```
*****************************************************************
*                                                               *
* CBIOS FOR CP/M VER 2.2 FOR DISK JOCKEY 2D CONTROLLER (ALL     *
* REVS, AND MODELS A & B). HANDLES DISKETTES WITH SECTOR SIZES  *
* OF 128 BYTES SINGLE DENSITY, 256, 512, 1Ø24 BYTES DOUBLE      *
* DENSITY. THERE ARE CONDITIONAL ASSEMBLIES FOR DISKUS HARD     *
* DISK CONTROLLER.                                              *
*                                                               *
* WRITTEN BY BOBBY DALE GIFFORD.                                *
* 12/8/8Ø                                                       *
*                                                               *
* CUSTOMIZED BY JAY O'BRIEN                                     *
* 1/16/82                                                       *
*                                                               *
* DISK MAP OF SECTORS USED BY COLD BOOT, WARM BOOT, FIRMWARE,   *
* AND CP/M:                                                     *
*                                                               *
* TRK Ø SEC  1 = FIRST SECTOR OF COLD BOOT.          E7ØØH      *
*     Ø      2 = COLD BOOT 256.                        8ØH      *
*     Ø      3 = COLD BOOT 512.                        8ØH      *
*     Ø      4 = COLD BOOT 1Ø24.                       8ØH      *
*     Ø      5 = WARM BOOT 256.                        8ØH      *
*     Ø      6 = WARM BOOT 512.                        8ØH      *
*     Ø      7 = WARM BOOT 1Ø24.                       8ØH      *
*     Ø      8 = COLD/WARM BOOT.                    2C ØØH      *
*     Ø      9 = FIRMWARE.                          E4ØØH       *
*     Ø     1Ø = FIRMWARE+8ØH.                      E48ØH       *
*     Ø     11 = FIRMWARE+1ØØH                      E5ØØH       *
*     Ø     12 = FIRMWARE+18ØH.                     E58ØH       *
*     Ø     13 = FIRMWARE+2ØØH.                     E6ØØH       *
*     Ø     14 = FIRMWARE+28ØH.                     E68ØH       *
*     Ø     15 = FIRMWARE+3ØØH.                     E7ØØH       *
*     Ø     16 = FIRMWARE+38ØH.                     E78ØH       *
*     Ø     17 = CCP.                               2700H       *
*     Ø     1Ø = CCP+8ØH.                           2780H       *
*     Ø     12 = CCP+1ØØH.                          2800H       *
*     Ø     14 = CCP+18ØH.                          2880H       *
*     Ø     16 = CCP+2ØØH.                          2900H       *
*     Ø     18 = CCP+28ØH.                          2980H       *
*     Ø     2Ø = CCP+3ØØH.                          2A00H       *
*     Ø     22 = CCP+38ØH.                          2A8ØH       *
*     Ø     24 = CCP+4ØØH.                          2B00H       *
*     Ø     26 = CCP+48ØH.                          2B8ØH       *
*     1        = REST OF CP/M.                2C ØØH-4FFFH      *
*                                                               *
*****************************************************************

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

*****************************************************************
*                                                               *
* THE FOLLOWING REVISION NUMBER IS IN REFERENCE TO THE CP/M     *
* 2.2 CBIOS.                                                    *
*                                                               *
*****************************************************************
```

CBIOS & PRN

only goes to TRACK 1 Sector 52

Really C700 Bios C700-2700 = 4000

CC00 - EFFF

2400h Bytes

9216d ÷ 128 = 72 Sectors !

Signon at E60B.

