MORROW CBIOS 2.9 11/81

```
*****************************************************************
*                                                               *
* Cbios for CP/M Ver 2.2 for Disk Jockey 2D Mod. B              *
* controller.  Handles diskettes with sector sizes of 128       *
* bytes single density, 256, 512, 1Ø24 bytes double             *
* density. There are conditional assemblies for the             *
* Diskus Hard Disk Controller.                                  *
*                                                               *
* Note: The system 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 can be 128, 256, or 1Ø24 bytes in             *
*       either single or double density.                        *
*                                                               *
* Software engineering, Morrow Designs 11/81                    *
*                                                               *
*****************************************************************

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

revnum  equ     29              ;Cbios revision number 2.9
cpmrev  equ     22              ;CP/M revision number 2.2

*****************************************************************
* 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.                            *
*                                                               *
*****************************************************************

contyp  equ     2

*****************************************************************
*                                                               *
* Define the printer driver to be used.                         *
*                                                               *
* lsttyp is:    Ø       Nothing, used for patching to PROM's.   *
*               1       Provide for 128 bytes of patch space.   *
*               2       Multi I/O serial, no protocol.          *
*               3       Multi I/O serial, CTS protocol.         *
*               4       Multi I/O serial, DSR protocol.         *
*               5       Multi I/O serial, Xon / Xoff protocol.  *
*               6       Multi I/O parallel, Centronics.         *
*               7       Multi I/O parallel, Diablo HyType II.   *
*                                                               *
* Note: The Decision board is functionally identical to the     *
*       Multi I/O board for printer I/O.  Selections 2 - 6      *
*       will work on the Decision I.                            *
*                                                               *
*****************************************************************

lsttyp  equ     3
```

```
*****************************************************************
*                                                               *
* 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 then you should set multr3 to Ø.                          *
*                                                               *
*****************************************************************

multr3  equ     Ø               ;Ø = Decision, 1 = Multi I/O

        if      contyp eq 2
congrp  equ     1               ;Cosole port (1 = p1, 2 = p2, 3 = p3)
        endif

        if      lsttyp ge 2
lstgrp  equ     3               ;Printer port (1 = p1, 2 = p2, 3 = p3)
        endif

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

first   equ     1               ;Ø = Floppies are A,B,C,D drives and
                                ;       Hard Disk are E,F,G,H
                                ;1 = Hard Disks are A,B,C,D drives and
                                ;       Floppies are E,F,G,H
maxhd   equ     1               ;Set to number of hard disks
maxflop equ     4               ;Set to number of floppies

M26     equ     Ø               ;Set only one of these variables
M2Ø     equ     1
M1ØF    equ     Ø
M1ØM    equ     Ø

m1Ø     equ     m1Øf or m1Øm

*****************************************************************
*                                                               *
* The next equate will set the number of logical disks on a     *
* physical hard disk drive.  The user must set stdlog to a      *
* vaule greater than or equale to 2 for an m1Ø or 3 for an      *
* m2Ø or m26.  The reason for this is that CP/M can not         *
* address more than 8 megabytes per logical disk and            *
* splitting a disk to less then 2 or 3 parts would make         *
* partitions that are greater than 8 megabytes in length.       *
*                                                               *
*****************************************************************

stdlog  equ     Ø                       ;Set to Ø to use standard logical disks

        if      stdlog ne Ø
logdsk  equ     stdlog                  ;Set to number of user selected
        else                            ; logical disks
logdsk  equ     3*m26+3*m2Ø+2*m1Ø       ;Default logical disks per drive
        endif

fujitsu equ     m2Ø  or m1Øf
mrev    equ     26*m26+2Ø*m2Ø+1Ø*m1Ø    ;Hard disk type

hdspt   equ     32*m26+21*m2Ø+21*m1Ø    ;Sectors per track
```

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

        if      maxflop ne Ø    ;Include Discus 2D ?
origin  equ     ØF8ØØH
djram   equ     origin+4ØØh     ;Disk Jockey 2D RAM address
djboot  equ     origin          ;Disk Jockey 2D initialization
djcin   equ     origin+3h       ;Disk Jockey 2D character input routine
djcout  equ     origin+6h       ;Disk Jockey 2D character output routine
djhome  equ     origin+9h       ;Disk Jockey 2D track zero seek
djtrk   equ     origin+Øch      ;Disk Jockey 2D track seek routine
djsec   equ     origin+Øfh      ;Disk Jockey 2D set sector routine
djdma   equ     origin+Ø12h     ;Disk Jockey 2D set DMA address
djread  equ     origin+15h      ;Disk Jockey 2D read routine
djwrite equ     origin+18h      ;Disk Jockey 2D write routine
djsel   equ     origin+1bh      ;Disk Jockey 2D select drive routine
djtstat equ     origin+21h      ;Disk Jockey 2D terminal status routine
djstat  equ     origin+27h      ;Disk Jockey 2D status routine
djerr   equ     origin+2ah      ;Disk Jockey 2D error, flash led
djden   equ     origin+2dh      ;Disk Jockey 2D set density routine
djside  equ     origin+3Øh      ;Disk Jockey 2D set side routine
dblsid  equ     8               ;Side bit from controller
io      equ     origin+3f8h     ;Start of I/O registers
dreg    equ     io+1
cmdreg  equ     io+4
clrcmd  equ     ØdØh
        endif

;*****************************************************************
;*                                                               *
;* The following block will define cerain 2DB entry points in    *
;* case the user is not actually using the 2DB's disks but is    *
;* using the 2DB's console driver PROM.                          *
;*                                                               *
;*****************************************************************

        if (maxflop eq Ø) and (contyp eq 3)
origin  equ     ØF8ØØH
djcin   equ     origin+3h       ;Disk Jockey 2D character input routine
djcout  equ     origin+6h       ;Disk Jockey 2D character output routine
djtstat equ     origin+21h      ;Disk Jockey 2D terminal status routine
        endif

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

        if      maxhd ne Ø      ;Want Hard Disk included ?
hdorg   equ     5Øh             ;Hard Disk Controller origin
hdstat  equ     hdorg           ;Hard Disk Status
hdcntl  equ     hdorg           ;Hard Disk Control
hddata  equ     hdorg+3         ;Hard Disk Data
hdfunc  equ     hdorg+2         ;Hard Disk Function
hdcmnd  equ     hdorg+1         ;Hard Disk Command
hdreslt equ     hdorg+1         ;Hard Disk Result
retry   equ     2               ;Retry bit of result
tkzero  equ     1               ;Track zero bit of status
opdone  equ     2               ;Operaction done bit of status
complt  equ     4               ;Complete bit of status
```

```
tmout   equ     8               ;Time out bit of status
wfault  equ     10h             ;Write fault bit of status
drvrdy  equ     20h             ;Drive ready bit of status
index   equ     40h             ;Index bit of status
pstep   equ     4               ;Step bit of function
nstep   equ     0fbh            ;Step bit mask of function
hdrlen  equ     4               ;Sector header length
seclen  equ     512             ;Sector data length
wenabl  equ     0fh             ;Write enable
wreset  equ     0bh             ;Write reset of function
scenbl  equ     5               ;Controller control
dskclk  equ     7               ;Disk clock for control
mdir    equ     0f7h            ;Direction mask for function
null    equ     0fch            ;Null command
idbuff  equ     0               ;Initialize data command
isbuff  equ     8               ;Initialize header command
rsect   equ     1               ;Read sector command
wsect   equ     5               ;Write sector command
        endif

;*****************************************************************
;*                                                               *
;* 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
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     80h             ;Divisor latch access bit
thre    equ     20h             ;Status line THRE bit
cts     equ     10h             ;Clear to send
dsr     equ     20h             ;Data set ready
dr      equ     1               ;Line status DR bit
wls0    equ     1               ;Word length select bit 0
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     0
daisy0  equ     mbase           ;Daisy input ports
daisy1  equ     mbase+1
sensesw equ     mbase+1         ;Sense switches

        if      multr3 eq 0     ;Daisy output ports are different
daisi0  equ     mbase           ;   for Decision I and Multi I/O.
daisi1  equ     mbase+1         ;These two are the Decision I ports
        else
daisi0  equ     mbase+1         ;   and these are the Multi I/O's.
daisi1  equ     mbase
        endif

; Define daisy 0 status input bits
```

```
ribbon  equ     01h             ;End of ribbon
paper   equ     02h             ;Paper out
cover   equ     04h             ;Cover open
pfrdy   equ     08h             ;Paper feed ready
crrdy   equ     10h             ;Carriage ready
pwrdy   equ     20h             ;Print wheel ready
check   equ     40h             ;Printer check (error)
ready   equ     80h             ;Printer ready

; Define daisy 0 status input bits for Diablo HyType II driver

crstrd  equ     1020h           ;Carriage ready
pfstrd  equ     810h            ;Paper feed ready
pwstrd  equ     2040h           ;Print wheel ready

; Define daisy 0 output bits

d9      equ     01h             ;Data bit  9
d10     equ     02h             ;Data bit 10
d11     equ     04h             ;Data bit 11
d12     equ     08h             ;Data bit 12

pfstb   equ     10h             ;Paper feed strobe
crstb   equ     20h             ;Carriage strobe
pwstb   equ     40h             ;Print wheel strobe
restore equ     80h             ;Printer restore (Ribbon lift on Multi I/O)

; Define clock select bits

rlift   equ     40h             ;Ribbon lift
pselect equ     80h             ;Select (Not used by Diablo)

; Define group select bits

s0      equ     01h             ;Group number (0-3)
s1      equ     02h
smask   equ     03h
bank    equ     04h
enint   equ     08h
restor  equ     10h             ;Printer restore on Multi I/O
denable equ     20h             ;Driver enable on Multi I/O

; Define special constants for the HyTyp II driver

cperi   equ     10              ;Default to 10 characters per inch
lperi   equ     6               ;Default lines per inch
hinc    equ     120             ;Horizontal increments per inch
vinc    equ     48              ;Vertical increments per inch
numtabs equ     160             ;Number of horizontal tabs
maxchrs equ     1024            ;Maximum number of printer characters to queue
maxrgt  equ     1584            ;Maximum carriage position
dfrmln  equ     110             ;Forms length times 10
autolf  equ     0               ;Default to no Auto line feed.

        endif

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

msize   equ     48              ;Memory size of target CP/M
```

```
bias    equ     (msize-20)*1024 ;Memory offset from 20k system
ccp     equ     2500h+bias      ;Console command processor
bdos    equ     ccp+800h        ;BDOS address
bios    equ     ccp+1600h       ;Cbios address
offsetc equ     2100h-bios      ;Offset for sysgen
cdisk   equ     4               ;Address of last logged disk
buff    equ     80h             ;Default buffer address
tpa     equ     100h            ;Transient memory
intioby equ     0               ;Initial IOBYTE
iobyte  equ     3               ;IOBYTE location
wbot    equ     0               ;Warm boot jump address
entry   equ     5               ;BDOS entry jump address

