf          ;I/O configuration table pointer
        if      (maxdm ne Ø) or (maxmf ne Ø)    ;DJDMA present?
        dw      dmarap          ;Drive parameter table pointer
        else
        dw      Ø
        endif
        dw      autost          ;Auto command structure pointer
        dw      Ø               ;End of table marker

************************************************************************
*                                                                      *
* The following code performs the mapping of logical to physical       *
* serial I/O devices.  The physical entry points are CONIN, CONOUT,    *
* CONIST, RDRIN, PUNOUT, LSTOUT, and LSTOST.  These entry points       *
* are mapped via the Intel standard I/O byte (IOBYTE) at location 3    *
* in the base page to the low level device drivers.                    *
*                                                                      *
* Note:  A naming convention has been chosen to reduce label           *
* colisions.  The first three characters of a name indicate the        *
* device drivers name, the following three characters indicated the   *
* function performed by that particular device routine.  The device    *
* names are defined and described in the "An Introduction to CP/M      *
* Features and Facilities" manual in the section on the STAT           *
* command and in the "CP/M Interface Guide" in the IOBYTE section.     *
* The device function postfixes are as follows.                        *
*                                                                      *
* devSET        Initial device setup and initialzation                 *
* devIN Read one character from the device                             *
* devOUT        Write one character to the device                      *
* devIST        Return the device character input ready status         *
* devOST        Return the device character output ready status        *
*                                                                      *
* The setup routine initializes the device and returns.  The input     *
* routine returns one character in the A register (parity reset).      *
* The output routine write one character from the C register.  The     *
* input status routine returns in the A register a Ø if the device     *
* does not have a character ready for input for Øffh if a character    *
* is ready for input.  The output status routine returns in the A       *
* register a Ø if the device is not ready accept a character and a      *
* Øffh if the device is ready.  The input and output routines           *
* should wait untill the device is ready for the desired operation      *
* before the doing the operation and returning.                         *
*                                                                       *
* Not all of these functions need to be implemented for all the         *
* devices.  The following is a table of the entry points needed for     *
* each device handler.                                                  *
*                                                                       *
*       device  setup   input   output  input   output                  *
*       name                            status  status                  *
*                                                                       *
*       CON:            CONIN   CONOUT  CONIST                          *
*       RDR:            RDRIN           RDRIST                          *
*       PUN:                    PUNOUT                                  *
*       LST:                    LSTOUT          LSTOST                  *
*                                                                       *
*       TTY:    TTYSET  TTYIN   TTYOUT  TTYIST  TTYOST                  *
*       CRT:    CRTSET  CRTIN   CRTOUT  CRTIST  CRTOST                  *
*       UC1:    UC1SET  UC1IN   UC1OUT  UC1IST                          *
*                                                                       *
*       PTR:    PTRSET  PTRIN           PTRIST                          *
*       UR1:    UR1SET  UR1IN           UR1IST                          *
*       UR2:    UR2SET  UR2IN           UR2IST                          *
*                                                                       *
*       PTP:    PTPSET          PTPOUT                                  *
*       UP1:    UP1SET          UP1OUT                                  *
*       UP2:    UP2SET          UP2OUT                                  *
*                                                                       *
*       LPT:    LPTSET          LPTOUT          LPTOST                  *
*       UL1:    UL1SET          UL1OUT          UL1OST                  *
*                                                                       *
* The CONIN, CONOUT, CONIST, RDRIN, RDRIST, PUNOUT, LSTOUT, and         *
* LSTOST routines are the logical device driver entry points            *
* provided by this device mapper.  The other entry names must be        *
* provided by the physical device drivers.                              *
*                                                                       *
*************************************************************************

        if      contyp eq 6             ;I/O byte implemented for North Star
                                        ;   drivers.  Other drivers to follow

conin:  mvi     e,1                     ;Console input
        call    redir                   ;       IOBYTE: 7654321Ø
        dw      ttyin                   ;CON: = TTY:    xxxxxxØØ
        dw      crtin                   ;CON: = CRT:    xxxxxxØ1
        dw      rdrin                   ;CON: = BAT:    xxxxxx1Ø
        dw      uc1in                   ;CON: = UC1:    xxxxxx11

conout: mvi     e,1                     ;Console output
        call    redir                   ;       IOBYTE: 7654321Ø
        dw      ttyout                  ;CON: = TTY:    xxxxxxØØ
        dw      crtout                  ;CON: = CRT:    xxxxxxØ1
        dw      lstout                  ;CON: = BAT:    xxxxxx1Ø
        dw      uc1out                  ;CON: = UC1:    xxxxxx11

conist: mvi     e,1                     ;Console input status
        call    redir                   ;       IOBYTE: 7654321Ø
        dw      ttyist                  ;CON: = TTY:    xxxxxxØØ
        dw      crtist                  ;CON: = CRT:    xxxxxxØ1
        dw      rdrist                  ;CON: = BAT:    xxxxxx1Ø
        dw      uc1ist                  ;CON: = UC1:    xxxxxx11

rdrin:  mvi     e,7                     ;Reader input
        call    redir                   ;       IOBYTE: 7654321Ø
```

```
         dw      ttyin                    ;RDR: = TTY:     xxxxØØxx
         dw      ptrin                    ;RDR: = PTR:     xxxxØ1xx
         dw      urlin                    ;RDR: = UR1:     xxxx1Øxx
         dw      ur2in                    ;RDR: = UR2:     xxxx11xx

rdrist:  mvi     e,7                      ;Reader input status
         call    redir                    ;        IOBYTE: 7654321Ø
         dw      ttyist                   ;RDR: = TTY:     xxxxØØxx
         dw      ptrist                   ;RDR: = PTR:     xxxxØ1xx
         dw      urlist                   ;RDR: = UR1:     xxxx1Øxx
         dw      ur2ist                   ;RDR: = UR2:     xxxx11xx

punout:  mvi     e,5                      ;Punch output
         call    redir                    ;        IOBYTE: 7654321Ø
         dw      ttyout                   ;PUN: = TTY:     xxØØxxxx
         dw      ptpout                   ;PUN: = PTP:     xxØ1xxxx
         dw      uplout                   ;PUN: = UP1:     xx1Øxxxx
         ~~dw~~      up2out                   ;PUN: = UP2:     xx11xxxx
      endif
lstout:  ~~mvi~~     e,3                      ;List output
         call    redir                    ;        IOBYTE: 7654321Ø
         dw      ttyout OKOUT             ;LST: = TTY:     ØØxxxxxx
         dw      crtout OKOUT             ;LST: = CRT:     Ø1xxxxxx  OKIDATA
         dw      lptout ~~DIOUT~~ SZQOUT  6/10/84 ;LST: = LPT:     1Øxxxxxx  SZQ
         dw      ullout DIOUT             ;LST: = UL1:     11xxxxxx  DIABLO

lstost:  mvi     e,3                      ;List output status
         call    redir                    ;        IOBYTE: 7654321Ø
         dw      ttyost OKIOST            ;LST: = TTY:     ØØxxxxxx
         dw      crtost OKIOST            ;LST: = CRT:     Ø1xxxxxx  OKIDATA
         dw      lptost ~~DIAOST~~ SZQOST  6/10/84 ;LST: = LPT:     1Øxxxxxx  SZQ
         dw      ullost DIAOST            ;LST: = UL1:     11xxxxxx  DIABLO

redir:   lda     iobyte                   ;Get the INTEL standard iobyte
redirØ:  rlc                              ;Shift the next field in
         dcr     e                        ;Bump the shift count
         jnz     redirØ

redirl:  ani     11Øb                     ;Mask the redirection field
         mov     e,a                      ;Make the word table offset
         mvi     d,Ø
         pop     h                        ;Get the table base
         dad     d                        ;Offset into our table
         mov     a,m                      ;Load the low level i/o routine pointer
         inx     h
         mov     h,m
         mov     l,a
         pchl                             ;Execute the low level i/o driver

         endif                            ;IOBYTE redirector

*********************************************************************
*                                                                   *
* CONTYP: 1      Blank space for console driver                     *
*                                                                   *
* The driver entries CONOUT, CONIN, CONIST are defined in the CP/M  *
* alternation guide.  Eg.  Input parameters are in register C and   *
* results are returned in register A.  The TTYSET routine is used   *
* for initialization code.  It should execute a RET when complete.  *
*                                                                   *
* The TTYSET routine could be placed just below the CBOOT routine.  *
* This space (below CBOOT) is recyled for use as a disk buffer      *
* after CBOOT is done.                                              *
*                                                                   *
*********************************************************************
```

ADDED
6/10/84

```
SZQ100    EQU = 10       ; SPOOL-Z-Q at PORT 10
SZQOUT    CALL SZQOST    ; GET STATUS
          ORA A          ; TEST FOR 0 IN ACC
          JZ SZQOUT      ; WAIT UNTIL READY
          MOV A,C        ; PUT CHAR IN A
          OUT SZQ100     ; SEND IT
          RET
SZQOST    IN SZQ100      ; GET SPOOL Z Q STATUS
          ANI 1          ; CHECK BIT 0
          MVI A,0        ; ZERO IN ACC MEANS NOT READY
          RNZ            ; RETURN WITH 0 IF SZQ IS BUSY
          CMA            ; MAKE ACC FFH
          RET            ; READY RETURN, A=FFH
OKOUT     IN 2           ; INPUT FROM PORT 2
          ANI 8          ; Wait until OK to send
          JZ OKOUT
OKOUT1    IN 5           ; BUFFER FULL?
          ANI 1          ;
          JZ OKOUT1      ; Wait until printer ready
          MOV a,C        ; OUTPUT The character
          OUT Ø          ;
          RET

diouta    IN 2           ; OUTPUT FROM PTR-GetStatus
          ANI 80h        ; wait until OK to send
          JZ DIOUTA
          MOV A,C        ; output The character
          Out 1
          Ret

diout     call DIOUTA    ; OUTPUT The character
          Lda COUNT
          Dcr a
          Sta COUNT
          RNZ
          MVI A,78
          STA COUNT
          MVI C,aETX
          Call diouta
dioutb    IN 2           ; input Fm Diablo
          ani 40h
          Jz dioutb
          IN 1
```

```
        if      contyp eq 1             ;User defined IO area
ttyset  equ     $                       ;Console initialization
conout  equ     $                       ;Console output
conin   equ     $                       ;Console input
conist  equ     $                       ;Console input status
        jmp     $
        ds      125
        endif                           ;User IO

***********************************************************************
*                                                                     *
* CONYTP: 6     North Star                                            *
*                                                                     *
* The following code implements the North Star console I/O system.   *
* This system is for users who purchase a Morrow Designs disk         *
* system to replace their North Star disk system.  The Mapping of     *
* the logical to physical entry points is performed as follows:       *
*                                                                     *
* Device name           Left    Right   Parallel                      *
*                       serial  serial  port                          *
*                                                                     *
* Console       CON: =  TTY:    CRT:    UC1:                          *
* Reader        RDR: =  TTY:    PTR:    UR1:                          *
* Punch         PUN: =  TTY:    PTP:    UP1:                          *
* List          LST: =  TTY:    CRT:    UL1:                          *
*                                                                     *
* For example, to use a printer connected to the right serial port,   *
* use the CP/M command:                                               *
*                                                                     *
*       STAT LST:=CRT:                                                *
*                                                                     *
* Likewise, the CP/M command "STAT LST:=UL1:" is used if you have a   *
* printer connected to the parallel port.                             *
*                                                                     *
***********************************************************************

        if      contyp eq 6             ;Use North Star I/O?

nsldat  equ     2                       ;Left serial port data port
nslsta  equ     3                       ;Left serial port status port

nsrdat  equ     4                       ;Right serial port data port
nsrsta  equ     5                       ;Right serial port status port

nsstbe  equ     1                       ;Transmitter buffer empty status bit
nssrbr  equ     2                       ;Reciever buffer ready status bit

                                        ;See the 8251 data sheets for more
                                        ;   configuration information.

nslinl  equ     Øceh                    ;Left serial port initialization # 1
nsrinl  equ     Øceh                    ;Right serial port initialization # 1
                                        ;7654321Ø Bit definations
                                        ;11ØØ111Ø Default configuration
                                        ;xxxxxxØØ Synchronous mode
                                        ;xxxxxxØ1 1X clock rate
                                        ;xxxxxx1Ø 16X clock rate
                                        ;xxxxxx11 64X clock rate
                                        ;xxxxØØxx 5 bit characters
                                        ;xxxxØ1xx 6 bit characters
                                        ;xxxx1Øxx 7 bit characters
                                        ;xxxx11xx 8 bit characters
                                        ;xxxØxxxx Parity disbable
                                        ;xxx1xxxx Parity enable
                                        ;xxØxxxxx Odd parity generation/check
                                        ;xx1xxxxx Even parity generation/check
```

```
ani   7FH      ; Strip parity
Cpi   aack
JNZ   dioutb
Ret
```

```
                                         ;ØØxxxxxx Invalid
                                         ;Ølxxxxxx l stop bit
                                         ;lØxxxxxx 1.5 stop bits
                                         ;llxxxxxx 2 stop bits

nslin2  equ     37h                      ;Left serial port initialization # 2
nsrin2  equ     37h                      ;Right serial port initialization # 2
                                         ;7654321Ø Bit definations
                                         ;ØØllØlll Default configuration
                                         ;xxxxxxxl Enable transmitter
                                         ;xxxxxxlx Assert DTR*
                                         ;xxxxxlxx Enable reciever
                                         ;xxxxlxxx Send break character, TxD low
                                         ;xxxlxxxx Reset PE, OE, FE error flags
                                         ;xxlxxxxx Assert RTS*
                                         ;xlxxxxxx Internal reset
                                         ;lxxxxxxx Enter hunt mode (for sync)

nspdat  equ     Ø                        ;Parallel data port
nspsta  equ     6                        ;Parallel status port

nsprbr  equ     l                        ;Reciever buffer ready status bit
nsptbe  equ     2                        ;Transmitter buffer empty status bit

nsram   equ     ØcØh                     ;North Star memory parity port,
                                         ;   set to Ø for no North Star RAM

***********************************************************************
*                                                                     *
* Left serial port routines.  Use TTY: device.                        *
*                                                                     *
***********************************************************************

ttyin:                                   ;Read a character
        in      nslsta
        ani     nssrbr
        jz      ttyin                    ;Wait till a character is ready
        in      nsldat                   ;Get the character
        ani     7fh                      ;Strip parity
        ret

ttyout:                                  ;Write a character
        in      nslsta
        ani     nsstbe
        jz      ttyout                   ;Wait till the buffer is empty
        mov     a,c                      ;Write the character
        out     nsldat
        ret

ttyist:                                  ;Return input buffer status
        in      nslsta
        ani     nssrbr
        rz                               ;Return not ready
        mvi     a,Øffh
        ret                              ;There is a character ready

ttyost:                                  ;Return output buffer status
        in      nslsta
        ani     nsstbe
        rz                               ;Return not ready
        mvi     a,Øffh
        ret                              ;Return ready

***********************************************************************
*                                                                     *
* Right serial port routines.  Use CRT:, PTR:, and PTP: devices.      *
*                                                                         *
***************************************************************************

crtin:                                   ;Read a character
ptrin:
        in      nsrsta
        ani     nssrbr
        jz      crtin                    ;Wait till a character is ready
        in      nsrdat                   ;Get the character
        ani     7fh                      ;Strip parity
        ret

crtout:                                  ;Write a character
ptpout:
        in      nsrsta
        ani     nsstbe
        jz      crtout                   ;Wait till the buffer is empty
        mov     a,c                      ;Write the character
        out     nsrdat
        ret

crtist:                                  ;Return input buffer status
ptrist:
        in      nsrsta
        ani     nssrbr
        rz                               ;Return not ready
        mvi     a,Øffh
        ret                              ;There is a character ready