— CC00

1A06 or 6667 or 53+ Sectors !

```
001C =          REVNUM  EQU     28              ;CBIOS REVISION NUMBER
0016 =          CPMREV  EQU     22              ;CP/M REVISION NUMBER

                ********************************************************************
                *                                                                  *
                * THE FOLLOWING EQUATES SET UP THE RELATIONSHIP BETWEEN THE        *
                * 2D FLOPPIES AND THE HARD DISK CONTROLLERS.                       *
                *                                                                  *
                ********************************************************************

0000 =          FIRST   EQU     0               ;0 = 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
0001 =          MAXHD   EQU     1               ;SET TO NUMBER OF HARD DISKS
0002 =          MAXFLOP EQU     2               ;SET TO NUMBER OF FLOPPIES

0001 =          M26     EQU     1               ;SET ONLY ONE OF THESE VARIABLES
0000 =          M20     EQU     0
0000 =          M10     EQU     0

                        IF      M10 OR M20
                SDELAY  EQU     0               ;SOFTWARE HEAD SETTLE DELAY (0 = NO, 1 = YES)
                        ELSE
0001 =          SDELAY  EQU     1
                        ENDIF

001A =          MREV    EQU     26*M26+20*M20+10*M10    ;HARD DISK TYPE
0003 =          LOGDSK  EQU     3*M26+3*M20+2*M10       ;LOGICAL DISKS PER DRIVE
0020 =          HDSPT   EQU     32*M26+21*M20+21*M10    ;SECTORS PER TRACK


                ********************************************************************
                *                                                                  *
                * THE FOLLOWING EQUATES RELATE THE THINKER TOYS 2D CONTROLLER.     *
                * IF THE CONTROLLER IS NON STANDARD (0E000H) ONLY THE ORIGIN       *
                * EQUATE NEED BE CHANGED. THIS VERSION OF THE CBIOS WILL WORK      *
                * WITH 2D CONTROLLER BOARDS REV 0, 1, 3, 3.1, 4, MODEL B.          *
                *                                                                  *
                ********************************************************************


                        IF      MAXFLOP NE 0    ;INCLUDE DISCUS 2D ?
F000 =          ORIGIN  EQU     0F000H
F400 =          DJRAM   EQU     ORIGIN+400H     ;DISK JOCKEY 2D RAM ADDRESS
F400 =          DJBOOT  EQU     DJRAM           ;DISK JOCKEY 2D INITIALIZATION
F003 =          DJCIN   EQU     ORIGIN+3H       ;DISK JOCKEY 2D CHARACTER INPUT ROUTINE
F006 =          DJCOUT  EQU     ORIGIN+6H       ;DISK JOCKEY 2D CHARACTER OUTPUT ROUTINE
F409 =          DJHOME  EQU     DJRAM+9H        ;DISK JOCKEY 2D TRACK ZERO SEEK
F40C =          DJTRK   EQU     DJRAM+0CH       ;DISK JOCKEY 2D TRACK SEEK ROUTINE
F40F =          DJSEC   EQU     DJRAM+0FH       ;DISK JOCKEY 2D SET SECTOR ROUTINE
F412 =          DJDMA   EQU     DJRAM+012H      ;DISK JOCKEY 2D SET DMA ADDRESS
F415 =          DJREAD  EQU     DJRAM+15H       ;DISK JOCKEY 2D READ ROUTINE
F418 =          DJWRITE EQU     DJRAM+18H       ;DISK JOCKEY 2D WRITE ROUTINE
F41B =          DJSEL   EQU     DJRAM+1BH       ;DISK JOCKEY 2D SELECT DRIVE ROUTINE
F021 =          DJTSTAT EQU     ORIGIN+21H      ;DISK JOCKEY 2D TERMINAL STATUS ROUTINE
```

```
F427 =          DJSTAT  EQU     DJRAM+27H       ;DISK JOCKEY 2D STATUS ROUTINE
F42A =          DJERR   EQU     DJRAM+2AH       ;DISK JOCKEY 2D ERROR, FLASH LED
F42D =          DJDEN   EQU     DJRAM+2DH       ;DISK JOCKEY 2D SET DENSITY ROUTINE
F430 =          DJSIDE  EQU     DJRAM+30H       ;DISK JOCKEY 2D SET SIDE ROUTINE
0008 =          DBLSID  EQU     8               ;SIDE BIT FROM CONTROLLER
                ENDIF

                ;*****************************************************************
                ;*                                                               *
                ;* THE FOLLOWING EQUATES ARE FOR THE DISKUS HARD DISK WANTED.    *
                ;*                                                               *
                ;*****************************************************************

                IF      MAXHD NE 0      ;WANT HARD DISK INCLUDED ?
0050 =          HDORG   EQU     50H             ;HARD DISK CONTROLLER ORIGIN
0050 =          HDSTAT  EQU     HDORG           ;HARD DISK STATUS
0050 =          HDCNTL  EQU     HDORG           ;HARD DISK CONTROL
0053 =          HDDATA  EQU     HDORG+3         ;HARD DISK DATA
0052 =          HDFUNC  EQU     HDORG+2         ;HARD DISK FUNCTION
0051 =          HDCMND  EQU     HDORG+1         ;HARD DISK COMMAND
0051 =          HDRESLT EQU     HDORG+1         ;HARD DISK RESULT
0002 =          RETRY   EQU     2               ;RETRY BIT OF RESULT
0001 =          TKZERO  EQU     1               ;TRACK ZERO BIT OF STATUS
0002 =          OPDONE  EQU     2               ;OPERACTION DONE BIT OF STATUS
0004 =          COMPLT  EQU     4               ;COMPLETE BIT OF STATUS
0008 =          TMOUT   EQU     8               ;TIME OUT BIT OF STATUS
0010 =          WFAULT  EQU     10H             ;WRITE FAULT BIT OF STATUS
0020 =          DRVRDY  EQU     20H             ;DRIVE READY BIT OF STATUS
0040 =          INDEX   EQU     40H             ;INDEX BIT OF STATUS
0004 =          PSTEP   EQU     4               ;STEP BIT OF FUNCTION
00FB =          NSTEP   EQU     0FBH            ;STEP BIT MASK OF FUNCTION
0004 =          HDRLEN  EQU     4               ;SECTOR HEADER LENGTH
0200 =          SECLEN  EQU     512             ;SECTOR DATA LENGTH
000F =          WENABL  EQU     0FH             ;WRITE ENABLE
000B =          WRESET  EQU     0BH             ;WRITE RESET OF FUNCTION
0005 =          SCENBL  EQU     5               ;CONTROLLER CONTROL
0007 =          DSKCLK  EQU     7               ;DISK CLOCK FOR CONTROL
00F7 =          MDIR    EQU     0F7H            ;DIRECTION MASK FOR FUNCTION
00FC =          NULL    EQU     0FCH            ;NULL COMMAND
0000 =          IDBUFF  EQU     0               ;INITIALIZE DATA COMMAND
0008 =          ISBUFF  EQU     8               ;INITIALIZE HEADER COMMAND
0001 =          RSECT   EQU     1               ;READ SECTOR COMMAND
0005 =          WSECT   EQU     5               ;WRITE SECTOR COMMAND
                ENDIF

                ;*****************************************************************
                ;*                                                               *
                ;* CP/M SYSTEM EQUATES. IF RECONFIGURATION OF THE CP/M SYSTEM    *
                ;* IS BEING DONE, THE CHANGES CAN BE MADE TO THE FOLLOWING       *
                ;* EQUATES.                                                      *
                ;*                                                               *
                ;*****************************************************************

003C =          MSIZE   EQU     60              ;MEMORY SIZE OF TARGET CP/M
A000 =          BIAS    EQU     (MSIZE-20)*1024 ;MEMORY OFFSET FROM 20K SYSTEM
C700 =          CCP     EQU     2700H+BIAS      ;CONSOLE COMMAND PROCESSOR
```

```
CF00 =          BDOS    EQU     CCP+800H        ;BDOS ADDRESS
DD00 =          BIOS    EQU     CCP+1600H       ;CBIOS ADDRESS
4A00 =          OFFSETC EQU     2700H-BIOS      ;OFFSET FOR SYSGEN
0004 =          CDISK   EQU     4               ;ADDRESS OF LAST LOGGED DISK
0080 =          BUFF    EQU     80H             ;DEFAULT BUFFER ADDRESS
0100 =          TPA     EQU     100H            ;TRANSIENT MEMORY
00C0 =          INTIOBY EQU     192             ;INITIAL IOBYTE
0003 =          IOBYTE  EQU     3               ;IOBYTE LOCATION
0000 =          WBOT    EQU     0               ;WARM BOOT JUMP ADDRESS
0005 =          ENTRY   EQU     5               ;BDOS ENTRY JUMP ADDRESS

                ;*****************************************************************
                ;*                                                               *
                ;* THE FOLLOWING ARE INTERNAL CBIOS EQUATES. MOST ARE MISC.      *
                ;* CONSTANTS.                                                    *
                ;*                                                               *
                ;*****************************************************************

000A =          RETRIES EQU     10              ;MAX RETRIES ON DISK I/O BEFORE ERROR
000D =          ACR     EQU     0DH             ;A CARRIAGE RETURN
000A =          ALF     EQU     0AH             ;A LINE FEED
001A =          CLEAR   EQU     1AH             ;CLEAR SCREEN FOR VIO-X
0003 =          AETX    EQU     3               ;ETX CHARACTER
0006 =          AACK    EQU     6               ;ACK CHARACTER

                ;*****************************************************************
                ;*                                                               *
                ;* THE JUMP TABLE BELOW MUST REMAIN IN THE SAME ORDER, THE       *
                ;* ROUTINES MAY BE CHANGED, BUT THE FUNCTION EXECUTED MUST BE    *
                ;* THE SAME.                                                     *
                ;*                                                               *
                ;*****************************************************************

DD00                    ORG     BIOS            ;CBIOS STARTING ADDRESS

DD00 C370E6             JMP     CBOOT           ;COLD BOOT ENTRY POINT
DD03 C352DF     WBOOTE  JMP     WBOOT           ;WARM BOOT ENTRY POINT
DD06 C336DD             JMP     CONST           ;CONSOLE STATUS ROUTINE
DD09 C342DD             JMP     CONIN           ;CONSOLE INPUT
DD0C C357DD     COUT    JMP     CONOUT          ;CONSOLE OUTPUT
DD0F C377DD             JMP     LIST            ;LIST DEVICE OUTPUT
DD12 C36CDD             JMP     PUNCH           ;PUNCH DEVICE OUTPUT
DD15 C362DD             JMP     READER          ;READER DEVICE INPUT
DD18 C3E7DF             JMP     HOME            ;HOME DRIVE
DD1B C329E0             JMP     SETDRV          ;SELECT DISK
DD1E C3E9DF             JMP     SETTRK          ;SET TRACK
DD21 C3DBDF             JMP     SETSEC          ;SET SECTOR
DD24 C3E1DF             JMP     SETDMA          ;SET DMA ADDRESS
DD27 C32FE1             JMP     READ            ;READ THE DISK
DD2A C328E1             JMP     WRITE           ;WRITE THE DISK
DD2D C382DD             JMP     LISTST          ;LIST DEVICE STATUS
DD30 C3EEDF             JMP     SECTRAN         ;SECTOR TRANSLATION

                        IF      MAXFLOP NE 0
DD33 C31BF4     DJDRV   JMP     DJSEL           ;HOOK FOR SINGLE.COM PROGRAM
                        ELSE
```

```
                        JMP     DONOP
                        ENDIF

                ;*****************************************************************
                ;*                                                               *
                ;* TERMINAL DRIVER ROUTINES. IOBYTE IS INITIALIZED BY THE COLD   *
                ;* BOOT ROUTINE, TO MODIFY, CHANGE THE "INTIOBY" EQUATE. THE     *
                ;* I/O ROUTINES THAT FOLLOW ALL WORK EXACTLY THE SAME WAY. USING *
                ;* IOBYTE, THEY OBTAIN THE ADDRESS TO JUMP TO IN ORDER TO EXECUTE*
                ;* THE DESIRED FUNCTION. THERE IS A TABLE WITH FOUR ENTRIES FOR  *
                ;* EACH OF THE POSSIBLE ASSIGNMENTS FOR EACH DEVICE. TO MODIFY   *
                ;* THE I/O ROUTINES FOR A DIFFERENT I/O CONFIGURATION, JUST      *
                ;* CHANGE THE ENTRIES IN THE TABLES.                             *
                ;*                                                               *
                ;*****************************************************************

F003 =          CITTY   EQU     DJCIN           ;INPUT FROM THE DISK JOCKEY 2D
F006 =          COTTY   EQU     DJCOUT          ;OUTPUT TO THE DISK JOCKEY 2D

                ;*****************************************************************
                ;*                                                               *
                ;* CONST: GET THE STATUS FOR THE CURRENTLY ASSIGNED CONSOLE      *
                ;*        DEVICE. THE CONSOLE DEVICE CAN BE GOTTEN FROM IOBYTE,  *
                ;*        THEN A JUMP TO THE CORRECT CONSOLE STATUS ROUTINE IS   *
                ;*        PERFORMED.                                             *
                ;*                                                               *
                ;*****************************************************************

DD36 21B0DD     CONST   LXI     H,CSTBLE        ;BEGINNING OF JUMP TABLE
DD39 C348DD             JMP     CONIN1          ;SELECT CORRECT JUMP

                ;*****************************************************************
                ;*                                                               *
                ;* CSREADER: IF THE CONSOLE IS ASSIGNED TO THE READER THEN A     *
                ;*           JUMP WILL BE MADE HERE, WHERE ANOTHER JUMP WILL     *
                ;*           OCCUR TO THE CORRECT READER STATUS.                 *
                ;*                                                               *
                ;*****************************************************************

DD3C 21B8DD     CSREADR LXI     H,CSRTBLE       ;BEGINNING OF READER STATUS TABLE
DD3F C365DD             JMP     READERA

                ;*****************************************************************
                ;*                                                               *
                ;* CONIN: TAKE THE CORRECT JUMP FOR THE CONSOLE INPUT ROUTINE.   *
                ;*        THE JUMP IS BASED ON THE TWO LEAST SIGNIFICANT BITS OF *
                ;*        IOBYTE.                                                *
                ;*                                                               *
                ;*****************************************************************

DD42 CDA2E1     CONIN   CALL    FLUSH           ;FLUSH THE DISK BUFFER
DD45 2188DD             LXI     H,CITBLE        ;BEGINNING OF CHARACTER INPUT TABLE

                ;
                ; ENTRY AT CONIN1 WILL DECODE THE TWO LEAST SIGNIFICANT BITS
                ; OF IOBYTE. THIS IS USED BY CONIN,CONOUT, AND CONST.
```

```
                *
DD48 3A0300     CONIN1  LDA     IOBYTE
DD4B 17                 RAL
                *
                * ENTRY AT SELDEV WILL FORM AN OFFSET INTO THE TABLE POINTED
                * TO BY H&L AND THEN PICK UP THE ADDRESS AND JUMP THERE.
                *
DD4C E606       SELDEV  ANI     6H              ;STRIP OFF UNWANTED BITS
DD4E 1600               MVI     D,0             ;FORM OFFSET
DD50 5F                 MOV     E,A
DD51 19                 DAD     D               ;ADD OFFSET
DD52 7E                 MOV     A,M             ;PICK UP HIGH BYTE
DD53 23                 INX     H
DD54 66                 MOV     H,M             ;PICK UP LOW BYTE
DD55 6F                 MOV     L,A             ;FORM ADDRESS
DD56 E9                 PCHL                    ;GO THERE !
                ****************************************************************
                *                                                              *
                * CONOUT: TAKE THE PROPER BRANCH ADDRESS BASED ON THE TWO LEAST *
                *         SIGNIFICANT BITS OF IOBYTE.                          *
                *                                                              *
                ****************************************************************
DD57 C5         CONOUT  PUSH    B               ;SAVE THE CHARACTER
DD58 CDA2E1             CALL    FLUSH           ;FLUSH THE DISK BUFFER
DD5B C1                 POP     B               ;RESTORE THE CHARACTER
DD5C 2190DD             LXI     H,COTBLE        ;BEGINNING OF THE CHARACTER OUT TABLE
DD5F C348DD             JMP     CONIN1          ;DO THE DECODE
                ****************************************************************
                *                                                              *
                * READER: SELECT THE CORRECT READER DEVICE FOR INPUT. THE      *
                *         READER IS SELECTED FROM BITS 2 AND 3 OF IOBYTE.      *
                *                                                              *
                ****************************************************************
DD62 21A8DD     READER  LXI     H,RTBLE         ;BEGINNING OF READER INPUT TABLE
                *
                * ENTRY AT READERA WILL DECODE BITS 2 & 3 OF IOBYTE, USED
                * BY CSREADER.
                *
DD65 3A0300     READERA LDA     IOBYTE
                *
                * ENTRY AT READER1 WILL SHIFT THE BITS INTO POSITION, USED
                * BY LIST AND PUNCH.
                *
DD68 1F         READR1  RAR
DD69 C34CDD             JMP     SELDEV
```

```
                ;*****************************************************************
                ;*                                                               *
                ;* PUNCH: SELECT THE CORRECT PUNCH DEVICE. THE SELECTION COMES   *
                ;*        FROM BITS 4&5 OF IOBYTE.                               *
                ;*                                                               *
                ;*****************************************************************

DD6C 21A0DD     PUNCH   LXI     H,PTBLE         ;BEGINNING OF PUNCH TABLE
DD6F 3A0300             LDA     IOBYTE

                ;
                ; ENTRY AT PNCH1 ROTATES BITS A LITTLE MORE IN PREP FOR
                ; SELDEV, USED BY LIST.
                ;
DD72 1F         PNCH1   RAR
DD73 1F                 RAR
DD74 C368DD             JMP     READR1

                ;*****************************************************************
                ;*                                                               *
                ;* LIST: SELECT A LIST DEVICE BASED ON BITS 6&7 OF IOBYTE        *
                ;*                                                               *
                ;*****************************************************************

DD77 2198DD     LIST    LXI     H,LTBLE         ;BEGINNING OF THE LIST DEVICE ROUTINES
DD7A 3A0300     LIST1   LDA     IOBYTE
DD7D 1F                 RAR
DD7E 1F                 RAR
DD7F C372DD             JMP     PNCH1

                ;*****************************************************************
                ;*                                                               *
                ;* LISTST: GET THE STATUS OF THE CURRENTLY ASSIGNED LIST DEVICE  *
                ;*                                                               *
                ;*****************************************************************

DD82 21C0DD     LISTST  LXI     H,LSTBLE        ;BEGINNING OF THE LIST DEVICE STATUS
DD85 C37ADD             JMP     LIST1

                ;*****************************************************************
                ;*                                                               *
                ;* IF CUSTOMIZING I/O ROUTINES IS BEING PERFORMED, THE TABLE     *
                ;* BELOW SHOULD BE MODIFIED TO REFLECT THE CHANGES. ALL I/O      *
                ;* DEVICES ARE DECODED OUT OF IOBYTE AND THE JUMP IS TAKEN FROM  *
                ;* THE FOLLOWING TABLES.                                         *
                ;*                                                               *
                ;*****************************************************************

                ;
                ; CONSOLE INPUT TABLE
                ;
DD88 F3DD       CITBLE  DW      CIUC1           ;INPUT FROM USER CONSOLE 1 (CURRENTLY
                                                ;       SWBD PARALLEL PORT 4)
```

```
DD8A ØØDE              DW      CICRT           ;INPUT FROM CRT (CURRENTLY SWITCHBOARD
                                               ;       SERIAL PORT 1)
DD8C 62DD              DW      READER          ;INPUT FROM READER (DEPENDS ON READER
                                               ;       SELECTION)
DD8E Ø3FØ              DW      CITTY           ;INPUT FROM TTY (CURRENTLY INPUT FROM
                                               ;       DISK JOCKEY 2D)

                *
                * CONSOLE OUTPUT TABLE
                *

DD9Ø 3BDE       COTBLE DW      COCRT           ;OUTPUT TO CRT
                                               ;
DD92 3BDE              DW      COCRT           ;OUTPUT TO CRT
                                               ;