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

retries equ     10              ;Max retries on disk i/o before error
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
acr     equ     'J'-64          ;Carriage return
avt     equ     'K'-64          ;Vertical tab
aff     equ     'L'-64          ;Form Feed
alf     equ     'M'-64          ;Line feed
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 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     cstty           ;Console status routine
cin     jmp     ciflsh          ;Console input
cout    jmp     cotty           ;Console output
        else
        jmp     $               ;Console status routine PROM pointer
cin     jmp     $               ;Console input PROM pointer
cout    jmp     $               ;Console output PROM pointer
        endif

        if      lsttyp ne 0
```

```
            else
            jmp     cout            ;List device output
            endif

            jmp     punch           ;Punch device output
            jmp     reader          ;Reader device input
            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 Ø
            jmp     listst          ;List device status
            else
            jmp     $               ;List device status
            endif

            jmp     sectran         ;Sector translation

            if      maxflop ne Ø
djdrv       jmp     djsel           ;Hookup for SINGLE.COM program
            else
            jmp     donop
            endif

;***************************************************************
;                                                              *
; The following two words define the default baud rate for     *
; the console and the LST: devices.  These words must          *
; immediatly follow the Cbios jump table so that the user      *
; can easly 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       *
; value needed for the defcon or deflst words.                 *
;                                                              *
; Baud rate     defcon          Baud rate       defcon         *
;       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          *
;                                                              *
;***************************************************************

defcon  dw      12              ;Console baud rate

deflst  dw      96              ;Printer baud rate

;***************************************************************
;                                                              *
; 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 a driver-enable)      *
; then you should modify the same bit in the group byte        *
; provided.  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.                                        *
*                                                              *
****************************************************************

group   db      Ø               ;Group byte

****************************************************************
*                                                              *
* 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.                           *
*                                                              *
****************************************************************

ciflsh  call    flush           ;Flush disk buffers on input
        jmp     citty

****************************************************************
*                                                              *
* contyp: 1     Blank space for console driver                 *
*                                                              *
* Note: If the user plans to utilize this space then the       *
* one time code such as tinit sould be placed just below       *
* the cboot routine.  This space (belowe cboot) is recyled     *
* for use as a disk buffer after cboot is done.                *
*                                                              *
****************************************************************

        if      contyp eq 1

tinit   equ     $               ;Make it easy to find this place
cotty   equ     $
citty   equ     $
cstty   equ     $
        ret
        ds      127

        endif                   ;Blank space

****************************************************************
*                                                              *
* 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 0-6 then the     *
; console baud rate will be taken from the rate table.           *
; Otherwise the current divisor latch value will be checked.     *
; If the divisor seems to be ok then no action will be taken     *
; as far as the baud rate setting goes.  If the divisor is not   *
; ok then 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  (0 = off, 1 = on)                          *
;               000 = 110                                       *
;               001 = 300                                       *
;               010 = 1200                                      *
;               011 = 2400                                      *
;               100 = 4800                                      *
;               101 = 9600                                      *
;               110 = 19200                                     *
;            defcon = 9600                                      *
;                                                               *
; Note: If you are compiling with multr3 (a Multi I/O) then     *
;       the switches will not be available so the baud rate     *
;       will be taken from defcon.                              *
;                                                               *
;*****************************************************************

;*****************************************************************
;                                                               *
; Due to its length, the tinit routine driver is belowe the     *
; cboot routine.                                                *
;                                                               *
;*****************************************************************

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

citty   lda     group           ;Get group byte
        ori     congrp          ;Select console
        out     grpsel

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.                            *
;                                                               *
;*****************************************************************

cotty   lda     group           ;Get group byte
        ori     congrp          ;Select console
        out     grpsel

conoutl in      lsr             ;Read status
        ani     thre            ;Wait till transmitter buffer empty
        jz      conoutl
        mov     a,c             ;Character is in (c)
```

```
        out     tnr             ;Output to transmitter buffer
        ret

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

cstty   lda     group           ;Get group byte
        ori     congrp          ;Select console
        out     grpsel

        in      lsr             ;Read status register
        ani     dr
        rz                      ;No charactter ready
        mvi     a,0ffh          ;Character ready
        ret

        endif                   ;Multi I/O or Decision I

;*****************************************************************
;*                                                               *
;* contyp: 3     2DB console driver                              *
;*                                                               *
;*****************************************************************

        if      contyp eq 3

cotty   jmp     djcout          ;Console output

citty   jmp     djcin           ;Console input

cstty   call    djtstat         ;Console status
        mvi     a,0ffh
        rz
        inr     a
        ret

        endif                   ;2DB

;*****************************************************************
;*                                                               *
;* LST: device driver routines.                                  *
;*                                                               *
;* Routine used depends on the value of lsttyp.  Possible        *
;* lsttyp values are listed as follows:                          *
;*                                                               *
;* lsttyp is:    0       Nothing, used for patching to PROM's.   *
;*               1       Provide for 128 bytes of patch space.   *
;*               2       Multi I/O serial, no protocol.          *
;*               3       Multi I/O serial, CTS protocol.         *
;*               4       Multi I/O serial, DSR protocol.         *
;*               5       Multi I/O serial, Xon / Xoff protocol.  *
;*               6       Multi I/O parallel, Centronics.         *
;*               7       Multi I/O parallel, Diablo HyType II.   *
;*                                                               *
;*****************************************************************

;*****************************************************************
;*                                                               *
;* lsttyp: 1     Blank space for printer driver                  *
;*                                                               *
;* Note: If the user plans to utilize this space then the        *
;* one time code such as linit sould be placed just below        *
```

```
* the cboot routine.  This space (belowe cboot) is recyled      *
* for use as a disk buffer after cboot is done.                 *
*                                                               *
*****************************************************************

        if      lsttyp eq 1

linit   equ     $               ;Make it easy to find this place
list    equ     $
listst  equ     $
        ret
        ds      127

        endif                   ;Blank space

*****************************************************************
*                                                               *
* lsttyp: 2     Serial printer, no protocol                     *
*                                                               *
* lsttyp: 3     Serial printer, CTS protocol                    *
*                                                               *
* lsttyp: 4     Serial printer, DSR protocol                    *
*                                                               *
* lsttyp: 5     Serial printer, Xon / Xoff protocol             *
*                                                               *
*****************************************************************

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

list    lda     group           ;Get group byte
        ori     lstgrp          ;Select list device
        out     grpsel

ll      in      lsr
        ani     thre            ;Wait till transmitter buffer empty
        jz      ll

*****************************************************************
*                                                               *
* The CTS driver is used for a printer with hardware            *
* handshaking (TI 810).  It should be connected to the CTS      *
* input on the list device serial port.                         *
*                                                               *
*****************************************************************

        if      lsttyp eq 3     ;CTS protocol
        in      msr
        ani     cts             ;Wait till clear to send
        jz      ll
        endif

*****************************************************************
*                                                               *
* The DSR driver is used for a printer with hardware            *
* handshaking (TI 810).  It should be connected to the DSR      *
* input on the list device serial port.                         *
*                                                               *
*****************************************************************

        if      lsttyp eq 4     ;DSR protocol
        in      msr
        ani     dsr
        jz      ll              ;Wail till DSR comes up
        endif