~~crtost:~~ OKIOST                       ;Return output buffer status          diaOST   IN   2
        in      nsrsta                                                                  ANI  80H
        ani     nsstbe                                    }  MVI A,OFFH
        rz                               ;Return not ready     Ret                      RZ
        mvi     a,Øffh                                                          OKIOST  MVI  A,OFFH
        ret                              ;Return ready                                  ret
***************************************************************************
*                                                                         *
* Parallel port routines.  Use UC1:, UR1:, UR2:, UP1:, UP2:, LPT:,        *
* and UL1: devices.                                                       *
*                                                                         *
***************************************************************************

uclin:                                   ;Read a character
urlin:
ur2in:
        in      nspsta
        ani     nsprbr
        jz      uclin                    ;Wait till a character is ready
        in      nspdat                   ;Get the character
        push    psw
        mvi     a,3Øh                    ;Reset the parallel input flag
        out     nspsta
        pop     psw
        ani     7fh                      ;Strip parity
        ret

uclout:                                  ;Write a character
uplout:
up2out:
lptout:
ullout:
        in      nspsta
        ani     nsptbe
        jz      uclout                   ;Wait till the buffer is empty
        mvi     a,2Øh                    ;Reset the parallel output flag
        out     nspsta
        mov     a,c                     ;Write the character, strobe bit 7
nspout: ori     8Øh
        out     nspdat
        ani     7fh
        out     nspdat
        ori     8ØH
        out     nspdat
        ret

uclist:                                 ;Return input buffer status
urlist:
ur2ist:
        in      nspsta
        ani     nsprbr
        rz                              ;Return not ready
        mvi     a,Øffh
        ret                             ;Return ready

lptost:                                 ;Return output buffer status
ullost:
        in      nspsta
        ani     nsptbe                                                  OUT
        rz                              ;Return not ready
        mvi     a,Øffh
        ret                             ;Return ready

        endif                           ;North Star I/O configuration
*************************************************************************
*                                                                       *
* LST: device driver routines.                                          *
*                                                                       *
* Routine used depends on the value of lsttyp.  Possible LSTTYP         *
* values are listed as follows:                                         *
*                                                                       *
* LSTTYP is:    Ø       Nothing, used for patching to PROM's            *
*               1       Provide for 128 bytes of patch space            *
*               2       Multio serial, no protocol                      *
*               3       Multio serial, Clear To Send protocol           *
*               4       Multio serial, Data Set Ready protocol          *
*               5       Multio serial, Xon/Xoff protocol                *
*                                                                       *
*************************************************************************

*************************************************************************
*                                                                       *
* lsttyp: 1     Blank space for printer driver                          *
*                                                                       *
* The driver entries LSTOUT and LSTOST are defined in the CP/M          *
* alternation guide.  Eg.  Input parameters are in register C and       *
* results are returned in register A.  The LSTSET routine is used       *
* for initialization code.  It should execute a RET when complete.      *
*                                                                       *
* The LSTSET routine could be placed just below the CBOOT routine.      *
* This space (below CBOOT) is recyled for use as a disk buffer          *
* after CBOOT is done.                                                  *
*                                                                       *
*************************************************************************

        if      lsttyp eq 1
lstset  equ     $                       ;Printer initialization
lstout  equ     $                       ;Printer output
lstost  equ     $                       ;Printer output status
        jmp     $
        ds      125
        endif                           ;User IO

****************************************************************
*                                                              *
* lsttyp: 2, 3, 4, or 5 Serial printer, multi protocol         *
*                                                              *
****************************************************************

        if      (lsttyp ge 2) and (lsttyp le 5)

lstout: call    lstost          ;Check printer status
        ora     a
        jz      lstout          ;Loop if not ready

        mov     a,c             ;Print the character
        out     thr
        ret

lstost: call    sellst          ;Printer status routine

        in      lsr             ;Check if transmitter buffer empty
        ani     thre
        rz                      ;Return busy if buffer is not empty

        lhld    lstand          ;Fetch handshake mask bits

        in      msr             ;Get MODEM Status Register
        ana     l               ;Strip out hand-shake lines
        xra     h               ;Invert status
        rz                      ;Return busy if printer is busy

        lda     lastch          ;Get last character recieved from the printer
        mov     b,a
        in      lsr             ;Check for a character from the printer
        ani     dr
        jz      xskip           ;Skip if no character present
        in      rbr             ;Get the character
        ani     7fh             ;Strip parity
        sta     lastch          ;Save last character recieved
        mov     b,a
xskip:  mov     a,b
        sui     xoff            ;Check for Xoff char (control S)
        jnz     xsdone          ;Printer ready
        ret                     ;Printer not ready (return zero)

xsdone: mvi     a,Øffh          ;Printer ready for data
        ret

        endif                   ;Multi I/O serial driver

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

gocpm:  lxi     h,buff          ;Set up initial DMA address
        call    setdma
        mvi     a,(jmp)         ;Initialize jump to warm boot
        sta     wbot
        sta     entry           ;Initialize jump to BDOS
        lxi     h,wboote        ;Set up low memory entry to CBIOS warm boot
        shld    wbot+1
        lxi     h,bdos+6        ;Set up low memory entry to BDOS
        shld    entry+1
        xra     a               ;A <- Ø
        sta     bufsec          ;Set buffer to unknown state
        sta     bufwrtn         ;Set buffer not dirty flag
        sta     error           ;Clear buffer error flag

        lda     cwflg           ;Get cold/warm boot flag
        ora     a
        lxi     h,coldmes       ;Pointer to initial cold command
        jz      cldcmnd
        lxi     h,warmes        ;Pointer to initial warm command
cldcmnd:mov     e,m             ;Do one level of indirection
        inx     h
        mov     d,m
        ldax    d               ;Get command length
        inr     a               ;Bump length to include length byte itself
        lxi     h,ccp+7         ;Command buffer (includes length byte)
        mov     c,a             ;Set up for block move
        mvi     b,Ø
        call    movbyt          ;Move command to internal CCP buffer
        lda     cwflg           ;Figure out whether or not to send message
        ora     a
        lda     autoflg
        jz      cldbot
        rar
cldbot: rar
        lda     cdisk           ;Jump to CP/M with currently selected disk in C
        mov     c,a
        jc      ccp             ;Enter CP/M, send message
        jmp     ccp+3           ;Enter CP/M, no message

cwflg:  db      Ø               ;Cold/warm boot flag

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

wboot:  lxi     sp,tpa          ;Set up stack pointer
        mvi     a,1
        sta     cwflg           ;Set cold/warm boot flag

        mvi     h,wmdrive       ;Move drive to warm boot off of into (h)
        mvi     l,d$wboot       ;Peform warm boot operation
        call    jumper
        jnc     gocpm           ;No error