DD94 77DD              DW      LIST            ;OUTPUT TO LIST DEVICE (DEPENDS ON
                                               ;       BITS 6&7 OF IOBYTE)
DD96 Ø6FØ              DW      COTTY           ;OUTPUT TO TTY (CURRENTLY OUTPUT TO
                                               ;       DISK JOCKEY 2D)
                *
                * LIST DEVICE TABLE
                *

DD98 Ø6FØ       LTBLE  DW      COTTY           ;OUTPUT TO TTY (CURRENTLY ASSIGNED
                                               ;       BY INTIOBY,OUTPUT TO 2D)
DD9A 46DE              DW      COPTR           ;OUTPUT TO PRINTER
                                               ;
DD9C C9DD              DW      COLPT           ;OUTPUT TO LINE PRINTER (CURRENTLY
                                               ;       SWITCHBOARD SERIAL PORT 1)
DD9E D4DD              DW      COUL1           ;OUTPUT TO USER LINE PRINTER 1 (CURRENTLY
                                               ;       SWITCHBOARD SERIAL PORT 1)
                *
                * PUNCH DEVICE TABLE
                *

DDAØ Ø6FØ       PTBLE  DW      COTTY           ;OUTPUT TO THE TTY (CURRENTLY ASSIGNED
                                               ;       BY INTIOBY,OUTPUT TO 2D)
DDA2 46DE              DW      COPTR           ;OUTPUT TO PRINTER
                                               ;
DDA4 C9DD              DW      COUP1           ;OUTPUT TO USER PUNCH 1 (CURRENTLY
                                               ;       SWITCHBOARD SERIAL PORT 1)
DDA6 C9DD              DW      COUP2           ;OUTPUT TO USER PUNCH 2 (CURRNTLLY
                                               ;       SWITCHBOARD SERIAL PORT 1)
                *
                * READER DEVICE INPUT TABLE
                *

DDA8 Ø3FØ       RTBLE  DW      CITTY           ;INPUT FROM TTY (CURRENTLY ASSIGNED
                                               ;       BY INTIOBY, INPUT FROM 2D)
DDAA ØØDE              DW      CIPTR           ;INPUT FROM PAPER TAPE READER (CURRENTLY
                                               ;       SWITCHBOARD SERIAL PORT 1)
DDAC ØØDE              DW      CIUR1           ;INPUT FROM USER READER 1 (CURRENTLY
                                               ;       SWITCHBOARD SERIAL PORT 1)
DDAE ØØDE              DW      CIUR2           ;INPUT FROM USER READER 2 (CURRENTLY
```

```
                                        ;       SWITCHBOARD SERIAL PORT 1)

                *
                * CONSOLE STATUS TABLE
                *

DDBØ FFDD       CSTBLE  DW      CSUC1           ;STATUS FROM SWBD PARALLEL PORT 4, AS
                                                ;       READ FROM ATTN BIT Ø)
DDB2 1CDE               DW      CSCRT           ;STATUS FROM CRT (CURRENTLY SWITCHBOARD
                                                ;       SERIAL PORT 1)
DDB4 3CDD               DW      CSREADR         ;STATUS FROM READER (DEPENDS ON READER DEVICE )
                                                ;
DDB6 14DE               DW      CSTTY           ;STATUS OF TTY (CURRENTLY STSTUS FROM
                                                ;       DISK JOCKEY 2D)
                *
                * STATUS FROM READER DEVICE
                *

DDB8 14DE       CSRTBLE DW      CSTTY           ;STATUS FROM TTY (CURRENTLY ASSIGNED
                                                ;       BY INTIOBY, STATUS OF 2D)
DDBA 1CDE               DW      CSPTR           ;STATUS FROM PAPER TAPE READER (CURRENTLY
                                                ;       SWITCHBOARD SERIAL PORT 1)
DDBC 1CDE               DW      CSUR1           ;STATUS FROM USER READER 1 (CURRENTLY
                                                ;       SWITCHBOARD SERIAL PORT 1)
DDBE 1CDE               DW      CSUR2           ;STATUS OF USER READER 2 (CURRENTLY
                                                ;       SWITCHBOARD SERIAL PORT 1)

                *
                * STATUS FROM LIST DEVICE
                *

DDCØ 2ADE       LSTBLE  DW      READY           ;CONSOLE ALWAYS READY
DDC2 2ADE               DW      READY           ;GET LIST STATUS
DDC4 25DE               DW      LSLPT
DDC6 25DE               DW      LSLPT

                ********************************************************************
                *                                                                  *
                * ROUTINES FOR MY SYSTEM.  J. J. O'BRIEN                           *
                *                                                                  *
                ********************************************************************
DDC8 ØØ                 NOP
                ********************************************************************
                *                                                                  *
                * THE FOLLOWING EQUATES SET OUTPUT DEVICE TO OUTPUT TO THE         *
                * SWITCHBOARD SERIAL PORT 1.                                       *
                *                                                                  *
                ********************************************************************

DDC9 =          COPTP   EQU     $               ;OUTPUT FROM PAPER TAPE PUNCH
DDC9 =          COUP1   EQU     $               ;OUTPUT FROM USER PUNCH 1
DDC9 =          COUP2   EQU     $               ;OUTPUT FROM USER PUNCH 2
DDC9 DBØ2       COLPT   IN      2               ;OUTPUT FROM LINE PRINTER,GET STATUS
DDCB E68Ø               ANI     8ØH             ;WAIT UNTIL OK TO SEND
DDCD CAC9DD             JZ      COLPT
DDDØ 79                 MOV     A,C             ;OUTPUT THE CHARACTER
```

```
DDD1 D301              OUT     1
DDD3 C9                RET

                ********************************************************************
                *                                                                  *
                * CUSTOM I/O PRINTER DRIVER FOR DIABLO PRINTER WITH 1200 BAUD      *
                * ETX/ACK HANDSHAKE.                                               *
                *                                                                  *
                ********************************************************************

DDD4 CDC9DD    COUL1   CALL    COLPT           ;OUTPUT THE CHARACTER
DDD7 3AF2DD            LDA     COUNT
DDDA 3D                DCR     A
DDDB 32F2DD            STA     COUNT
DDDE C0                RNZ
DDDF 3E4E              MVI     A,78
DDE1 32F2DD            STA     COUNT
DDE4 0E03              MVI     C,AETX
DDE6 CDC9DD            CALL    COLPT
DDE9 CD08DE    PWAIT   CALL    CIPTR
DDEC FE06              CPI     AACK
DDEE C2E9DD            JNZ     PWAIT
DDF1 C9                RET

DDF2 32        COUNT   DB      50

                ********************************************************************
                *                                                                  *
                * THE FOLLOWING EQUATES SET THE INPUT TO COME FROM THE SWBD        *
                * PARALLEL PORT 4, WITH STATUS ON ATTENTION PORT BIT 0.            *
                *                                                                  *
                ********************************************************************

DDF3 DB03      CIUC1   IN      3               ;GET ATTENTION BYTE
DDF5 E601              ANI     1               ;GET BIT 0 ONLY
DDF7 CAF3DD            JZ      CIUC1           ;WAIT FOR CHARACTER
DDFA DB04              IN      4               ;GET CHARACTER
DDFC E67F              ANI     7FH             ;STRIP OFF THE PARITY
DDFE C9                RET

DDFF DB03      CSUC1   IN      3               ;GET ATTENTION BYTE
DE01 E601              ANI     1               ;GET BIT 0 ONLY
DE03 EE01              XRI     1               ;CHANGE POLARITY
DE05 C317DE            JMP     STAT            ;RETURN PROPER INDICATION

                ********************************************************************
                *                                                                  *
                * THE FOLLOWING EQUATES SET THE INPUT FROM THE DEVICES TO COME     *
                * FROM THE SWITCHBOARD SERIAL PORT 1.                              *
                *                                                                  *
                ********************************************************************

DE08 =         CICRT   EQU     $               ;INPUT FROM CRT
DE08 =         CIUR1   EQU     $               ;INPUT FROM USER READER 1
DE08 =         CIUR2   EQU     $               ;INPUT FROM USER READER 2
DE08 DB02      CIPTR   IN      2               ;INPUT FROM PAPER TAPE READER, GET STATUS
```

```
DEØA E64Ø           ANI     4ØH             ;WAIT FOR CHARACTER
DEØC CAØ8DE         JZ      CIPTR
DEØF DBØ1           IN      1
DE11 E67F           ANI     7FH             ;STRIP OFF THE PARITY
DE13 C9             RET

                    ;*****************************************************************
                    ;*                                                               *
                    ;* CONSOLE STATUS ROUTINES, TEST IF A CHARACTER HAS ARRIVED.     *
                    ;*                                                               *
                    ;*****************************************************************

DE14 CD21FØ     CSTTY   CALL    DJTSTAT         ;STATUS FROM DISK JOCKEY 2D
DE17 3EØØ       STAT    MVI     A,Ø             ;PREP FOR ZERO RETURN
DE19 CØ                 RNZ                     ;NOTHING FOUND
DE1A 3D                 DCR     A               ;RETURN WITH ØFFH
DE1B C9                 RET

                    ;*****************************************************************
                    ;*                                                               *
                    ;* THE FOLLOWING EQUATES CAUSE THE DEVICES TO GET STATUS FROM    *
                    ;* THE SWITCHBOARD SERIAL PORT 1.                                *
                    ;*                                                               *
                    ;*****************************************************************

DE1C =          CSUR1   EQU     $               ;STATUS OF USER READER 1
DE1C =          CSUR2   EQU     $               ;STATUS OF USER READER 2
DE1C =          CSPTR   EQU     $               ;STATUS OF PAPER TAPE READER
DE1C DBØ2       CSCRT   IN      2               ;STATUS FROM CRT, GET STATUS
DE1E E64Ø               ANI     4ØH             ;STRIP OF DATA READY BIT
DE2Ø EE4Ø               XRI     4ØH             ;MAKE CORRECT POLARITY
DE22 C317DE             JMP     STAT            ;RETURN PROPER INDICATION

                    ;*****************************************************************
                    ;*                                                               *
                    ;* LIST DEVICE STATUS ROUTINES.                                  *
                    ;*                                                               *
                    ;*****************************************************************

DE25 DBØ2       LSLPT   IN      2               ;ALL OTHER DEVICES WAIT
DE27 E68Ø               ANI     8ØH
DE29 C8                 RZ
DE2A 3EFF       READY   MVI     A,ØFFH
DE2C C9                 RET

                    ;*****************************************************************
                    ;*                                                               *
                    ;* THIS INITIALLIZING ROUTINE SAMPLES BIT Ø OF SWBD PORT 7 TO    *
                    ;* DETERMINE IF THE KEYBOARD IS PLUGGED IN.  IF THE KEYBOARD IS  *
                    ;* PLUGGED IN, THE LSB RETURNS A Ø. OTHERWISE, IT IS A 1.        *
                    ;* THIS 1 IS ADDED TO IOBYTE TO CHANGE THE CONSOLE INPUT FROM    *
                    ;* THE SWBD PARALLEL PORT 4 (THE KEYBOARD) TO THE SWBD SERIAL    *
                    ;* PORT THAT RECEIVES RS232 DATA FROM THE RS232 TERMINAL.        *
                    ;*                                                               *
                    ;*****************************************************************
```

```
DE2D ØE1A       TINIT   MVI     C,CLEAR         ;INITIALIZE THE TERMINAL ROUTINE
DE2F DBØ7               IN      7               ;GET KEYBOARD INTERLOCK BYTE
DE31 E6Ø1               ANI     1               ;GET BIT 1 ONLY
DE33 C6CØ               ADI     INTIOBY         ;ADD INTIOBY TO KEYBOARD BIT
DE35 32Ø3ØØ             STA     IOBYTE          ;INITIALIZE IOBYTE
DE38 C3ØCDD             JMP     COUT

                ********************************************************************
                *                                                                  *
                * VIO-X VIDEO DRIVER                                               *
                *                                                                  *
                ********************************************************************

DE3B DBØ9       COCRT   IN      9               ;READ STATUS PORT
DE3D E6Ø1               ANI     1               ;MASK TXRDY BIT
DE3F CA3BDE             JZ      COCRT           ;WAIT FOR READY
DE42 79                 MOV     A,C             ;GET CHAR
DE43 D3Ø8               OUT     8               ;OUTPUT IT
DE45 C9                 RET                     ;ALL DONE

                ********************************************************************
                *                                                                  *
                * ROUTINE FOR OKIDATA PRINTER                                      *
                * PRINTER IS ON PORT Ø WITH PRINTER READY ON PORT 5 BIT 1          *
                *                                                                  *
                ********************************************************************

DE46 DBØ2       COPTR   IN      2               ;INPUT FROM PORT 2
DE48 E6Ø8               ANI     8               ;WAIT UNTIL OK TO SEND
DE4A CA46DE             JZ      COPTR
DE4D DBØ5       COPTR1  IN      5               ;BUFFER FULL?
DE4F E6Ø1               ANI     1
DE51 CA4DDE             JZ      COPTR1          ;WAIT UNTIL PRINTER READY
DE54 79                 MOV     A,C             ;OUTPUT THE CHARACTER
DE55 D3ØØ               OUT     Ø
DE57 C9                 RET

                ********************************************************************
                *                                                                  *
                * 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).                                       *
                *                                                                  *
                ********************************************************************

DE58 218ØØØ     GOCPM   LXI     H,BUFF          ;SET UP INITIAL DMA ADDRESS
DE5B CDE1DF             CALL    SETDMA
DE5E 3EC3               MVI     A,(JMP)         ;INITIALIZE JUMP TO WARM BOOT
DE6Ø 32ØØØØ             STA     WBOT
DE63 32Ø5ØØ             STA     ENTRY           ;INITIALIZE JUMP TO BDOS
DE66 21Ø3DD             LXI     H,WBOOTE        ;ADDRESS IN WARM BOOT JUMP
DE69 22Ø1ØØ             SHLD    WBOT+1
DE6C 21Ø6CF             LXI     H,BDOS+6        ;ADDRESS IN BDOS JUMP
DE6F 22Ø6ØØ             SHLD    ENTRY+1
DE72 AF                 XRA     A               ;A <- Ø
DE73 329BE5             STA     BUFSEC          ;DISK JOCKEY BUFFER EMPTY
```

```
DE76 32A3E1             STA     BUFWRTN         ;SET BUFFER NOT DIRTY FLAG
DE79 3A0400             LDA     CDISK           ;JUMP TO CP/M WITH CURRENTLY SELECTED DISK IN C
DE7C 4F                 MOV     C,A
DE7D 3ABCDE             LDA     CWFLG
                                                ;
                                                ;USER Ø ENHANCEMENTS FROM MICROSYSTEMS
                                                ;VOL3/NO2  MAR/APR 1982  PAGE 36
                                                ;
DE80 21E1DE             LXI     H,CHECK         ;LOAD ADR OF CHECK ROUTINE
DE83 22DCCD             SHLD    CCP+6DCH        ;STORE AT CP/M PATCH 2
DE86 2121DF             LXI     H,USRRST        ;LOAD ADR OF USRRST ROUTINE
DE89 22EDCD             SHLD    CCP+6EDH        ;STORE AT CP/M PATCH 3
DE8C 21BDDE             LXI     H,PROMP1        ;LOAD ADR OF PROMPT ROUTINE
DE8F 2289CA             SHLD    CCP+389H        ;STORE AT CP/M PATCH 1
                                                ;
                                                ;END OF ENHANCEMENTS
                                                ;
DE92 A7                 ANA     A
DE93 1142DF             LXI     D,COLDBEG       ;BEGINNING OF INITIAL COMMAND
DE96 3E0F               MVI     A,COLDEND-COLDBEG+1 ;LENGTH OF COMMAND
DE98 CAA0DE             JZ      CLDCMND
DE9B 1151DF             LXI     D,WARMBEG
DE9E 3E01               MVI     A,WARMEND-WARMBEG+1
DEA0 2108C7     CLDCMND LXI     H,CCP+8         ;COMMAND BUFFER
DEA3 3207C7             STA     CCP+7
DEA6 47                 MOV     B,A
DEA7 CD6AE2             CALL    MOVLOP
DEAA 3ABCDE             LDA     CWFLG
DEAD A7                 ANA     A
DEAE 3A41DF             LDA     AUTOFLG
DEB1 CAB5DE             JZ      CLDBOT
DEB4 1F                 RAR
DEB5 1F         CLDBOT  RAR
DEB6 DA00C7             JC      CCP
DEB9 C303C7             JMP     CCP+3           ;ENTER CP/M

DEBC 00         CWFLG   DB      Ø               ;COLD/WARM BOOT FLAG

                ****************************************************************
                *                                                              *
                * THE FOLLOWING ROUTINES: PROMP1, CHECK, USRRST ARE FROM THE   *
                * MAR/APR 1982 MICROSYSTEMS, VOL3/NO2. IF CP/M CANNOT FIND THE *
                * REQUESTED PROGRAM IN THE DIRECTORY FOR THE LOGGED USER,      *
                * THEN THE CURRENT USER NUMBER IS SAVED, THE USER NUMBER       *
                * CHANGED TO Ø AND THE SEARCH DONE AGAIN. IF FOUND, LOADED.    *
                * THE ORIGINAL USER NUMBER IS RESTORED AND OPERATION RESUMES.  *
                * NOTE: THIS ONLY WORKS IF THE CCP IS SEARCHING.  TRANSIENT    *
                * PROGRAMS SUCH AS BASIC WILL NOT GET THIS TREATMENT.  FURTHER, *
                * THE PROMPT IS CHANGED TO INDICATE THE USER NUMBER. '5A>' IS  *
                * THE NEW PROMPT FOT USER 5, DISK A.    JJO 3/24/82            *
                ****************************************************************

0005 =          BDOSE   EQU     5               ;ADR OF BDOS ENTRY POINT
                                                ;USED FOR SYSTEM CALLS
C7D0 =          OPEN    EQU     CCP+0D0H        ;CALL THIS LOCATION TO
                                                ;RE-INITIATE THE SEARCH
```

```
                                        ;FOR FILE IN DIR Ø
CE6B =          NFOUND  EQU     CCP+76BH        ;ADR TO RETURN TO IF
                                        ;FILE NOT FOUND
CEØ1 =          EOF     EQU     CCP+7Ø1H        ;JUMP TO THIS ADR AFTER USER
                                        ;NUMBER RESTORE OPERATION
CDDE =          RFILE   EQU     CCP+6DEH        ;JUMP TO THIS ADR TO READ FILE
                                        ;
C78C =          PCHAR   EQU     CCP+8CH         ;CALL THIS LOCATION TO PRINT
                                        ;USER NUMBER PROMPT CHARACTERS
C8DØ =          CPMPT   EQU     CCP+1DØH        ;JUMP TO THIS LOCATION AFTER
                                        ;PRINTING USER NUMBER PROMPT

                ****************************************************************
                *                                                              *
                *              ROUTINE TO MODIFY CP/M PROMPT                   *
                *                                                              *
                ****************************************************************

DEBD C5         PROMP1: PUSH    B
DEBE D5                 PUSH    D
DEBF ØE2Ø               MVI     C,2ØH           ;SYSTEM CALL TO GET USER NUMBER
DEC1 1EFF               MVI     E,ØFFH          ;GETS CURRENT USER IN A
DEC3 CDØ5ØØ             CALL    BDOSE
DEC6 FEØA               CPI     ØAH             ;IS USER NUMBER > 1Ø ?
DEC8 D2D5DE             JNC     CHAR2           ;IF SO MUST PRINT TWO NUMBERS
DECB C63Ø       PMT1:   ADI     3ØH             ;OTHERWISE MAKE ASCII
DECD CD8CC7             CALL    PCHAR           ;OUTPUT NUMBER TO CONSOLE
DEDØ D1         EXIT1:  POP     D
DED1 C1                 POP     B