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

```
*                                                                     *
* The Xon/Xoff driver is used for a printer with software             *
* handshaking (Diablo 630).                                           *
*                                                                     *
***********************************************************************

        if      lsttyp eq 5     ;Xon / Xoff protocol
xloop   call    listst          ;Check printer status
        ora     a
        jz      xloop           ;Loop if not ready
        endif

        mov     a,c
        out     thr
        ret

listst  lda     group           ;Get group byte
        ori     lstgrp          ;Select list device
        out     grpsel

        in      lsr             ;Check if transmitter buffer empty
        ani     thr
        rz                      ;Return not ready

        if      lsttyp eq 3     ;CTS protocol
        in      msr
        ani     cts
        rz                      ;Return not ready if CTS is false
        endif

        if      lsttyp eq 4     ;DSR protocol
        in      msr
        ani     dsr
        rz                      ;Return not ready if DSR is true
        endif

        if      lsttyp eq 5     ;Xon / Xoff protocol
        mvi     b,xon           ;Last character recieved from printer
lstflg  equ     $-1
        in      lsr
        ani     dr              ;Check for a character
        jz      xskip           ;No character present
        in      rbr             ;Get character
        ani     7fh
        mov     b,a             ;Save
        sta     lstflg          ;Kludge flag (last character recieved)
xskip   mov     a,b
        sui     xoff            ;Check for Xoff char (control S)
        jnz     xsdone          ;Printer ready
        ret                     ;Printer not ready

xsdone  equ     $               ;Printer ready for data
        endif

        mvi     a,0ffh
        ret                     ;Printer ready

        endif                   ;Multi I/O serial

***********************************************************************
*                                                                     *
* lsttyp: 6     Centronics parallel printer driver.                   *
*                                                                     *
***********************************************************************

        if      lsttyp eq 6
```

```
*****************************************************************
*                                                               *
* Decision I Diablo parallel to Centronics parallel interface.  *
*                                                               *
* The following cable must be made from the 50 pin Diablo       *
* conector to the 35 pin Centronics.                            *
*                                                               *
* Centronics                    Multi I/O                       *
* pin   signal                  pin     signal                  *
* 1     /strobe         <-      46      /d9                     *
* 2     data1           <-      37      /d1                     *
* 3     data2           <-      36      /d2                     *
* 4     data3           <-      39      /d3                     *
* 5     data4           <-      33      /d4                     *
* 6     data5           <-      40      /d5                     *
* 7     data6           <-      42      /d6                     *
* 8     data7           <-      43      /d7                     *
* 9     data8           <-      45      /d8                     *
* 10    /acknlg         ->      12      /check                  *
* 11    busy            ->      28      /printer ready          *
* 12    pe              ->      3       /paper                  *
* 13    slct            ->      4       /ribbon                 *
* 14    /auto feed xt   <-      1       /d10                    *
* 15    NC                                                      *
* 16    0V              <->     2       GND                     *
* 17    chassis gnd                                             *
* 18    NC                                                      *
* 19    /strobe rtn     <->     8       GND                     *
* 20    data1 rtn       <->     11      GND                     *
* 21    data2 rtn       <->     14      GND                     *
* 22    data3 rtn       <->     16      GND                     *
* 23    data4 rtn       <->     18      GND                     *
* 24    data5 rtn       <->     20      GND                     *
* 25    data6 rtn       <->     22      GND                     *
* 26    data7 rtn       <->     25      GND                     *
* 27    data8 rtn       <->     38      GND                     *
* 28    /acknlg rtn     <->     41      GND                     *
* 29    busy rtn        <->     44      GND                     *
* 30    pe rtn          <->     47      GND                     *
* 31    /init           <-      9       /d11                    *
* 32    /error          ->      5       /cover                  *
* 33    GND             <->     35      GND                     *
* 34    NC                                                      *
* 35    /slct in        <-      10      /d12                    *
*                               35 <-> 24 /Select               *
*                                                               *
* IMPORTANT: For this interface to work /select (24) on the     *
* parallel conector must be tied to ground (35).                *
*                                                               *
*****************************************************************

list    lda     group           ;Get group byte
        out     grpsel

rl      in      daisy0          ;Wait till printer ready and selected
        ani     ready+paper
        jz      rl
pl      in      daisy0          ;Test if out of paper
        ani     ribbon
        jnz     pl
el      in      daisy0
        ani     cover
        jnz     el
        mov     a,c             ;Move character into (a)
        out     daisil          ;Latch data
```

```
            mvi     a,dll+dlØ+d9    ;Make sure strobe is high
            out     daisiØ
            dcr     a               ;Pulse strobe low
            out     daisiØ
            inr     a
            out     daisiØ

ack         in      daisyØ          ;Wait till ready again
            ani     ready
            jz      ack
            ret

listst      lda     group           ;Get group byte
            out     grpsel          ;Select group zero

            in      daisyØ          ;Wait till printer ready and selected
            ani     ready+paper
            rz
            in      daisyØ          ;Test if out of paper
            ani     ribbon
            rz
            in      daisyØ
            ani     cover
            xri     cover
            rz
            dcr     a
            ret

            endif                   ;Centronics parallel

            if      lsttyp eq 7     ;Diablo HyTyp II

;*****************************************************************
;* Diablo 161Ø simulator for the Morrow Designs / Thinker Toys   *
;* Mult I/O board. The simulator makes the parallel Hytyp II     *
;* look like a serial 161Ø.                                      *
;*****************************************************************

;*****************************************************************
;*                                                               *
;* This routine does all of the character decoding, escape       *
;* sequences forward, backward, etc. The list of escape          *
;* sequences, and special characters recognized is:              *
;*                                                               *
;*       adel            ignored                                 *
;*       anul            ignored                                 *
;*       aack            ignored (when received)                 *
;*       abel            ignored                                 *
;*       aff             form feed                               *
;*       aetx            etx/ack handshake                       *
;*       aht             horizontal tab                          *
;*       alf             line feed                               *
;*       asp             space                                   *
;*       abs             backspace                               *
;*       acr             carriage return                         *
;*       aesc Ø          ignored                                 *
;*       aesc 1          set tab stop at current print position  *
;*       aesc 2          clear all tab stops                     *
;*       aesc 3          graphics mode on                        *
;*       aesc 4          graphics mode off                       *
;*       aesc 5          forward print                           *
;*       aesc 6          backward print                          *
;*       aesc 8          clear tab stop                          *
;*       aesc 9          set left margin                         *
;*       aesc A          ignored                                 *
;*       aesc B          ignored                                 *
```

```
*       aesc D          negative half line feed                 *
*       aesc U          half line feed                          *
*       aesc alf        negative line feed                      *
*       aesc aht c      absolute horizontal tab                 *
*       aesc avt c      absolute vertical tab                   *
*       aesc ars c      set vmi                                 *
*       aesc aus c      set hmi                                 *
*                                                               *
*****************************************************************

list    lda     group           ;Set printer initialized flag
        ori     denable
        sta     group
        mov     a,c             ;Get the character to print
        ani     7fh             ;Strip off parity
        rz
        cpi     adel            ;Ignore delete
        rz
        mov     c,a             ;Save character
        lda     escflg
        lxi     h,level0        ;Level zero characters
        ana     a
        mov     a,c             ;Scan for char in A
        jz      lookup          ;Look up activity for this character
        lda     escflg
        lxi     h,level1        ;Single character escape sequences
        cpi     aesc
        mov     a,c             ;Scan for char in A
        jz      lookup          ;Execute single level escape sequence
        lxi     h,level2        ;Two character escape sequence
        lda     escflg

*****************************************************************
* Lookup scans the table pointed at by HL looking for a match   *
* of the character in register A.                               *
*****************************************************************

lookup  dcr     m               ;Test if end of table
        inr     m
        jz      gother          ;Execute the default function
        cmp     m               ;Otherwise test for a match
        jz      gother
        inx     h               ;Bump over character
        inx     h               ;Bump over function address
        inx     h
        jmp     lookup
gother  inx     h               ;Bump over character
        mov     a,m             ;Get low byte of function address
        inx     h
        mov     h,m             ;Get high byte of function address
        mov     l,a             ;Form Address of function
        pchl                    ;Execute it

*****************************************************************
* Each of the following tables contains entries of the form:    *
*       1 byte character to match                               *
*       2 bytes of address to execute                           *
* terminated by a first byte of 0.                              *
*****************************************************************

level0  db      aesc
        dw      doaesc          ;Beginning of an escape sequence
        db      aff
        dw      doaff           ;Form feed
        db      aetx
        dw      doaetx
```

```
            db      aht
            dw      doaht           ;horizontal tab
            db      alf
            dw      doalf           ;Line feed
            db      asp
            dw      doasp           ;Space
            db      abs
            dw      doabs           ;Back space
            db      acr
            dw      doacr           ;Carriage return
            db      Ø
            dw      dochar          ;Any other character

level1      db      '1'
            dw      sethtab         ;Set horizontal tab
            db      '2'
            dw      clrall          ;Clear all horizontal tabs
            db      '3'
            dw      setgrp          ;Graphics mode
            db      '4'
            dw      clrgrp          ;Clear graphics mode
            db      '5'
            dw      clrdir          ;Forward printing
            db      '6'
            dw      setdir          ;Backward printing
            db      '8'
            dw      clrhtab         ;Clear horizontal tab
            db      '9'
            dw      setlmar         ;Set left margin
            db      'Ø'
            dw      funcl           ;No operation level 1
            db      'A'
            dw      funcl
            db      'B'
            dw      funcl
            db      'a'
            dw      funcl
            db      'b'
            dw      funcl
            db      'D'
            dw      neghlf          ;Negative half line feed
            db      'U'
            dw      poshlf          ;Half line feed
            db      alf
            dw      neglf           ;Negative line feed
            db      aht
            dw      settwo          ;Two character escape sequence
            db      avt
            dw      settwo
            db      ars
            dw      settwo
            db      aus
            dw      settwo
            db      Ø
            dw      funcl

level2      db      aht
            dw      abshtab         ;Absolute horizontal tab
            db      avt
            dw      absvtab         ;Absolute vertical tab
            db      ars
            dw      setvmi
            db      aus
            dw      sethmi
            db      Ø
            dw      func2
```

```
;*********************************************************************
;* The following routines execute escape sequences, etc.             *
;*********************************************************************

settwo
doaesc  mov     a,c             ;Get the escape character
        sta     escflg
funcØ   ret

doaetx  ret

doalf   call    lfvmi           ;Get line feed vmi
adjvp   xchg
        lhld    dlvpos          ;Get vertical motion displacement
        dad     d
        shld    dlvpos
        ret

lfvmi   lda     grhflg
        ana     a
        lxi     h,1             ;Only 1/48 if in graphics mode
        rnz
        lhld    vmi             ;Get vertical motion index
        ret

neglf   call    lfvmi           ;Get line feed vmi
        call    neghl
        call    adjvp
        jmp     funcl

doasp   call    sphmi           ;Get space horizontal motion
spdir   lda     dirflg          ;Forward or backwards ?
        ana     a
        cnz     neghl           ;Negate HL
adjhp   xchg                    ;Adjust Horizontal position
        lhld    dlhpos          ;Get current adjustment
        dad     d               ;Update it
        shld    dlhpos          ;And save
        ret

sphmi   lda     grhflg          ;In graphics mode ?
        ana     a
        lxi     h,2             ;Only 1/6Ø if in graphics mode
        rnz
        lhld    hmi
        ret

doabs   call    sphmi           ;Space increment
        call    neghl           ;Negative to start with
        jmp     spdir           ;Adjust backwards

doacr   xra     a
        sta     dirflg          ;Forward printing
        sta     grhflg          ;No graphics mode
        lhld    hpos            ;Get current offset
        xchg
        lhld    lmar            ;Get left margin
        call    hlmde
        shld    dlhpos          ;Don't move yet though
        mvi     a,autolf        ;In Auto line feed mode ?
        ana     a
        jnz     doalf           ;Do line feed also
        ret

dochar  mov     l,c
```

```
                mvi     h,Ø
                call    wheel           ;Print the character in register C
                lda     grhflg
                ana     a
                lxi     h,Ø             ;Don't move if in graphics mode
                jnz     spdir
                lhld    hmi
                jmp     spdir

clrall          equ     $               ;Clear all horizontal tabs
                lxi     h,tabstp        ;Beginning of tab stop array
                mvi     d,tablen        ;Size of tab array (bytes)
notblp          mvi     m,8Øh           ;Reset tabs (reset to Ø later)