        hlt                     ;Halt computer
        db      Ø

        jmp     wboot           ;In case user restarts the computer

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

setsec: mov     h,b             ;Enter with sector number in (bc)
        mov     l,c
        shld    cpmsec
donop:  ret
```

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

setdma: mov     h,b             ;Enter with DMA address in (bc)
        mov     l,c
        shld    cpmdma          ;CP/M dma address
        ret

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

home:   lda     bufwrtn         ;Test buffer dirty flag
        ora     a
        jnz     dohome          ;Skip buffer disable if buffer dirty
        xra     a               ;Invalidate buffer on home call
        sta     bufsec
dohome: lxi     b,Ø             ;Track to seek to

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

settrk: mov     h,b     ;Enter with track number in (bc)
        mov     l,c
        shld    cpmtrk
        ret

*****************************************************************
*                                                               *
* Sectran translates a logical sector number into a physical    *
* sector number.                                                *
*                                                               *
*****************************************************************

sectran:lda     cpmdrv          ;Get the Drive Number
        mov     h,a             ;Drive in (h)
        mvi     l,d$stran
        jmp     jumper          ;See device level sector translation routines

*****************************************************************
*                                                               *
* Setdrv selects the next drive to be used in read/write        *
* operations.  If the drive has never been selected it calls    *
* a low level drive select routine that should perform some     *
* sort of check if the device is working.  If not working then  *
* it should report an error.  If the logical drive has been     *
* selected before then setdrv just returns the DPH without      *
* checking the drive.                                           *
*                                                               *
*****************************************************************

setdrv: mov     a,c             ;Save the logical drive number
        sta     cpmdrv
        cpi     maxlog          ;Check for a valid drive number
        jnc     zret            ;Illegal drive
        mov     a,e             ;Check if bit Ø of (e) = 1
        ani     1
        jnz     setd3           ;Drive has allready been accessed

        mov     h,c             ;Move logical drive into (h)
        mvi     l,d$sel1
        call    jumper          ;Call low level drive select
        mov     a,h             ;Check if the low level drive select returned
        ora     l               ; zero to indicate an error
        jz      zret            ;Yes, an error so report to CP/M

        push    h               ;Save DPH address
        call    gdph            ;Get entry if DPH save table
        pop     d               ;DPH -> (de)
        mov     m,e             ;Put address of DPH in table
        inx     h
        mov     m,d
        inx     h
        mov     m,c             ;Put sector size in table
        inx     h
        mov     a,m             ;Check if bad map has ever been read for this
        ora     a               ; drive
        cz      getbad          ;Never been read so read in bad map
        xchg                    ;DPH -> (hl)

setdØ:  mov     a,c             ;Move sector size code into (a)
        sta     secsiz          ;Save sector size
        xra     a
setd1:  dcr     c               ;Create number of (128 bytes/physical sector)-1
        jz      setd2
        rlc
        ori     1
        jmp     setd1
setd2:  sta     secpsec         ;Save for deblocking
        lda     cpmdrv          ;Save current drive as old drive
        sta     lastdrv         ; in case of select errors
        ret

setd3:  push    d               ;Save DPH address
        mov     h,c             ;Drive in (h)
        mvi     l,d$sel2        ;Select drive
        call    jumper
        call    gdph            ;Quick select
        pop     d
        mov     e,m             ;DPH -> (de)
        inx     h
        mov     d,m
        inx     h
        mov     c,m             ;Sector size -> (c)
        xchg                    ;DPH -> (hl)
        jmp     setdØ

gdph:   lda     cpmdrv          ;Return pointer to DPH save area
        rlc                     ;Each entry is 4 bytes long
        rlc
        mov     e,a
        mvi     d,Ø
        lxi     h,dphtab        ;DPH save area table
        dad     d               ;Add offset
        ret                     ;(hl) = DPH save area for current drive

zret:   lxi     h,Ø             ;Seldrv error exit
        lda     lastdrv         ;Get last selected drive
        mov     c,a
        lda     cdisk           ;Pick up user/drive
        ani     ØfØh            ;Save user number
        ora     c               ;Put together with old drive
        sta     cdisk
        ret

*****************************************************************
*                                                               *
* DPH save area.  Each entry is 4 bytes long:                   *
*       Ø - LSB of DPH address                                  *
*       1 - MSB of DPH address                                  *
*       2 - Sector size code (1 = 128, 2 = 256, 3 = 512...      *
*       3 - Bad map has been initilized (Ø = Uninitilized)      *
*                                                               *
*****************************************************************

dphtab: rept    maxlog*4
        db      Ø
        endm

*****************************************************************
*                                                               *
* Getbad - Check if a device has a bad map.  If the device has  *
* a bad sector map then append bad entries to end of badmap     *
* table.                                                        *
*                                                               *
*****************************************************************

getbad: mvi     m,1             ;Set drive initilized
        push    b
        push    d
        lda     cpmdrv          ;Pick up current drive
        mov     h,a             ;Call drive routine to return a pointer to
        mvi     l,d$bad         ;the track and sector of the bad map
        call    jumper

        mov     a,h             ;If routine returns Ø then the device has
        ora     l               ; no bad sector map
        jz      badret

        mov     e,m             ;Pick up track number of bad sector map -> (de)
        inx     h
        mov     d,m
        inx     h
        xchg
        shld    cpmtrk
        xchg
        mov     a,m             ;Pick up sector number of of bad sector map
        inx     h
        mov     h,m
        mov     l,a
        shld    truesec
        call    fill            ;Read in bad sector map into the buffer
        rc

        lhld    badptr          ;Pick up bad map pointer
        lxi     d,buffer        ;Start at beginning of buffer
badl:   ldax    d               ;Pick up an entry from the buffer
        ora     a
        jz      bade            ;All done
        mov     a,m             ;Pick up entry from bad map table
        inr     a
        jz      overflo         ;Bad map overflow
        lda     cpmdrv          ;Put drive in table
        mov     m,a
        inx     h
        lxi     b,8
        call    movbyt          ;Move the rest of information into the table
        jmp     badl

bade:   shld    badptr          ;Restore new bad map pointer
badret: pop     d
        pop     b
        ret

overflo:lxi     h,omes
        call    message
        jmp     badret

omes:   db      Ødh, Øah, 'BAD MAP OVERFLOW!', Ødh, Øah, Ø

nobad:  lxi     h,Ø             ;Used by device drives to indicate no bad
        ret                     ; sector map

badptr: dw      badmap          ;Pointer to next available bad map entry

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

write:  mov     a,c             ;Save write command type
        sta     writtyp
        mvi     a,1             ;Set write command
        jmp     rwent

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

read:   xra     a               ;Set the command type to read
        if      nostand ne Ø
        sta     unaloc          ;Clear unallocated write flag
        endif
rwent:  sta     rdwr            ;Save command type

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

redwrt: mvi     b,Ø             ;The Ø is modified to contain the log2
secsiz  equ     $-1             ;       of the physical sector size/128
                                ;       on the currently selected disk
        lhld    cpmsec          ;Get the desired CP/M sector #
        mov     a,h
        ani     8Øh             ;Save only the side bit
        mov     c,a             ;Remember the side
        mov     a,h
        ani     7fh             ;Forget the side bit
        mov     h,a
        dcx     h               ;Temporary adjustment
divloop:dcr     b               ;Update repeat count
        jz      divdone
        ora     a
        mov     a,h
        rar
        mov     h,a
        mov     a,l
        rar                     ;Divide the CP/M sector # by the size
                                ;       of the physical sectors
        mov     l,a
        jmp     divloop         ;
divdone:inx     h
        mov     a,h
        ora     c               ;Restore the side bit
        mov     h,a
        shld    truesec         ;Save the physical sector number
        lxi     h,cpmdrv        ;Pointer to desired drive,track, and sector
        lxi     d,bufdrv        ;Pointer to buffer drive,track, and sector
        mvi     b,6             ;Count loop
dtslop: dcr     b               ;Test if done with compare
        jz      move            ;Yes, match. Go move the data
        ldax    d               ;Get a byte to compare
        cmp     m               ;Test for match
        inx     h               ;Bump pointers to next data item
        inx     d
        jz      dtslop          ;Match, continue testing

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

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

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

move:   lda     cpmsec          ;Get the CP/M sector to transfer
        dcr     a               ;Adjust to proper sector in buffer
        ani     Ø               ;Strip off high ordered bits
secpsec equ     $-1             ;The Ø is modified to represent the # of
                                ;       CP/M sectors per physical sectors
        mov     l,a             ;Put into HL
        mvi     h,Ø
        dad     h               ;Form offset into buffer
        dad     h
        dad     h
        dad     h
        dad     h
        dad     h
        dad     h
        lxi     d,buffer        ;Beginning address of buffer
        dad     d               ;Form beginning address of sectgr to transfer
        xchg                    ;DE = address in buffer
        lxi     h,Ø             ;Get DMA address, the Ø is modified t/
                                ;       contain the DMA address
cpmdma  equ     $-2
        mvi     a,Ø             ;The zero gets modified to contain
                                ;       a zero if a read, or a 1 if write
rdwr    equ     $-1
        ana     a               ;Test which kind of operation
        jnz     into            ;Transfer data into the buffer
outof:  call    mov128
        lda     error           ;Get the buffer error flag
        ret

into:   xchg                    ;
        call    mov128          ;Move the data, HL = destination
                                ;       DE = source
        mvi     a,1
        sta     bufwrtn         ;Set buffer written into flag
        mvi     a,Ø             ;Check for directory write
writtyp equ     $-1
        dcr     a               ;Test for a directory write
        mvi     a,Ø
        rnz                     ;No error exit

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

flush:  mvi     a,Ø             ;The Ø is modified to reflect if
                                ;       the buffer has been written into
bufwrtn equ     $-1
        ora     a               ;Test if written into
        rz                      ;Not written, all done
        mvi     a,d$write
        sta     rwop+1
        call    prep            ;Do the physical write
        sta     error           ;Set up the error flag
        ret

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

prep:   call    alt             ;Check for alternate sectors
        di                      ;Reset interrupts
        xra     a               ;Reset buffer written flag
        sta     bufwrtn

        mvi     b,retries       ;Maximum number of retries to attempt
retrylp:push    b               ;Save the retry count

        mvi     l,d$sel2        ;Select drive
        call    jumpbuf

        lhld    alttrk          ;Track number -> (hl)

        mov     a,h             ;Test for track zero
        ora     l

        push    h               ;Save track number
        mvi     l,d$home
        cz      jumpbuf
        pop     b               ;Restore track #

        mvi     l,d$strk
        call    jumpbuf

        lhld    altsec          ;Sector -> (hl)
        mov     b,h
        mov     c,l

        mvi     l,d$ssec
        call    jumpbuf

        lxi     b,buffer        ;Set the DMA address
        mvi     l,d$sdma
        call    jumpbuf

rwop:   mvi     l,Ø             ;Get operation address
        call    jumpbuf

        pop     b               ;Restore the retry counter
        mvi     a,Ø             ;No error exit status
        rnc                     ;Return no error
        dcr     b               ;Update the retry counter
        stc                     ;Assume retry count expired
        mvi     a,Øffh          ;Error return
        rz                      ;Return sad news
        mov     a,b
        cpi     retries/2
        jnz     retrylp         ;Try again
        push    b               ;Save retry count
        mvi     l,d$home        ;Home drive after (retries/2) errors
        call    jumpbuf
        pop     b
        jmp     retrylp         ;Try again

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

fill:   call    flush           ;Flush buffer first
        rc                      ;Check for error
        lxi     d,cpmdrv        ;Update the drive, track, and sector
        lxi     h,bufdrv
        lxi     b,5             ;Number of bytes to move
        call    movbyt          ;Copy the data

        lda     rdwr            ;Test read write flag
        ora     a
        jz      fread           ;Skip write type check if reading
        lda     writtyp         ;Ø = alloc, 1 = dir, 2 = unalloc

        if      nostand ne Ø    ;Do non standard (but quick and dirty) check
        ora     a
        jnz     fnaloc          ;Skip if not an allocated write

        lda     unaloc          ;Check unallocated write in progress flag
        ora     a
        jz      fwritin         ;We are doing an allocated write
        lhld    cblock          ;Get current block address
        xchg
        lhld    oblock          ;   and old block address
        mov     a,d             ;Compare old versus new
        cmp     h
        jnz     awritin         ;Different, clear unallocated writting mode
        mov     a,e
        cmp     l
        jnz     awritin
        lxi     h,cpmdrv        ;Test for different drive
        lda     unadrv
        cmp     m
        jnz     awritin         ;Drive is different, clear unallocated mode
        ret                     ;Unallocated write, do nothing...

fnaloc: dcr     a
        jz      awritin         ;Do a directory write
                                ;We are now doing an unallocated write
        lhld    cblock          ;Save current block number
        shld    oblock
        lda     cpmdrv          ;Save drive that this block belongs to
        sta     unadrv
        mvi     a,1             ;Set unallocated write flag
        sta     unaloc
        ret                     ;   and we do nothing about the write

awritin:xra     a               ;Clear unallocated writting mode
        sta     unaloc

        else                    ;Do standard unallocated test

        sui     2               ;Test for an unallocated write
        rz

        endif

fwritin:lda     secsiz          ;Check for 128 byte sectors
        dcr     a
        rz                      ;No deblocking needed

fread:  mvi     a,d$read
        sta     rwop+1
        call    prep            ;Read the physical sector the buffer
        sta     error           ;Set the error status
        ret

******************************************************************
*                                                                *
* Jumpbuf, jumper are used to dispatch to a low level device     *
* subroutine.  Jumper is called with the drive in (h) and the    *
* routine number (see description above) in (l).  It passes      *
* along the (bc) and (de) registers unaltered.  Jumpbuf is       *
* a call to jumper with the drive number from bufdrv.            *
*                                                                *
******************************************************************

jumpbuf:lda     bufdrv          ;Dispatch with bufdrv for drive
        mov     h,a

jumper: push    d
        push    b
        push    h
        mov     a,h             ;Logical drive into (a)
        lxi     d,dsttab        ;Drive specification pointer table
jumpl:  mov     c,a             ;Save logical in (c)
        ldax    d
        mov     l,a
        inx     d
        ldax    d
        mov     h,a             ;Get a DST pointer in (hl)
        inx     d
        mov     a,c             ;Logical in (a)
        sub     m               ;Subtract from first entry in DST
        jnc     jumpl           ;Keep scanning table till correct driver found

        inx     h               ;Bump (hl) to point to start of dispatch table
        pop     d               ;Real (hl) -> (de)
        mov     a,e             ;Move offset number into (a)
        rlc                     ;Each entry is 2 bytes
        mov     e,a             ;Make an offset
        mvi     d,Ø
        dad     d               ;(hl) = **Routine
        mov     a,m             ;Pick up address of handler for selected
        inx     h               ; function
        mov     h,m
        mov     l,a             ;(hl) = *routine
        mov     a,c             ;Logical in (a)
        pop     b               ;Restore saved registers
        pop     d
        pchl

*****************************************************************
*                                                               *
* Check for alternate sectors in bad sector table.  If an       *
* alternate sector is found replace alttrk and altsec with      *
* new sector number else pass along unaltered.                  *
*                                                               *
*****************************************************************

alt:    lxi     h,badmap        ;Address of bad map -> (hl)
        lda     bufdrv          ;Pick up drive number currently working on
        mov     c,a             ;Move drive into (c) for speed in search
all:    xchg
        lhld    badptr          ;Get bad map pointer
        xchg                    ; -> (de)
        mov     a,d             ;Check if at end of bad map table
        cmp     h
        jnz     alt2            ;Still more
        mov     a,e
        cmp     l
        jnz     alt2            ;Still more
        lhld    buftrk          ;No alternate sector so use selected sector
        shld    alttrk
        lhld    bufsec
        shld    altsec
        ret

alt2:   push    h               ;Save current bad map entry address
        mov     a,c             ;Move drive into (a)
        cmp     m               ;Check if drive in table matches
        jnz     altmis          ;Does not match skip this entry
        inx     h               ;Point to LSB of alternate track
        lda     buftrk          ;Pick up LSB of buffer track
        cmp     m
        jnz     altmis
        inx     h               ;Point to MSB alternate track
        lda     buftrk+l        ;Pick up MSB of buffer track
        cmp     m
        jnz     altmis
        inx     h               ;Point to LSB of alternate sector
        lda     bufsec          ;Pick up LSB of buffer sector
        cmp     m
        jnz     altmis
        inx     h               ;Point to MSB of alternate sector
        lda     bufsec+1        ;Pick up MSB of buffer sector
        cmp     m
        jnz     altmis          ;Found an alternate sector
        inx     h               ;Point to real info on the alternate sector
        lxi     d,alttrk
        xchg                    ;MOVLOP (de) = source, (hl) = dest
        push    b
        lxi     b,4
        call    movbyt          ;Move alternate sector info in correct place
        pop     b
        pop     h
        ret

altmis: pop     h               ;Current alternate did not match
        lxi     d,9             ;Bump pointer by the length of an entry
        dad     d
        jmp     all             ;Loop for more

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

movl28: lxi     b,128           ;Length of transfer
movbyt: xra     a               ;Check if host processor is a Z8Ø
        adi     3
        jpo     z8Ømov          ;Yes, Its a Z8Ø so use block move

m8Ø8Ø:  ldax    d               ;Get a byte of source
        mov     m,a             ;Move it
        inx     d               ;Bump pointers
        inx     h
        dcx     b               ;Update counter
        mov     a,b             ;Test for end
        ora     c
        jnz     m8Ø8Ø
        ret

z8Ømov: xchg                    ;Source in (hl), Destination in (de)
        dw      ØbØedh          ;ldir
        xchg
        ret

*****************************************************************
*                                                               *
* Return DPH pointer.  Enter with (de) with DPH base address    *
* and (a) with logical drive number.  Returns with DPH address  *
* in (hl).                                                      *
*                                                               *
*****************************************************************

retdph  mov     l,a             ;Move logical drive into (l)
        mvi     h,Ø
        dad     h               ;Multiply by 16 (size of DPH)
        dad     h
        dad     h
        dad     h
        dad     d               ;(hl) = pointer to DPH
        ret

*****************************************************************
*                                                               *
* Utility routine to output the message pointed at by (hl)      *
* terminated with a null.                                       *
*                                                               *
*****************************************************************

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

*****************************************************************
*                                                               *
* The following code is for the Diskus Hard disk                *
*                                                               *
*****************************************************************

        if      hdca ne Ø               ;Want HDC3 or 4 controller included ?

hdorg   equ     5Øh                     ;Hard Disk Controller origin

hdstat  equ     hdorg                   ;Disk Status
hdcntl  equ     hdorg                   ;Disk Control
hdreslt equ     hdorg+1                 ;Disk Results
hdcmnd  equ     hdorg+1                 ;Disk Commands
hdskomp equ     hdorg+2                 ;Seek complete clear port (on HDC4)
hdfunc  equ     hdorg+2                 ;Function port
hddata  equ     hdorg+3                 ;Data port

;       Status port (5Ø)

tkzero  equ     Ø1h                     ;Track zero
opdone  equ     Ø2h                     ;Operation done
complt  equ     Ø4h                     ;Seek complete
tmout   equ     Ø8h                     ;Time out
wfault  equ     1Øh                     ;Write fault
drvrdy  equ     2Øh                     ;Drive ready
index   equ     4Øh                     ;Delta index

;       Control port (5Ø)

hdfren  equ     Ø1h                     ;Enable external drivers
hdrun   equ     Ø2h                     ;Enable controllers state machine
hdclok  equ     Ø4h                     ;Clock source control bit, high = disk
hdwprt  equ     Ø8h                     ;Write protect a drive

;       Result port (51)

retry   equ     Ø2h                     ;Retry flag

;       Command port (51)

idbuff  equ     Ø                       ;Initialize data buffer pointer
rsect   equ     1                       ;Read sector
wsect   equ     5                       ;Write sector
isbuff  equ     8                       ;Initialize header buffer pointer

;       Function port (52)

pstep   equ     Ø4h                     ;Step bit
nstep   equ     Øffh-pstep              ;Step bit mask
null    equ     Øfch                    ;Null command
```

```
;       Misc constants

hdrlen  equ     4                       ;Sector header length
seclen  equ     512                     ;Sector data length

*****************************************************************
*                                                               *
* Device Specification Table for HDCA controller driver         *
*                                                               *
*****************************************************************

hddst:  db      maxhd*hdlog             ;Number of logical drives
        dw      hdwarm                  ;Warm boot
        dw      hdtran                  ;Sector translation
        dw      hdldrv                  ;First time select
        dw      hddrv                   ;General select
        dw      hdhome                  ;Home current selected drive
        dw      hdseek                  ;Seek to selected track
        dw      hdsec                   ;Select sector
        dw      hddma                   ;Set DMA address
        dw      hdread                  ;Read a sector
        dw      hdwrite                 ;Write a sector
        dw      nobad                   ;No bad sector map

hdwarm: call    divlog                  ;Get physical drive number in (c)
        xra     a
        lxi     h,ccp-2ØØh              ;Initial DMA address
        push    h
        sta     head                    ;Select head zero
        inr     a                       ; 1 -> (a)
        push    psw                     ;Save first sector - 1
        call    hdd2                    ;Select drive
        mvi     c,Ø
        call    hdhome                  ;Home the drive
hdwrld: pop     psw                     ;Restore sector
        pop     h                       ;Restore DMA address
        inr     a
        sta     hdsect
        cpi     13                      ;Past BDOS ?
        rz                              ;Yes, all done
        inr     h                       ;Update DMA address
        inr     h
        shld    hdadd
        push    h
        push    psw
hdwrrd: lxi     b,retries*1ØØh+Ø        ;Retry counter
hdwr:   push    b                       ;Save the retry count
        call    hdread                  ;Read the sector
        pop     b
        jnc     hdwrld                  ;Test for error
        dcr     b                       ;Update the error count
        jnz     hdwr                    ;Keep trying if not too many errors
        stc                             ;Error flag
        ret

hdtran: mov     h,b                     ;Sector translation is handled via
        mov     l,c                     ;   physical sector header skewwing
        inx     h
        ret

hdldrv: sta     hdcur                   ;Save logical disk
        call    divlog                  ;Divide by logical disks per drive
        mov     a,c
        sta     hddisk                  ;Save new physical drive
        call    hdptr                   ;Get track pointers
        mov     a,m                     ;Get current track
        inr     a                       ;Check if -1
        jnz     hdl2                    ;Nope, allready accessed
        ori     null                    ;Select drive
        out     hdfunc
        mvi     a,hdfren+hdclok         ;Enable drivers
        out     hdcntl
        mvi     c,239                   ;Wait 2 minutes for disk ready
        lxi     h,Ø
hdtdel: dcx     h
        mov     a,h
        ora     l
        cz      dcrc
        jz      zret                    ;Drive not ready error
        in      hdstat                  ;Test if ready yet
        ani     drvrdy
        jnz     hdtdel

        if      not fujitsu
        lxi     h,Ø                     ;Time one revolution of the drive
        mvi     c,index
        in      hdstat
        ana     c
        mov     b,a                     ;Save current index level in B
hdinxdl:in      hdstat
        ana     c
        cmp     b                       ;Loop untill index level changes
        jz      hdinxdl
hdindx2:inx     h
        in      hdstat                  ;Start counting untill index returns to
        ana     c                       ;       previous state
        cmp     b
        jnz     hdindx2

        if      m1Ø                     ;Memorex M1Ø's have 4Ø ms head settle
        dad     h                       ;HL*2
        endif

        if      m26                     ;Shugart M26's have 3Ø ms head settle
        xra     a                       ;HL/2 + HL (same as HL*1.5)
        mov     a,h
        rar
        mov     d,a
        mov     a,l
        rar
        mov     e,a
        dad     d
        endif

        shld    settle                  ;Save the count for timeout delay
        endif

        call    hdhome

hdl2:   lda     hdcur                   ;Load logical drive
        lxi     d,dphhdØ                ;Start of hard disk DPH's
        mvi     c,3                     ;Hard disk sector size equals 512 bytes
        jmp     retdph

dcrc:   dcr     c                       ;Conditional decrement C routine
        ret

divlog: mvi     c,Ø
divlx:  sui     hdlog
        rc
        inr     c
        jmp     divlx
```

```
hddrv:  sta     hdcur
        call    divlog                  ;Get the physical drive #
hdd2:   mov     a,c
        sta     hddisk                  ;Select the drive
        ori     null
        out     hdfunc
        mvi     a,hdfren+hdrun+hdclok+hdwprt    ;Write protect
        out     hdcntl
        ret

hdhome: call    hdptr                   ;Get track pointer
        mvi     m,Ø                     ;Set track to zero
        in      hdstat                  ;Test status
        ani     tkzero                  ;At track zero ?
        rz                              ;Yes

        if      not fujitsu
hdstepo:in      hdstat                  ;Test status
        ani     tkzero                  ;At track zero ?
        jz      hddelay
        mvi     a,1
        stc
        call    accok                   ;Take one step out
        jmp     hdstepo

        else

        xra     a
        jmp     accok
        endif

        if      not fujitsu
hddelay:lhld    settle                  ;Get hddelay
deloop: dcx     h                       ;Wait 2Øms
        mov     a,h
        ora     l
        inx     h
        dcx     h
        jnz     deloop
        ret
        endif

hdseek: call    hdptr                   ;Get pointer to current track
        mov     e,m                     ;Get current track
        mov     m,c                     ;Update the track
        mov     a,e                     ;Need to seek at all ?
        sub     c
        rz
        cmc                             ;Get carry into direction
        jc      hdtrk2
        cma
        inr     a
        if      fujitsu
hdtrk2: jmp     accok
        else
hdtrk2: call    accok
        jmp     hddelay
        endif

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

hddma:  mov     h,b                     ;Save the DMA address
        mov     l,c
        shld    hdadd
        ret

wsdone: in      hdstat                  ;Wait for seek complete to finish
        ani     complt
        jz      wsdone
        in      hdskomp                 ;Clear sdone bit on an HDCA4
        ret

        if      m26
hdsec:  mvi     a,Ø1fh                  ;For compatibility with Cbios revs.
                                        ;  2.3 and 2.4
        ana     c                       ;Mask in sector number (Ø-31)
        cz      getspt                  ;Translate sector Ø to sector 32
        sta     hdsect                  ;Save translated sector number (1-32)
        mvi     a,ØeØh                  ;Get the head number
        ana     c
        rlc
        rlc
        rlc
        sta     head                    ;Save the head number
getspt: mvi     a,hdspt
        ret

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

divspt: mvi     c,Ø
divsx:  sui     hdspt
        rc
        inr     c
        jmp     divsx
        endif

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

hdwrite:call    hdprep                  ;Prepare header
        rc
        xra     a
        out     hdcmnd
        lhld    hdadd
        mvi     b,seclen/4
wtloop: mov     a,m                     ;Move 4 bytes
        out     hddata
        inx     h
        mov     a,m
        out     hddata
        inx     h
        mov     a,m
        out     hddata
        inx     h
        mov     a,m
        out     hddata
        inx     h
        dcr     b
        jnz     wtloop
        mvi     a,wsect                 ;Issue write sector command
        out     hdcmnd
        call    process
        rc
        mvi     a,wfault
        ana     b
        stc
        rz
        xra     a
        ret

process:in      hdstat                  ;Wait for command to finish
        mov     b,a
        ani     opdone
        jz      process
        mvi     a,hdfren+hdrun+hdclok   ;Write protect
        out     hdcntl
        in      hdstat
        ani     tmout                   ;Timed out ?
        stc
        rnz
        in      hdreslt
        ani     retry                   ;Any retries ?
        stc
        rnz
        xra     a
        ret
```

```
hdprep: in      hdstat
        ani     drvrdy
        stc
        rnz
        mvi     a,isbuff                ;Initialize pointer
        out     hdcmnd
        call    build
        ori     Øch
        out     hdfunc
        lda     head
        out     hddata                  ;Form head byte
        call    hdptr                   ;Get pointer to current drives track
        mov     a,m                     ;Form track byte
        out     hddata
        ana     a
        mvi     b,8Øh
        jz      zkey
        mvi     b,Ø
zkey:   lda     hdsect                  ;Form sector byte
        out     hddata
        mov     a,b
        out     hddata
        mvi     a,hdfren+hdrun+hdclok   ;Write protect
        out     hdcntl
        mvi     a,hdfren+hdrun+hdclok+hdwprt    ;Write protect
        out     hdcntl
        xra     a
        ret

hdptr:  lhld    hddisk                  ;Get a pointer to the current drives
        mvi     h,Ø                     ;   track position
        xchg
        lxi     h,hdtrak
        dad     d
        ret

build:  lda     head                    ;Build a controller command byte
        ral
        ral
        ral
        ral
        lxi     h,hddisk
        ora     m
        xri     ØfØh
        ret

hdcur:  db      Ø                       ;Current logical disk
hdadd:  dw      Ø                       ;DMA address
hddisk: db      Ø                       ;Current physical disk number
head:   db      Ø                       ;Current physical head number
hdsect: db      Ø                       ;Current physical sector number

hdtrak: db      Øffh                    ;Track pointer for each drive
        db      Øffh                    ;All drive default to an uncalibrated
        db      Øffh                    ;   state (ff)
        db      Øffh

settle: dw      Ø                       ;Time delay constant for head settle

        endif

****************************************************************
*                                                              *
* The following equates relate the Morrow Designs 2D/B         *
* controller. If the controller is non standard (ØF8ØØH)       *
* only the FDORIG equate need be changed.                      *
*                                                               *
*****************************************************************

        if      maxfd ne Ø      ;Include Discus 2D ?
fdorig  equ     Øf8ØØH          ;Origin of Disk Jockey PROM
fdboot  equ     fdorig+ØØh      ;Disk Jockey 2D initialization
fdcin   equ     fdorig+Ø3h      ;Disk Jockey 2D character input routine
fdcout  equ     fdorig+Ø6h      ;Disk Jockey 2D character output routine
fdhome  equ     fdorig+Ø9h      ;Disk Jockey 2D track zero seek
fdseek  equ     fdorig+Øch      ;Disk Jockey 2D track seek routine
fdsec   equ     fdorig+Øfh      ;Disk Jockey 2D set sector routine
fddma   equ     fdorig+12h      ;Disk Jockey 2D set DMA address
fdread  equ     fdorig+15h      ;Disk Jockey 2D read routine
fdwrite equ     fdorig+18h      ;Disk Jockey 2D write routine
fdsel   equ     fdorig+1bh      ;Disk Jockey 2D select drive routine
fdtstat equ     fdorig+21h      ;Disk Jockey 2D terminal status routine
fdstat  equ     fdorig+27h      ;Disk Jockey 2D status routine
fderr   equ     fdorig+2ah      ;Disk Jockey 2D error, flash led
fdden   equ     fdorig+2dh      ;Disk Jockey 2D set density routine
fdside  equ     fdorig+3Øh      ;Disk Jockey 2D set side routine
fdram   equ     fdorig+4ØØh     ;Disk Jockey 2D RAM address
dblsid  equ     2Øh             ;Side bit from controller
io      equ     fdorig+3f8h     ;Start of I/O registers
dreg    equ     io+1
cmdreg  equ     io+4
clrcmd  equ     ØdØh

*****************************************************************
*                                                               *
* Device Specification Table for the Disk Jockey 2D/B           *
*                                                               *
*****************************************************************

fddst:  db      maxfd           ;Number of logical drives
        dw      fdwarm          ;Warm boot
        dw      fdtran          ;Sector translation
        dw      fdldrv          ;Select drive 1
        dw      fdsel2          ;Select drive 2
        dw      fdlhome         ;Home drive
        dw      fdseek          ;Seek to specified track
        dw      fdssec          ;Set sector
        dw      fddma           ;Set DMA address
        dw      fdread          ;Read a sector
        dw      fdwrite         ;Write a sector
        dw      nobad           ;No bad sector map

*****************************************************************
*                                                               *
* Floppy disk warm boot loader                                  *
*                                                               *
*****************************************************************

fdwarm: mov     c,a
        call    fdsel           ;Select drive A
        mvi     c,Ø             ;Select side Ø
        call    fdside
wrmfail:call    fdhome          ;Track Ø, single density
        jc      wrmfail         ;Loop if error

                                ;The next block of code re-initializes
                                ;   the warm boot loader for track Ø
        mvi     a,5-2           ;Initialize the sector to read - 2
        sta     newsec
        lxi     h,ccp-1ØØh      ;First revolution DMA - 1ØØh
        shld    newdma
                                ;Load all of track Ø
```

```
tØboot: mvi     a,5-2           ;First sector - 2
newsec  equ     $-1
        inr     a               ;Update sector #
        inr     a
        cpi     27              ;Size of track in sectors + 1
        jc      nowrap          ;Skip if not at end of track
        jnz     t1boot          ;Done with this track
        sui     27-6            ;Back up to sector 6
        lxi     h,ccp-8Øh       ;Memory address of sector - 1ØØh
        shld    newdma
nowrap: sta     newsec          ;Save the updated sector #
        mov     c,a
        call    fdsec           ;Set up the sector
        lxi     h,ccp-1ØØh      ;Memory address of sector - 1ØØh
newdma  equ     $-2
        lxi     d,1ØØh          ;Update DMA address
        dad     d
nowrp:  shld    newdma          ;Save the updated DMA address
        mov     b,h
        mov     c,l
        call    fddma           ;Set up the new DMA address
        lxi     b,retries*1ØØh+Ø;Maximum # of errors, track #
wrmfred:push    b
        call    fdseek          ;Set up the proper track
        call    fdread          ;Read the sector
        pop     b
        jnc     tØboot          ;Continue if no error
        dcr     b
        jnz     wrmfred         ;Keep trying if error
        jmp     fderr           ;Too many errors, flash the light

;Load track 1, sector 1, sector 3 (partial), sector 2 (1Ø24 byte sectors)

t1boot: mvi     c,1             ;Track 1
        call    fdseek
        lxi     b,ccp+Øb00h     ;Address for sector 1
        lxi     d,1Ø*1ØØh+1     ;Retry count + sector 1
        call    wrmread
        lxi     b,ccp+ØfØØh     ;Address for sector 2
        lxi     d,1Ø*1ØØh+3     ;Retry count + sector 3
        call    wrmread

        lxi     b,Ø3ØØh         ;Size of partial sector
        lxi     d,ccp+13ØØh     ;Address for sector 3
        lxi     h,ccp+ØfØØh     ;Address of sector 3

wrmcpy: mov     a,m             ;Get a byte and
        stax    d               ;   save it
        inx     d               ;Bump pointers
        inx     h
        dcx     b               ;Bump counter
        mov     a,b             ;Check if done
        ora     c
        jnz     wrmcpy          ;   if not, loop

        lxi     b,ccp+ØfØØh     ;Address for sector 2
        lxi     d,1Ø*1ØØh+2     ;Retry count + sector 2
        call    wrmread

        xra     a               ;Clear error indicator
        ret

wrmread:push    d
        call    fddma           ;Set DMA address
        pop     b
        call    fdsec           ;Set sector
wrmfrd: push    b               ;Save error count
        call    fdread          ;Read a sector
        jc      wrmerr          ;Do retry stuff on error
        call    fdstat          ;Sector size must be 1Ø24 bytes
        ani     Øch             ;Mask length bits
        sui     Øch             ;Carry (error) will be set if < ØcØh
wrmerr: pop     b               ;Fetch retry count
        rnc                     ;Return if no error
        dcr     b               ;Bump error count
        jnz     wrmfrd
        jmp     fderr           ;Error, flash the light

fdtran: inx     b
        push    d               ;Save table address
        push    b               ;Save sector #
        call    fdget           ;Get DPH for current drive
        lxi     d,1Ø            ;Load DPH pointer
        dad     d
        mov     a,m
        inx     h
        mov     h,m
        mov     l,a
        mov     a,m             ;Get # of CP/M sectors/track
        ora     a               ;Clear carry
        rar                     ;Divide by two
        sub     c               ;Subtract sector number
        push    psw             ;Save adjusted sector
        jm      sidetwo
sidea:  pop     psw             ;Discard adjusted sector
        pop     b               ;Restore sector requested
        pop     d               ;Restore address of xlt table
sideone:xchg                    ;hl <- &(translation table)
        dad     b               ;bc = offset into table
        mov     l,m             ;hl <- physical sector
        mvi     h,Ø
        ret

sidetwo:call    fdgsid          ;Check out number of sides
        jz      sidea           ;Single sided
        pop     psw             ;Retrieve adjusted sector
        pop     b
        cma                     ;Make sector request positive
        inr     a
        mov     c,a             ;Make new sector the requested sector
        pop     d
        call    sideone
        mvi     a,8Øh           ;Side two bit
        ora     h               ;       and sector
        mov     h,a
        ret

fdldrv: sta     fdlog           ;Save logical drive
        mov     c,a             ;Save drive #
        mvi     a,Ø             ;Have the floppies been accessed yet ?
flopflg equ     $-1
        ana     a
        jnz     flopok

        mvi     b,17            ;Floppies havn't been accessed
        lxi     h,fdboot        ;Check if 2D controller is installed
        mvi     a,(jmp)
clopp:  cmp     m               ;Must have 17 jumps
        jnz     zret
        inx     h
        inx     h
        inx     h
        dcr     b
        jnz     clopp
        lxi     d,fdinit        ;Initialization sequence
        lxi     h,fdorig+7e2h   ;Load address
        lxi     b,3Ø            ;Byte count
        call    movbyt          ;Load controller RAM
        mvi     a,Øffh          ;Start 1791
        sta     dreg
        mvi     a,clrcmd        ;1791 reset
        sta     cmdreg
        mvi     a,1             ;Set 2D initialized flag
        sta     flopflg

flopok: call    flush           ;Flush buffer since we are using it
        lda     fdlog           ;Select new drive
        mov     c,a
        call    fdsel
        call    fdlhome         ;Recalibrate the drive
        lxi     h,1             ;Select sector 1 of track 2
        shld    truesec
        inx     h
        shld    cpmtrk
        xra     a               ;Make sure we are doing a read
        sta     rdwr
        call    fill            ;Fill in buffer with sector
        jc      zret            ;Test for error return
        call    fdstat          ;Get status on current drive
        sta     fdldst          ;Save drive status
        ani     Øch             ;Mask in sector size bits
        push    psw             ;Used to select a DPB
        rar
        lxi     h,xlts          ;Table of XLT addresses
        mov     e,a
        mvi     d,Ø
        dad     d
        push    h               ;Save pointer to proper XLT
        call    fdget           ;Get pointer to proper DPH
        pop     d
        lxi     b,2             ;Copy XLT pointer into DPH
        call    movbyt
        lxi     d,8             ;Offset to DPB pointer in DPH
        dad     d               ;HL <- &DPH.DPB
        push    h
        call    fdgsid          ;Get pointer to side flag table entry
        lda     fdldst          ;Get drive status
        ani     dblsid          ;Check double sided bit
        mov     m,a             ;Save sides flag
        lxi     d,dpb128s       ;Base for single sided DPB's
        jz      sideok
        lxi     d,dpb128d       ;Base of double sided DPB's
sideok: xchg
        pop     d               ;(HL) -> DPB base, (DE) -> &DPH.DPB
        pop     psw             ;Offset to correct DPB
        ral
        ral                     ;Make Ø, 1Ø, 2Ø, 3Ø
        mov     c,a
        mvi     b,Ø             ;Make offset
        dad     b               ;(hl) is now a DPB pointer
        xchg                    ;Put proper DPB address in DPH.DPB
        mov     m,e
        inx     h
        mov     m,d
        lxi     h,15            ;Offset to DPB.SIZ
        dad     d
        mov     c,m             ;Fetch sector size code
```

```
fdget:  lda     fdlog           ;Return proper DPH
        lxi     d,dphfdØ
        jmp     retdph

fdsel2: sta     fdlog
        mov     c,a
        jmp     fdsel

fdlhome:mvi     c,Ø             ;Select side Ø
        call    fdside
        jmp     fdhome          ;Do actual home

fdssec: push    b               ;Save sector number
        mov     a,b             ;Check side select bit
        rlc                     ;Move high bit to bit zero
        ani     1
        mov     c,a
        call    fdside          ;Call select side Ø = side A, 1 = Side B
        pop     b
        jmp     fdsec

fdgsid: lxi     h,fdlsid        ;Side flag table
        lda     fdlog           ;Drive number
        push    d
        mov     e,a             ;Make offset
        mvi     d,Ø
        dad     d               ;Offset to proper entry
        pop     d
        mov     a,m             ;Set up flags
        ora     a
        ret

fdinit: dw      Ø               ;Initialization bytes loaded onto 2D/B
        dw      18ØØh           ;Head loaded timeout
        dw      Ø               ;DMA address
        db      Ø               ;Double sided flag
        db      Ø               ;Read header flag
        db      Ø7eh            ;Drive select constant
        db      Ø               ;Drive number
        db      8               ;Current disk
        db      Ø               ;Head loaded flag
        db      9               ;Drive Ø parameters
        db      Øffh            ;Drive Ø track address
        db      9               ;Drive 1 parameters
        db      Øffh            ;Drive 1 track address
        db      9               ;Drive 2 parameters
        db      Øffh            ;Drive 2 track address
        db      9               ;Drive 3 parameters
        db      Øffh            ;Drive 3 track address
        db      9               ;Current parameters
        db      Ø               ;Side desired
        db      1               ;Sector desired
        db      Ø               ;Track desired

        db      Ø               ;Header image, track
        db      Ø               ;Sector
        db      Ø               ;Side
        db      Ø               ;Sector
        dw      Ø               ;CRC

fdlog:  db      Ø
fdldst: db      Ø               ;Floppy drive status byte

fdlsid: rept    maxfd
        db      Øffh            ;Double sided flag Ø = single, 1 = double
        endm
        endif

        if      (maxfd ne Ø) or (maxdm ne Ø)

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

xlts:   dw      xlt128          ;Xlt for 128 byte sectors
        dw      xlt256          ;Xlt for 256 byte sectors
        dw      xlt512          ;Xlt for 512 byte sectors
        dw      xlt124          ;Xlt for 1Ø24 byte sectors

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

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

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

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

xlt124: db      Ø
        db      1,2,3,4,5,6,7,8
        db      25,26,27,28,29,3Ø,31,32
        db      49,5Ø,51,52,53,54,55,56
        db      9,1Ø,11,12,13,14,15,16
        db      33,34,35,36,37,38,39,4Ø
        db      57,58,59,6Ø,61,62,63,64
        db      17,18,19,2Ø,21,22,23,24
        db      41,42,43,44,45,46,47,48
```