DED2 C3DØC8             JMP     CPMPT           ;ALL DONE
DED5 D6ØA       CHAR2   SUI     ØAH             ;USER IS > 1Ø SO SUBTRACT 1Ø
DED7 F5                 PUSH    PSW             ;SAVE RESULT ON STACK
DED8 3E31               MVI     A,31H           ;SEND ASCII 1 TO CONSOLE
DEDA CD8CC7             CALL    PCHAR
DEDD F1                 POP     PSW             ;RECOVER REMAINDER
DEDE C3CBDE             JMP     PMT1            ;JUMP TO OUTPUT REMAINDER

                ****************************************************************
                *                                                              *
                *              ROUTINE TO CHECK DIR Ø FOR FILE                 *
                *                                                              *
                ****************************************************************

DEE1 D5         CHECK:  PUSH    D
DEE2 C5                 PUSH    B
DEE3 F5                 PUSH    PSW
DEE4 3EØØ               MVI     A,Ø             ;RESET FLAG
DEE6 323FDF             STA     FLAG            ;FLAG INDICATES THAT THE USER
                                                ;NUMBER CHANGED IF IT IS SET
DEE9 1EFF               MVI     E,ØFFH          ;INTERROGATE CURRENT USER NO.
DEEB ØE2Ø               MVI     C,2ØH
DEED CDØ5ØØ             CALL    BDOSE
DEFØ 3240DF             STA     USER            ;STORE CURRENT USER NUMBER
DEF3 B7                 ORA     A               ;CHECK FOR USER Ø
DEF4 C2FADE             JNZ     NUSERØ          ;NOT USER Ø
DEF7 C31ØDF             JMP     EXIT            ;IF USER Ø GO TO CCP
```

```
                                        ;
DEFA 0E20       NUSER0: MVI     C,20H   ;SET USER NUMBER TO 0
DEFC 1E00               MVI     E,0
DEFE CD0500             CALL    BDOSE
DF01 CDD0C7             CALL    OPEN    ;RE-INITIATE SEARCH
DF04 C216DF             JNZ     FOUND   ;IF A RETURNED NON-ZERO THEN FILE
                                        ;WAS FOUND IN USER 0 DIRECTORY
DF07 3A40DF             LDA     USER    ;OTHERWISE FILE NOT FOUND IN
                                        ;USER 0 SO RESTORE USER NUMBER
DF0A 5F                 MOV     E,A
DF0B 0E20               MVI     C,20H
DF0D CD0500             CALL    BDOSE
DF10 F1         EXIT:   POP     PSW
DF11 C1                 POP     B
DF12 D1                 POP     D
DF13 C36BCE             JMP     NFOUND  ;TO CCP IF FILE NOT FOUND
                                        ;
DF16 F1         FOUND:  POP     PSW     ;FILE FOUND IN USER 0 DIRECTORY
DF17 C1                 POP     B
DF18 D1                 POP     D
DF19 3E01               MVI     A,1     ;SET FLAG FOR RESTORE OPERATION
DF1B 323FDF             STA     FLAG    ;AFTER FILE IS LOADED
DF1E C3DECD             JMP     RFILE   ;GO CCP AND READ FILE

                ;*****************************************************************
                ;*                                                               *
                ;*       ROUTINE TO RESTORE USER NUMBER AFTER LOADING FILE        *
                ;*       FROM USER 0 DIRECTORY                                   *
                ;*                                                               *
                ;*****************************************************************

DF21 D5         USRRST: PUSH    D
DF22 C5                 PUSH    B
DF23 F5                 PUSH    PSW
DF24 3A3FDF             LDA     FLAG    ;CHECK USER NUMBER FLAG
DF27 B7                 ORA     A
DF28 CA39DF             JZ      RSTR1   ;IF FLAG NOT SET NO RESTORE RQ'D
DF2B 3A40DF             LDA     USER    ;IF FLAG NON-ZERO THEN GET USER
                                                ;NUMBER
DF2E 5F                 MOV     E,A     ;RESTORE USER NUMBER
DF2F 0E20               MVI     C,20H
DF31 CD0500             CALL    BDOSE
DF34 3E00               MVI     A,0     ;RESET FLAG
DF36 323FDF             STA     FLAG
DF39 F1         RSTR1:  POP     PSW
DF3A C1                 POP     B
DF3B D1                 POP     D
DF3C C301CE             JMP     EOF     ;RETURN TO CCP AT EOF
                                        ;
DF3F 00         FLAG:   DB      0       ;SET FLAG INITIALLY TO 0
DF40 00         USER:   DB      0       ;SET TEMP USER INITIALLY TO 0

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

DF41 Ø1         AUTOFLG DB      1               ;AUTO COMMAND FEATURE

                ********************************************************************
                *                                                                  *
                * IF THERE IS A COMMAND INSERTED HERE, IT WILL BE GIVEN IF THE     *
                * AUTO FEATURE IS ENABLED.                                         *
                *       FOR EXAMPLE:                                               *
                *                                                                  *
                *       COLDBEG DB      'MBASIC MYPROG'                            *
                *       COLDEND DB      Ø                                          *
                *                                                                  *
                * WILL EXECUTE MICROSOFT BASIC, AND MBASIC WILL EXECUTE THE        *
                * "MYPROG" BASIC PROGRAM.                                          *
                *                                                                  *
                ********************************************************************

DF42 5355424D49COLDBEG DB       'SUBMIT STARTUP';COLD BOOT COMMAND
DF5Ø ØØ         COLDEND DB      Ø
                WARMBEG DB      ''              ;WARM BOOT COMMAND GOES HERE
DF51 ØØ         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.     *
                *                                                                  *
                ********************************************************************

DF52 31ØØØ1     WBOOT   LXI     SP,TPA          ;SET UP STACK POINTER
DF55 3EØ1               MVI     A,1
DF56 =          WFLG    EQU     $-1             ;TEST IF BEGINNING OR
DF57 A7                 ANA     A               ;       ENDING A WARM BOOT
DF58 3EØ1               MVI     A,1
DF5A 3256DF             STA     WFLG
DF5D 32BCDE             STA     CWFLG           ;SET COLD/WARM BOOT FLAG
DF6Ø CA58DE             JZ      GOCPM
DF63 AF                 XRA     A
DF64 3256DF             STA     WFLG
DF67 4F                 MOV     C,A

                        IF      (MAXHD NE Ø) AND FIRST  ;SUPPLY WARM BOOT FROM HARD DISK ?
                        LXI     H,CCP-2ØØH      ;INITIAL DMA ADDRESS
                        PUSH    H
                        STA     HEAD
                        MVI     A,4
                        PUSH    PSW             ;SAVE FIRST SECTOR
```

```
                        *** Cbios For CP/M Ver. 2.2 ***

                        CALL    HDDRV           ;SELECT DRIVE A
                        MVI     C,Ø
                        CALL    HDTRK           ;HOME THE DRIVE
                WARMLOD POP     PSW             ;RESTORE SECTOR
                        POP     H               ;RESTORE DMA ADDRESS
                        INR     A
                        STA     HDSECTR
                        CPI     16              ;PAST BDOS ?
                        JZ      WBOOT           ;YES, ALL DONE
                        INR     H               ;UPDATE DMA ADDRESS
                        INR     H
                        SHLD    HDADD
                        PUSH    H
                        PUSH    PSW
                WARMRD  LXI     B,RETRIES*1ØØH+Ø ;RETRY COUNTER
                WRMREAD PUSH    B               ;SAVE THE RETRY COUNT
                        CALL    HDREAD          ;READ THE SECTOR
                        POP     B
                        JNC     WARMLOD         ;TEST FOR ERROR
                        DCR     B               ;UPDATE THE ERROR COUNT
                        JNZ     WRMREAD         ;KEEP TRYING IF NOT TO MANY ERRORS
                        HLT                     ;CAN'T WARM BOOT
                        ENDIF

                        IF      (MAXFLOP NE Ø) AND NOT FIRST ;SUPPLY WARM BOOT FROM 2D ?
DF68 CD33DD             CALL    DJDRV           ;SELECT DRIVE A
DF6B ØE ØØ              MVI     C,Ø             ;SELECT SINGLE DENSITY
DF6D CD2DF4             CALL    DJDEN
DF7Ø ØE ØØ              MVI     C,Ø             ;SELECT SIDE Ø
DF72 CD3ØF4             CALL    DJSIDE
DF75 3EØF               MVI     A,15            ;INITIALIZE THE SECTOR TO READ
DF77 3295DF             STA     NEWSEC
DF7A 21ØØC6             LXI     H,CCP-1ØØH      ;AND THE DMA ADDRESS
DF7D 22B4DF             SHLD    NEWDMA
DF8Ø CD94DF             CALL    WARMLOD         ;READ IN CP/M
DF83 Ø1ØØCC             LXI     B,CCP+5ØØH      ;LOAD ADDRESS FOR REST OF WARM BOOT
DF86 CD12F4             CALL    DJDMA
DF89 ØE Ø8              MVI     C,8
DF8B CDØFF4             CALL    DJSEC
DF8E CDC8DF             CALL    WARMRD
DF91 C3Ø3CC             JMP     CCP+5Ø3H

DF94 3EØF       WARMLOD MVI     A,15            ;PREVIOUS SECTOR
DF95 =          NEWSEC  EQU     $-1
DF96 3C                 INR     A               ;UPDATE THE PREVIOUS SECTOR
DF97 3C                 INR     A
DF98 FE1B               CPI     27              ;WAS IT THE LAST ?
DF9A DAACDF             JC      NOWRAP
DF9D D6Ø9               SUI     9               ;YES
DF9F FE13               CPI     19
DFA1 C8                 RZ
DFA2 2AB4DF             LHLD    NEWDMA
DFA5 1180FB             LXI     D,-480H
DFA8 19                 DAD     D
DFA9 22B4DF             SHLD    NEWDMA
DFAC 3295DF     NOWRAP  STA     NEWSEC          ;SAVE THE NEW SECTOR TO READ
```

```
DFAF 4F                 MOV     C,A
DFB0 CD0FF4             CALL    DJSEC
DFB3 2100C6             LXI     H,CCP-100H      ;GET THE PREVIOUS DMA ADDRESS
DFB4 =          NEWDMA  EQU     $-2
DFB6 110001             LXI     D,100H          ;UPDATE THE DMA ADDRESS
DFB9 19                 DAD     D
DFBA 22B4DF             SHLD    NEWDMA          ;SAVE THE DMA ADDRESS
DFBD 44                 MOV     B,H
DFBE 4D                 MOV     C,L
DFBF CD12F4             CALL    DJDMA           ;SET THE DMA ADDRESS
DFC2 CDC8DF             CALL    WARMRD
DFC5 C394DF             JMP     WARMLOD

DFC8 01000A     WARMRD  LXI     B,RETRIES*100H+0;MAXIMUM # OF ERRORS
DFCB C5         WRMREAD PUSH    B
DFCC CD0CF4             CALL    DJTRK           ;SET THE TRACK
DFCF CD15F4             CALL    DJREAD          ;READ THE SECTOR
DFD2 C1                 POP     B
DFD3 D0                 RNC                     ;CONTINUE IF SUCCESSFUL
DFD4 05                 DCR     B
DFD5 C2CBDF             JNZ     WRMREAD         ;KEEP TRYING
DFD8 C32AF4             JMP     DJERR
                        ENDIF

                ;*****************************************************************
                ;*                                                               *
                ; * SETSEC JUST SAVES THE DESIRED SECTOR TO SEEK TO UNTIL AN      *
                ; * ACTUAL READ OR WRITE IS ATTEMPTED.                           *
                ;*                                                               *
                ;*****************************************************************

DFDB 60         SETSEC  MOV     H,B
DFDC 69                 MOV     L,C
DFDD 2293E5             SHLD    CPMSEC
DFE0 C9         DONOP   RET

                ;*****************************************************************
                ;*                                                               *
                ; * SETDMA SAVES THE DMA ADDRESS FOR THE DATA TRANSFER.           *
                ;*                                                               *
                ;*****************************************************************

DFE1 60         SETDMA  MOV     H,B             ;HL <- BC
DFE2 69                 MOV     L,C
DFE3 2283E1             SHLD    CPMDMA          ;CP/M DMA ADDRESS
DFE6 C9                 RET

                ;*****************************************************************
                ;*                                                               *
                ; * HOME IS TRANSLATED INTO A SEEK TO TRACK ZERO.                 *
                ;*                                                               *
                ;*****************************************************************

DFE7 0E00       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.          *
                ;*                                                               *
                ;*****************************************************************

DFE9 79         SETTRK  MOV     A,C             ;A <- TRACK #
DFEA 3296E5             STA     CPMTRK          ;CP/M TRACK #
DFED C9                 RET

                ;*****************************************************************
                ;*                                                               *
                ;* SECTRAN TRANSLATES A LOGICAL SECTOR # INTO A PHYSICAL SECTOR  *
                ;* #.                                                            *
                ;*                                                               *
                ;*****************************************************************

                        IF      (MAXHD NE Ø) AND (MAXFLOP NE Ø) ;BOTH TYPES ?
DFEE 3A95E5     SECTRAN LDA     CPMDRV          ;GET THE DRIVE NUMBER

                        IF      FIRST
                        CPI     MAXHD*LOGDSK    ;OVER THE # OF HARD DISKS ?
                        JC      TRANHD
                        ELSE
DFF1 FEØ2               CPI     MAXFLOP         ;OVER THE # OF FLOPPIES ?
DFF3 D225EØ             JNC     TRANHD
                        ENDIF
                        ENDIF

                        IF      (MAXHD EQ Ø) OR (MAXFLOP EQ Ø) ;JUST ONE TYPE ?

                SECTRAN EQU     $
                        ENDIF

                        IF      MAXFLOP NE Ø    ;FLOPPY TRANSLATION
DFF6 Ø3         TRANFP  INX     B
DFF7 D5                 PUSH    D               ;SAVE TABLE ADDRESS
DFF8 C5                 PUSH    B               ;SAVE SECTOR #
DFF9 CD07E1             CALL    GETDPB          ;GET DPB ADDRESS INTO HL
DFFC 7E                 MOV     A,M             ;GET # OF CP/M SECTORS/TRACK
DFFD B7                 ORA     A               ;CLEAR CARY
DFFE 1F                 RAR                     ;DIVIDE BY TWO
DFFF 91                 SUB     C
E000 F5                 PUSH    PSW             ;SAVE ADJUSTED SECTOR
E001 FAØDEØ             JM      SIDETWO
E004 F1         SIDEA   POP     PSW             ;DISCARD ADJUSTED SECTOR
E005 C1                 POP     B               ;RESTORE SECTOR REQUESTED
E006 D1                 POP     D               ;RESTOR ADDRESS OF XLT TABLE
E007 EB         SIDEONE XCHG                    ;HL <- &(TRANSLATION TABLE)
E008 09                 DAD     B               ;BC = OFFSET INTO TABLE
E009 6E                 MOV     L,M             ;HL <- PHYSICAL SECTOR
E00A 2600               MVI     H,Ø
E00C C9                 RET

E00D Ø10FØØ     SIDETWO LXI     B,15            ;OFFSET TO SIDE BIT
E010 09                 DAD     B
```

```
E011 7E                 MOV     A,M
E012 E608               ANI     8               ;TEST FOR DOUBLE SIDED
E014 CA04E0             JZ      SIDEA           ;MEDIA IS ONLY SINGLE SIDED
E017 F1                 POP     PSW             ;RETRIEVE ADJUSTED SECTOR
E018 C1                 POP     B
E019 2F                 CMA                     ;MAKE SECTOR REQUEST POSITIVE
E01A 3C                 INR     A
E01B 4F                 MOV     C,A             ;MAKE NEW SECTOR THE REQUESTED SECTOR
E01C D1                 POP     D
E01D CD07E0             CALL    SIDEONE
E020 3E80               MVI     A,80H           ;SIDE TWO BIT
E022 B4                 ORA     H               ;       AND SECTOR
E023 67                 MOV     H,A
E024 C9                 RET
                        ENDIF

                        IF      MAXHD NE 0      ;HARD DISK TRANSLATION ROUTINE
E025 60         TRANHD  MOV     H,B
E026 69                 MOV     L,C
E027 23                 INX     H
E028 C9                 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.                           *
                ;*                                                               *
                ;*****************************************************************

E029 79         SETDRV  MOV     A,C             ;SAVE THE DRIVE #
E02A 3295E5             STA     CPMDRV
E02D FE05               CPI     MAXFLOP+(MAXHD*LOGDSK) ;CHECK FOR A VALID DRIVE #
E02F D2F8E0             JNC     ZRET            ;ILLEGAL DRIVE #
E032 7B                 MOV     A,E             ;TEST IF DRIVE EVER LOGGED IN BEFORE
E033 E601               ANI     1
E035 C2DFE0             JNZ     SETDRV1         ;BIT 0 OF E = 0 -> NEVER SELECTED BEFORE

                        IF      (MAXHD NE 0) AND (MAXFLOP NE 0) ;BOTH TYPES ?
E038 3A95E5             LDA     CPMDRV          ;GET THE DRIVE NUMBER

                        IF      FIRST
                        CPI     MAXHD*LOGDSK    ;OVER THE # OF HARD DISKS ?
                        JC      DRVHD
                        SUI     MAXHD*LOGDSK
                        ELSE
E03B FE02               CPI     MAXFLOP         ;OVER THE # OF FLOPPIES ?
E03D D295E0             JNC     SUBFP
                        ENDIF
```

```
                        ENDIF

                        IF      (MAXFLOP NE Ø) AND FIRST
                        MOV     C,A             ;SAVE DRIVE #
                        MVI     A,Ø             ;HAVE THE FLOPPIES BEEN ACCESSED YET ?
                FLOPFLG EQU     $-1
                        ANA     A
                        JNZ     FLOPOK
                        MVI     B,17            ;FLOPPIES HAVN'T BEEN ACCESSED
                        LXI     H,DJBOOT        ;CHECK IF 2D CONTROLLER IS INSTALLED
                        MVI     A,(JMP)
                CLOPP   CMP     M
                        JNZ     ZRET
                        DCR     B
                        JNZ     CLOPP
                        CALL    DJBOOT          ;INITIALIZE THE CONTROLLER
                        MVI     A,1             ;SAVE 2D INITIALIZED FLAG
                        STA     FLOPFLG
                        ENDIF
                        IF      MAXFLOP NE Ø
E04Ø 21Ø1ØØ     FLOPOK  LXI     H,1             ;SELECT SECTOR 1 OF TRACK 1
E043 2297E5             SHLD    TRUESEC
E046 3EØ1               MVI     A,1
E048 3296E5             STA     CPMTRK
E04B CD34E2             CALL    FILL            ;FLUSH BUFFER AND REFILL
E04E DAF8EØ             JC      ZRET            ;TEST FOR ERROR RETURN
E051 CD27F4             CALL    DJSTAT          ;GET STATUS ON CURRENT DRIVE
E054 E6ØC               ANI     ØCH             ;STRIP OFF UNWANTED BITS
E056 F5                 PUSH    PSW             ;USED TO SELECT A DPB
E057 1F                 RAR
E058 2121ØE1            LXI     H,XLTS          ;TABLE OF XLT ADDRESSES
E05B 5F                 MOV     E,A
E05C 16ØØ               MVI     D,Ø
E05E 19                 DAD     D
E05F E5                 PUSH    H               ;SAVE POINTER TO PROPER XLT
E06Ø CDØ7E1             CALL    GETDPB          ;GET DPH POINTER INTO DE
E063 EB                 XCHG                    ;
E064 D1                 POP     D
E065 Ø6Ø2               MVI     B,2             ;NUMBER OF BYTES TO MOVE
E067 CD6AE2             CALL    MOVLOP          ;MOVE THE ADDRESS OF XLT
E06A 11Ø8ØØ             LXI     D,8             ;OFFSET TO DPB POINTER
E06D 19                 DAD     D               ;HL <- &DPH.DPB
E06E E5                 PUSH    H
E06F 2AØ7FØ             LHLD    ORIGIN+7        ;GET ADDRESS OF DJ TERMINAL OUT ROUTINE
E072 23                 INX     H               ;BUMP TO LOOK AT ADDRESS OF
                                                ;       UART STATUS LOCATION
E073 7E                 MOV     A,M
E074 EEØ3               XRI     3               ;ADJUST FOR PROPER REV DJ
E076 6F                 MOV     L,A
E077 26F3               MVI     H,(ORIGIN+3ØØH)/1ØØH
E079 7E                 MOV     A,M
E07A E6Ø8               ANI     DBLSID          ;CHECK DOUBLE SIDED BIT
E07C 1193E4             LXI     D,DPB128S       ;BASE FOR SINGLE SIDED DPB'S
E07F C285EØ             JNZ     SIDEOK
E082 11D3E4             LXI     D,DPB128D       ;BASE OF DOUBLE SIDED DPB'S
E085 EB         SIDEOK  XCHG                    ;HL <- DBP BASE, DE <- &DPH.DPB
```