kludge          equ     $-1             ;Used on first reset (warmboot)
                inx     h               ;Next tab stop
                dcr     d               ;Update repeat count
                jnz     notblp          ;Continue zeroing
func2           equ     $
func1           xra     a               ;Clear escape sequence flag
                sta     escflg
                ret

setgrp          mvi     a,1             ;Set graphics mode on
                sta     grhflg
                jmp     func1

clrgrp          xra     a               ;Turn graphics mode off
                sta     grhflg
                jmp     func1

clrdir          xra     a               ;Forward print mode
                sta     dirflg
                jmp     func1

setdir          mvi     a,a             ;Set backward printing mode
                sta     dirflg
                jmp     func1

setlmar         lhld    hpos            ;Get current position
                xchg
                lhld    dlhpos          ;Get offset
                dad     d
                shld    lmar
                jmp     func1

setvmi          mov     l,c             ;Set the motion index
                mvi     h,Ø
                dcx     h
                shld    vmi
                jmp     func2

sethmi          mov     l,c
                mvi     h,Ø
                dcx     h
                shld    hmi
                jmp     func2

poshlf          call    hlfvmi          ;Half line feed vmi
                call    adjvp
                jmp     func1

neghlf          call    hlfvmi          ;Negative half line feed
                call    neghl
                call    adjvp
                jmp     func1
```

```
hlfvmi  lhld    vmi             ;Get vmi for full line feed
divid2  mov     a,h             ;High byte
        ora     a               ;Clear the carry
        rar
        mov     h,a
        mov     a,l
        rar
        mov     l,a
        ret

abshtab mov     e,c             ;Absolute horizontal tab
        mvi     d,Ø
        dcx     d               ;Form 16 bit tab column
        call    newdlh
        jmp     func2

newdlh  lhld    hmi
        call    hltde           ;Multiply by hmi
        xchg
        lhld    hpos            ;And subtract current horizontal position
        xchg
        call    hlmde
        shld    dlhpos
        ret

absvtab mov     e,c             ;Absolute vertical tab
        mvi     d,Ø
        dcx     d
        lhld    vmi
        call    hltde           ;Multiply by vmi
        xchg
        lhld    vpos            ;And subtract the current vertical position
        xchg
        call    hlmde
        xchg
        call    hlmde
        shld    dlvpos
        jmp     func2

sethtab call    tabcol          ;Set horizontal tab
        ora     m               ;OR in tab stop
        mov     m,a             ;   and save
        jmp     funcl

tabcol  lhld    hpos            ;Compute address of current character col
        xchg
        lhld    dlhpos
        dad     d               ;Get logical position
        xchg
        lhld    hmi             ;And divide by hmi to get character column
        xchg
        call    hldde

mtabp   ;Make a tab pointer
        ;HL -> Tab column desired (1-16Ø)
        ;HL <- address of tab stop
        ; A <- bit mask for tab stop
        lxi     d,8             ;Number of stops per byte
        call    hldde           ;HL/DE -> HL, HL mod DE -> DE
        mov     c,e             ;Save
        inr     c               ;Make range (1-8)
        lxi     d,tabstp        ;Tab array
        dad     d               ;Make array pointer
        xra     a
        stc
mtabØ   rar
        dcr     c               ;Bump bit counter
        jnz     mtabØ
```

```
        ret

clrhtab call    tabcol          ;Clear horizontal tab
        cma
        ana     m               ;Mask out tab stop
        mov     m,a
        jmp     funcl

doaht   lhld    hpos            ;Compute address of current character col
        xchg
        lhld    dlhpos
        dad     d               ;Get logical position
        xchg
        lhld    hmi             ;And divide by hmi to get character column
        xchg
        call    hldde
tablop  lxi     d,numtabs
        inx     h               ;Start with next position
        call    hlcde           ;Compare position with number of tabs
        jnc     tofar           ;Past last tab
        push    h               ;Save col pointer
        call    mtabp           ;Generate tab pointer
        ana     m               ;Check out tab stop
        pop     h               ;Restore col pointer
        jz      tablop          ;Loop if stop not set
        xchg
        jmp     newdlh          ;Set new col position and return
tofar   lhld    hpos            ;Go all the way to the right
        xchg
        lxi     h,maxrgt
        call    hlmde
        shld    dlhpos
        ret

doaff   lxi     h,dfrmln        ;Multiply forms length by 48
        lxi     d,48
        call    hltde
        lxi     d,10
        call    hldde           ;And divide it by 10
        push    h               ;Save this result
        lhld    vpos            ;Get logical vertical position
        xchg
        lhld    dlvpos
        dad     d
        pop     d
        push    d               ;Get copy of forms length
        call    hldde           ;HL mod DE
        xchg
        pop     d
        xchg
        call    hlmde
        xchg
        lhld    dlvpos
        dad     d
        shld    dlvpos
        jmp     papr

;*****************************************************************
;* Neghl forms the twos complement of HL.                        *
;*****************************************************************

neghl   mov     a,h
        cma
        mov     h,a
        mov     a,l
        cma
```

```
        mov     l,a
        inx     h
        ret

;*****************************************************************
;* Hlmde subtracts DE from HL and returns.                       *
;*****************************************************************

hlmde   xchg
        call    neghl
        xchg
        dad     d
        ret

;*****************************************************************
;* Hlcde compares HL with DE. On return the Z flag is set if     *
;* they are equal, the Carry flag is set if HL is less than DE.  *
;*****************************************************************

hlcde   mov     a,h
        cmp     d
        rnz
        mov     a,l
        cmp     e
        ret

;*****************************************************************
;* Divide the number in HL by the number in DE.  Return the      *
;* quotient in HL and the remainder in DE.                       *
;*****************************************************************

hldde   mov     a,d             ;Start by negating DE and
        cma                     ;       moving the left operand to BC
        mov     b,a
        mov     a,e
        cma
        mov     c,a
        inx     b
        mvi     a,16            ;Repeat count in reg A
        lxi     d,0             ;Initial remainder is zero
div3    dcr     a               ;Test if done
        rm                      ;All done ?
        dad     h               ;Shift right operand to the left
        xchg
        push    psw             ;Save carry
        dad     h               ;Shift left operand to the left
        pop     psw
        jnc     div1            ;Does it fit ?
        inx     h
div1    push    h
        dad     b
        jnc     div2
        xchg
        inx     h
        xthl
        pop     h
        jmp     div3
div2    pop     h
        xchg
        jmp     div3

;*****************************************************************
;* Multiply the contents of HL by the contents of DE.            *
;*****************************************************************

hltde   mov     c,l
```

```
          mov     b,h
          lxi     h,Ø
mult      mov     a,b
          ora     c
          rz
          mov     a,b
          ora     a
          rar
          mov     b,a
          mov     a,c
          rar
          mov     c,a
          cc      dadde
          xchg
          dad     h
          xchg
          jmp     mult
dadde     dad     d
          ret

;*****************************************************************
;* The routines below actually interface to the printer,         *
;* causing paper feed, carriage, and print wheel motion.         *
;*****************************************************************

carrg     lhld    dlhpos          ;Check for any accumulated motion
          mov     a,h
          ora     l
          rz
          lhld    hpos            ;Check for too much motion
          xchg
          lhld    dlhpos
          dad     d
          mov     a,h
          ana     a
          jp      lftok
          lhld    hpos
          call    neghl
          shld    dlhpos
lftok     lhld    hpos
          xchg
          lhld    dlhpos
          dad     d
          lxi     d,maxrgt
          call    hlcde
          jc      rgtok
          lhld    hpos            ;Otherwise move only to maxright
          xchg
          lxi     h,maxrgt
          call    hlmde
          shld    dlhpos
rgtok     lhld    hpos            ;Update the horizontal position
          xchg
          lhld    dlhpos
          dad     d
          shld    hpos
          lhld    dlhpos          ;check if required motion is to the left
          mov     a,h
          ana     a
          mvi     c,Ø
          jp      posh
          call    neghl
          mvi     c,dll
posh      xchg
          lxi     h,Ø
          shld    dlhpos          ;Reset the horizontal increment
```

```
            xchg
            mov     a,l
            ani     1
            jz      nohhlf          ;No half spaces
            mov     a,c
            ori     dl2
            mov     c,a
nohhlf      call    divid2
            mov     a,h
            ani     d9+dlØ
            ora     c
            mov     h,a
            lxi     d,crstrd
            jmp     cmnd

papr        lhld    dlvpos          ;Check for any paper motion
            mov     a,h
            ora     l
            rz                      ;No motion
            mov     a,h
            ana     a
            mvi     c,Ø
            jp      posv
            call    neghl
            mvi     c,dll
posv        mov     a,h
            ani     d9+dlØ
            ora     c
            mov     h,a
            push    h               ;Save paper motion
            lhld    vpos
            xchg
            lhld    dlvpos          ;Get logical position
            dad     d
            push    h               ;Save for now
            lxi     h,dfrmln        ;Get default form length
            lxi     d,48
            call    hltde           ;Multiply by 48
            lxi     d,1Ø
            call    hldde           ;Divide by 1Ø
            pop     d
            xchg
            call    hldde           ;Compute HL mod DE
            xchg
            shld    vpos            ;Save new vertical position
            lxi     h,Ø
            shld    dlvpos          ;Reset vertical motion
            pop     h
            lxi     d,pfstrd        ;Paper feed strobe
            jmp     cmnd

wheel       push    h
            call    carrg           ;Position the carriage first
            call    papr
            pop     h
            lxi     d,pwstrd

cmnd        lda     group           ;Get group byte
            out     grpsel          ;Select group zero

cmndØ       in      daisyØ
            ana     d
            jz      cmndØ
            mov     a,l             ;Negate low data bits
            cma
            mov     l,a
```

```
        mov     a,h
        ani     d9+d10+d11+d12  ;Mask in data bits only
        cma
        if      multr3          ;Mask out ribbon lift bit on Multi I/O
        ani     0ffh-restore
        endif
        mov     h,a
        mov     a,l
        out     daisil          ;Output low bits
        mov     a,h
        out     daisi0          ;Output high bits
        xra     e               ;Slap strobe bits in
        out     daisi0
        mov     a,h             ;And drop strobes back down
        out     daisi0
        ret

;*****************************************************************
;* New list device status routine. Returns 0ffh if the printer   *
;* can except another character, otherwise it returns 0.         *
;*****************************************************************

listst  lda     group           ;Check printer initialized flag
        ani     denable
        rz                      ;0 = printer not initialized
        lda     group           ;Get group byte
        out     grpsel          ;Select group zero
        lxi     d,pwstrd
        in      daisy0
        ana     d
        xra     a
        rz
        cma
        ret

;*****************************************************************
;* Dynamic data locations used by the simulator.                 *
;*****************************************************************

hmi     dw      0               ;Horizontal motion index. Set by linit
                                ;       and escape sequences.
vmi     dw      0               ;Vertical motion index. Set by linit
                                ;       and escape sequences.
vpos    dw      0               ;Vertical position. Set by platen motion
dlvpos  dw      0               ;Delta vpos. Set by platen motion
hpos    dw      0               ;Horizontal position. Set by carriage motion
dlhpos  dw      0               ;Delta hpos. Set by carriage motion
lmar    dw      0               ;Left margin
dirflg  db      0               ;Direction flag
grnflg  db      0               ;Graphics mode flag
escflg  db      0               ;Escape sequence in progress flag

tabstp  ds      numtabs/8+1     ;Tab stops bit array
tablen  equ     numtabs/8+1     ;Length of tabs array

        endif

;*****************************************************************
;*                                                               *
;* The following routines are used to make the reader and punch  *
;* devices peform I/O through the console.  The user may patch   *
;* here for their particular devices.                            *
;*                                                               *
;*****************************************************************

punch   jmp     cout
```

```
reader  jmp     cin

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

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

cwflg   db      Ø               ;Cold/warm boot flag

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

autoflg db      Ø               ;Auto command feature

;*****************************************************************
;*                                                               *
;* If there is a command inserted here, it will be given if the  *
;* auto feature is enabled.                                      *
```

```
*       For Example:                                                  *
*                                                                     *
*       coldbeg db      'MBASIC MYPROG'                               *
*       coldend db      Ø                                             *
*                                                                     *
* will execute microsoft basic, and mbasic will execute the           *
* "MYPROG" basic program.                                             *
*                                                                     *
***********************************************************************

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

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

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

        if      (maxhd ne Ø) and first  ;Supply Warm Boot from Hard Disk ?
        xra     a
        mov     c,a
        lxi     h,ccp-2ØØh      ;Initial DMA address
        push    h
        sta     head
        mvi     a,1
        push    psw             ;Save first sector - 1
        call    hddrv           ;Select drive A
        mvi     c,Ø
        call    hdtrk           ;Home the drive
warmlod pop     psw             ;Restore sector
        pop     h               ;Restore DMA address
        inr     a
        sta     hdsectr
        cpi     12              ;Past BDOS ?
        jz      gocpm           ;Yes, all done
        inr     h               ;Update DMA address
        inr     h
        shld    hdadd
        push    h
        push    psw
warmrd  lxi     b,retries*1ØØh+Ø ;Retry counter
wrmread push    b               ;Save the retry count
        call    hdread          ;Read the sector
        pop     b
        jnc     warmlod         ;Test for error
        dcr     b               ;Update the error count
        jnz     wrmread         ;Keep trying if not to many errors
        hlt                     ;Error halt
        db      Ø               ;Try not to screw up Decision CPU's
        endif

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