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

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

dpb128s:dw      26              ;CP/M sectors/track
        db      3               ;BSH
        db      7               ;BLM
        db      0               ;EXM
        dw      242             ;DSM
        dw      63              ;DRM
        db      0c0h            ;AL0
        db      0               ;AL1
        dw      16              ;CKS
        dw      2               ;OFF
        db      1               ;128 byte sectors

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

dpb256s:dw      52              ;CP/M sectors/track
        db      4               ;BSH
        db      15              ;BLM
        db      1               ;EXM
        dw      242             ;DSM
        dw      127             ;DRM
        db      0c0h            ;AL0
        db      0               ;AL1
        dw      32              ;CKS
        dw      2               ;OFF
        db      2               ;256 byte sectors

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

dpb512s:dw      60              ;CP/M sectors/track
        db      4               ;BSH
        db      15              ;BLM
        db      0               ;EXM
        dw      280             ;DSM
        dw      127             ;DRM
        db      0c0h            ;AL0
        db      0               ;AL1
        dw      32              ;CKS
        dw      2               ;OFF
        db      3               ;512 byte sectors

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

dp1Ø24s:dw      64              ;CP/M sectors/track
        db      4               ;BSH
        db      15              ;BLM
        db      Ø               ;EXM
        dw      299             ;DSM
        dw      127             ;DRM
        db      ØcØh            ;ALØ
        db      Ø               ;AL1
        dw      32              ;CKS
        dw      2               ;OFF
        db      4               ;1Ø24 byte sectors