; ; out

```
E086 D1                 POP     D               ;RESTORE DE (POINTER INTO DPH)
E087 F1                 POP     PSW             ;OFFSET TO CORRECT DPB
E088 17                 RAL
E089 17                 RAL
E08A 4F                 MOV     C,A
E08B 0600               MVI     B,0
E08D 09                 DAD     B
E08E EB                 XCHG                    ;PUT DPB ADDRESS IN DPH
E08F 73                 MOV     M,E
E090 23                 INX     H
E091 72                 MOV     M,D
                        ENDIF

                        IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E092 C3DFE0             JMP     SETDRV1         ;SKIP OVER THE HARD DISK SELECT
                        IF      NOT FIRST
E095 D602       SUBFP   SUI     MAXFLOP         ;ADJUST THE DRIVE #
                        ENDIF
                        ENDIF

                        IF      MAXHD NE 0
E097 CDFEE0     DRVHD   CALL    DIVLOG          ;DIVIDE BY LOGICAL DISKS PER DRIVE
E09A 79                 MOV     A,C
E09B 32C0E3             STA     HDDISK
E09E CDAEE3             CALL    DRVPTR
E0A1 7E                 MOV     A,M
E0A2 3C                 INR     A
E0A3 C2DFE0             JNZ     SETDRV1
E0A6 F6FC               ORI     NULL            ;SELECT DRIVE
E0A8 D352               OUT     HDFUNC
E0AA 3E05               MVI     A,SCENBL        ;ENABLE THE CONTROLLER
E0AC D350               OUT     HDCNTL
E0AE 0EEF               MVI     C,239           ;WAIT APPROX 2 MINUTES FOR DISK TO READY
E0B0 210000             LXI     H,0
E0B3 2B         TDELAY  DCX     H
E0B4 7C                 MOV     A,H
E0B5 B5                 ORA     L
E0B6 CCFCE0             CZ      DCRC
E0B9 C8                 RZ
E0BA DB50               IN      HDSTAT          ;TEST IF READY YET
E0BC E620               ANI     DRVRDY
E0BE C2B3E0             JNZ     TDELAY

                        IF      SDELAY
E0C1 210000             LXI     H,0             ;TIME ONE REVOLUTION OF THE DRIVE
E0C4 0E40               MVI     C,INDEX
E0C6 DB50               IN      HDSTAT
E0C8 A1                 ANA     C
E0C9 47                 MOV     B,A             ;SAVE CURRENT INDEX LEVEL IN B
E0CA DB50       INDX1   IN      HDSTAT
E0CC A1                 ANA     C
E0CD B8                 CMP     B               ;LOOP UTIL INDEX LEVEL CHANGES
E0CE CACAE0             JZ      INDX1
E0D1 23         INDX2   INX     H
E0D2 DB50               IN      HDSTAT          ;START COUNTING UNTIL INDEX RETURNS TO
E0D4 A1                 ANA     C               ;       PREVIOUS STATE
```

```
E0D5 B8                 CMP     B
E0D6 C2D1E0             JNZ     INDX2
                        IF      M10
                        DAD     H
                        ENDIF
E0D9 22A6E2             SHLD    SETTLE          ;SAVE THE COUNT FOR TIMEOUT DELAY
                        ENDIF
E0DC CD90E2             CALL    HDHOME
                        ENDIF

E0DF CD07E1     SETDRV1 CALL    GETDPB          ;GET ADDRESS OF DPB IN HL
E0E2 010F00             LXI     B,15            ;OFFSET TO SECTOR SIZE
E0E5 09                 DAD     B
E0E6 7E                 MOV     A,M             ;GET SECTOR SIZE
E0E7 E607               ANI     7H
E0E9 3234E1             STA     SECSIZ
E0EC 7E                 MOV     A,M
E0ED 1F                 RAR
E0EE 1F                 RAR
E0EF 1F                 RAR
E0F0 1F                 RAR
E0F1 E60F               ANI     0FH
E0F3 3272E1             STA     SECPSEC
E0F6 EB                 XCHG                    ;HL <- DPH
E0F7 C9                 RET

E0F8 210000     ZRET    LXI     H,0             ;SELDRV ERROR EXIT
E0FB C9                 RET

                        IF      MAXHD NE 0
E0FC 0D         DCRC    DCR     C               ;CONDITIONAL DECREMENT C ROUTINE
E0FD C9                 RET

E0FE 0E00       DIVLOG  MVI     C,0
E100 D603       DIVLOGX SUI     LOGDSK
E102 D8                 RC
E103 0C                 INR     C
E104 C300E1             JMP     DIVLOGX
                        ENDIF
                ********************************************************************
                *                                                                  *
                * GETDPB RETURNS HL POINTING TO THE DPB OF THE CURRENTLY           *
                * SELECTED DRIVE, DE POINTING TO DPH.                              *
                *                                                                  *
                ********************************************************************

E107 3A95E5     GETDPB  LDA     CPMDRV
E10A 6F                 MOV     L,A             ;FORM OFFSET
E10B 2600               MVI     H,0
E10D 29                 DAD     H
E10E 29                 DAD     H
E10F 29                 DAD     H
E110 29                 DAD     H
E111 1143E5             LXI     D,DPBASE        ;BASE OF DPH'S
E114 19                 DAD     D
```

```
E115 E5              PUSH    H               ;SAVE ADDRESS OF DPH
E116 110A00          LXI     D,10            ;OFFSET TO DPB
E119 19              DAD     D
E11A 7E              MOV     A,M             ;GET LOW BYTE OF DPB ADDRESS
E11B 23              INX     H
E11C 66              MOV     H,M             ;GET LOW BYTE OF DPB
E11D 6F              MOV     L,A
E11E D1              POP     D
E11F C9              RET

                     ********************************************************************
                     *                                                                  *
                     * XLTS IS A TABLE OF ADDRESS THAT POINT TO EACH OF THE XLT         *
                     * TABLES FOR EACH SECTOR SIZE.                                     *
                     *                                                                  *
                     ********************************************************************

                     IF      MAXFLOP NE Ø
E120 C5E3      XLTS  DW      XLT128          ;XLT FOR 128 BYTE SECTORS
E122 E0E3            DW      XLT256          ;XLT FOR 256 BYTE SECTORS
E124 15E4            DW      XLT512          ;XLT FOR 512 BYTE SECTORS
E126 52E4            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.           *
                     *                                                                  *
                     ********************************************************************

E128 79        WRITE MOV     A,C             ;SAVE WRITE COMMAND TYPE
E129 329AE1          STA     WRITTYP
E12C 3E01            MVI     A,1             ;SET WRITE COMMAND
E12E 06              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.                                             *
                     *                                                                  *
                     ********************************************************************

E12F AF        READ  XRA     A               ;SET THE COMMAND TYPE TO READ
```

```
E130 3286E1             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.                                               *
                *                                                              *
                ****************************************************************

E133 0600       REDWRT  MVI     B,0             ;THE 0 IS MODIFIED TO CONTAIN THE LOG2
E134 =          SECSIZ  EQU     $-1             ;       OF THE PHYSICAL SECTOR SIZE/128
                                                ;       ON THE CURRENTLY SELECTED DISK.
E135 2A93E5             LHLD    CPMSEC          ;GET THE DESIRED CP/M SECTOR #
E138 7C                 MOV     A,H
E139 E680               ANI     80H             ;SAVE ONLY THE SIDE BIT
E13B 4F                 MOV     C,A             ;REMEMBER THE SIDE
E13C 7C                 MOV     A,H
E13D E67F               ANI     7FH             ;FORGET THE SIDE BIT
E13F 67                 MOV     H,A
E140 2B                 DCX     H               ;TEMPORARY ADJUSTMENT
E141 05         DIVLOOP DCR     B               ;UPDATE REPEAT COUNT
E142 CA4FE1             JZ      DIVDONE
E145 B7                 ORA     A
E146 7C                 MOV     A,H
E147 1F                 RAR
E148 67                 MOV     H,A
E149 7D                 MOV     A,L
E14A 1F                 RAR                     ;DIVIDE THE CP/M SECTOR # BY THE SIZE
                                                ;       OF THE PHYSICAL SECTORS
E14B 6F                 MOV     L,A
E14C C341E1             JMP     DIVLOOP         ;
E14F 23         DIVDONE INX     H
E150 7C                 MOV     A,H
E151 B1                 ORA     C               ;RESTORE THE SIDE BIT
E152 67                 MOV     H,A
E153 2297E5             SHLD    TRUESEC         ;SAVE THE PHYSICAL SECTOR NUMBER
E156 2195E5             LXI     H,CPMDRV        ;POINTER TO DESIRED DRIVE,TRACK, AND SECTOR
E159 1199E5             LXI     D,BUFDRV        ;POINTER TO BUFFER DRIVE,TRACK, AND SECTOR
E15C 0605               MVI     B,5             ;COUNT LOOP
E15E 05         DTSLOP  DCR     B               ;TEST IF DONE WITH COMPARE
E15F CA6DE1             JZ      MOVE            ;YES, MATCH. GO MOVE THE DATA
E162 1A                 LDAX    D               ;GET A BYTE TO COMPARE
E163 BE                 CMP     M               ;TEST FOR MATCH
E164 23                 INX     H               ;BUMP POINTERS TO NEXT DATA ITEM
E165 13                 INX     D
E166 CA5EE1             JZ      DTSLOP          ;MATCH, CONTINUE TESTING

                ****************************************************************
                *                                                              *
                * DRIVE, TRACK, AND SECTOR DON'T MATCH, FLUSH THE BUFFER IF    *
                * NECESSARY AND THEN REFILL.                                   *
                *                                                              *
                ****************************************************************
```

```
E169 CD34E2             CALL    FILL            ;FILL THE BUFFER WITH CORRECT PHYSICAL SECTOR
E16C D8                 RC                      ;NO GOOD, RETURN WITH ERROR INDICATION

                ****************************************************************
                *                                                              *
                * MOVE HAS BEEN MODIFIED TO CAUSE EITHER A TRANSFER INTO OR OUT *
                * THE BUFFER.                                                  *
                *                                                              *
                ****************************************************************

E16D 3A93E5     MOVE    LDA     CPMSEC          ;GET THE CP/M SECTOR TO TRANSFER
E170 3D                 DCR     A               ;ADJUST TO PROPER SECTOR IN BUFFER
E171 E600               ANI     0               ;STRIP OFF HIGH ORDERED BITS
E172 =          SECPSEC EQU     $-1             ;THE 0 IS MODIFIED TO REPRESENT THE # OF
                                                ;       CP/M SECTORS PER PHYSICAL SECTORS
E173 6F                 MOV     L,A             ;PUT INTO HL
E174 2600               MVI     H,0
E176 29                 DAD     H               ;FORM OFFSET INTO BUFFER
E177 29                 DAD     H
E178 29                 DAD     H
E179 29                 DAD     H
E17A 29                 DAD     H
E17B 29                 DAD     H
E17C 29                 DAD     H
E17D 119DE5             LXI     D,BUFFER        ;BEGINNING ADDRESS OF BUFFER
E180 19                 DAD     D               ;FORM BEGINNING ADDRESS OF SECTOR TO TRANSFER
E181 EB                 XCHG                    ;DE = ADDRESS IN BUFFER
E182 210000             LXI     H,0             ;GET DMA ADDRESS, THE 0 IS MODIFIED TO
                                                ;       CONTAIN THE DMA ADDRESS
E183 =          CPMDMA  EQU     $-2
E185 3E00               MVI     A,0             ;THE ZERO GETS MODIFIED TO CONTAIN
                                                ;       A ZERO IF A READ, OR A 1 IF WRITE
E186 =          RDWR    EQU     $-1
E187 A7                 ANA     A               ;TEST WHICH KIND OF OPERATION
E188 C290E1             JNZ     INTO            ;TRANSFER DATA INTO THE BUFFER
E18B CD68E2     OUTOF   CALL    MOVER
E18E AF                 XRA     A
E18F C9                 RET

E190 EB         INTO    XCHG                    ;
E191 CD68E2             CALL    MOVER           ;MOVE THE DATA, HL = DESTINATION
                                                ;       DE = SOURCE
E194 3E01               MVI     A,1
E196 32A3E1             STA     BUFWRTN         ;SET BUFFER WRITTEN INTO FLAG
E199 3E00               MVI     A,0             ;CHECK FOR DIRECTORY WRITE
E19A =          WRITTYP EQU     $-1
E19B 3D                 DCR     A
E19C 3E00               MVI     A,0
E19E 329AE1             STA     WRITTYP         ;SET NO DIRECTORY WRITE
E1A1 C0                 RNZ                     ;NO ERROR EXIT

                ****************************************************************
                *                                                              *
                * FLUSH WRITES THE CONTENTS OF THE BUFFER OUT TO THE DISK IF    *
                * IT HAS EVER BEEN WRITTEN INTO.                               *
```

```
                ;                                                       *
                ;***************************************************************
                ;
E1A2 3E00       FLUSH   MVI     A,0             ;THE 0 IS MODIFIED TO REFLECT IF
                        ;                       ;       THE BUFFER HAS BEEN WRITTEN INTO
E1A3 =          BUFWRTN EQU     $-1
E1A4 A7                 ANA     A               ;TEST IF WRITTEN INTO
E1A5 C8                 RZ                      ;NOT WRITTEN, ALL DONE

                        IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E1A6 2118F4             LXI     H,DJWRITE       ;WRITE OPERATION FOR DISK JOCKEY
E1A9 1130E3             LXI     D,HDWRITE       ;WRITE OPERATION FOR HARD DISK
E1AC CD77E2             CALL    DECIDE
                        ELSE
                        IF      MAXHD NE 0
                        LXI     H,HDWRITE
                        ENDIF
                        IF      MAXFLOP NE 0
                        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.                                                     *
                ;                                                               *
                ;***************************************************************

E1AF AF         PREP    XRA     A               ;RESET BUFFER WRITTEN FLAG
E1B0 32A3E1             STA     BUFWRTN
E1B3 2215E2             SHLD    RETRYOP         ;SET UP THE READ/WRITE OPERATION
E1B6 060A               MVI     B,RETRIES       ;MAXIMUM NUMBER OF RETRIES TO ATTEMPT
E1B8 C5         RETRYLP PUSH    B               ;SAVE THE RETRY COUNT
E1B9 3A99E5             LDA     BUFDRV          ;GET DRIVE NUMBER INVOLVED IN THE OPERATION

                        IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
                        IF      FIRST
                        CPI     MAXHD*LOGDSK
                        JC      NOADJST
                        SUI     MAXHD*LOGDSK
                        ELSE
E1BC FE02               CPI     MAXFLOP
E1BE DAC3E1             JC      NOADJST
E1C1 D602               SUI     MAXFLOP
                        ENDIF

E1C3 4F         NOADJST MOV     C,A
E1C4 2133DD             LXI     H,DJDRV         ;SELECT DRIVE
E1C7 117FE2             LXI     D,HDDRV
E1CA CD73E2             CALL    DECIDGO
                        ELSE
                        MOV     C,A
                        IF      MAXHD NE 0
                        CALL    HDDRV
```

```
                        ENDIF
                        IF      MAXFLOP NE Ø
                        CALL    DJDRV           ;SELECT THE DRIVE
                        ENDIF
                        ENDIF

 E1CD 3A9AE5            LDA     BUFTRK
 E1DØ A7                ANA     A               ;TEST FOR TRACK ZERO
 E1D1 4F                MOV     C,A
 E1D2 C5                PUSH    B

                        IF      (MAXHD NE Ø) AND (MAXFLOP NE Ø)
 E1D3 21Ø9F4            LXI     H,DJHOME
 E1D6 119ØE2            LXI     D,HDHOME
 E1D9 CC73E2            CZ      DECIDGO
                        ELSE
                        IF      MAXHD NE Ø
                        CZ      HDHOME
                        ENDIF
                        IF      MAXFLOP NE Ø
                        CZ      DJHOME          ;HOME THE DRIVE IF TRACK Ø
                        ENDIF
                        ENDIF

 E1DC C1                POP     B               ;RESTORE TRACK #

                        IF      (MAXHD NE Ø) AND (MAXFLOP NE Ø)
 E1DD 21ØCF4            LXI     H,DJTRK
 E1EØ 11B1E2            LXI     D,HDTRK
 E1E3 CD73E2            CALL    DECIDGO
                        ELSE
                        IF      MAXHD NE Ø
                        CALL    HDTRK
                        ENDIF
                        IF      MAXFLOP NE Ø
                        CALL    DJTRK           ;SEEK TO PROPER TRACK
                        ENDIF
                        ENDIF

 E1E6 2A9BE5            LHLD    BUFSEC
 E1E9 7C                MOV     A,H             ;GET SECTOR INVOLVED IN OPERATION
 E1EA Ø7                RLC                     ;BIT Ø OF A EQUALS SIDE #
 E1EB E6Ø1              ANI     1               ;STRIP OFF UNNECESSARY BITS
 E1ED 4F                MOV     C,A             ;C <- SIDE #

                        IF      (MAXHD NE Ø) AND (MAXFLOP NE Ø)
 E1EE 213ØF4            LXI     H,DJSIDE
 E1F1 11DDE2            LXI     D,HDSIDE
 E1F4 CD73E2            CALL    DECIDGO
                        ELSE
                        IF      MAXHD NE Ø
                        CALL    HDSIDE
                        ENDIF
                        IF      MAXFLOP NE Ø
                        CALL    DJSIDE          ;SELECT THE SIDE
                        ENDIF
```

```
                        ENDIF

 E1F7 2A9BE5            LHLD    BUFSEC
 E1FA 7C                MOV     A,H
 E1FB E67F              ANI     7FH             ;STRIP OFF SIDE BIT
 E1FD 47                MOV     B,A             ;C <- SECTOR #
 E1FE 4D                MOV     C,L

                        IF      (MAXHD NE Ø) AND (MAXFLOP NE Ø)
 E1FF 210FF4            LXI     H,DJSEC
 E202 11E6E2            LXI     D,HDSEC
 E205 CD73E2            CALL    DECIDGO
                        ELSE
                        IF      MAXHD NE Ø
                        CALL    HDSEC
                        ENDIF
                        IF      MAXFLOP NE Ø
                        CALL    DJSEC           ;SELECT THE SIDE
                        ENDIF
                        ENDIF

 E208 Ø19DE5            LXI     B,BUFFER        ;SET THE DMA ADDRESS

                        IF      (MAXHD NE Ø) AND (MAXFLOP NE Ø)
 E20B 2112F4            LXI     H,DJDMA
 E20E 11D8E2            LXI     D,HDDMA
 E211 CD73E2            CALL    DECIDGO
                        ELSE
                        IF      MAXHD NE Ø
                        CALL    HDDMA
                        ENDIF
                        IF      MAXFLOP NE Ø
                        CALL    DJDMA           ;SELECT THE SIDE
                        ENDIF
                        ENDIF

 E214 CDØØØØ            CALL    Ø               ;GET OPERATION ADDRESS
 E215 =         RETRYOP EQU     $-2
 E217 C1                POP     B               ;RESTORE THE RETRY COUNTER
 E218 3EØØ              MVI     A,Ø             ;NO ERROR EXIT STATUS
 E21A DØ                RNC                     ;RETURN NO ERROR
 E21B Ø5                DCR     B               ;UPDATE THE RETRY COUNTER
 E21C 37                STC                     ;ASSUME RETRY COUNT EXPIRED
 E21D 3EFF              MVI     A,ØFFH          ;ERROR RETURN
 E21F C8                RZ
 E22Ø 78                MOV     A,B
 E221 FEØ5              CPI     RETRIES/2
 E223 C2B8E1            JNZ     RETRYLP         ;TRY AGAIN

 E226 C5                PUSH    B
                        IF      (MAXHD NE Ø) AND (MAXFLOP NE Ø)
 E227 2109F4            LXI     H,DJHOME
 E22A 1190E2            LXI     D,HDHOME
 E22D CD73E2            CALL    DECIDGO
                        ELSE
                        IF      MAXHD NE Ø
```

```
                        CALL    HDHOME
                        ENDIF
                        IF      MAXFLOP NE Ø
                        CALL    DJHOME          ;HOME THE DRIVE IF TRACK Ø
                        ENDIF
                        ENDIF

E23Ø C1                 POP     B
E231 C3B8E1             JMP     RETRYLP

                ;*****************************************************************
                ;*                                                               *
                ;* FILL FILLS THE BUFFER WITH A NEW SECTOR FROM THE DISK.        *
                ;*                                                               *
                ;*****************************************************************