```
            if      (maxflop ne 0) and not first ;Supply Warm Boot from 2D ?
            mvi     c,0
            call    djsel           ;Select drive A
wrmfail     call    djhome          ;Track 0, single density
            jc      wrmfail         ;Loop if error
            mvi     c,0             ;Select side 0
            call    djside
                                    ;The next block of code re-initializes
                                    ;   the warm boot loader for track 0.

            mvi     a,5-2           ;Initialize the sector to read - 2
            sta     newsec
            lxi     h,ccp-100h      ;First revolution DMA - 100h
            shld    newdma
                                    ;Load all of track 0

t0boot      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-80h       ;Memory address of sector - 100h
            shld    newdma
nowrap      sta     newsec          ;Save the updated sector #
            mov     c,a
            call    djsec           ;Set up the sector
            lxi     h,ccp-100h      ;Memory address of sector - 100h
newdma      equ     $-2
            lxi     d,100h          ;Update DMA address
            dad     d
nowrp       shld    newdma          ;Save the updated DMA address
            mov     b,h
            mov     c,l
            call    djdma           ;Set up the new DMA address
            lxi     b,retries*100h+0;Maximum # of errors, track #
wrmfred     push    b
            call    djtrk           ;Set up the proper track
            call    djread          ;Read the sector
            pop     b
            jnc     t0boot          ;Continue if no error
            dcr     b
            jnz     wrmfred         ;Keep trying if error
            jmp     djerr           ;To many errors, flash the light

;Load track 1, sector 1, sector 3 (partial), sector 2 (1024 byte sectors)

t1boot      mvi     c,1             ;Track 1
            call    djtrk
            lxi     b,ccp+0b00h     ;Address for sector 1
            lxi     d,10*100h+1     ;Retry count + sector 1
            call    wrmread
            lxi     b,ccp+0f00h     ;Address for sector 2
            lxi     d,10*100h+3     ;Retry count + sector 3
            call    wrmread

            lxi     b,0300h         ;Size of partial sector
            lxi     d,ccp+1300h     ;Address for sector 3
            lxi     h,ccp+0f00h     ;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+0f00h     ;Address for sector 2
        lxi     d,10*100h+2     ;Retry count + sector 2
        call    wrmread

        jmp     gocpm           ;All done, do last inits...

wrmread push    d
        call    djdma           ;Set DMA address
        pop     b
        call    djsec           ;Set sector
wrmfrd  push    b               ;Save error count
        call    djread          ;Read a sector
        jc      wrmerr          ;Do retry stuff on error
        call    djstat          ;Sector size must be 1024 bytes
        ani     0ch             ;Mask length bits
        sui     0ch             ;Carry (error) will be set if < 0c0h
wrmerr  pop     b               ;Fetch retry count
        rnc                     ;Return if no error
        dcr     b               ;Bump error count
        jnz     wrmfrd
        jmp     djerr           ;Error, flash the light

        endif

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

setsec  mov     h,b
        mov     l,c
        shld    cpmsec
donop   ret                     ;Null SINGLE.COM hookup for no floppies

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

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

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

home    mvi     c,0             ;Track to seek to

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

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

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

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

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

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

sectran equ     $
        endif

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

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

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

;*****************************************************************
;*                                                               *
;* Setdrv selects the next drive to be used in read/write        *
;* operations. If the drive has never been selected before, a    *
;* parameter table is created which correctly describes the      *
;* diskette currently in the drive. Diskettes can be of four     *
;* different sector sizes:                                       *
;*       1) 128 bytes single density.                            *
;*       2) 256 bytes double density.                            *
;*       3) 512 bytes double density.                            *
;*       4) 1024 bytes double density.                           *
;*                                                               *
;*****************************************************************

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

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

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

        if      (maxflop ne Ø) and first
        mov     c,a             ;Save drive #
        mvi     a,Ø             ;Have the floppies been accessed yet ?
flopflg equ     $-1
        ana     a
        jnz     flopok
        mvi     b,17            ;Floppies havn't been accessed
        lxi     h,djboot        ;Check if 2D controller is installed
        mvi     a,(jmp)
clopp   cmp     m
        jnz     zret
        inx     h
        inx     h
        inx     h
        dcr     b
        jnz     clopp
        lxi     d,djinit        ;Initialization sequence
        lxi     h,origin+7e2h   ;Load address
        mvi     b,3Ø            ;Byte count
        call    movlop
        mvi     a,Øffh          ;Start 1791
        sta     dreg
        mvi     a,clrcmd        ;1791 reset
```

```
          sta     cmdreg
          jmp     djnext

djinit    db      Ø, Ø, Ø, 18h, Ø, Ø, 8, Ø, 7eh, Ø, 8, Ø, 9, Øffh, 9, Øffh
          db      9, Øffh, 9, Øffh, 9, Ø, 1, Ø, Ø, Ø, Ø, Ø, Ø, Ø

djnext    mvi     a,1             ;Save 2D initialized flag
          sta     flopflg
          endif
          if      maxflop ne Ø
flopok    lxi     h,1             ;Select sector 1 of track 1
          shld    truesec
          mvi     a,1
          sta     cpmtrk
          call    fill            ;Flush buffer and refill
          jc      zret            ;Test for error return
          call    djstat          ;Get status on current drive
          ani     Øch             ;Strip off unwanted bits
          push    psw             ;Used to select a DPB
          rar
          lxi     h,xlts          ;Table of XLT addresses
          mov     e,a
          mvi     d,Ø
          dad     d
          push    h               ;Save pointer to proper XLT
          call    getdpb          ;Get DPH pointer into DE
          xchg                    ;
          pop     d
          mvi     b,2             ;Number of bytes to move
          call    movlop          ;Move the address of XLT
          lxi     d,8             ;Offset to DPB pointer
          dad     d               ;HL <- &DPH.DPB
          push    h
          lhld    origin+7        ;Get address of DJ terminal out routine
          inx     h               ;Bump to look at address of
                                  ;       uart status location
          mov     a,m
          xri     3               ;Adjust for proper rev DJ
          mov     l,a
          mvi     h,(origin+3ØØh)/1ØØh
          mov     a,m
          ani     dblsid          ;Check double sided bit
          lxi     d,dpbl28s       ;Base for single sided DPB's
          jnz     sideok
          lxi     d,dpbl28d       ;Base of double sided DPB's
sideok    xchg                    ;HL <- DBP base, DE <- &DPH.DPB
          pop     d               ;Restore DE (pointer into DPH)
          pop     psw             ;Offset to correct DPB
          ral
          ral
          mov     c,a
          mvi     b,Ø
          dad     b
          xchg                    ;Put DPB address in DPH
          mov     m,e
          inx     h
          mov     m,d
          endif