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

dpb128d:dw      52              ;CP/M sectors/track
        db      4               ;BSH
        db      15              ;BLM
        db      1               ;EXM
        dw      242             ;DSM
        dw      127             ;DRM
        db      ØcØh            ;ALØ
        db      Ø               ;AL1
        dw      32              ;CKS
        dw      2               ;OFF
        db      1               ;128 byte sectors

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

dpb256d:dw      1Ø4             ;CP/M sectors/track
        db      4               ;BSH
        db      15              ;BLM
        db      Ø               ;EXM
        dw      486             ;DSM
        dw      255             ;DRM
        db      ØfØh            ;ALØ
        db      Ø               ;AL1
        dw      64              ;CKS
        dw      2               ;OFF
        db      2               ;256 byte sectors

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

dpb512d:dw      12Ø             ;CP/M sectors/track
        db      4               ;BSH
        db      15              ;BLM
        db      Ø               ;EXM
        dw      561             ;DSM
        dw      255             ;DRM
        db      ØfØh            ;ALØ
        db      Ø               ;AL1
        dw      64              ;CKS
        dw      2               ;OFF
        db      3               ;512 byte sectors

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

dp1Ø24d:dw      128             ;CP/M sectors/track
        db      4               ;BSH
        db      15              ;BLM
        db      Ø               ;EXM
        dw      599             ;DSM
        dw      255             ;DRM
        db      ØfØh            ;ALØ
        db      Ø               ;AL1
        dw      64              ;CKS
        dw      2               ;OFF
        db      4               ;1Ø24 byte sectors

        endif

****************************************************************
*                                                              *
* The following equates relate the Morrow Designs DJDMA        *
* controller.                                                  *
*                                                              *
****************************************************************

        if      (maxdm ne Ø) or (maxmf ne Ø)
dmchan  equ     5Øh             ;Default channel address
dmkick  equ     Øefh            ;Kick I/O port address

rdsect  equ     2Øh             ;Read sector command
wrsect  equ     21h             ;Write a sector command
gstat   equ     22h             ;Get drive status
dmsdma  equ     23h             ;Set DMA address
intrqc  equ     24h             ;Set Interrupt request
dmhaltc equ     25h             ;Halt command
bracha  equ     26h             ;Channel branch
setcha  equ     27h             ;Set channel address
setcrc  equ     28h             ;Set CRC retry count
rdtrck  equ     29h             ;Read track command
wrtrck  equ     2ah             ;Write track command
serout  equ     2bh             ;Serial ouput through bit banger serial port
senabl  equ     2ch             ;Enable serial input
trksiz  equ     2dh             ;Set number of tracks
setlog  equ     2eh             ;Set logical drives
readm   equ     ØaØh            ;Read from controller memory
writem  equ     Øa1h            ;Write to controller memory

dmfstp  equ     3*341/1Ø        ;Fast stepping rate constant is 3 ms * 34.1
dmfset  equ     15*341/1Ø       ;Fast settling rate constant is 15 ms * 34.1

n$dubl  equ     8Øh             ;Double density
n$2side equ     4Øh             ;2 sided drive

serin   equ     Ø3eh            ;Address of serial input data, (status - 1)
```

```
****************************************************************
*                                                              *
* Device Specification Table for the Disk Jockey DMA floppy    *
*                                                              *
****************************************************************

        if      maxdm ne Ø
dmdst:  db      maxdm           ;Number of logical drives
        dw      dmwarm          ;Warm boot
        dw      dmtran          ;Sector translation
        dw      dmldrv          ;Select drive 1
        dw      dmselr          ;Select drive 2
        dw      dmhome          ;Home drive
        dw      dmseek          ;Seek to specified track
        dw      dmssec          ;Set sector
        dw      dmdma           ;Set DMA address
        dw      dmread          ;Read a sector
        dw      dmwrite         ;Write a sector
        dw      nobad           ;No bad sector map

dmtrck  equu    22*128          ;Amount of code on track Ø

dmwarm: call    dmselr          ;Select drive Ø
        lxi     h,dmchan        ;Set up branch
        mvi     m,bracha
        inx     h
        mvi     m,(low dmwchn)  ;Low address byte
        inx     h
        mvi     m,(high dmwchn) ;High address byte
        inx     h
        mvi     m,Ø
dmwbad: lxi     h,dmwend-1      ;Pointer to end of command structure
        call    docmd           ;Read in tracks
        lda     dmwst           ;Get track read status
        ani     4Øh
        jz      dmwbad          ;Loop on 'terrible' errors like no disk
        lxi     b,3ØØh          ;3/4 K bytes of sector 3 needs to be moved
        lxi     d,buffer        ;Sector 3 is in our buffer
        lxi     h,ccp+13ØØh     ;  and this is where we want it to go...
        call    movbyt
        xra     a
        ret

dmwchn: db      dmsdma          ;Set track Ø DMA address
        dw      ccp-512         ;First track DMA address - boot loader
        db      Ø
        db      rdtrck          ;Read track command
        db      Ø               ;Track Ø
        db      Ø               ;Side Ø
        db      Ø               ;Drive Ø
        dw      dmwsec          ;Sector load/status map
        db      Ø
dmwst:  db      Ø               ;Track read status
        db      dmsdma
        dw      ccp+dmtrck      ;DMA address for track 1
        db      Ø
        db      rdtrck
        db      1               ;Track 1
        db      Ø               ;Side Ø
        db      Ø               ;Drive Ø
        dw      dmwsec+26       ;Map is loaded right after track Ø status map
        db      Ø
        db      Ø               ;Track read status
        db      dmsdma
        dw      buffer          ;Sector 3 gets loaded in system buffer
        db      Ø
        db      rdsect
        db      1               ;Track 1
        db      3               ;Side Ø, sector 3
        db      Ø               ;Drive Ø
dmwend: db      Ø               ;Read status
        dw      Ø               ;Room for the halt

dmwsec: dw      Øffffh, Øffffh                  ;Do not load boot loader
        dw      Ø, Ø, Ø, Ø, Ø, Ø, Ø, Ø, Ø, Ø, Ø ;22 sectors to be loaded
        dw      Ø, Øffffh, Øffffh, Øffffh       ;First 2 sectors on track 2

dmselr: sta     dmlog
        mvi     b,Ø             ;8 inch logical drives start at zero
        jmp     dmsel2

dmtran: inx     b
        push    d               ;Save table address
        push    b               ;Save sector #
        call    dmget
        lxi     d,1Ø
        dad     d
        mov     a,m
        inx     h
        mov     h,m
        mov     l,a
        mov     a,m             ;Get # of CP/M sectors/track
        ora     a               ;Clear cary
        rar                     ;Divide by two
        sub     c
        push    psw             ;Save adjusted sector
        jm      dmside2
dmsidea:pop     psw             ;Discard adjusted sector
        pop     b               ;Restore sector requested
        pop     d               ;Restor address of xlt table
dmsidel:xchg                    ;hl <- &(translation table)
        dad     b               ;bc = offset into table
        mov     l,m             ;hl <- physical sector
        mvi     h,Ø
        ret

dmside2:call    dmstat
        ani     2Øh
        jz      dmsidea
        pop     psw             ;Retrieve adjusted sector
        pop     b
        cma                     ;Make sector request positive
        inr     a
        mov     c,a             ;Make new sector the requested sector
        pop     d
        call    dmsidel
        mvi     a,8Øh           ;Side two bit
        ora     h               ;       and sector
        mov     h,a
        ret

dmldrv: sta     dmlog
        call    dminit          ;Test for a drive
        jc      zret
        lxi     h,1             ;Select sector 1 of track 2
        shld    truesec
        inx     h
        shld    cpmtrk
        xra     a               ;Make sure we are doing a read
        sta     rdwr
        call    fill            ;Flush buffer and refill
        jc      zret            ;Test for error return
        call    dmstat          ;Get status on current drive
        ani     Øch             ;Mask in sector size bits
        push    psw             ;Used to select a DPB
        rar
        lxi     h,xlts          ;Table of XLT addresses
        mov     e,a
        mvi     d,Ø
        dad     d
        push    h               ;Save pointer to proper XLT
        call    dmget
        pop     d
        lxi     b,2             ;Number of bytes to move
        call    movbyt          ;Move the address of XLT
        lxi     d,8             ;Offset to DPB pointer
        dad     d               ;HL <- &DPH.DPB
        push    h
        call    dmstat
        ani     2Øh             ;Check double sided bit
        lxi     d,dpb128s       ;Base for single sided DPB's
        jz      dmsok
        call    sethigh         ;Set controller to know about fast steping
        lxi     d,dpb128d       ;Base of double sided DPB's
dmsok:  xchg                    ;HL <- DBP base, DE <- &DPH.DPB
        pop     d               ;Restore DE (pointer into DPH)
        pop     psw             ;Offset to correct DPB
        ral
        ral
        mov     c,a
        mvi     b,Ø
        dad     b
        xchg                    ;Put DPB address in DPH
        mov     m,e
        inx     h
        mov     m,d
        lxi     h,15
        dad     d
        mov     c,m
dmget:  lda     dmlog
        lxi     d,dphdmØ
        jmp     retdph

;
;       The current drive is double sided.  Thus is it safe to set the
;       stepping rate to 3 ms with 15 ms settling.
;

sethigh:lhld    dmlog                   ;Get the current drive number
        mvi     h,Ø                     ;Drive number is a byte
        dad     h                       ;Ten bytes per parameter table entry
        mov     d,h
        mov     e,l
        dad     h
        dad     h
        dad     d
        lxi     d,dparam+1              ;Parameter table address
        dad     d                       ;Skip the track size byte
        mvi     m,Ø                     ;Force reparamitization of this drive
        inx     h                       ;Offset to the Stepping rate constant
        mvi     m,(low dmfstp)          ;Fast stepping rate constant
        inx     h
        mvi     m,(high dmfstp)
        lxi     d,5                     ;Skip over the reserved fields
        dad     d
        mvi     m,(low dmfset)          ;Fast settling rate constant
        inx     h
        mvi     m,(high dmfset)
        call    dmparm                  ;Set drive parameters for the SA85Ø
        ret

        endif

*****************************************************************
*                                                               *
* Drive specification table for DJDMA 5 1/4 inch drives         *
*                                                               *
*****************************************************************

        if      maxmf ne Ø
mfdst:  db      maxmf           ;Number of logical drives
        dw      mfwarm          ;Warm boot
        dw      mftran          ;Sector translation
        dw      mfldrv          ;Select drive 1
        dw      mfsel2          ;Select drive 2
        dw      dmhome          ;Home drive
        dw      mfseek          ;Seek to specified track
        dw      mfssec          ;Set sector
        dw      dmdma           ;Set DMA address
        dw      dmread          ;Read a sector
        dw      dmwrite         ;Write a sector
        dw      nobad           ;No bad sector map

mftrck  equ     9*512           ;Amount of code on track Ø

mfwarm: call    mfsel2          ;Select drive Ø
        lxi     h,dmchan        ;Set up branch
        mvi     m,bracha
        inx     h
        mvi     m,(low mfwchn)  ;Low address byte
        inx     h
        mvi     m,(high mfwchn) ;High address byte
        inx     h
        mvi     m,Ø
mfwfal: lxi     h,mfwend-1      ;Pointer to end of command structure
        call    docmd           ;Read in tracks
        lda     mfwst           ;Check out drive status
        ani     4Øh             ;Test for ok
        jz      mfwfal          ;Failed, loop
        xra     a               ;Return no error
        ret

mfwchn: db      dmsdma          ;Set track Ø DMA address
        dw      ccp-512         ;First track DMA address - boot loader
        db      Ø
        db      rdtrck          ;Read track command
        db      Ø               ;Track Ø
        db      Ø               ;Side Ø
        db      Ø               ;Drive Ø
        dw      mfwsec          ;Sector load/status map
        db      Ø
mfwst   db      Ø               ;Track read status
        db      dmsdma
        dw      ccp+mftrck      ;DMA address for track 1
        db      Ø
        db      rdtrck
        db      1               ;Track 1
        db      Ø               ;Side Ø
        db      Ø               ;Drive Ø
        dw      mfwsec+1Ø       ;Map is loaded right after track Ø status map
        db      Ø
mfwend: db      Ø               ;Track read status
        dw      Ø               ;Room for the halt
```

```
mfwsec: dw      Øffh, Ø, Ø, Ø, Ø        ;Do not load boot loader
        dw      Ø, Øffffh, Øffffh, Øffffh, Øffffh ;first two sectors loaded

mfssec: dcr     c               ;Minnie floppy sectors start at zero
        lda     dblflg          ;Get double sided flags
        ora     a
        jz      dmssec          ;Nope, single sided
        mvi     b,8Øh           ;Set high bit for double sided select
        jmp     dmssec

dblflg: db      Ø

mfseek: xra     a               ;Clear double sided select
        sta     dblflg
        lda     mfpcon
        ani     n$2side
        jz      dmseek          ;Only single sided

        mov     a,c             ;Move selected track in (a)
        sbi     35              ;Subtract by track by number of tracks
        jc      dmseek          ;Less than track 35

        mov     d,a             ;Save adjusted track number
        mvi     a,34
        sub     d               ;Adjust to count tracks back out
        mov     c,a             ;Resave new track number

        mvi     a,Øffh          ;Set double sided flag
        sta     dblflg
        jmp     dmseek

mfsel2: sta     mflog

        mov     c,a             ;Get proper physical configuration byte
        mvi     b,Ø
        lxi     h,mfscon
        dad     b
        mov     a,m
        sta     mfpcon
        mov     a,c             ;Shhh, pretend that nothing happened

        mvi     b,4             ;5 1/4 inch drives start at drive 4
        jmp     dmsel2

mftran: lda     mfpcon
        ani     n$dubl
        lxi     h,mfxltd        ;Point to double sided sector translation table
        jnz     mftdubl         ;Single density sector translation
        lxi     h,mfxlts
mftdubl:dad     b               ;Add offset sector number to table
        mov     l,m             ;Pick up sector number from table
        mvi     h,Ø             ;MSB of sector number equal Ø
        ret

mfldrv: sta     mflog
        call    dminit          ;Test for a controller
        jc      zret

        lda     mflog           ;Get proper physical configuration byte
        mov     c,a
        mvi     b,Ø
        lxi     h,mfscon
        dad     b
        mvi     a,n$dubl
        mov     m,a
        sta     mfpcon

        lxi     h,1             ;Select sector 1 of track Ø
        shld    truesec
        dcx     h
        shld    cpmtrk
        xra     a               ;Make sure we are doing a read
        sta     rdwr
        call    fill            ;Flush buffer and refill
        jc      zret            ;Test for error return
        lda     buffer+5ch      ;Get diskette configuration byte

        push    psw             ;Save configuration byte
        lxi     h,1
        shld    cpmtrk          ;Load track 1 sector 1
        call    fill            ;This is to fix bug with DJDMA firmware on
        jc      zret            ; returning single density status on track Ø
        pop     psw

        ora     a
        jnz     mfl9            ;Non zero

        mvi     a,9Øh           ;Double density default configuration
        call    dmstat          ;If zero then determine sector size
        ani     8Øh             ;Check density bit
        jnz     mfl9            ;Its double density
        mvi     a,1Øh           ;Single density default configuration byte

mfl9:   mov     c,a             ;Move configuration byte into (c)

        lxi     h,mfs           ;Address of configuration table -> (hl)
mfl2:   mov     a,m             ;Get an entry
        ora     a               ;Check for end of the table
        jz      zret            ;Yes, select error
        cmp     c               ;Check if entry matches selected drive
        jz      mfl3
        inx     h               ;Skip onfiguration byte
        inx     h               ;Skip drive type
        inx     h               ;Skip DPB address
        inx     h
        jmp     mfl2

mfl3:   inx     h
        mov     a,m             ;Pick up drive type
        sta     mfpcon
        mov     e,a

        push    h
        lda     mflog           ;Get proper physical configuration byte
        mov     c,a
        mvi     b,Ø
        lxi     h,mfscon
        dad     b
        mov     m,e
        pop     h

        inx     h
        mov     a,m
        inx     h
        mov     h,m
        mov     l,a             ;DPB address -> (hl)
        push    h               ;Save DPB address
        call    mfgdph          ;Get DPH
        lxi     d,1Ø            ;Offset to DPB address in DPH
        dad     d
        pop     d
        mov     m,e             ;Store DPB address in DPH
        inx     h
        mov     m,d
        call    mfgdph
        push    h
        call    dmstat          ;Get status
        pop     h
        ani     8Øh             ;Check density bit
        mvi     c,3             ;512 byte sectors
        rnz
        mvi     c,2             ;256 byte sectors
        ret

mfgdph  lda     mflog
        lxi     d,dphmfØ
        jmp     retdph

mfpcon: db      Ø               ;Physical configuration byte
mflog:  db      Ø

mfscon: db      Ø, Ø, Ø, Ø      ;Saved physical configuration bytes

mfs:    db      1Øh             ;North Star CP/M 1.4
        db      Ø               ;Single density, 35 tracks, single sided
        dw      dpbmfØ          ;1K groups

        db      9Øh             ;North Star CP/M 1.4
        db      n$dubl          ;Double density, 35 tracks, single sided
        dw      dpbmf1          ;1K groups

        db      ØbØh            ;North Star CP/M 2.x
        db      n$dubl          ;Double density, 35 tracks, single sided
        dw      dpbmf2          ;2K groups

        db      ØfØh            ;North Star CP/M 2.x
        db      n$dubl+n$2side  ;Double density, 35 tracks, double sided
        dw      dpbmf3          ;2K groups

        db      Øe5h            ;North Star CP/M 1.4
        db      n$dubl          ;Double density, 35 tracks, single sided
        dw      dpbmf1          ;1K groups

        db      ØaØh            ;North Star CP/M 2.x  (fake 4Ø track)
        db      n$dubl          ;Double density, 35 tracks, single sided
        dw      dpbmf2          ;2K groups

        db      ØdØh            ;North Star CP/M 2.x (fake 4Ø track)
        db      n$dubl+n$2side  ;Double density, 35 tracks, double sided
        dw      dpbmf3          ;2K groups

        db      Ø               ;End of configuration table

mfxltd  db       1, 2, 3, 4
        db      21,22,23,24
        db       5, 6, 7, 8
        db      25,26,27,28
        db       9,1Ø,11,12
        db      29,3Ø,31,32
        db      13,14,15,16
        db      33,34,35,36
        db      17,18,19,2Ø
        db      37,38,39,4Ø

mfxlts  db       1, 2
        db       3, 4
        db       5, 6
        db       7, 8
        db       9,1Ø
        db      11,12
        db      13,14
        db      15,16
        db      17,18
        db      19,2Ø
        endif