E234 CDA2E1     FILL    CALL    FLUSH           ;FLUSH BUFFER FIRST
E237 D8                 RC                      ;CHECK FOR ERROR
E238 1195E5             LXI     D,CPMDRV        ;UPDATE THE DRIVE, TRACK, AND SECTOR
E23B 2199E5             LXI     H,BUFDRV
E23E Ø6Ø4               MVI     B,4             ;NUMBER OF BYTES TO MOVE
E24Ø CD6AE2             CALL    MOVLOP          ;COPY THE DATA

E243 3A86E1             LDA     RDWR
E246 A7                 ANA     A
E247 CA5CE2             JZ      FREAD
E24A 3A9AE1             LDA     WRITTYP
E24D 3D                 DCR     A
E24E 3D                 DCR     A
E24F C8                 RZ
E25Ø CDØ7E1             CALL    GETDPB
E253 11ØFØØ             LXI     D,15
E256 19                 DAD     D
E257 7E                 MOV     A,M
E258 E6Ø3               ANI     3
E25A 3D                 DCR     A
E25B C8                 RZ

E25C =          FREAD   EQU     $
                        IF      (MAXHD NE Ø) AND (MAXFLOP NE Ø)
E25C 2115F4             LXI     H,DJREAD
E25F 11FBE2             LXI     D,HDREAD
E262 CD77E2             CALL    DECIDE
                        ELSE
                        IF      MAXHD NE Ø
                        LXI     H,HDREAD
                        ENDIF
                        IF      MAXFLOP NE Ø
                        LXI     H,DJREAD        ;SELECT THE SIDE
                        ENDIF
                        ENDIF
E265 C3AFE1             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.                                                   *
                *                                                                  *
                ********************************************************************

E268 0680       MOVER   MVI     B,128           ;LENGTH OF TRANSFER
E26A 1A         MOVLOP  LDAX    D               ;GET A BTE OF SOURCE
E26B 77                 MOV     M,A             ;MOVE IT
E26C 13                 INX     D               ;BUMP POINTERS
E26D 23                 INX     H
E26E 05                 DCR     B               ;UPDATE COUNTER
E26F C26AE2             JNZ     MOVLOP          ;CONTINUE MOVING UNTIL DONE
E272 C9                 RET

                ********************************************************************
                *                                                                  *
                * ROUTINES TO DECIDE WHICH CONTROLLER TO USE.                      *
                *                                                                  *
                ********************************************************************

                        IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E273 CD77E2     DECIDGO CALL    DECIDE  ;WHICH CONTROLLER ?
E276 E9                 PCHL
                        ENDIF

                        IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E277 3A99E5     DECIDE  LDA     BUFDRV          ;GET PROPER ROUTINE INTO H&L, BASED
                        IF      FIRST           ;   ON CURRENTLY SELECTED DRIVE
                        CPI     MAXHD*LOGDSK
                        RNC
                        ELSE
E27A FE02               CPI     MAXFLOP
E27C D8                 RC
                        ENDIF
E27D EB                 XCHG
E27E C9                 RET
                        ENDIF

                ********************************************************************
                *                                                                  *
                * THE FOLLOWING IS THE EQUIVALENT OF THE LOWEST LEVEL DRIVERS      *
                * FOR THE HARD DISK.                                               *
                *                                                                  *
                ********************************************************************

                        IF      MAXHD NE 0
E27F 79         HDDRV   MOV     A,C             ;SELECT HARD DISK DRIVE
E280 CDFEE0             CALL    DIVLOG          ;GET THE PHYSICAL DRIVE #
E283 79                 MOV     A,C
E284 32C0E3             STA     HDDISK          ;SELECT THE DRIVE
E287 F6FC               ORI     NULL
E289 D352               OUT     HDFUNC
E28B 3E0F               MVI     A,WENABL
E28D D350               OUT     HDCNTL
E28F C9                 RET
```

```
E290 CDAEE3     HDHOME  CALL    DRVPTR
E293 3600               MVI     M,0             ;SET TRACK TO ZERO

                        IF      SDELAY
E295 DB50       STEPO   IN      HDSTAT          ;TEST STATUS
E297 E601               ANI     TKZERO          ;AT TRACK ZERO ?
E299 CAA5E2             JZ      DELAY
E29C 3E01               MVI     A,1
E29E 37                 STC
E29F CDC5E2             CALL    ACCOK           ;TAKE ONE STEP OUT
E2A2 C395E2             JMP     STEPO

                        ELSE

                        IN      HDSTAT
                        ANI     TKZERO
                        RZ
                        XRA     A
                        JMP     ACCOK
                        ENDIF

                        IF      SDELAY
E2A5 210000     DELAY   LXI     H,0             ;GET DELAY
E2A6 =          SETTLE  EQU     $-2
E2A8 2B         DELOOP  DCX     H               ;WAIT 20MS
E2A9 7C                 MOV     A,H
E2AA B5                 ORA     L
E2AB 23                 INX     H
E2AC 2B                 DCX     H
E2AD C2A8E2             JNZ     DELOOP
E2B0 C9                 RET
                        ENDIF

E2B1 CDAEE3     HDTRK   CALL    DRVPTR          ;GET POINTER TO CURRENT TRACK
E2B4 5E                 MOV     E,M             ;GET CURRENT TRACK
E2B5 71                 MOV     M,C             ;UPDATE THE TRACK
E2B6 7B                 MOV     A,E             ;NEED TO SEEK AT ALL ?
E2B7 91                 SUB     C
E2B8 C8                 RZ
E2B9 3F                 CMC                     ;GET CARRY INTO DIRECTION
E2BA DABFE2             JC      HDTRK2
E2BD 2F                 CMA
E2BE 3C                 INR     A
                        IF      NOT SDELAY
                HDTRK2  JMP     ACCOK
                        ELSE
E2BF CDC5E2     HDTRK2  CALL    ACCOK
E2C2 C3A5E2             JMP     DELAY
                        ENDIF

E2C5 47         ACCOK   MOV     B,A             ;PREP FOR BUILD
E2C6 CDB9E3             CALL    BUILD
E2C9 E6FB       SLOOP   ANI     NSTEP           ;GET STEP PULSE LOW
E2CB D352               OUT     HDFUNC          ;OUTPUT LOW STEP LINE
E2CD F604               ORI     PSTEP           ;SET STEP LINE HIGH
E2CF D352               OUT     HDFUNC          ;OUTPUT HIGH STEP LINE
```

```
E2D1 Ø5              DCR     B               ;UPDATE REPEAT COUNT
E2D2 C2C9E2          JNZ     SLOOP           ;KEEP GOING THE REQUIRED # OF TRACKS
E2D5 C3DEE2          JMP     WSDONE

E2D8 6Ø        HDDMA MOV     H,B             ;SAVE THE DMA ADDRESS
E2D9 69              MOV     L,C
E2DA 2215E3          SHLD    HDADD
E2DD =         HDSIDE EQU    $
E2DD C9              RET

E2DE DB5Ø      WSDONE IN     HDSTAT          ;WAIT FOR SEEK COMPLETE TO FINISH
E2EØ E6Ø4            ANI     COMPLT
E2E2 CADEE2          JZ      WSDONE
E2E5 C9              RET

                     IF      M26
E2E6 3E1F      HDSEC MVI     A,Ø1FH          ;FOR COMPATIBILITY WITH CBIOS REV 2.3, 2.4
E2E8 A1              ANA     C
E2E9 CCF8E2          CZ      GETSPT
E2EC 329EE3          STA     HDSECTR
E2EF 3EEØ            MVI     A,ØE0H
E2F1 A1              ANA     C
E2F2 Ø7              RLC
E2F3 Ø7              RLC
E2F4 Ø7              RLC
E2F5 32BAE3          STA     HEAD
E2F8 3E2Ø      GETSPT MVI    A,HDSPT
E2FA C9              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

E2FB CD79E3     HDREAD CALL   HDPREP
E2FE D8                RC
E2FF AF                XRA    A
E3ØØ D351              OUT    HDCMND
E3Ø2 2F                CMA
E3Ø3 D353              OUT    HDDATA
```

```
E3Ø5 D353              OUT     HDDATA
E3Ø7 3EØ1              MVI     A,RSECT         ;READ SECTOR COMMAND
E3Ø9 D351              OUT     HDCMND
E3ØB CD5FE3            CALL    PROCESS
E3ØE D8                RC
E3ØF AF                XRA     A
E31Ø D351              OUT     HDCMND
E312 Ø68Ø              MVI     B,SECLEN/4
E314 21ØØØØ            LXI     H,Ø
E315 =         HDADD   EQU     $-2
E317 DB53              IN      HDDATA
E319 DB53              IN      HDDATA
E31B DB53      RTLOOP  IN      HDDATA          ;MOVE FOUR BYTES
E31D 77                MOV     M,A
E31E 23                INX     H
E31F DB53              IN      HDDATA
E321 77                MOV     M,A
E322 23                INX     H
E323 DB53              IN      HDDATA
E325 77                MOV     M,A
E326 23                INX     H
E327 DB53              IN      HDDATA
E329 77                MOV     M,A
E32A 23                INX     H
E32B Ø5                DCR     B
E32C C21BE3            JNZ     RTLOOP
E32F C9                RET

E33Ø CD79E3    HDWRITE CALL    HDPREP          ;PREPARE HEADER
E333 D8                RC
E334 AF                XRA     A
E335 D351              OUT     HDCMND
E337 2A15E3            LHLD    HDADD
E33A Ø68Ø              MVI     B,SECLEN/4
E33C 7E        WTLOOP  MOV     A,M             ;MOVE 4 BYTES
E33D D353              OUT     HDDATA
E33F 23                INX     H
E34Ø 7E                MOV     A,M
E341 D353              OUT     HDDATA
E343 23                INX     H
E344 7E                MOV     A,M
E345 D353              OUT     HDDATA
E347 23                INX     H
E348 7E                MOV     A,M
E349 D353              OUT     HDDATA
E34B 23                INX     H
E34C Ø5                DCR     B
E34D C23CE3            JNZ     WTLOOP
E35Ø 3EØ5              MVI     A,WSECT         ;ISSUE WRITE SECTOR COMMAND
E352 D351              OUT     HDCMND
E354 CD5FE3            CALL    PROCESS
E357 D8                RC
E358 3E1Ø              MVI     A,WFAULT
E35A AØ                ANA     B
E35B 37                STC
E35C C8                RZ
```

```
E35D AF                 XRA     A
E35E C9                 RET

E35F DB50       PROCESS IN      HDSTAT          ;WAIT FOR COMMAND TO FINISH
E361 47                 MOV     B,A
E362 E602               ANI     OPDONE
E364 CA5FE3             JZ      PROCESS
E367 3E07               MVI     A,DSKCLK
E369 D350               OUT     HDCNTL
E36B DB50               IN      HDSTAT
E36D E608               ANI     TMOUT           ;TIMED OUT ?
E36F 37                 STC
E370 C0                 RNZ
E371 DB51               IN      HDRESLT
E373 E602               ANI     RETRY           ;ANY RETRIES ?
E375 37                 STC
E376 C0                 RNZ
E377 AF                 XRA     A
E378 C9                 RET

E379 DB50       HDPREP  IN      HDSTAT
E37B E620               ANI     DRVRDY
E37D 37                 STC
E37E C0                 RNZ
E37F 3E08               MVI     A,ISBUFF        ;INITIALIZE POINTER
E381 D351               OUT     HDCMND
E383 CDB9E3             CALL    BUILD
E386 F60C               ORI     0CH
E388 D352               OUT     HDFUNC
E38A 3ABAE3             LDA     HEAD
E38D D353               OUT     HDDATA          ;FORM HEAD BYTE
E38F CDAEE3             CALL    DRVPTR
E392 7E                 MOV     A,M             ;FORM TRACK BYTE
E393 D353               OUT     HDDATA
E395 A7                 ANA     A
E396 0680               MVI     B,80H
E398 CA9DE3             JZ      ZKEY
E39B 0600               MVI     B,0
E39D 3E00       ZKEY    MVI     A,0             ;FORM SECTOR BYTE
E39E =          HDSECTR EQU     $-1
E39F D353               OUT     HDDATA
E3A1 78                 MOV     A,B
E3A2 D353               OUT     HDDATA
E3A4 3E07               MVI     A,DSKCLK
E3A6 D350               OUT     HDCNTL
E3A8 3E0F               MVI     A,WENABL
E3AA D350               OUT     HDCNTL
E3AC AF                 XRA     A
E3AD C9                 RET

E3AE 2AC0E3     DRVPTR  LHLD    HDDISK
E3B1 EB                 XCHG
E3B2 1600               MVI     D,0
E3B4 21C4E3             LXI     H,DRIVES
E3B7 19                 DAD     D
E3B8 C9                 RET
```

```
E3B9 3E00       BUILD   MVI     A,0
E3BA =          HEAD    EQU     $-1
E3BB 17                 RAL
E3BC 17                 RAL
E3BD 17                 RAL
E3BE 17                 RAL
E3BF F600               ORI     0
E3C0 =          HDDISK  EQU     $-1
E3C1 EEF0               XRI     0F0H
E3C3 C9                 RET

E3C4 =          DRIVES  EQU     $
                        REPT    MAXHD
                        DB      0FFH
                        ENDM
E3C4+FF                 DB      0FFH
                        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
E3C5 00         XLT128  DB      0
E3C6 01070D1319         DB      1,7,13,19,25
E3CB 050B1117           DB      5,11,17,23
E3CF 03090F15           DB      3,9,15,21
E3D3 02080E141A         DB      2,8,14,20,26
E3D8 060C1218           DB      6,12,18,24
E3DC 040A1016           DB      4,10,16,22

E3E0 00         XLT256  DB      0
E3E1 0102131425         DB      1,2,19,20,37,38
E3E7 0304151627         DB      3,4,21,22,39,40
E3ED 0506171829         DB      5,6,23,24,41,42
E3F3 0708191A2B         DB      7,8,25,26,43,44
E3F9 090A1B1C2D         DB      9,10,27,28,45,46
E3FF 0B0C1D1E2F         DB      11,12,29,30,47,48
E405 0D0E1F2031         DB      13,14,31,32,49,50
E40B 0F10212233         DB      15,16,33,34,51,52
E411 11122324           DB      17,18,35,36

E415 00         XLT512  DB      0
E416 0102030411         DB      1,2,3,4,17,18,19,20
E41E 2122232431         DB      33,34,35,36,49,50,51,52
E426 0506070815         DB      5,6,7,8,21,22,23,24
E42E 2526272835         DB      37,38,39,40,53,54,55,56
E436 090A0B0C19         DB      9,10,11,12,25,26,27,28
```

```
E43E 292A2B2C39          DB      41,42,43,44,57,58,59,60
E446 0D0E0F101D          DB      13,14,15,16,29,30,31,32
E44E 2D2E2F30            DB      45,46,47,48

E452 00          XLT124  DB      0
E453 0102030405          DB      1,2,3,4,5,6,7,8
E45B 191A1B1C1D          DB      25,26,27,28,29,30,31,32
E463 3132333435          DB      49,50,51,52,53,54,55,56
E46B 090A0B0C0D          DB      9,10,11,12,13,14,15,16
E473 2122232425          DB      33,34,35,36,37,38,39,40
E47B 393A3B3C3D          DB      57,58,59,60,61,62,63,64
E483 1112131415          DB      17,18,19,20,21,22,23,24
E48B 292A2B2C2D          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.                             *
                ;*                                                               *
                ;*****************************************************************

E493 1A00       DPB128S DW      26              ;CP/M SECTORS/TRACK
E495 03                 DB      3               ;BSH
E496 07                 DB      7               ;BLM
E497 00                 DB      0               ;EXM
E498 F200               DW      242             ;DSM
E49A 3F00               DW      63              ;DRM
E49C C0                 DB      0C0H            ;AL0
E49D 00                 DB      0               ;AL1
E49E 1000               DW      16              ;CKS
E4A0 0200               DW      2               ;OFF
E4A2 01                 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.                             *
                ;*                                                               *
                ;*****************************************************************

E4A3 3400       DPB256S DW      52              ;CP/M SECTORS/TRACK
E4A5 04                 DB      4               ;BSH
E4A6 0F                 DB      15              ;BLM
E4A7 00                 DB      0               ;EXM
E4A8 F200               DW      242             ;DSM
E4AA 7F00               DW      127             ;DRM
```

```
E4AC C0                 DB      0C0H            ;AL0
E4AD 00                 DB      0               ;AL1
E4AE 2000               DW      32              ;CKS
E4B0 0200               DW      2               ;OFF
E4B2 12                 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.                            *
                *                                                              *
                ****************************************************************

E4B3 3C00       DPB512S DW      60              ;CP/M SECTORS/TRACK
E4B5 04                 DB      4               ;BSH
E4B6 0F                 DB      15              ;BLM
E4B7 00                 DB      0               ;EXM
E4B8 1801               DW      280             ;DSM
E4BA 7F00               DW      127             ;DRM
E4BC C0                 DB      0C0H            ;AL0
E4BD 00                 DB      0               ;AL1
E4BE 2000               DW      32              ;CKS
E4C0 0200               DW      2               ;OFF
E4C2 33                 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 1024 BYTE SECTORS,   *
                * DOUBLE DENSITY, AND SINGLE SIDED.                            *
                *                                                              *
                ****************************************************************

E4C3 4000       DP1024S DW      64              ;CP/M SECTORS/TRACK
E4C5 04                 DB      4               ;BSH
E4C6 0F                 DB      15              ;BLM
E4C7 00                 DB      0               ;EXM
E4C8 2B01               DW      299             ;DSM
E4CA 7F00               DW      127             ;DRM
E4CC C0                 DB      0C0H            ;AL0
E4CD 00                 DB      0               ;AL1
E4CE 2000               DW      32              ;CKS
E4D0 0200               DW      2               ;OFF
E4D2 74                 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.                            *
                *                                                              *
                ****************************************************************
```

```
E4D3 3400       DPB128D DW      52              ;CP/M SECTORS/TRACK
E4D5 04                 DB      4               ;BSH
E4D6 0F                 DB      15              ;BLM
E4D7 01                 DB      1               ;EXM
E4D8 F200               DW      242             ;DSM
E4DA 7F00               DW      127             ;DRM
E4DC C0                 DB      0C0H            ;AL0
E4DD 00                 DB      0               ;AL1
E4DE 2000               DW      32              ;CKS
E4E0 0200               DW      2               ;OFF
E4E2 09                 DB      9H

                ;*****************************************************************
                ;*                                                               *
                ;* THE FOLLOWING DPB DEFINES A DISKETTE AS 256 BYTE SECTORS,     *
                ;* DOUBLE DENSITY, AND DOUBLE SIDED.                             *
                ;*                                                               *
                ;*****************************************************************

E4E3 6800       DPB256D DW      104             ;CP/M SECTORS/TRACK
E4E5 04                 DB      4               ;BSH
E4E6 0F                 DB      15              ;BLM
E4E7 00                 DB      0               ;EXM
E4E8 E601               DW      486             ;DSM
E4EA FF00               DW      255             ;DRM
E4EC F0                 DB      0F0H            ;AL0
E4ED 00                 DB      0               ;AL1
E4EE 4000               DW      64              ;CKS
E4F0 0200               DW      2               ;OFF
E4F2 1A                 DB      1AH

                ;*****************************************************************
                ;*                                                               *
                ;* THE FOLLOWING DPB DEFINES A DISKETTE AS 512 BYTE SECTORS,     *
                ;* DOUBLE DENSITY, AND DOUBLE SIDED.                             *
                ;*                                                               *
                ;*****************************************************************

E4F3 7800       DPB512D DW      120             ;CP/M SECTORS/TRACK
E4F5 04                 DB      4               ;BSH
E4F6 0F                 DB      15              ;BLM
E4F7 00                 DB      0               ;EXM
E4F8 3102               DW      561             ;DSM
E4FA FF00               DW      255             ;DRM
E4FC F0                 DB      0F0H            ;AL0
E4FD 00                 DB      0               ;AL1
E4FE 4000               DW      64              ;CKS
E500 0200               DW      2               ;OFF
E502 3B                 DB      3BH

                ;*****************************************************************
                ;*                                                               *
                ;* THE FOLLOWING DPB DEFINES A DISKETTE AS 1024 BYTE SECTORS,    *
                ;* DOUBLE DENSITY, AND DOUBLE SIDED.                             *
                ;*                                                               *
```