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

```
            if      maxhd ne Ø
drvhd       call    divlog          ;Divide by logical disks per drive
            mov     a,c
            sta     hddisk
            call    drvptr
            mov     a,m
            inr     a
            jnz     setdrv1
            ori     null            ;Select drive
            out     hdfunc
            mvi     a,scenbl        ;Enable the controller
            out     hdcntl
            mvi     c,239           ;Wait approx 2 minutes for Disk to ready
            lxi     h,Ø
tdelay      dcx     h
            mov     a,h
            ora     l
            cz      dcrc
            rz
            in      hdstat          ;Test if ready yet
            ani     drvrdy
            jnz     tdelay
            if      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
indx1       in      hdstat
            ana     c
            cmp     b               ;Loop util index level changes
            jz      indx1
indx2       inx     h
            in      hdstat          ;Start counting until index returns to
            ana     c               ;       previous state
            cmp     b
            jnz     indx2

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

            if      m26             ;Shugart M26's have 3Ø ms head settle
            xra     a
            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
            endif

setdrv1     call    getdpb          ;Get address of DPB in HL
            lxi     b,15            ;Offset to sector size
            dad     b
            mov     a,m             ;Get sector size
            ani     7h
            sta     secsiz
            mov     a,m
            rar
```

```
        rar
        rar
        rar
        ani     0fh
        sta     secpsec
        xchg                    ;HL <- DPH
        ret

zret    lxi     h,0             ;Seldrv error exit
        ret

        if      maxhd ne 0
dcrc    dcr     c               ;Conditional decrement C routine
        ret

divlog  mvi     c,0
divlogx sui     logdsk
        rc
        inr     c
        jmp     divlogx
        endif

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

getdpb  lda     cpmdrv
        mov     l,a             ;Form offset
        mvi     h,0
        dad     h
        dad     h
        dad     h
        dad     h
        lxi     d,dpbase        ;Base of DPH's
        dad     d
        push    h               ;Save address of DPH
        lxi     d,10            ;Offset to DPB
        dad     d
        mov     a,m             ;Get low byte of DPB address
        inx     h
        mov     h,m             ;Get low byte of DPB
        mov     l,a
        pop     d
        ret

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

        if      maxflop ne 0
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 1024 byte sectors
        endif

;*****************************************************************
;*                                                               *
;* Write routine moves data from memory into the buffer. If the  *
;* desired CP/M sector is not contained in the disk buffer, the  *
```

```
* buffer is first flushed to the disk if it has ever been       *
* written into, then a read is performed into the buffer to get *
* the desired sector. Once the correct sector is in memory, the *
* buffer written indicator is set, so the buffer will be        *
* flushed, then the data is transferred into the buffer.        *
*                                                               *
*****************************************************************

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

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

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

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

redwrt  mvi     b,0             ;The 0 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     80h             ;Save only the side bit
        mov     c,a             ;Remember the side
        mov     a,h
        ani     7fh             ;Forget the side bit
        mov     h,a
        dcx     h               ;Temporary adjustment
divloop dcr     b               ;Update repeat count
        jz      divdone
        ora     a
        mov     a,h
        rar
        mov     h,a
        mov     a,l
        rar                     ;Divide the CP/M sector # by the size
                                ;       of the physical sectors
        mov     l,a
        jmp     divloop         ;
divdone inx     h
        mov     a,h
        ora     c               ;Restore the side bit
        mov     h,a
```

```
        shld    truesec         ;Save the physical sector number
        lxi     h,cpmdrv        ;Pointer to desired drive,track, and sector
        lxi     d,bufdrv        ;Pointer to buffer drive,track, and sector
        mvi     b,5             ;Count loop
dtslop  dcr     b               ;Test if done with compare
        jz      move            ;Yes, match. Go move the data
        ldax    d               ;Get a byte to compare
        cmp     m               ;Test for match
        inx     h               ;Bump pointers to next data item
        inx     d
        jz      dtslop          ;Match, continue testing

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

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

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

move    lda     cpmsec          ;Get the CP/M sector to transfer
        dcr     a               ;Adjust to proper sector in buffer
        ani     Ø               ;Strip off high ordered bits
secpsec equ     $-1             ;The Ø is modified to represent the # of
                                ;       CP/M sectors per physical sectors
        mov     l,a             ;Put into HL
        mvi     h,Ø
        dad     h               ;Form offset into buffer
        dad     h
        dad     h
        dad     h
        dad     h
        dad     h
        dad     h
        lxi     d,buffer        ;Beginning address of buffer
        dad     d               ;Form beginning address of 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    mover
        xra     a
        ret

into    xchg                    ;
        call    mover           ;Move the data, HL = destination
                                ;       DE = source
        mvi     a,1
        sta     bufwrtn         ;Set buffer written into flag
        mvi     a,Ø             ;Check for directory write
writtyp equ     $-1
        dcr     a
```

```
        mvi     a,Ø
        sta     writtyp         ;Set no directory write
        rnz                     ;No error exit

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

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

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

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

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

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

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

```
        if      maxflop ne Ø
        call    djdrv           ;Select the drive
        endif
        endif

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

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

        pop     b               ;Restore track #

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

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

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

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

        if      (maxhd ne Ø) and (maxflop ne Ø)
        lxi     h,djsec
        lxi     d,hdsec
```

```
            call    decidgo
            else
            if      maxhd ne 0
            call    hdsec
            endif
            if      maxflop ne 0
            call    djsec           ;Select the side
            endif
            endif

            lxi     b,buffer        ;Set the DMA address

            if      (maxhd ne 0) and (maxflop ne 0)
            lxi     h,djdma
            lxi     d,hddma
            call    decidgo
            else
            if      maxhd ne 0
            call    hddma
            endif
            if      maxflop ne 0
            call    djdma           ;Select the side
            endif
            endif

            call    0               ;Get operation address
retryop     equ     $-2
            pop     b               ;Restore the retry counter
            mvi     a,0             ;No error exit status
            rnc                     ;Return no error
            dcr     b               ;Update the retry counter
            stc                     ;Assume retry count expired
            mvi     a,0ffh          ;Error return
            rz                      ;Return sad news
            mov     a,b
            cpi     retries/2       ;reseek after half retries done
            jnz     retrylp         ;Try again
            push    b

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

            pop     b
            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
            mvi     b,4             ;Number of bytes to move
```

```
        call    movlop          ;Copy the data

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

fread   equ     $
        if      (maxhd ne 0) and (maxflop ne 0)
        lxi     h,djread
        lxi     d,hdread
        call    decide
        else
        if      maxhd ne 0
        lxi     h,hdread
        endif
        if      maxflop ne 0
        lxi     h,djread        ;Select the side
        endif
        endif
        jmp     prep            ;Select drive, track, and sector.
                                ;       Then read the buffer

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

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

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

        if      (maxhd ne 0) and (maxflop ne 0)
decidgo call    decide  ;which controller ?
        pchl
        endif

        if      (maxhd ne 0) and (maxflop ne 0)
decide  lda     bufdrv          ;Get proper routine into H&L, based
        if      first           ;   on currently selected drive
        cpi     maxhd*logdsk
        rnc
        else
```

```
        cpi     maxflop
        rc
        endif
        xchg
        ret
        endif

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

        if      maxhd ne 0
hddrv   mov     a,c             ;Select Hard Disk drive
        call    divlog          ;Get the physical drive #
        mov     a,c
        sta     hddisk          ;Select the drive
        ori     null
        out     hdfunc
        mvi     a,wenabl
        out     hdcntl
        ret

hdhome  call    drvptr
        mvi     m,0             ;Set track to zero
        in      hdstat          ;Test status
        ani     tkzero          ;At track zero ?
        rz                      ;Yes

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

        else

        xra     a
        jmp     accok
        endif

        if      not fujitsu
delay   lxi     h,0             ;Get delay
settle  equ     $-2
deloop  dcx     h               ;Wait 20ms
        mov     a,h
        ora     l
        inx     h
        dcx     h
        jnz     deloop
        ret
        endif

hdtrk   call    drvptr          ;Get pointer to current track
        mov     e,m             ;Get current track
        mov     m,c             ;Update the track
        mov     a,e             ;Need to seek at all ?
        sub     c
        rz
        cmc                     ;Get carry into direction
        jc      hdtrk2
```

```
            cma
            inr     a
            if      fujitsu
hdtrk2      jmp     accok
            else
hdtrk2      call    accok
            jmp     delay
            endif

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

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

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

            if      m26
hdsec       mvi     a,Ø1fh          ;For compatibility with Cbios rev 2.3, 2.4
            ana     c
            cz      getspt
            sta     hdsectr
            mvi     a,Øe0h
            ana     c
            rlc
            rlc
            rlc
            sta     head
getspt      mvi     a,hdspt
            ret

            else

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

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

hdread      call    hdprep
```

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

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

process   in      hdstat          ;Wait for command to finish
          mov     b,a
```

```
            ani     opdone
            jz      process
            mvi     a,dskclk
            out     hdcntl
            in      hdstat
            ani     tmout           ;Timed out ?
            stc
            rnz
            in      hdreslt
            ani     retry           ;Any retries ?
            stc
            rnz
            xra     a
            ret

hdprep      in      hdstat
            ani     drvrdy
            stc
            rnz
            mvi     a,isbuff        ;Initialize pointer
            out     hdcmnd
            call    build
            ori     0ch
            out     hdfunc
            lda     head
            out     hddata          ;Form head byte
            call    drvptr
            mov     a,m             ;Form track byte
            out     hddata
            ana     a
            mvi     b,80h
            jz      zkey
            mvi     b,0
zkey        mvi     a,0             ;Form sector byte
hdsectr     equ     $-1
            out     hddata
            mov     a,b
            out     hddata
            mvi     a,dskclk
            out     hdcntl
            mvi     a,wenabl
            out     hdcntl
            xra     a
            ret

drvptr      lhld    hddisk
            xchg
            mvi     d,0
            lxi     h,drives
            dad     d
            ret

build       mvi     a,0
head        equ     $-1
            ral
            ral
            ral
            ral
            ori     0
hddisk      equ     $-1
            xri     0f0h
            ret

drives      equ     $
            rept    maxhd
            db      0ffh
```

```
        endm
        endif

;*****************************************************************
;*                                                               *
;* Xlt tables (sector skew tables) for CP/M 2.0. 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 0 sectors 6 and 8. They are       *
;* loaded into memory in the Cbios ram by the cold boot routine. *
;*                                                               *
;*****************************************************************

        if      maxflop ne 0
xlt128  db      0
        db      1,7,13,19,25
        db      5,11,17,23
        db      3,9,15,21
        db      2,8,14,20,26
        db      6,12,18,24
        db      4,10,16,22

xlt256  db      0
        db      1,2,19,20,37,38
        db      3,4,21,22,39,40
        db      5,6,23,24,41,42
        db      7,8,25,26,43,44
        db      9,10,27,28,45,46
        db      11,12,29,30,47,48
        db      13,14,31,32,49,50
        db      15,16,33,34,51,52
        db      17,18,35,36

xlt512  db      0
        db      1,2,3,4,17,18,19,20
        db      33,34,35,36,49,50,51,52
        db      5,6,7,8,21,22,23,24
        db      37,38,39,40,53,54,55,56
        db      9,10,11,12,25,26,27,28
        db      41,42,43,44,57,58,59,60
        db      13,14,15,16,29,30,31,32
        db      45,46,47,48

xlt124  db      0
        db      1,2,3,4,5,6,7,8
        db      25,26,27,28,29,30,31,32
        db      49,50,51,52,53,54,55,56
        db      9,10,11,12,13,14,15,16
        db      33,34,35,36,37,38,39,40
        db      57,58,59,60,61,62,63,64
        db      17,18,19,20,21,22,23,24
        db      41,42,43,44,45,46,47,48

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

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

```
;*****************************************************************

dpb128s dw      26              ;CP/M sectors/track
        db      3               ;BSH
        db      7               ;BLM
        db      Ø               ;EXM
        dw      242             ;DSM
        dw      63              ;DRM
        db      Øc0h            ;AL0
        db      Ø               ;AL1
        dw      16              ;CKS
        dw      2               ;OFF
        db      1h              ;16*((#cpm sectors/physical sector) -1) +
                                ;log2(#bytes per sector/128) + 1 +
                                ;8 if double sided.

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

dpb256s dw      52              ;CP/M sectors/track
        db      4               ;BSH
        db      15              ;BLM
        db      Ø               ;EXM
        dw      242             ;DSM
        dw      127             ;DRM
        db      Øc0h            ;AL0
        db      Ø               ;AL1
        dw      32              ;CKS
        dw      2               ;OFF
        db      12h             ;16*((#cpm sectors/physical sector) -1) +
                                ;log2(#bytes per sector/128) + 1 +
                                ;8 if double sided.