*****************************************************************
*                                                               *
* Common routines for the DJDMA with 8 and 5 1/4 inch drives    *
*                                                               *
*****************************************************************

dmsel2: mov     c,a             ;Move drive into (c)
        lxi     h,dmchan
        mvi     m,setlog        ;Set logical drives
        inx     h
        mov     m,b             ;Drive in (b)
        push    b
        call    docmd
        pop     b
        jmp     dmsel

dmssec: push    b               ;Save sector number
        mov     a,b
        rlc
        ani     1
        mov     c,a
        call    dmside
        pop     b
        jmp     dmsec

dmdma   lxi     h,dmchan        ;Default channel address
        mvi     m,dmsdma        ;Set DMA address
        inx     h
        mov     m,c             ;Low byte first
        inx     h
        mov     m,b             ;High byte next

docmd   xra     a
        inx     h
        mov     m,a
docmd2  inx     h
        mvi     m,dmhaltc
        inx     h
        mov     m,a
        out     dmkick
tests   ora     m
        jz      tests
        ret

dminit: lxi     h,dmchan        ;See if controller will halt
        mvi     m,dmhaltc
        inx     h
        mvi     m,Ø
        out     dmkick          ;Start controller
        lxi     d,Ø             ;Set up timeout counter
dminwt  mov     a,m
        ora     a
        jnz     dmiok           ;Controller has responded
        dcx     d               ;Bump timeout counter
        mov     a,d
        ora     e
        jnz     dminwt
        stc                     ;Set error flag
        ret

dmiok   push    h               ;Set drive parameters
        call    dmparm
        pop     h
        dcx     h               ;Back to start of command
        mvi     m,setcrc        ;Set CRC error retry count to one
        inx     h
        mvi     m,1
        xra     a
        jmp     docmd2          ;Do command

;
;       Set floppy drive parameters
;
;       This routine reads the dparam table and if the a drive has not
;       previously been calibrated then that drives track count,
;       stepping rate, and head settling time are loaded.
;

dmparm: mvi     a,8                     ;Eight drives
        lxi     d,134Øh                 ;Start with drive Ø's table
        lxi     h,dparam+1              ;Drive parameter table

dmstrØ: push    psw                     ;Save the drive count
        mov     a,m                     ;Load flags
        ora     a                       ;Does the drive need to be calibrated?
        jnz     dmstr1                  ;No, do not fiddle around
        push    h                       ;Save the parameter table pointer
        push    d                       ;Save the controllers table pointer
        dcr     m                       ;Set to calibrated mode (Øffh)
        dcx     h                       ;Back up to the track size byte
        shld    dmntrk                  ;Set the number of tracks pointer
        inx     h
        inx     h
        shld    dmspar                  ;Set the stepping constants pointer
        xchg                            ;Set the local parameter table pointer
        shld    dmlocØ
        inx     h                       ;Offset to the stepping parameters
        inx     h
        inx     h
        inx     h
        shld    dmloc1
        lxi     h,dmwcon                ;Write the drive constants out
        lxi     d,17                    ;Halt status offset
        call    dmdoit
        pop     d                       ;Retrieve the table pointers
        pop     h

dmstr1: lxi     b,1Ø                    ;Bump parameter table pointer
        dad     b
        xchg
        lxi     b,16                    ;Bump controller tables pointer
        dad     b
        xchg

        pop     psw                     ;Retrieve drive count
        dcr     a                       ;Bump count
        jnz     dmstrØ                  ;Set up next drive

        ret

dmhome  xra     a
        mov     c,a             ;Put a zero into (c) for track zero
```

```
dmseek  mov     a,c             ;Enter with track in (c)
        sta     lltrk           ;Save for use later
        ret

dmsec   lda     llss            ;Load sector
        ani     8Øh             ;Save side select bit
stores  ora     c
        sta     llss
        ret

dmside: mov     a,c             ;Move side bit into (a)
        ani     1
        rrc                     ;Move around to bit 7
        mov     c,a             ;Resave in (c)
        lda     llss
        ani     7fh             ;Mask out old side select bit
        jmp     stores

dmsel:  mov     a,c             ;Move drive into (a)
        sta     lldrv
dmden:  ret                     ;Double density only

;
; Return status in the (a) register in the form:
;
;                       7  6  5  5  3  2  1  Ø
;                       ^  ^  ^  ^  ^  ^  ^  ^
; Density --------------+  |  |  |  |  |  |  |
; Side select -------------+  |  |  |  |  |  |
; Double sided ---------------+  |  |  |  |  |
; 5 1/4 -------------------------+  |  |  |  |
; Sector size MSB ------------------+  |  |  |
; Sector size LSB ---------------------+  |  |
; Drive select MSB -----------------------+  |
; Drive select LSB --------------------------+
;

dmstat  lxi     h,dmchan
        mvi     m,gstat         ;Set up read status
        inx     h
        lda     lldrv           ;Get last selected drive
        mov     m,a             ;Store drive in command
        inx     h               ;Skip over returned status
        inx     h
        inx     h
        call    docmd           ;Issue command
        lda     llss            ;Get side bit of last operation
        ani     8Øh
        rrc                     ;Move to bit 7
        mov     c,a
        lxi     h,dmchan+1      ;Point to drive
        mov     a,m             ;Load drive
        ora     c
        ani     4               ;Mask upper drive select bit for 5 1/4
        rlc
        rlc                     ;Move to bit 4
        ora     m               ;Put together with lower drive bits
        ora     c
        mov     c,a
        inx     h
        mvi     a,1Øh           ;Double density bit
        ana     m
        rlc                     ;2Øh
        rlc                     ;4Øh
        rlc                     ;8Øh for density bit
        ora     c
        mov     c,a
        inx     h
        mvi     a,3             ;Sector length mask
        ana     m               ;And in
        rlc                     ;Move to bits 2 & 3
        rlc
        ora     c
        mov     c,a
        inx     h
        mvi     a,4             ;Mask for double sided bit
        ana     m
        rlc                     ;8
        rlc                     ;1Ø
        rlc                     ;2Ø
        ora     c
        ret

dmwrite mvi     a,wrsect
        db      Ø1              ;Ugh...
dmread  mvi     a,rdsect
        lxi     h,dmchan
        lxi     d,lltrk-1
        mvi     b,4
cload   mov     m,a
        inx     h
        inx     d
        ldax    d
        dcr     b
        jnz     cload
        dcx     h
        call    docmd
        lda     dmchan+4
        cpi     8Øh
        cmc
        ret

;
;       Execute a DJDMA command, no command status is returned
;
;       Entry:
;               DE = offset to the halt status
;               HL = pointer to the start of the command
;
;       Returns:
;               nothing
;

dmdoit: mvi     a,bracha        ;Branch channel command
        sta     dmchan
        shld    dmchan+1        ;Load command vector
        xra     a               ;Clear extended address
        sta     dmchan+3

        dad     d               ;Offset to the halt status
        mov     m,a             ;Clear the halt status indicator

        out     dmkick          ;Start the controller

dmwait: ora     m               ;Wait for the operation complete status
        jz      dmwait

        ret

dmwcon: db      writem                  ;Write track size
dmntrk: dw      Ø                       ;Number of tracks + desync
        db      Ø                       ;X-address
        dw      2                       ;Two bytes
dmlocØ: dw      Ø                       ;Local controller address

        db      writem                  ;Write stepping rate data
dmspar: dw      Ø                       ;Pointer to the stepping parameters
        db      Ø
        dw      8
dmlocl: dw      Ø

        db      dmhaltc                 ;Controller halt
        db      Ø                       ;Status

;
;       Driver variables
;

lltrk   db      Ø
llss    db      1
lldrv   db      Ø
dmlog   db      Ø

        endif

*****************************************************************
*                                                               *
* The follwing equates are for the HDDMA hard disk controller   *
*                                                               *
*****************************************************************

        if      maxmw ne Ø      ;HDDMA controller present ?
        if      st5Ø6           ;Specifications for a Seagate Technology 5Ø6
cyl     equ     153             ;Number of cylinders
heads   equ     4               ;Number of heads per cylinder
precomp equ     64              ;Cylinder to start write precomensation
lowcurr equ     128             ;Cylinder to start low current
stepdly equ     3Ø              ;Step delay (Ø-12.7 milliseconds)
steprcl equ     3Ø              ;Recalibrate step delay
headdly equ     Ø               ;Settle delay (Ø-25.5 milliseconds)
        endif

        if      st412           ;Specifications for a Seagate ST412
cyl     equ     3Ø6
heads   equ     4
precomp equ     128
lowcurr equ     128
stepdly equ     Ø
steprcl equ     3Ø
headdly equ     Ø
        endif

        if      cm5619          ;Specifications for an CMI 5619
cyl     equ     3Ø6
heads   equ     6
precomp equ     128
lowcurr equ     128
stepdly equ     2
steprcl equ     3Ø
headdly equ     Ø
        endif

sectsiz equ     7               ;Sector size code (must be 7 for this Cbios)
                                ; Ø =  128 byte sectors
                                ; 1 =  256 byte sectors
                                ; 3 =  512 byte sectors
                                ; 7 = 1Ø24 byte sectors (default)
                                ; f = 2Ø48 byte sectors
                                ;Define controller commands
dmaread equ     Ø               ;Read sector
dmawrit equ     1               ;Write sector
dmarhed equ     2               ;Find a sector
dmawhed equ     3               ;Write headers (format a track)
dmalcon equ     4               ;Load disk parameters
dmassta equ     5               ;Sense disk drive status
dmanoop equ     6               ;Null controller operation

reset   equ     54h             ;Reset controller
attn    equ     55h             ;Send a controller attention

chan    equ     5Øh             ;Default channel address
stepout equ     1Øh             ;Step direction out
stepin  equ     Ø               ;Step direction in
band1   equ     4Øh             ;No precomp, high current
band2   equ     ØcØh            ;Precomp, high current
band3   equ     8Øh             ;precomp, low current
trackØ  equ     1               ;Track zero status
wflt    equ     2               ;Write fault from drive
dready  equ     4               ;Drive ready
sekcmp  equ     8               ;Seek complete

*****************************************************************
*                                                               *
* Drive Specification Table for the HD DMA hard disk controller *
*                                                               *
*****************************************************************

mwdst:  db      maxmw*mwlog     ;Number of logical drives
        dw      mwwarm          ;Warm boot
        dw      mwtran          ;Sector translation
        dw      mwldrv          ;Select logical drive 1 (First time select)
        dw      mwdrv           ;Select logical drive 2 (General select)
        dw      mwhome          ;Home current selected drive
        dw      mwseek          ;Seek to selected track
        dw      mwsec           ;Select sector
        dw      mwdma           ;Set DMA address
        dw      mwread          ;Read a sector
        dw      mwwrite         ;Write a sector
        if      heads > 2       ;Test if drive is big enough for a bad spot map
        dw      mwbad           ;Return bad sector map info
        else
        dw      nobad
        endif

*****************************************************************
*                                                               *
* The following are the lowest level drivers for the Morrow     *
* Designs Hard Disk DMA controller.                             *
*                                                               *
*****************************************************************

mwwarm  xra     a
        call    mwdrv           ;Select drive A
        call    mwhome          ;Home and reset the drive
        lxi     b,Ø             ;Make sure we are on track Ø
        call    mwseek
        xra     a
        sta     mwhead          ;Select head zero
        sta     mwsectr         ;Select sector 1
        lxi     h,buffer        ;Load sector 1 into buffer
        shld    dmadma
        call    mwread          ;Read CCP into buffer
        rc                      ;Return if error
        lxi     d,buffer+2ØØh
        lxi     h,ccp
        lxi     b,2ØØh          ;Move 2ØØh bytes
        call    movbyt
        lxi     h,ccp-2ØØh      ;Initial DMA address
        push    h
        xra     a
        push    a               ;Save first sector -1
mwwlod  pop     psw             ;Restore sector
        pop     h               ;Restore DMA address
        inr     a
        sta     mwsectr
        cpi     6               ;Past BDOS ?
        rz                      ;Yes, all done
        inr     h               ;Update DMA address by 1Ø24 bytes
        inr     h
        inr     h
        inr     h
        shld    dmadma
        push    h
        push    psw
        call    mwwread         ;Read in a sector
        jnc     mwwlod
        ret                     ;Return with error

mwwread mvi     c,retries       ;Retry counter
mwwerr  push    b               ;Save the retry count
        call    mwread          ;Read the sector
        pop     b
        rnc
        dcr     c               ;Update the error count
        jnz     mwwerr          ;Keep trying if not too many errors
        stc                     ;Set error flag
        ret

mwldrv  sta     mwcurl          ;Save current logical drive
        call    mwreset         ;Reset controller card
        jc      zret            ;Controller failure

        lda     mwcurl
        call    mwdrv           ;Select drive
        jc      zret            ;Select error

        call    mwstat          ;Get drive status
        ani     dready          ;Check if drive ready
        jnz     zret

        call    mwhome          ;Home drive

        lxi     d,dphmwØ        ;Start of hard disk DPH's
        lda     mwcurl
        mov     l,a
        mvi     h,Ø
        dad     h
        dad     h
        dad     h
        dad     h
        dad     d               ;(hl) = pointer to DPH
        mvi     c,4             ;Return sector size of 1Ø24
        ret

mwdrv   sta     mwcurl
        call    mwdlog
        mov     a,c
        sta     mwdrive         ;Save new selected drive
mwsel   mvi     a,dmanoop
```

