

PROC
DDT CPM62.COM
IABOOTH1.HEX
R0D00
ICBOSH3.HEX
R3E00
TC
SYSEEN

I/O PARTS 9/2/82!

00-07 SWBD
08-09 VIO-X
0A - SIOOL ZB
48-4F MULTI I/O
50-53 - H32A
C0-C1 PMMI

CB10SH3.PRN
8/25/82

INCLUDES I/O BYTE

OLD DECISION
THIS IS REVISION 01 M20

Removal of some configurations
with the I/O

also CB10SH14 (DO NOT USE)

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

```

001D = REVNUM EQU 29 ;CBIOS REVISION NUMBER 2.9
0016 = CPMREV EQU 22 ;CP/M REVISION NUMBER 2.2
0001 = CONGRP EQU 1 ;COSOLE PORT (1 = P1, 2 = P2, 3 = P3)
0003 = LSTGRP EQU 3 ;PRINTER PORT (1 = P1, 2 = P2, 3 = P3)
*0001 = MAXHD EQU 1 ;SET TO NUMBER OF HARD DISKS
*0001 = MAXFLOP EQU 1 ;SET TO NUMBER OF FLOPPIES
0003 = LOGDSK EQU 3 ;DEFAULT LOGICAL DISKS PER DRIVE
0014 = MREV EQU 20 ;HARD DISK TYPE
0015 = HDSPT EQU 21 ;SECTORS PER TRACK

```

```

*****
*
* THE FOLLOWING EQUATES RELATE THE MORROW DESIGNS 2D
* CONTROLLER. IF THE CONTROLLER IS NON STANDARD (0F800H)
* ONLY THE ORIGIN EQUATE NEED BE CHANGED.
*
*****

```

```

* F800 = ORIGIN EQU 0F800H
FC00 = DJRAM EQU ORIGIN+400H ;DISK JOCKEY 2D RAM ADDRESS
F800 = DJBOOT EQU ORIGIN ;DISK JOCKEY 2D INITIALIZATION
F803 = DJCIN EQU ORIGIN+3H ;DISK JOCKEY 2D CHARACTER INPUT ROUTINE
F806 = DJCOUT EQU ORIGIN+6H ;DISK JOCKEY 2D CHARACTER OUTPUT ROUTINE
F809 = DJHOME EQU ORIGIN+9H ;DISK JOCKEY 2D TRACK ZERO SEEK
F80C = DJTRK EQU ORIGIN+0CH ;DISK JOCKEY 2D TRACK SEEK ROUTINE
F80F = DJSEC EQU ORIGIN+0FH ;DISK JOCKEY 2D SET SECTOR ROUTINE
F812 = DJDMA EQU ORIGIN+012H ;DISK JOCKEY 2D SET DMA ADDRESS
F815 = DJREAD EQU ORIGIN+15H ;DISK JOCKEY 2D READ ROUTINE
F818 = DJWRITE EQU ORIGIN+18H ;DISK JOCKEY 2D WRITE ROUTINE
F81B = DJSEL EQU ORIGIN+1BH ;DISK JOCKEY 2D SELECT DRIVE ROUTINE
F821 = DJTSTAT EQU ORIGIN+21H ;DISK JOCKEY 2D TERMINAL STATUS ROUTINE
F827 = DJSTAT EQU ORIGIN+27H ;DISK JOCKEY 2D STATUS ROUTINE
F82A = DJERR EQU ORIGIN+2AH ;DISK JOCKEY 2D ERROR, FLASH LED
F82D = DJDEN EQU ORIGIN+2DH ;DISK JOCKEY 2D SET DENSITY ROUTINE
F830 = DJSIDE EQU ORIGIN+30H ;DISK JOCKEY 2D SET SIDE ROUTINE
0008 = DBLSID EQU 8 ;SIDE BIT FROM CONTROLLER
FBF8 = IO EQU ORIGIN+3F8H ;START OF I/O REGISTERS
FBF9 = DREG EQU IO+1
FBFC = CMDREG EQU IO+4
00D0 = CLRCMD EQU 0D0H