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

E503 8000       DP1024D DW      128             ;CP/M SECTORS/TRACK
E505 04                 DB      4               ;BSH
E506 0F                 DB      15              ;BLM
E507 00                 DB      0               ;EXM
E508 5702               DW      599             ;DSM
E50A FF00               DW      255             ;DRM
E50C F0                 DB      0F0H            ;AL0
E50D 00                 DB      0               ;AL1
E50E 4000               DW      64              ;CKS
E510 0200               DW      2               ;OFF
E512 7C                 DB      7CH
                        ENDIF

                ;*****************************************************************
                ;                                                               *
                ; THE FOLLOWING DPB DEFINES A 10 MEGABYTE HARD DISK, WITH 512   *
                ; BYTE SECTORS.                                                 *
                ;                                                               *
                ;*****************************************************************

                        IF      MAXHD NE 0
                        IF      M26 NE 0
E513 0004       DPBHD1  DW      1024            ;CP/M SECTORS/TRACK
E515 05                 DB      5               ;BSH
E516 1F                 DB      31              ;BLM
E517 01                 DB      1               ;EXM
E518 B507               DW      1973            ;DSM
E51A FF01               DW      511             ;DRM
E51C FF                 DB      0FFH            ;AL0
E51D FF                 DB      0FFH            ;AL1
E51E 0000               DW      0               ;CKS
E520 0100               DW      1               ;OFF
E522 33                 DB      33H             ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
                                                ;LOG2(#BYTES PER SECTOR/128) + 1 +
                                                ;8 IF DOUBLE SIDED.
E523 0004       DPBHD2  DW      1024            ;CP/M SECTORS/TRACK
E525 05                 DB      5               ;BSH
E526 1F                 DB      31              ;BLM
E527 01                 DB      1               ;EXM
E528 B507               DW      1973            ;DSM
E52A FF01               DW      511             ;DRM
E52C FF                 DB      0FFH            ;AL0
E52D FF                 DB      0FFH            ;AL1
E52E 0000               DW      0               ;CKS
E530 4000               DW      64              ;OFF
E532 33                 DB      33H             ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
                                                ;LOG2(#BYTES PER SECTOR/128) + 1 +
                                                ;8 IF DOUBLE SIDED.

E533 0004       DPBHD3  DW      1024            ;CP/M SECTORS/TRACK
E535 05                 DB      5               ;BSH
E536 1F                 DB      31              ;BLM
E537 01                 DB      1               ;EXM
E538 B507               DW      1973            ;DSM
```

```
E53A FF01               DW      511             ;DRM
E53C FF                 DB      0FFH            ;AL0
E53D FF                 DB      0FFH            ;AL1
E53E 0000               DW      0               ;CKS
E540 7F00               DW      127             ;OFF
E542 33                 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      ØFFH            ;ALØ
                        DB      ØFFH            ;AL1
                        DW      Ø               ;CKS
                        DW      98              ;OFF
                        DB      33H             ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
                                                ;LOG2(#BYTES PER SECTOR/128) + 1 +
                                                ;8 IF DOUBLE SIDED.

                DPBHD3  DW      672             ;CP/M SECTORS/TRACK
                        DB      5               ;BSH
                        DB      31              ;BLM
                        DB      1               ;EXM
                        DW      1Ø28            ;DSM
                        DW      511             ;DRM
                        DB      ØFFH            ;ALØ
                        DB      ØFFH            ;AL1
                        DW      Ø               ;CKS
                        DW      195             ;OFF
                        DB      33H             ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
                                                ;LOG2(#BYTES PER SECTOR/128) + 1 +
                                                ;8 IF DOUBLE SIDED.
                        ENDIF
                        ENDIF

                ;*****************************************************************
                ;                                                               *
                ; CP/M DISK PARAMETER HEADERS, UNITIALIZED.                     *
                ;                                                               *
                ;*****************************************************************

                HEADER  MACRO   ND,DPB
                        DW      Ø               ;TRANSLATION TABLE FILLED IN LATER
                        DW      Ø,Ø,Ø           ;SCRATCH
                        DW      DIRBUF          ;DIRECTORY BUFFER
                        DW      DPB             ;DPB FILLED IN LATER
                        DW      CSV&ND          ;DIRECTORY CHECK VECTOR
                        DW      ALV&ND          ;ALLOCATION VECTOR
                        ENDM

E543 =          DPBASE  EQU     $
Ø000 #          DN      SET     Ø
                        IF      FIRST
                        REPT    MAXHD           ;GENERATE HARD DISK DPH'S FOLLOWED
                        HEADER  %DN,DPBHD1      ;       BY FLOPPY DPH'S
                DN      SET     DN+1
                        HEADER  %DN,DPBHD2
                DN      SET     DN+1
                        IF      (M26 NE Ø) OR (M2Ø NE Ø)
                        HEADER  %DN,DPBHD3
                DN      SET     DN+1
                        ENDIF
                        ENDM
                        REPT    MAXFLOP
                        HEADER  %DN,Ø
                DN      SET     DN+1
```

```
                        ENDM
                        ELSE
                        REPT    MAXFLOP         ;GENERATE FLOPPY DPH'S FOLLOWED BY
                        HEADER  %DN,0           ;       HARD DISK DPH'S
                DN      SET     DN+1
                        ENDM
 E543+0000              DW      0               ;TRANSLATION TABLE FILLED IN LATER
 E545+0000000000        DW      0,0,0           ;SCRATCH
 E54B+9DE9              DW      DIRBUF          ;DIRECTORY BUFFER
 E54D+0000              DW      0               ;DPB FILLED IN LATER
 E54F+68EA              DW      CSV0            ;DIRECTORY CHECK VECTOR
 E551+1DEA              DW      ALV0            ;ALLOCATION VECTOR
 E553+0000              DW      0               ;TRANSLATION TABLE FILLED IN LATER
 E555+0000000000        DW      0,0,0           ;SCRATCH
 E55B+9DE9              DW      DIRBUF          ;DIRECTORY BUFFER
 E55D+0000              DW      0               ;DPB FILLED IN LATER
 E55F+F3EA              DW      CSV1            ;DIRECTORY CHECK VECTOR
 E561+A8EA              DW      ALV1            ;ALLOCATION VECTOR
                        REPT    MAXHD
                        HEADER  %DN,DPBHD1
                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
 E563+0000              DW      0               ;TRANSLATION TABLE FILLED IN LATER
 E565+0000000000        DW      0,0,0           ;SCRATCH
 E56B+9DE9              DW      DIRBUF          ;DIRECTORY BUFFER
 E56D+13E5              DW      DPBHD1          ;DPB FILLED IN LATER
 E56F+2AEC              DW      CSV2            ;DIRECTORY CHECK VECTOR
 E571+33EB              DW      ALV2            ;ALLOCATION VECTOR
 E573+0000              DW      0               ;TRANSLATION TABLE FILLED IN LATER
 E575+0000000000        DW      0,0,0           ;SCRATCH
 E57B+9DE9              DW      DIRBUF          ;DIRECTORY BUFFER
 E57D+23E5              DW      DPBHD2          ;DPB FILLED IN LATER
 E57F+21ED              DW      CSV3            ;DIRECTORY CHECK VECTOR
 E581+2AEC              DW      ALV3            ;ALLOCATION VECTOR
 E583+0000              DW      0               ;TRANSLATION TABLE FILLED IN LATER
 E585+0000000000        DW      0,0,0           ;SCRATCH
 E58B+9DE9              DW      DIRBUF          ;DIRECTORY BUFFER
 E58D+33E5              DW      DPBHD3          ;DPB FILLED IN LATER
 E58F+18EE              DW      CSV4            ;DIRECTORY CHECK VECTOR
 E591+21ED              DW      ALV4            ;ALLOCATION VECTOR
                        ENDIF

                ;*****************************************************************
                ;                                                               *
                ; CBIOS RAM LOCATIONS THAT DON'T NEED INITIALIZATION.           *
                ;                                                               *
                ;*****************************************************************