```
        jmp     mwprep          ;Execute disk command

mwdlog: mvi     c,Ø
mwllx:  sui     mwlog
        rc
        inr     c
        jmp     mwllx

mwstat  mvi     a,dmassta       ;Sense status operation code
        jmp     mwprep          ;Execute disk command

mwhome  call    mwreset         ;Reset controller, do a load constants
        lxi     h,dmargl        ;Load arguments
        mvi     m,stepracl      ;Load step delay (slow rate)
        inx     h
        mvi     m,headdly       ;Head settle delay
        call    mwissue         ;Do load constants again
        call    mwptr           ;Get pointer to current cylinder number
        mvi     m,Øffh          ;Fake at cylinder 65535 for max head travel
        inx     h
        mvi     m,Øffh
        lxi     b,Ø             ;Seek to cylinder Ø
        call    mwseek          ;Recal slowly
        jmp     mwreset         ;Back to fast stepping mode

mwbad:  lxi     h,mwbtab        ;Return pointer to bad sector location
        ret

mwbtab: dw      Ø               ;Track Ø
        dw      19              ;Head 2, sector Ø  = (2 * SPT + Ø) + 1

mwseek  call    mwptr           ;Get track pointer
        mov     e,m             ;Get old track number
        inx     h
        mov     d,m
        dcx     h
        mov     m,c             ;Store new track number
        inx     h
        mov     m,b
        mov     l,c             ;Build cylinder word
        mov     h,b
        shld    dmargØ          ;Set command channel cylinder number
        mov     a,d
        inr     a
        lxi     h,Øffffh
        jnz     mwskipØ
        mvi     c,stepout
        jmp     mwskip

mwskipØ:mov     h,b             ;(hl) = new track, (de) = old track
        mov     l,c
        call    mwhlmde
        mvi     c,stepout
        mov     a,h
        ani     8Øh             ;Check hit bit for negitive direction
        jnz     mwsout          ;Step in
        mvi     c,Ø
        jmp     mwskip
mwsout: call    mwneghl
mwskip: shld    dmastep
        lda     mwdrive
        ora     c
        sta     dmaselØ

        mvi     a,dmanoop       ;No-operation command for the channel
        call    mwprep          ;Step to proper track
        lxi     h,Ø             ;Clear step counter
        shld    dmastep
        ret

mwdma   mov     h,b             ;Set DMA address
        mov     l,c
        shld    dmadma
        ret

mwsec   mov     a,c             ;Load sector number
        dcr     a               ;Range is actaully Ø-16
        call    mwdspt          ;Figure out head number -> (c)
        adi     mwspt           ;Make sector number
        sta     mwsectr
        mov     a,c
        sta     mwhead          ;Save head number
        ret

mwdspt  mvi     c,Ø             ;Clear head counter
mwdsptx sui     mwspt           ;Subtract a tracks worth of sectors
        rc                      ;Return if all done
        inr     c               ;Bump to next head
        jmp     mwdsptx

mwreset lhld    chan            ;Save the command channel for a while
        shld    tempb
        lda     chan+2
        sta     tempb+2
        out     reset           ;Send reset pulse to controller
        lxi     h,dmachan       ;Address of command channel
        shld    chan            ;Default channel address
        xra     a
        sta     chan+2          ;Clear extended address byte
        shld    4Øh             ;Set up a pointer to the command channel
        sta     42h
        lhld    dmargØ          ;Save the track number
        push    h
        lxi     h,dmasell       ;Load arguments
        lda     mwdrive         ;Get the currently selected drive
        ori     Ø3ch            ;Raise *step and *dir
        mov     m,a             ;Save in drive select register
        lxi     d,5             ;Offset to dmargl
        dad     d
        mvi     m,stepdly       ;Load step delay
        inx     h
        mvi     m,headdly       ;Head settle delay
        inx     h
        mvi     m,sectsiz       ;Sector size code
        inx     h
        mvi     m,dmalcon       ;Load constants command
        call    mwissue         ;Do load constants
        pop     h               ;Restore the track number
        shld    dmargØ
        push    psw             ;Save status
        lhld    tempb           ;Restore memory used for the channel pointer
        shld    chan
        lda     tempb+2
        sta     chan+2
        pop     psw
        ret

mwread  mvi     a,dmaread       ;Load disk read command
        jmp     mwprep

mwwrite mvi     a,dmawrit       ;Load disk write command
```

```
mwprep: sta     dmaop           ;Save command channel op code

        mvi     c,band1
        lhld    dmargØ
        lxi     d,precomp
        call    mwhlcde
        jc      mwpreps

        mvi     c,band2
        lxi     d,lowcurr
        call    mwhlcde
        jc      mwpreps

        mvi     c,band3         ;cylinder > low_current
mwpreps lda     mwhead          ;Load head address
        sta     dmarg2
        cma                     ;Negative logic for the controller
        ani     7               ;3 bits of head select
        rlc                     ;Shove over to bits 2 - 4
        rlc
        ora     c               ;Add on low current and precomp bits
        mov     c,a
        lda     mwdrive         ;Load drive address
        ora     c               ;Slap in drive bits
        sta     dmasel1         ;Save in command channel head select
        lda     mwsectr         ;Load sector address
        sta     dmarg3

        if      Ø               ;Set to 1 for MW error reporter
mwissue call    mwdoit          ;Do desired operation
        rnc                     ;Do nothing if no error
        push    psw             ;Save error info
        call    hexout          ;Print status
        call    dspout          ;   and a space
        lxi     h,dmachan
        mvi     c,16            ;16 bytes of status
mwerr:  push    b
        push    h
        mov     a,m
        call    hexout          ;Print a byte of the status line
        call    spout
        pop     h
        pop     b
        inx     h               ;Bump command channel pointer
        dcr     c
        jnz     mwerr
        mvi     c,Øah           ;Terminate with a CRLF
        call    pout
        mvi     c,Ødh
        call    pout
        pop     psw             ;Restore error status
        ret

dspout: call    spout           ;Print two spaces
spout:  mvi     c,' '           ;Print a space
        jmp     pout

hexout: push    psw             ;Poor persons number printer
        rrc
        rrc
        rrc
        rrc
        call    nibout
        pop     psw
nibout: ani     Øfh
        adi     'Ø'
        cpi     '9'+1
        jc      nibok
        adi     27h
nibok:  mov     c,a
        jmp     pout

mwdoit  equ     $

        else

mwissue equ     $               ;Do a disk command, handle timeouts + errors

        endif

        lxi     h,dmastat       ;Clear status byte
        mvi     m,Ø
        out     attn            ;Start the controller
        lxi     d,Ø             ;Time out counter (65536 retries)
mwiloop mov     a,m             ;Get status
        ora     a               ;Set up CPU flags
        rm                      ;Return no error (carry reset)
        stc
        rnz                     ;Return error status
        xthl                    ;Waste some time
        xthl
        xthl
        xthl
        dcx     d               ;Bump timeout counter
        mov     a,d
        ora     e
        jnz     mwiloop         ;Loop if still busy
        stc                     ;Set error flag
        ret

mwptr   lda     mwdrive         ;Get currently select drives track address
        rlc
        mov     e,a
        mvi     d,Ø
        lxi     h,mwtab
        dad     d               ;Offset into track table
        ret

mwtran: mov     h,b
        mov     l,c
        inx     h
        ret

mwneghl:mov     a,h
        cma
        mov     h,a
        mov     a,l
        cma
        mov     l,a
        inx     h
        ret

mwhlmde:xchg
        call    mwneghl
        xchg
        dad     d
        ret

mwhlcde:mov     a,h
        cmp     d
        rnz
        mov     a,l
        cmp     e
        ret

mwtab   equ     $               ;Collection of track addresses
        rept    maxmw
        db      Øffh            ;Initialize to (way out on the end of the disk)
        db      Øffh
        endm
        db      Øffh

mwcurl  db      Ø               ;Current logical drive
mwdrive db      Øffh            ;Currently selected drive
mwhead  db      Ø               ;Currently selected head
mwsectr db      Ø               ;Currently selected sector

dmachan equ     $               ;Command channel area
dmaselØ db      Ø               ;Drive select
dmastep dw      Ø               ;Relative step counter
dmasel1 db      Ø               ;Head select
dmadma  dw      Ø               ;DMA address
        db      Ø               ;Extended address
dmargØ  db      Ø               ;First argument
dmarg1  db      Ø               ;Second argument
dmarg2  db      Ø               ;Third argument
dmarg3  db      Ø               ;Fourth argument
dmaop   db      Ø               ;Operation code
dmastat db      Ø               ;Controller status byte
dmalnk  dw      dmachan         ;Link address to next command channel
        db      Ø               ;extended address

        endif

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

        if      nostand ne Ø    ;Unallocated writting variables
unaloc: db      Ø               ;Unallocated write in progress flag
oblock: dw      Ø               ;Last unallocated block number written
unadrv: db      Ø               ;Drive that the block belongs to
        endif

cpmsec: dw      Ø               ;CP/M sector #

cpmdrv: db      Ø               ;CP/M drive #
cpmtrk: dw      Ø               ;CP/M track #
truesec:dw      Ø               ;Physical sector that contains CP/M sector

error:  db      Ø               ;Buffer's error status flag
bufdrv: db      Ø               ;Drive that buffer belongs to
buftrk: dw      Ø               ;Track that buffer belongs to
bufsec: dw      Ø               ;Sector that buffer belongs to

alttrk: dw      Ø               ;Alternate track
altsec: dw      Ø               ;Alterante sector
lastdrv:db      Ø               ;Last selected drive

*****************************************************************
*                                                               *
* DPB and DPH area.                                             *
*                                                               *
*****************************************************************

        if      maxhd ne Ø
```

```
dphdsk  set     Ø               ;Generate DPH's for the HDCA hard disks
        rept    maxhd
ldsk    set     Ø
        rept    hdlog
        dphgen  hd,%dphdsk,dpbhd,%ldsk
ldsk    set     ldsk+1
dphdsk  set     dphdsk+1
        endm
        endm

        if      hdpart ne Ø     ;Use non-standard partitioning

*****************************************************************
*                                                               *
* hdsectp is the number of 128 byte sectors per cylinder.       *
*                                                               *
* hdtrks is the total number of data cylinders.  Eg.  it is     *
* the number of cyliders on the drive minus the number of       *
* cylinders that are used for the system.  If the number of     *
* 'system tracks' is not one then the initial value of          *
* 'off' should be adjusted accordingly.                         *
*                                                               *
* hdtrks = tracks - 1                                           *
*                                                               *
*****************************************************************

        if      m1Ø ne Ø
        hdsectp equ     336             ;Sectors per track
        hdtrks  equ     243             ;Total data tracks
        endif

        if      m2Ø ne Ø
        hdsectp equ     672
        hdtrks  equ     243
        endif

        if      m26 ne Ø
        hdsectp equ     1Ø24
        hdtrks  equ     2Ø1
        endif

ldsk    set     Ø               ;Use non-standard partitioning
tracks  set     hdtrks/hdlog    ;Number of tracks per partition
dsm     set     hdsectp/8*tracks/4-1    ;Number of groups per partition
off     set     1

        rept    hdlog
        dpbgen  hd,%ldsk,%hdsectp,5,31,1,%dsm,511,Øffh,Øffh,Ø,%off,3
off     set     off+tracks
ldsk    set     ldsk+1
        endm

        else                    ;Else use standard DPB's

        if      m26 ne Ø
dpbhdØ  dw      1Ø24            ;CP/M sectors/track
        db      5               ;BSH
        db      31              ;BLM
        db      1               ;EXM
        dw      2Ø15            ;DSM
        dw      511             ;DRM
        db      Øffh            ;ALØ
        db      Øffh            ;AL1
        dw      Ø               ;CKS
        dw      1               ;OFF
        db      3               ;SECSIZ

dpbhd1  dw      1Ø24            ;CP/M sectors/track
        db      5               ;BSH
        db      31              ;BLM
        db      1               ;EXM
        dw      2Ø15            ;DSM
        dw      511             ;DRM
        db      Øffh            ;ALØ
        db      Øffh            ;AL1
        dw      Ø               ;CKS
        dw      64              ;OFF
        db      3               ;SECSIZ

dpbhd2  dw      1Ø24            ;CP/M sectors/track
        db      5               ;BSH
        db      31              ;BLM
        db      1               ;EXM
        dw      2Ø47            ;DSM
        dw      511             ;DRM
        db      Øffh            ;ALØ
        db      Øffh            ;AL1
        dw      Ø               ;CKS
        dw      127             ;OFF
        db      3               ;SECSIZ
        endif

        if      m1Ø ne Ø
dpbhdØ  dw      336             ;CP/M sectors/track
        db      5               ;BSH
        db      31              ;BLM
        db      1               ;EXM
        dw      1269            ;DSM
        dw      511             ;DRM
        db      Øffh            ;ALØ
        db      Øffh            ;AL1
        dw      Ø               ;CKS
        dw      1               ;OFF
        db      3               ;SECSIZ

dpbhd1  dw      336             ;CP/M sectors/track
        db      5               ;BSH
        db      31              ;BLM
        db      1               ;EXM
        dw      128Ø            ;DSM
        dw      511             ;DRM
        db      Øffh            ;ALØ
        db      Øffh            ;AL1
        dw      Ø               ;CKS
        dw      122             ;OFF
        db      3               ;SECSIZ
        endif

        if      m2Ø ne Ø
dpbhdØ  dw      672             ;CP/M sectors/track
        db      5               ;BSH
        db      31              ;BLM
        db      1               ;EXM
        dw      2Ø36            ;DSM
        dw      511             ;DRM
        db      Øffh            ;ALØ
        db      Øffh            ;AL1
        dw      Ø               ;CKS
        dw      1               ;OFF
        db      3               ;SECSIZ
```

```
dpbhd1  dw      672                     ;CP/M sectors/track
        db      5                       ;BSH
        db      31                      ;BLM
        db      1                       ;EXM
        dw      2Ø36                    ;DSM
        dw      511                     ;DRM
        db      Øffh                    ;ALØ
        db      Øffh                    ;AL1
        dw      Ø                       ;CKS
        dw      98                      ;OFF
        db      3                       ;SECSIZ

dpbhd2  dw      672                     ;CP/M sectors/track
        db      5                       ;BSH
        db      31                      ;BLM
        db      1                       ;EXM
        dw      1Ø28                    ;DSM
        dw      511                     ;DRM
        db      Øffh                    ;ALØ
        db      Øffh                    ;AL1
        dw      Ø                       ;CKS
        dw      195                     ;OFF
        db      3                       ;SECSIZ
        endif
        endif
        endif                           ;End of HD DPH's and DPB's

        if      maxmf ne Ø

        dpbgen  mf, Ø, 2Ø, 3,  7, Ø, Ø4fh,  63, ØcØh, Ø, 16, 3, 2
        dpbgen  mf, 1, 4Ø, 3,  7, Ø, Øa4h,  63, ØcØh, Ø, 16, 2, 3
        dpbgen  mf, 2, 4Ø, 4, 15, 1, Ø51h,  63,  8Øh, Ø, 16, 2, 3
        dpbgen  mf, 3, 4Ø, 4, 15, 1, Øa9h,  63,  8Øh, Ø, 16, 2, 3

dn      set     Ø
        rept    maxmf
        dphgen  mf,%dn,dpbmf,%dn
dn      set     dn+1
        endm
        endif

        if      maxfd ne Ø
dn      set     Ø
        rept    maxfd
        dphgen  fd,%dn,Ø,Ø
dn      set     dn+1
        endm
        endif

        if      maxdm ne Ø
dn      set     Ø
        rept    maxdm
        dphgen  dm,%dn,Ø,Ø
dn      set     dn+1
        endm
        endif

        if      maxmw ne Ø

*****************************************************************
*                                                               *
* mwsectp is the number of 128 byte sectors per cylinder.       *
* mwsectp = 72 * heads                                          *
*                                                               *
* mwtrks is the total number of data cylinders.                 *
* mwtrks = tracks - 1                                           *
*                                                               *
*****************************************************************

        if      st5Ø6 ne Ø
        mwsecpt equ     288             ;Sectors per track
        mwtrks  equ     152             ;Total data tracks
        endif

        if      st412 ne Ø
        mwsecpt set     288
        mwtrks  set     3Ø5
        endif

        if      cm5619 ne Ø
        mwsecpt set     432
        mwtrks  set     3Ø5
        endif

dphdsk  set     Ø               ;Generate DPH's for the HDDMA hard disks
        rept    maxmw
ldsk    set     Ø
        rept    mwlog
        dphgen  mw,%dphdsk,dpbmw,%ldsk
dphdsk  set     dphdsk+1
ldsk    set     ldsk+1
        endm
        endm

        if      mwpart ne Ø     ;Generate DPB's for a HDDMA hard disk

ldsk    set     Ø               ;Use non-standard partitioning
tracks  set     mwtrks/mwlog    ;Number of tracks per partition
dsm     set     mwsectp/8*tracks/4-1    ;Number of groups per partition
off     set     1

        rept    mwlog
        dpbgen  mw,%ldsk,%mwsecpt,5,31,1,%dsm,1Ø23,Øffh,Øffh,Ø,%off,4
off     set     off+tracks
ldsk    set     ldsk+1
        endm

        else                    ;Use standard partitioning

off     set     1                       ;Initial system track offset
trkoff  set     8192/(mwsecpt/8)+1      ;The number of tracks in a partition
blocks  set     mwsecpt/8*mwtrks        ;The number of blocks on the drive
psize   set     trkoff*(mwsecpt/8)      ;The number of blocks in a partition
ldsk    set     Ø

        rept    blocks/8192     ;Generate some 8 megabyte DPB's
        dpbgen  mw,%ldsk,%mwsecpt,5,31,1,2Ø47,1Ø23,Øffh,Øffh,Ø,%off,4
off     set     off+trkoff
blocks  set     blocks-psize
ldsk    set     ldsk+1
        endm
blocks  set     blocks/4
        if      blocks gt 256   ;If there is any stuff left, then use it
blocks  set     blocks-1
        dpbgen  mw,%ldsk,%mwsecpt,5,31,1,%blocks,1Ø23,Øffh,Øffh,Ø,%off,4
        endif
        endif
        endif

buffer  equ     $
*****************************************************************
```