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

dpb512s dw      60              ;CP/M sectors/track
        db      4               ;BSH
        db      15              ;BLM
        db      Ø               ;EXM
        dw      28Ø             ;DSM
        dw      127             ;DRM
        db      Øc0h            ;AL0
        db      Ø               ;AL1
        dw      32              ;CKS
        dw      2               ;OFF
        db      33h             ;16*((#cpm sectors/physical sector) -1) +
                                ;log2(#bytes per sector/128) + 1 +
                                ;8 if double sided.

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

dp1Ø24s dw      64              ;CP/M sectors/track
```

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

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

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

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

dpb256d dw      1Ø4             ;CP/M sectors/track
        db      4               ;BSH
        db      15              ;BLM
        db      Ø               ;EXM
        dw      486             ;DSM
        dw      255             ;DRM
        db      Øf Øh           ;ALØ
        db      Ø               ;AL1
        dw      64              ;CKS
        dw      2               ;OFF
        db      1ah

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

dpb512d dw      12Ø             ;CP/M sectors/track
        db      4               ;BSH
        db      15              ;BLM
        db      Ø               ;EXM
        dw      561             ;DSM
        dw      255             ;DRM
        db      Øf Øh           ;ALØ
        db      Ø               ;AL1
```

```
        dw      64              ;CKS
        dw      2               ;OFF
        db      3bh

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

dpl024d dw      128             ;CP/M sectors/track
        db      4               ;BSH
        db      15              ;BLM
        db      0               ;EXM
        dw      599             ;DSM
        dw      255             ;DRM
        db      0f0h            ;AL0
        db      0               ;AL1
        dw      64              ;CKS
        dw      2               ;OFF
        db      7ch
        endif

;*****************************************************************
;*                                                               *
;* The following DPB's are for the standard format to be         *
;* compatable with older versions of the Cbios.                  *
;*                                                               *
;*****************************************************************

        if      stdlog eq 0     ;Use standard format

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

dpbhd3  dw      1024            ;CP/M sectors/track
        db      5               ;BSH
        db      31              ;BLM
        db      1               ;EXM
```

```
            dw      1973            ;DSM
            dw      511             ;DRM
            db      0ffh            ;AL0
            db      0ffh            ;AL1
            dw      0               ;CKS
            dw      127             ;OFF
            db      33h             ;16*((#cpm sectors/physical sector) -1) +
                                    ;log2(#bytes per sector/128) + 1 +
                                    ;8 if double sided.
            endif
            if      m10 ne 0
dpbhd1      dw      336             ;CP/M sectors/track
            db      5               ;BSH
            db      31              ;BLM
            db      1               ;EXM
            dw      1269            ;DSM
            dw      511             ;DRM
            db      0ffh            ;AL0
            db      0ffh            ;AL1
            dw      0               ;CKS
            dw      1               ;OFF
            db      33h             ;16*((#cpm sectors/physical sector) -1) +
                                    ;log2(#bytes per sector/128) + 1 +
                                    ;8 if double sided.
dpbhd2      dw      336             ;CP/M sectors/track
            db      5               ;BSH
            db      31              ;BLM
            db      1               ;EXM
            dw      1280            ;DSM
            dw      511             ;DRM
            db      0ffh            ;AL0
            db      0ffh            ;AL1
            dw      0               ;CKS
            dw      122             ;OFF
            db      33h             ;16*((#cpm sectors/physical sector) -1) +
                                    ;log2(#bytes per sector/128) + 1 +
                                    ;8 if double sided.
            endif
            if      m20 ne 0
dpbhd1      dw      672             ;CP/M sectors/track
            db      5               ;BSH
            db      31              ;BLM
            db      1               ;EXM
            dw      2015            ;DSM
            dw      511             ;DRM
            db      0ffh            ;AL0
            db      0ffh            ;AL1
            dw      0               ;CKS
            dw      1               ;OFF
            db      33h             ;16*((#cpm sectors/physical sector) -1) +
                                    ;log2(#bytes per sector/128) + 1 +
                                    ;8 if double sided.
dpbhd2      dw      672             ;CP/M sectors/track
            db      5               ;BSH
            db      31              ;BLM
            db      1               ;EXM
            dw      2015            ;DSM
            dw      511             ;DRM
            db      0ffh            ;AL0
            db      0ffh            ;AL1
            dw      0               ;CKS
            dw      98              ;OFF
            db      33h             ;16*((#cpm sectors/physical sector) -1) +
                                    ;log2(#bytes per sector/128) + 1 +
                                    ;8 if double sided.
```

```
dpbhd3  dw      672             ;CP/M sectors/track
        db      5               ;BSH
        db      31              ;BLM
        db      1               ;EXM
        dw      1028            ;DSM
        dw      511             ;DRM
        db      0ffh            ;AL0
        db      0ffh            ;AL1
        dw      0               ;CKS
        dw      195             ;OFF
        db      33h             ;16*((#cpm sectors/physical sector) -1) +
                                ;log2(#bytes per sector/128) + 1 +
                                ;8 if double sided.
        endif
        endif
        endif

;*****************************************************************
;*                                                               *
;* The following DPB's are used when the user selectes the       *
;* number of logical drives. These macros divide evenly the      *
;* space per logical drive where the standard format tries       *
;* to create the least amount of logical drives with the         *
;* most space per logical drive.                                 *
;*                                                               *
;*****************************************************************

        if      stdlog ne 0

mdpbhd  macro   l,d
dpbhd&l dw      secpt
        db      bsh
        db      blm
        db      exm
        if      ldsk ne 0
        dw      (totbls/logdsk)
        else
        dw      (totbls/logdsk)-1               ;Reserved cpm track
        endif
        dw      drm
        db      al0
        db      all
        dw      cks
        dw      (tracks/logdsk)*&d+1
        db      slog
        endm

        if      maxhd ne 0
        if      ml0 ne 0
secpt   equ     336             ;Sectors per track
totbls  equ     2562            ;Total blocks (4096 byte)
tracks  equ     244             ;Total tracks
        endif

        if      m20 ne 0
secpt   equ     672
totbls  equ     5124
tracks  equ     244
        endif

        if      m26 ne 0
secpt   equ     1024
totbls  equ     6464
tracks  equ     202
        endif
```

```
        bsh     equ     5
        blm     equ     31
        exm     equ     1
        drm     equ     511
        al0     equ     0ffh
        all     equ     0ffh
        cks     equ     0
        slog    equ     33h

ldsk    set     0
        rept    maxhd
dpbdrv  set     0
        rept    stdlog
        mdpbhd  %ldsk,%dpbdrv
ldsk    set     ldsk+1
dpbdrv  set     dpbdrv+1
        endm
        endm
        endif
        endif

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

        if      stdlog eq 0
header  macro   nd,dpb
        dw      0               ;Translation table filled in later
        dw      0,0,0           ;Scratch
        dw      dirbuf          ;Directory buffer
        dw      dpb             ;DPB filled in later
        dw      csv&nd          ;Directory check vector
        dw      alv&nd          ;Allocation vector
        endm

        else

header  macro   nd,dpb,dpno
        dw      0               ;Translation table filled in later
        dw      0,0,0           ;Scratch
        dw      dirbuf          ;Directory buffer
        dw      dpb&dpno        ;DPB filled in later
        dw      csv&nd          ;Directory check vector
        dw      alv&nd          ;Allocation vector
        endm
        endif

dpbase  equ     $

        if      stdlog eq 0
dn      set     0
        if      first
        rept    maxhd           ;Generate Hard Disk DPH's followed
        header  %dn,dpbhd1      ;       by Floppy DPH's
dn      set     dn+1
        header  %dn,dpbhd2
dn      set     dn+1
        if      (m26 ne 0) or (m20 ne 0)
        header  %dn,dpbhd3
dn      set     dn+1
        endif
        endm
        rept    maxflop
        header  %dn,0
```

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

        if      stdlog ne Ø
        if      first
dn      set     maxflop
        rept    maxhd
        rept    stdlog                  ;Generate Hard Disk DPH's followed
        header  %dn,dpbhd,%(dn-maxflop) ;by Floppy DPH's
dn      set     dn+1
        endm
        endm
dn      set     Ø               ;Floppies always start at zero
        rept    maxflop
        header  %dn,Ø,Ø
dn      set     dn+1
        endm

        else                    ;Generate floppies before hard disk

dn      set     Ø
        rept    maxflop
        header  %dn,Ø,Ø
dn      set     dn+1
        endm
dn      set     maxflop
        rept    maxhd
        rept    stdlog
        header  %dn,dpbhd,%(dn-maxflop)
dn      set     dn+1
        endm
        endm
        endif
        endif

buffer  equ     $

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

prompt  db      8Øh, clear              ;Clean buffer and screen
        db      acr,alf,acr,alf,acr,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/10+'0'
        db      '.'
        db      (cpmrev mod 10)+'0'
        db      ', Cbios rev '
        db      revnum/10+'0','.'       ;Cbios revision number
        db      revnum mod 10+'0'

        if      maxhd ne 0
        db      '.'
        db      mrev/10+'0'
        db      mrev mod 10+'0'
        endif

        if      m10
        if      fujitsu
        db      'F'
        else
        db      'M'
        endif
        endif

        db      acr,alf
        db      'For '

        if      maxflop ne 0
        db      'a Disk Jockey 2D/B'
        endif

        if      (maxhd ne 0) and (maxflop ne 0)
        db      ' and '
        else
        db      '.'
        endif

        if      maxhd ne 0
        if      maxhd eq 1
        db      'a '
        endif
        if      maxhd eq 2
        db      'two '
        endif
        if      maxhd eq 3
        db      'three '
        endif
        if      maxhd eq 4
        db      'four '
        endif
        if      mrev eq 10
        if      fujitsu
        db      'Fujitsu '
        else
        db      'Memorex '
        endif
        db      'M10 '
        endif
        if      mrev eq 20
        db      'Fujitsu M20 '
        endif
        if      mrev eq 26
        db      'Shugart M26 '
        endif
        db      'hard disk'
        if      maxhd ne 1
        db      's'
        endif
        db      '.'
```

```
        endif
        db      acr,alf

        if      (contyp eq Ø) or (contyp eq 1)
        db      'Nothing', acr, alf
        endif

        if      contyp eq 2
        if      multr3
        db      'Multi I/O'
        else
        db      'Decision I'
        endif
        endif
        if      contyp eq 3
        db      '2D/B'
        endif

        db      ' as console'

        if      lsttyp ge 2
        db      ', '

        if      lsttyp eq 2
        db      'serial'
        endif
        if      lsttyp eq 3
        db      'CTS protocol serial'
        endif
        if      lsttyp eq 4
        db      'DSR protocol serial'
        endif
        if      lsttyp eq 5
        db      'Xon/Xoff protocol serial'
        endif
        if      lsttyp eq 6
        db      'Centronics parallel'
        endif
        if      lsttyp eq 7
        db      'Diablo HyType II parallel'
        endif

        db      ' printer as LST:'

        else
        db      '.'
        endif

        db      acr, alf

        db      Ø               ;End of message

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

message mov     a,m             ;Get a character of the message
        inx     h               ;Bump text pointer
        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