 E593 0000      CPMSEC  DW      0               ;CP/M SECTOR #
 E595 00        CPMDRV  DB      0               ;CP/M DRIVE #
```

```
E596 ØØ          CPMTRK  DB      Ø               ;CP/M TRACK #
E597 ØØØØ        TRUESEC DW      Ø               ;DISK JOCKEY SECTOR THAT CONTAINS CP/M SECTOR
E599 ØØ          BUFDRV  DB      Ø               ;DRIVE THAT BUFFER BELONGS TO
E59A ØØ          BUFTRK  DB      Ø               ;TRACK THAT BUFFER BELONGS TO
E59B ØØØØ        BUFSEC  DW      Ø               ;SECTOR THAT BUFFER BELONGS TO
E59D =           BUFFER  EQU     $

                 ;*****************************************************************
                 ;*                                                               *
                 ;* SIGNON MESSAGE OUTPUT DURING COLD BOOT.                       *
                 ;*                                                               *
                 ;*****************************************************************

                 HEXNUM  MACRO   NUM
                         IF      (NUM/16) > 9
                         DB      (NUM/16 AND ØFH) + 'A' - 1Ø
                         ELSE
                         DB      (NUM/16 AND ØFH) + 'Ø'
                         ENDIF
                         IF      (NUM AND ØFH) > 9
                         DB      (NUM AND ØFH) + 'A' - 1Ø
                         ELSE
                         DB      (NUM AND ØFH) + 'Ø'
                         ENDIF
                         ENDM

E59D ØDØAØA      PROMPT  DB      ACR,ALF,ALF
E5AØ 4D6F72726F          DB      'Morrow Designs '
E5AF 36                  DB      'Ø'+MSIZE/1Ø            ;CP/M MEMORY SIZE
E5BØ 3Ø                  DB      'Ø'+(MSIZE MOD 1Ø)
E5B1 4B2043Ø2F           DB      'K CP/M '               ;CP/M VERSION NUMBER
E5B8 32                  DB      CPMREV/1Ø+'Ø'
E5B9 2E                  DB      '.'
E5BA 32                  DB      (CPMREV MOD 1Ø)+'Ø'
E5BB 2C20436269          DB      ', Cbios rev '
E5C7 322E                DB      REVNUM/1Ø+'Ø','.'       ;CBIOS REVISION NUMBER
E5C9 38                  DB      REVNUM MOD 1Ø+'Ø'
                         IF      MAXHD NE Ø
E5CA 2E                  DB      '.'
E5CB 32                  DB      MREV/1Ø+'Ø'
E5CC 36                  DB      MREV MOD 1Ø+'Ø'
                         IF      (M1Ø OR M2Ø) AND SDELAY
                         DB      'M'
                         ENDIF
                         IF      (M1Ø OR M2Ø) AND NOT SDELAY
                         DB      'F'
                         ENDIF
                         ENDIF
E5CD ØDØA                DB      ACR,ALF
E5CF 466F7220            DB      'For '

                         IF      MAXFLOP NE Ø
E5D3 6120446973          DB      'a Disk Jockey 2D @ '
                         HEXNUM  %(ORIGIN/256)
E5E6+46                  DB      (24Ø/16 AND ØFH) + 'A' - 1Ø
E5E7+3Ø                  DB      (24Ø AND ØFH) + 'Ø'
```

```
CP/M MACRO ASSEM 2.0     #045    *** Cbios For CP/M Ver. 2.2 ***

 E5E8 30304820           DB      '00H '
                         ENDIF

                         IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
 E5EC 616E6420           DB      'and '
                         ENDIF

                         IF      MAXHD NE 0
                         IF      MAXHD EQ 1
 E5F0 616E20             DB      'an '
                         ENDIF
                         IF      MAXHD EQ 2
                         DB      'two '
                         ENDIF
                         IF      MAXHD EQ 3
                         DB      'three '
                         ENDIF
                         IF      MAXHD EQ 4
                         DB      'four '
                         ENDIF
                         IF      MREV EQ 10
                         DB      'M10 '
                         ENDIF
                         IF      MREV EQ 20
                         DB      'M20 '
                         ENDIF
                         IF      MREV EQ 26
 E5F3 4D323620           DB      'M26 '
                         ENDIF
 E5F7 6861726420         DB      'hard disk'
                         IF      MAXHD NE 1
                         DB      's'
                         ENDIF
 E600 204020             DB      ' @ '
                         HEXNUM  %HDORG
   E603+35               DB      (80/16 AND 0FH) + '0'
   E604+30               DB      (80 AND 0FH) + '0'
   E605 482E             DB      'H.'
                         ENDIF
   E607 0D0A             DB      ACR,ALF
   E609 0D0A             DB      ACR,ALF
   E60B 2020202020       DB      '       THE W6GO/K6HHD LIST'
   E625 0D0A             DB      ACR,ALF
   E627 2020202020       DB      '     Electronics Enterprises'
   E643 0D0A             DB      ACR,ALF
   E645 2020202020       DB      '      Rio Linda, California'
   E660 0D0A             DB      ACR,ALF
   E662 00               DB      0

                ********************************************************************
                *                                                                  *
                * UTILITY ROUTINE TO OUTPUT THE MESSAGE POINTED AT BY H&L,         *
                * TERMINATED WITH A NULL.                                          *
                *                                                                  *
                ********************************************************************
```

```
E663 7E        MESSAGE MOV     A,M             ;GET A CHARACTER OF THE MESSAGE
E664 23                INX     H               ;BUMP TEXT POINTER
E665 A7                ANA     A               ;TEST FOR END
E666 C8                RZ                      ;RETURN IF DONE
E667 E5                PUSH    H               ;SAVE POINTER TO TEXT
E668 4F                MOV     C,A             ;OUTPUT CHARACTER IN C
E669 CDØCDD            CALL    COUT            ;OUTPUT THE CHARACTER
E66C E1                POP     H               ;RESTORE THE POINTER
E66D C363E6            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.                                 *
                       *                                                              *
                       ****************************************************************

E67Ø 31ØØØ1    CBOOT   LXI     SP,TPA          ;SET UP STACK

E673 3ECØ              MVI     A,INTIOBY
E675 32Ø3ØØ            STA     IOBYTE
E678 CD2DDE            CALL    TINIT           ;INITIALIZE THE TERMINAL

E67B 219DE5            LXI     H,PROMPT        ;PREP FOR SENDING SIGNON MESSAGE
E67E CD63E6            CALL    MESSAGE         ;SEND THE PROMPT
E681 AF                XRA     A               ;SELECT DISK A
E682 3295E5            STA     CPMDRV
E685 32Ø4ØØ            STA     CDISK

                       IF      (MAXFLOP NE Ø) AND FIRST
                       STA     FLOPFLG
                       ENDIF
E688 2103DD            LXI     H,BIOS+3
E68B 2201DD            SHLD    BIOS+1
E68E C358DE            JMP     GOCPM

E691                   DS      512-($-BUFFER)  ;MAXIMUM SIZE BUFFER FOR 512 BYTE SECTORS

                       IF      MAXFLOP NE Ø
E79D                   DS      512             ;ADDITIONAL SPACE FOR FLOPPIES 1K SECTORS
                       ENDIF

                       IF      (MAXFLOP NE Ø) OR (MAXHD NE Ø)
E99D           DIRBUF  DS      128             ;DIRECTORY BUFFER
                       ENDIF

               ALLOC   MACRO   ND,AL,CS
               ALV&ND  DS      AL
               CSV&ND  DS      CS
                       ENDM

Ø000 #         DN      SET     Ø

                       IF      NOT FIRST
                       REPT    MAXFLOP
                       ALLOC   %DN,75,64
```

E690h
3090h in memory
TK1, SECTOR 54
BYTE 10h

```
                DN      SET     DN+1
                        ENDM
EA1D+           ALV0    DS      75
EA68+           CSV0    DS      64
EAA8+           ALV1    DS      75
EAF3+           CSV1    DS      64
                        REPT    MAXHD
                        IF      M26 NE 0
                        ALLOC   %DN,247,0
                DN      SET     DN+1
                        ALLOC   %DN,247,0
                DN      SET     DN+1
                        ALLOC   %DN,247,0
                DN      SET     DN+1
                        ENDIF
                        IF      M10 NE 0
                        ALLOC   %DN,159,0
                DN      SET     DN+1
                        ALLOC   %DN,161,0
                DN      SET     DN+1
                        ENDIF
                        IF      M20 NE 0
                        ALLOC   %DN,252,0
                DN      SET     DN+1
                        ALLOC   %DN,252,0
                DN      SET     DN+1
                        ALLOC   %DN,129,0
                DN      SET     DN+1
                        ENDIF
                        ENDM
EB33+           ALV2    DS      247
EC2A+           CSV2    DS      0
EC2A+           ALV3    DS      247
ED21+           CSV3    DS      0
ED21+           ALV4    DS      247
EE18+           CSV4    DS      0

                        ELSE

                        REPT    MAXHD
                        IF      M26 NE 0
                        ALLOC   %DN,247,0
                DN      SET     DN+1
                        ALLOC   %DN,247,0
                DN      SET     DN+1
                        ALLOC   %DN,247,0
                DN      SET     DN+1
                        ENDIF
                        IF      M10 NE 0
                        ALLOC   %DN,159,0
                DN      SET     DN+1
                        ALLOC   %DN,161,0
                DN      SET     DN+1
                        ENDIF
                        IF      M20 NE 0
                        ALLOC   %DN,252,0
```

```
                DN      SET     DN+1
                        ALLOC   %DN,252,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
 EE18                   END
```

0006 AACK	E2C5 ACCOK	000D ACR	0003 AETX	000A ALF
EA1D ALV0	EAA8 ALV1	EB33 ALV2	EC2A ALV3	ED21 ALV4
DF41 AUTOFLG	CF00 BDOS	0005 BDOSE	A000 BIAS	DD00 BIOS
E599 BUFDRV	0080 BUFF	E59D BUFFER	E59B BUFSEC	E59A BUFTRK
E1A3 BUFWRTN	E3B9 BUILD	E670 CBOOT	C700 CCP	0004 CDISK
DED5 CHAR2	DEE1 CHECK	DE08 CICRT	DE08 CIPTR	DD88 CITBLE
F003 CITTY	DDF3 CIUC1	DE08 CIUR1	DE08 CIUR2	DEB5 CLDBOT
DEA0 CLDCMND	001A CLEAR	DE3B COCRT	DF42 COLDBEG	DF50 COLDEND
DDC9 COLPT	0004 COMPLT	DD42 CONIN	DD48 CONIN1	DD57 CONOUT
DD36 CONST	DDC9 COPTP	DE46 COPTR	DE4D COPTR1	DD90 COTBLE
F006 COTTY	DDD4 COUL1	DDF2 COUNT	DDC9 COUP1	DDC9 COUP2
DD0C COUT	E183 CPMDMA	E595 CPMDRV	C8D0 CPMPT	0016 CPMREV
E593 CPMSEC	E596 CPMTRK	DE1C CSCRT	DE1C CSPTR	DD3C CSREADR
DDB8 CSRTBLE	DDB0 CSTBLE	DE14 CSTTY	DDFF CSUC1	DE1C CSUR1
DE1C CSUR2	EA68 CSV0	EAF3 CSV1	EC2A CSV2	ED21 CSV3
EE18 CSV4	DEBC CWFLG	0008 DBLSID	E0FC DCRC	E277 DECIDE
E273 DECIDGO	E2A5 DELAY	E2A8 DELOOP	E99D DIRBUF	E14F DIVDONE
E0FE DIVLOG	E100 DIVLOGX	E141 DIVLOOP	F400 DJBOOT	F003 DJCIN
F006 DJCOUT	F42D DJDEN	F412 DJDMA	DD33 DJDRV	F42A DJERR
F409 DJHOME	F400 DJRAM	F415 DJREAD	F40F DJSEC	F41B DJSEL
F430 DJSIDE	F427 DJSTAT	F40C DJTRK	F021 DJTSTAT	F418 DJWRITE
DFE0 DONOP	E503 DP1024D	E4C3 DP1024S	E4D3 DPB128D	E493 DPB128S
E4E3 DPB256D	E4A3 DPB256S	E4F3 DPB512D	E4B3 DPB512S	E543 DPBASE
E513 DPBHD1	E523 DPBHD2	E533 DPBHD3	E3C4 DRIVES	E097 DRVHD
E3AE DRVPTR	0020 DRVRDY	0007 DSKCLK	E15E DTSLOP	0005 ENTRY
CE01 EOF	DED0 EXIT1	DF10 EXIT	E234 FILL	0000 FIRST
DF3F FLAG	E040 FLOPOK	E1A2 FLUSH	DF16 FOUND	E25C FREAD
E107 GETDPB	E2F8 GETSPT	DE58 GOCPM	E315 HDADD	0051 HDCMND
0050 HDCNTL	0053 HDDATA	E3C0 HDDISK	E2D8 HDDMA	E27F HDDRV
0052 HDFUNC	E290 HDHOME	0050 HDORG	E379 HDPREP	E2FB HDREAD
0051 HDRESLT	0004 HDRLEN	E2E6 HDSEC	E39E HDSECTR	E2DD HDSIDE
0020 HDSPT	0050 HDSTAT	E2B1 HDTRK	E2BF HDTRK2	E330 HDWRITE
E3BA HEAD	DFE7 HOME	0000 IDBUFF	0040 INDEX	E0CA INDX1
E0D1 INDX2	00C0 INTIOBY	E190 INTO	0003 IOBYTE	0008 ISBUFF
DD77 LIST	DD7A LIST1	DD82 LISTST	0003 LOGDSK	DE25 LSLPT
DDC0 LSTBLE	DD98 LTBLE	0000 M10	0000 M20	0001 M26
0002 MAXFLOP	0001 MAXHD	00F7 MDIR	E663 MESSAGE	E16D MOVE
E268 MOVER	E26A MOVLOP	001A MREV	003C MSIZE	DFB4 NEWDMA
DF95 NEWSEC	CE6B NFOUND	E1C3 NOADJST	DFAC NOWRAP	00FB NSTEP
00FC NULL	DEFA NUSER0	4A00 OFFSETC	0002 OPDONE	C7D0 OPEN
F000 ORIGIN	E18B OUTOF	C78C PCHAR	DECB PMT1	DD72 PNCH1
E1AF PREP	E35F PROCESS	DEBD PROMP1	E59D PROMPT	0004 PSTEP
DDA0 PTBLE	DD6C PUNCH	DDE9 PWAIT	E186 RDWR	DD62 READER
E12F READ	DD65 READERA	DD68 READR1	DE2A READY	E133 REDWRT
000A RETRIES	0002 RETRY	E1B8 RETRYLP	E215 RETRYOP	001C REVNUM
CDDE RFILE	0001 RSECT	DF39 RSTR1	DDA8 RTBLE	E31B RTLOOP
0005 SCENBL	0001 SDELAY	0200 SECLEN	E172 SECPSEC	E134 SECSIZ
DFEE SECTRAN	DD4C SELDEV	DFE1 SETDMA	E029 SETDRV	E0DF SETDRV1
DFDB SETSEC	E2A6 SETTLE	DFE9 SETTRK	E004 SIDEA	E085 SIDEOK
E007 SIDEONE	E00D SIDETWO	E2C9 SLOOP	DE17 STAT	E295 STEPO
E095 SUBFP	E0B3 TDELAY	DE2D TINIT	0001 TKZERO	0008 TMOUT
0100 TPA	DFF6 TRANFP	E025 TRANHD	E597 TRUESEC	DF40 USER
DF21 USRRST	DF51 WARMBEG	DF51 WARMEND	DF94 WARMLOD	DFC8 WARMRD
DD03 WBOOTE	DF52 WBOOT	0000 WBOT	000F WENABL	0010 WFAULT
DF56 WFLG	000B WRESET	E128 WRITE	E19A WRITTYP	DFCB WRMREAD
E2DE WSDONE	0005 WSECT	E33C WTLOOP	E452 XLT124	E3C5 XLT128

```
CP/M MACRO ASSEM 2.Ø     #Ø5Ø     *** Cbios For CP/M Ver. 2.2 ***

E3EØ XLT256      E415 XLT512      E12Ø XLTS        E39D ZKEY        EØF8 ZRET
```