```
*                                                                  *
* Signon message output during cold boot.                          *
*                                                                  *
********************************************************************
                  18H, 7EH
prompt: db        8Øh, clear                ;Clean buffer and screen
        db        acr, alf, alf
        db        'Morrow Designs '
        db        'Ø'+msize/1Ø              ;CP/M memory size
        db        'Ø'+(msize mod 1Ø)
        db        'K CP/M '                 ;CP/M version number
        db        cpmrev/1Ø+'Ø'
        db        '.'
        db        (cpmrev mod 1Ø)+'Ø'
        db        ' '
        db        (revnum/1Ø)+'A'-1
        db        (revnum mod 1Ø)+'Ø'
        db        acr, alf

;
;       Print a message like:
;
;        AB: DJDMA 8", CD: DJDMA 5 1/4", E: HDDMA M5
;

msdrv   set       Ø                         ;Start with drive A:

msbump  macro     ndrives                   ;Print a drive name
        if        dn gt 1
        db        ', '
        endif
        rept      ndrives
        db        msdrv+'A'
msdrv   set       msdrv+1
        endm
        db        ': '
        endm

prhex   macro     digit                     ;Write a byte in hex
        prnib     digit/1Øh
        prnib     digit
        endm

prnib   macro     digit                     ;Write a digit in hex
temp    set       digit and Øfh
        if        temp < 1Ø
        db        temp + 'Ø'
        else
        db        temp - 1Ø + 'A'
        endif
        endm

dn      set       1                         ;Generate the drive messages

        rept      16                        ;Run off at least 16 drives

        if        dn eq hdorder             ;Generate the HDCA's message
        msbump    maxhd*hdlog
        db        'HDCA '
        if        maxhd gt 1
        db        '(', maxhd+'Ø', ')'
        endif
        if        m1Ø ne Ø
        if        m1Øm ne Ø
        db        'Memorex'
        else
        db      'Fujitsu'
        endif
        db      ' M1Ø'
        endif
        if      m2Ø ne Ø
        db      'Fujitsu M2Ø'
        endif
        if      m26 ne Ø
        db      'Shugart M26'
        endif
        endif

        if      dn eq mworder           ;Generate the HDDMA's message
        msbump  maxmw*mwlog
        db      'HDDMA'
        if      mwquiet eq Ø
        db      ' '
        if      maxmw gt 1
        db      '(', maxmw+'Ø', ')'
        endif
        if      st5Ø6 ne Ø
        db      'M5'
        endif
        if      st412 ne Ø
        db      'M1Ø'
        endif
        if      cm5619 ne Ø
        db      'M16'
        endif
        endif
        endif

        if      dn eq fdorder           ;Generate the 2D/B message
        msbump  maxfd
        db      'DJ2D/B @'
        prhex   fdorig/1ØØh
        prhex   fdorig
        endif

        if      dn eq dmorder           ;Generate the DJDMA 8 message
        msbump  maxdm
        db      'DJDMA 8"'
        endif

        if      dn eq mforder           ;Generate the DJDMA 5 1/4 message
        msbump  maxmf
        db      'DJDMA 5 1/4"'
        endif

dn      set     dn+1
        endm

        db      acr,alf
        db      Ø                       ;End of message

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

cboot:  lxi     sp,tpa                  ;Set up stack

        xra     a                       ;Clear cold boot flag
        sta     cwflg
        sta     group                   ;Clear group select byte
        sta     cpmdrv                  ;Select disk A:
        sta     cdisk

        lxi     h,bios+3                ;Patch cold boot to warm code
        shld    bios+1

        lda     iobyt                   ;Initialize the IOBYTE
        sta     iobyte

        lxi     d,badmap                ;Clear out bad map
        stax    d
        lxi     h,badmap+1
        lxi     b,9*badsiz              ;32 map entries
        call    movbyt
        mvi     m,Øffh                  ;End marker

        if      contyp ne 6             ;Non IOBYTE inits
        if      contyp ne Ø             ;Do not call TTYSET for PROM's
        call    ttyset                  ;Initialize the terminal
        endif

        if      lsttyp ne Ø             ;Do not call LSTSET for PROM's
        call    lstset                  ;Initialize the list device
        endif
        else                            ;Do IOBYTE inits
        lxi     h,devset                ;Device setup routine pointer table
cbootØ: mov     e,m                     ;Load a routine address
        inx     h
        mov     d,m
        inx     h
        mov     a,d                     ;Test for the end of the table
        ora     e
        jz      cboot2
        push    h                       ;Save the table pointer
        lxi     h,cboot1                ;Return address
        push    h
        xchg
        pchl                            ;'CALL' a device setup routine
cboot1: pop     h                       ;Restore the table pointer
        jmp     cbootØ

devset: dw      ttyset, crtset, uc1set  ;Device setup routine pointers
        dw      ptrset, ur1set, ur2set
        dw      ptpset, up1set, up2set
        dw      lptset, ul1set, Ø

cboot2  equ     $
        endif

        lxi     h,prompt                ;Prep for sending signon message
        call    message                 ;Send the prompt
        jmp     gocpm

*****************************************************************
*                                                               *
* Console and list device initialization routines follow.      *
*                                                               *
*****************************************************************

        if      contyp eq 2             ;Multi I/O, Decision I

*****************************************************************
*                                                               *
* Terminal initilization routine.  This routine reads the sense *
* switch on the WB-14 and sets the speed accordingly.           *
*                                                               *
*****************************************************************

ttyset: call    selgØ                   ;Select group Ø
        in      sensesw                 ;Get sense switch (ff on a Multio)
        push    psw
        call    selcon                  ;Select console
        pop     psw
        push    psw
        call    tiniØ                   ;Initialize the console
        pop     psw
        push    psw
        call    selrdr                  ;Select the reader/punch
        pop     psw
        call    tiniØ                   ;Initialize the reader/punch
        ret

tiniØ:  ani     ØeØh                    ;Mask in upper three bits
        rlc                             ;Move into lower 3 bits
        rlc
        rlc
        cpi     7                       ;check for sense = 7 (Default setting)
        jz      dfbaud                  ;Use default baud rate

        lxi     h,btab                  ;Pointer to baud rate table
        add     a                       ;Table of words so double
        mov     e,a                     ;Make a 16 bit number into (de)
        mvi     d,Ø
        dad     d                       ;Get a pointer into baud rate table
        mov     e,m                     ;Get lower byte of word
        inx     h                       ;Bump to high byte of word
        mov     d,m                     ;Get upper byte. (de) now has divisor
        jmp     setit                   ;Set baud rate

dfbaud: lhld    defcon                  ;Use default baud rate
        xchg

setit:  mvi     a,dlab+wls1+wlsØ+stb    ;Enable divisor access latch
        out     lcr                     ;Set the baud rate in (de)
        mov     a,d
        out     dlm                     ;Set upper divisor
        mov     a,e
        out     dll                     ;Set lower divisor

        mvi     a,wls1+wlsØ+stb         ;Clear Divisor latch
        out     lcr
        xra     a
        out     ier                     ;Set no interrupts
        out     lsr                     ;Clear status
        mvi     a,dtrenb+rtsenb         ;Enable DTR and RTS outputs to terminal
        out     mcr
        in      msr                     ;Clear MODEM Status Register
        in      lsr                     ;Clear Line Status Register
        in      rbr                     ;Clear reciever buffers
        in      rbr
        ret

btab:   dw      1Ø47                    ;11Ø Baud       ØØØ
        dw      384                     ;3ØØ            ØØ1
        dw      96                      ;12ØØ           Ø1Ø
        dw      48                      ;24ØØ           Ø11
        dw      24                      ;48ØØ           1ØØ
        dw      12                      ;96ØØ           1Ø1
        dw      6                       ;192ØØ          11Ø
                                        ;DEFCON         111
        endif                           ;Multi I/O, Decision I

        if      contyp eq 3             ;2D/B console initialization

ttyset: call    fdtstat                 ;Clean input buffer
        rnz                             ;All empty
        call    fdcin
        jmp     ttyset

        endif                           ;2D/B console

        if      contyp eq 4
ttyset: call    dminit                  ;See if controller present
        rc                              ;No controller, return
        lxi     d,dmaci                 ;Console initialization sequence
        lxi     h,dmchan
        lxi     b,1Ø                    ;Command length
        call    movbyt
        dcx     h
        xra     a                       ;Clear serial input status
        sta     serin+1
        jmp     docmd2                  ;Do stuff and return

dmaci:  db      writem                  ;Zot monitor disable flag
        dw      ttyset                  ;Any non-zero byte will do
        db      Ø
        dw      1                       ;One byte
        dw      13f5h                   ;Magical place in monitor
        db      senabl                  ;Enable serial input
        db      1

        endif

*************************************************************************
*                                                                       *
* Initialize the North Star Mother board, left serial port, right       *
* serial port, and North Star RAM parity.                               *
*                                                                       *
*************************************************************************

        if      contyp eq 6             ;North Star drivers

ttyset:                                 ;Set up the parallel port + motherboard
        xra     a                       ;Initialize mother board
        out     6
        out     6
        out     6
        out     6

        mvi     a,3Øh                   ;Reset the parallel port input flag
        out     nspsta
        mvi     a,6Øh                   ;Set the parallel port output flag
        out     nspsta
        mvi     a,acr                   ;Force a CR out the parallel port
        call    nspout

                                        ;Initialize the left serial port
        mvi     a,nslinl                ;See the equates for bit definations
        out     nslsta
        mvi     a,nslin2
        out     nslsta
        xra     a                       ;Clear the input/output buffers
        out     nsldat
        in      nsldat
        in      nsldat
```

```
                                        ;Initialize the right serial port
        mvi     a,nsrin1                ;See the equates for bit definations
        out     nsrsta
        mvi     a,nsrin2
        out     nsrsta
        xra     a                       ;Clear the input/output buffers
        out     nsrdat
        in      nsrdat
        in      nsrdat

        if      nsram ne Ø              ;Reset parity on North Star RAMs
        mvi     a,4Øh                   ;Disable parity logic
        out     nsram
        lxi     h,Ø                     ;Starting address

nsetØ:  mov     a,m                     ;Get a byte
        mov     m,a                     ;Rewrite, set proper parity
        inr     l                       ;Bump the address pointer
        jnz     nsetØ

nset1:  inr     h                       ;Skip to the next memory page
        jz      nset2                   ;Skip if all done
        mvi     a,(high $) + 1          ;Is the pointer above us?
        cmp     h                       ;Set carry if pointer is <= our page+1
        jc      nsetØ                   ;Reset the next pages parity
        mov     a,m                     ;Test for a PROM or no memory
        mov     b,a                     ;Save the original byte
        cma                             ;See if this location will change
        mov     m,a
        cmp     m                       ;Test for a change
        mov     m,b                     ;Restore the original value
        jz      nsetØ                   ;Value complemented, must be RAM
        ora     a                       ;Test for no memory present
        jz      nset1                   ;Skip to the next page if no memory
        lxi     d,7ØØh                  ;Skip 2K bytes of 'PROM'
        dad     d
        jnc     nset1                   ;Do a page check if no overflow

nset2:  mvi     a,41h                   ;Re-enable parity on the memory boards
        out     nsram
        endif

crtset:                                 ;Null routines
ptrset:
ptpset:
uc1set:
ur1set:
ur2set:
up1set:
up2set:
lptset:
ul1set:
        ret
        endif                           ;North Star drivers

        if      (lsttyp ge 2) and (lsttyp le 5) ;Serial Multi I/O list drivers

lstset: call    sellst                  ;Select printer group
        mvi     a,dlab                  ;Access divisor latch
        out     lcr
        lhld    deflst                  ;Get LST: baud rate divisor
        mov     a,h
        out     dlm                     ;Set upper baud rate
        mov     a,l
        out     dll
        mvi     a,stb+wlsØ+wls1         ;2 stop bits + 8 bit word
        out     lcr
        mvi     a,dtrenb+rtsenb         ;DTR + RTS enabled
        out     mcr
        in      rbr                     ;Clear input buffer
        xra     a
        out     ier                     ;No interrupts
        ret

        endif

        db      Ø,Øffh,Ø

codelen equ     ($-bios)                ;Length of Cbios code

        if      codelen gt 1ØØØh        ;Test for SYSGEN problems
        'FATAL ERROR, system is too big for SYSGEN rev. 4.X'
dbgtmp  set     codelen         ;Cbios code length   !   <debug>
        endif

        if      debug
dbgtmp  set     codelen         ;Cbios code length   !   <debug>
        endif

        ds      512-($-buffer)          ;Buffer for 512 byte sectors

        if      (maxfd ne Ø) or (maxdm ne Ø) or (maxmw ne Ø)
        ds      512                     ;Additional space for 1k sector devices
        endif

*****************************************************************
*                                                               *
* Each bad map entry consists of 9 bytes:                       *
*       Logical drive number (1 byte)                           *
*       Track number of bad sector (2 bytes)                    *
*       Sector number of bad sector (2 bytes)                   *
*       Track number of alternate sector (2 bytes)              *
*       Sector number of alternate sector (2 bytes)             *
*                                                               *
*****************************************************************

badmap: ds      badsiz*9+1              ;32 entries + end marker

dirbuf: ds      128                     ;Directory buffer

tempb:  ds      16                      ;A little temporary buffer

*****************************************************************
*                                                               *
* Allocation and checked directory table area                   *
*                                                               *
*****************************************************************

        if      maxhd ne Ø
        if      hdpart ne Ø             ;Use non-standard partitioning

tracks  set     hdtrks/hdlog            ;Number of tracks per partition
dsm     set     hdsectp/8*tracks/4-1    ;Number of groups per partition
alv     set     (dsm/8)+1

dn      set     Ø
        rept    maxhd*hdlog             ;Generate CKS and ALV tables
        alloc   hd,%dn,%alv,Ø
dn      set     dn+1
        endm

        else                            ;Standard partitioning
```

```
dn      set     0
        rept    maxhd
        if      m26 ne 0
        alloc   hd,%dn,252,0
dn      set     dn+1
        alloc   hd,%dn,252,0
dn      set     dn+1
        alloc   hd,%dn,256,0
dn      set     dn+1
        endif

        if      m10 ne 0
        alloc   hd,%dn,159,0
dn      set     dn+1
        alloc   hd,%dn,161,0
dn      set     dn+1
        endif

        if      m20 ne 0
        alloc   hd,%dn,255,0
dn      set     dn+1
        alloc   hd,%dn,255,0
dn      set     dn+1
        alloc   hd,%dn,129,0
dn      set     dn+1
        endif
        endm
        endif

        endif

        if      maxfd ne 0
dn      set     0

        rept    maxfd
        alloc   fd,%dn,75,64
dn      set     dn+1
        endm
        endif

        if      maxdm ne 0
dn      set     0

        rept    maxdm
        alloc   dm,%dn,75,64
dn      set     dn+1
        endm
        endif

        if      maxmf ne 0
dn      set     0
        rept    maxmf
        alloc   mf,%dn,22,16
dn      set     dn+1
        endm
        endif

        if      maxmw ne 0
        if      mwpart ne 0             ;Use non-standard partitioning

tracks  set     mwtrks/mwlog            ;Number of tracks per partition
dsm     set     mwsectp/8*tracks/4-1    ;Number of groups per partition
alv     set     (dsm/8)+1
dn      set     0
```

```
        rept    maxmw*mwlog             ;Generate CKS and ALV tables
        alloc   mw,%dn,%alv,Ø
dn      set     dn+1
        endm

        else                            ;Use standard partitioning

dn      set     Ø
trkoff  set     8192/(mwsecpt/8)+1
psize   set     trkoff*(mwsecpt/8)

        rept    maxmw

blocks  set     mwsecpt/8*mwtrks

        rept    blocks/8192             ;Generate some 8 megabyte ALV's
        alloc   mw,%dn,256,Ø
blocks  set     blocks-psize
dn      set     dn+1
        endm

blocks  set     blocks/4

        if      blocks gt 256           ;Use the remainder
blocks  set     blocks-1
alv     set     (blocks/8)+1
        alloc   mw,%dn,%alv,Ø
dn      set     dn+1
        endif
        endm

        endif
        endif

bioslen equ     (high ($-bios))+1       ;BIOS length in pages

        if      bioslen gt biosln       ;Test for overflow
        'FATAL ERROR, system overflow.  BIOSLN must be at least'
dbgtmp  set     bioslen         ;BIOSLN! <debug>
        endif

        if      debug
dbgtmp  set     biosln          ;Current BIOSLN! <debug>
        if      biosln gt bioslen
dbgtmp  set     bioslen         ;Optimal BIOSLN! <debug>
        endif
        endif

        end
```