;*****************************************************************
;*                                                               *
;* 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

        if      maxflop ne Ø    ;If 2D/B is there then make RAM copy
        lxi     h,djram         ;   of the jump table.
        lxi     d,origin
        mvi     b,33h           ;Size of jump table
        call    movlop          ;Copy table
        endif

        mvi     a,intioby
        sta     iobyte

        if      contyp ne Ø     ;Do not call tinit for PROM's
        call    tinit           ;Initialize the terminal
        endif

        if      lsttyp ne Ø     ;Do not call linit for PROM's
        call    linit           ;Initialize the list device
        endif

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

        if      (maxflop ne Ø) and first
        sta     flopflg
        endif
        lxi     h,bios+3        ;Patch cold boot to warm code
        shld    bios+1
        jmp     gocpm

        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.           *
;*                                                               *
;*****************************************************************

tinit   lda     group           ;Get group byte
        ori     congrp          ;Select console device
        out     grpsel

        in      rbr             ;Clear reciever buffers
        in      rbr
        xra     a
        out     lsr             ;Clear status
        out     ier             ;Set no interrupts
```

```
            if      not multr3      ;Multi I/O has no sense switches
            lda     group           ;Get group byte
            out     grpsel          ;Select group zero to read sense switch
            in      sensesw         ;Get sense switch.
            ani     0e0h            ;Mask in upper three bits
            rlc
            rlc
            rlc                     ;Move into lower 3 bits
            cpi     7               ;check for sense = 7
            push    psw             ;Save value
            lda     group           ;Get group byte
            ori     congrp          ;Reselect serial port group
            out     grpsel
            pop     psw

            jz      valid           ;Do default 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,0
            dad     d               ;Get a pointer into baud rate table
            mov     e,m             ;Get lower byte of word
            inx     h               ;Point to high byte of word
            mov     d,m             ;Get upper byte. (de) now has divisor
            jmp     setit           ;Set baud rate.

btab        dw      1047            ;110 Baud       000
            dw      384             ;300            001
            dw      96              ;1200           010
            dw      48              ;2400           011
            dw      24              ;4800           100
            dw      12              ;9600           101
            dw      6               ;19200          110
            endif

;*****************************************************************
;*                                                               *
;* The following is a list of valid baud rates.  The current     *
;* baud rate is checked on cold boot.  If it is not in the       *
;* vtab table then the baud rate will be set from the defcon     *
;* word found below the Cbios jump table.  If the user           *
;* happens to have a weird baud rate that is not in this         *
;* table or is looking for a way to save space then entries      *
;* can be added or deleted from the table.                       *
;*                                                               *
;*****************************************************************

vtab        dw      2304            ;50 baud
            dw      1536            ;75
            dw      1047            ;110
            dw      857             ;134.5
            dw      768             ;150
            dw      384             ;300
            dw      192             ;600
            dw      96              ;1200
            dw      64              ;1800
            dw      58              ;2000
            dw      48              ;2400
            dw      32              ;3600
            dw      24              ;4800
            dw      16              ;7200
            dw      12              ;9600
            dw      6               ;19200

svtab       equ     ($-vtab)/2      ;Length of the vtab table
```

```
;*****************************************************************
;*                                                               *
;* Valid checks if the divisor latch is a reasonable value.      *
;* If the value seems off then it will get the default baud      *
;* rate from defcon and jump to setit.                           *
;*                                                               *
;*****************************************************************

valid   mvi     a,dlab+wls0+wls1+stb
        out     lcr             ;Access divisor latch
        in      dll             ;Get lower divisor value
        mov     e,a
        in      dlm             ;Get upper divisor value
        mov     d,a
        mvi     a,wls1+wls0+stb
        out     lcr
        lxi     h,vtab          ;Valid baud rate table
        mvi     c,svtab         ;Length of the baud rate table
vloop   mov     a,e
        cmp     m               ;Check low byte
        jnz     vskip           ;First byte is bad
        inx     h
        mov     a,d
        cmp     m               ;Check high byte
        jz      done            ;Baud rate is OK...  Do cleanup
        dcx     h
vskip   inx     h               ;Skip to next entry
        inx     h
        dcr     c               ;Bump entry counter
        jnz     vloop

nvalid  lhld    defcon          ;Get default baud rate
        xchg

setit   mvi     a,dlab+wls1+wls0+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+wls0+stb
        out     lcr

done    xra     a               ;Clear status register
        out     lsr
        ret

        endif                   ;Multi I/O, Decision I

        if      contyp eq 3     ;2D/B console initialization

tinit   call    djtstat         ;Clean input buffer
        rnz                     ;All empty
        call    djcin
        jmp     tinit

        endif                   ;2D/B console

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

linit   lda     group           ;Get group byte
        ori     lstgrp          ;Select list device
        out     grpsel
        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+wls0+wls1
        out     lcr
        in      rbr             ;Clear input buffer
        xra     a
        out     ier             ;No interupts
        ret

        endif

        if      lsttyp eq 6     ;Multi I/O parallel, Centronics

linit   lda     group           ;Get group byte
        ori     denable         ;Set driver enable bit
        sta     group
        out     grpsel          ;Select group zero with drivers enabled
        xra     a
        out     daisil          ;Zero out data
        mvi     a,d9+dl0        ;Set strobe high, init low
        out     daisi0
        mvi     a,10            ;Wait about 50uS for printer to initilize
dloop   dcr     a
        jnz     dloop
        mvi     a,dll+d9+dl0
        out     daisi0
        ret

        endif

        if      lsttyp eq 7     ;Diablo HyType II

;*****************************************************************
;*                                                               *
;* Initialize Diablo HyType printer.  If the printer fails       *
;* to initialize then the output drivers will be turned off      *
;* and any attempts to print will result in redirection to       *
;* the console.                                                  *
;*                                                               *
;*****************************************************************

        if      multr3          ;Multi I/O initialization
linit   lda     group           ;Get group byte
        ori     denable         ;Add driver enable bit
        out     grpsel
        ori     restore         ;Toggle restore high
        out     grpsel
        mvi     a,10            ;Hold line up for 50uS
dloop   dcr     a
        jnz     dloop
        lda     group
        out     grpsel          ;Turn denable and restore off

        else                    ;Mother board initialization

linit   lda     group           ;Get group byte
        out     grpsel          ;Select group zero
        mvi     a,pselect+rlift ;Set select line active, rlift not active
        out     clk
        mvi     a,0ffh
        out     daisi0
        mvi     a,0ffh-restore  ;Strobe restore bit low
        out     daisi0
```

```
            mvi     a,10            ;Wait about 50uS
dloop       dcr     a
            jnz     dloop
            mvi     a,0ffh          ;Raise restore back up
            out     daisi0
            endif

            xra     a
            out     daisil          ;Clear data buffers

            if      multr3          ;Lift ribbon
            lda     group
            ori     denable
            out     grpsel          ;Re-enable the drivers
            mvi     a,0ffh-restore  ;Pull -ribbon lift down
            out     daisi0
            else
            mvi     a,pselect       ;Re-enable drivers and lift ribbon
            out     clk
            endif

linit9      lxi     h,hinc/cperi
            shld    hmi             ;Save hmi = 120/(characters per inch)

            lxi     h,vinc/lperi
            shld    vmi             ;Save vmi = 48/(lines per inch)

            lxi     h,0             ;Other variables default to zero
            shld    vpos
            shld    divpos
            shld    hpos
            shld    dlhpos
            shld    lmar

            call    clrall          ;Clear the TAB array

            xra     a
            sta     kludge          ;Reset TAB clear byte
            sta     dirflg
            sta     grhflg

            ret

            endif

            db      0,0ffh,0

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

            if      maxflop ne 0
            ds      512             ;Additional space for floppies 1k sectors
            endif

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

cpmsec      dw      0               ;CP/M sector #
cpmdrv      db      0               ;CP/M drive #
cpmtrk      db      0               ;CP/M track #
truesec     dw      0               ;Disk Jockey sector that contains CP/M sector
bufdrv      db      0               ;Drive that buffer belongs to
buftrk      db      0               ;Track that buffer belongs to
bufsec      dw      0               ;Sector that buffer belongs to
```

```
dirbuf  ds      128             ;Directory buffer

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

dn      set     Ø

        if      stdlog eq Ø
        if      not first
        rept    maxflop
        alloc   %dn,75,64
dn      set     dn+1
        endm
        rept    maxhd
        if      m26 ne Ø
        alloc   %dn,247,Ø
dn      set     dn+1
        alloc   %dn,247,Ø
dn      set     dn+1
        alloc   %dn,247,Ø
dn      set     dn+1
        endif
        if      m1Ø ne Ø
        alloc   %dn,159,Ø
dn      set     dn+1
        alloc   %dn,161,Ø
dn      set     dn+1
        endif
        if      m2Ø ne Ø
        alloc   %dn,252,Ø
dn      set     dn+1
        alloc   %dn,252,Ø
dn      set     dn+1
        alloc   %dn,129,Ø
dn      set     dn+1
        endif
        endm

        else

        rept    maxhd
        if      m26 ne Ø
        alloc   %dn,247,Ø
dn      set     dn+1
        alloc   %dn,247,Ø
dn      set     dn+1
        alloc   %dn,247,Ø
dn      set     dn+1
        endif
        if      m1Ø ne Ø
        alloc   %dn,159,Ø
dn      set     dn+1
        alloc   %dn,161,Ø
dn      set     dn+1
        endif
        if      m2Ø ne Ø
        alloc   %dn,252,Ø
dn      set     dn+1
        alloc   %dn,252,Ø
dn      set     dn+1
        alloc   %dn,129,Ø
dn      set     dn+1
        endif
```

```
        endm
        rept    maxflop
        alloc   %dn,75,64
dn      set     dn+1
        endm
        endif
        endif

        if      stdlog ne Ø
        if      maxhd ne Ø      ;Make up hard disk allocation vectors
dn      set     maxflop         ;Hard disks always start after floppies
        rept    maxhd
        rept    stdlog
        alloc   %dn,((totbls/logdsk)/8)+1,Ø
dn      set     dn+1
        endm
        endm
        endif

        if      maxflop ne Ø    ;Make up floppy allocation vectors
dn      set     Ø
        rept    maxflop                 ;Floppies first
        alloc   %dn,75,64
dn      set     dn+1
        endm
        endif
        endif

        end
```