```

```

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

```

```

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          ;OPERATION 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

0048 =      MBASE   EQU     48H        ;BASE ADDRESS OF MULTI I/O OR DECISION I
004F =      GRPSEL  EQU     MBASE+7    ;GROUP SELECT PORT
0048 =      DLL     EQU     MBASE      ;DIVISOR (LSB)
0049 =      DLM     EQU     MBASE+1    ;DIVISOR (MSB)
0049 =      IER     EQU     MBASE+1    ;INTERUPT ENABLE REGISTER
004A =      CLK     EQU     MBASE+2    ;WB14 PRINTER SELECT PORT
004B =      LCR     EQU     MBASE+3    ;LINE CONTROL REGISTER
004D =      LSR     EQU     MBASE+5    ;LINE STATUS REGISTER
004E =      MSR     EQU     MBASE+6

0048 =      RBR     EQU     MBASE      ;READ DATA BUFFER
0048 =      THR     EQU     MBASE      ;TRANSMITTER DATA BUFFER
0080 =      DLAB   EQU     80H        ;DIVISOR LATCH ACCESS BIT
0020 =      THRE   EQU     20H        ;STATUS LINE THRE BIT
0010 =      CTS    EQU     10H        ;CLEAR TO SEND
0020 =      DSR    EQU     20H        ;DATA SET READY
0001 =      DR     EQU      1          ;LINE STATUS DR BIT
0001 =      WLS0   EQU      1          ;WORD LENGTH SELECT BIT 0
0002 =      WLS1   EQU      2          ;WORD LENGTH SELECT BIT 1 FOR 8 BIT WORD
0004 =      STB    EQU      4          ;STOP BIT COUNT - 2 STOP BITS

```

```

; DEFINE MULTI I/O PORTS ADDRESSES FOR GROUP ZERO

```

```

0000 =      GZERO   EQU      0
0048 =      DAISY0  EQU     MBASE      ;DAISY INPUT PORTS
0049 =      DAISY1  EQU     MBASE+1
0049 =      SENSESW EQU     MBASE+1    ;SENSE SWITCHES

0048 =      DAISI0  EQU     MBASE      ;   FOR DECISION I AND MULTI I/O.
0049 =      DAISI1  EQU     MBASE+1    ;THESE TWO ARE THE DECISION I PORTS

```

```

; DEFINE GROUP SELECT BITS

```

```

0001 =      S0     EQU     01H        ;GROUP NUMBER (0-3)

```

```

0002 = S1 EQU 02H
0003 = SMASK EQU 03H
0004 = BANK EQU 04H
0008 = ENINT EQU 08H
0010 = RESTOR EQU 10H ;PRINTER RESTORE ON MULTI I/O
0020 = DENABLE EQU 20H ;DRIVER ENABLE ON MULTI I/O
    
```

```

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

```

003E = MSIZE EQU 62 ;MEMORY SIZE OF TARGET CP/M
A800 = BIAS EQU (MSIZE-20)*1024 ;MEMORY OFFSET FROM 20K SYSTEM
CD00 = CCP EQU 2500H+BIAS ;CONSOLE COMMAND PROCESSOR
D500 = BDOS EQU CCP+800H ;BDOS ADDRESS
E300 = BIOS EQU CCP+1600H ;CBIOS ADDRESS
3E00 = OFFSETC EQU 2100H-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.
*
*****
    
```

```

0003 = AETX EQU 3 ;ETX CHARACTER
0006 = AACK EQU 6 ;ACK CHARACTER
000A = ACR EQU 'J'-64 ;CARRIAGE RETURN
000D = ALF EQU 'M'-64 ;LINE FEED
000A = RETRIES EQU 10 ;MAX RETRIES ON DISK I/O BEFORE ERROR
001A = CLEAR EQU 'Z'-64 ;CLEAR SCREEN ON AN ADM 3
    
```

```

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

```

E300 ORG BIOS ;CBIOS STARTING ADDRESS

E300 C3BFEB JMP CBOOT ;COLD BOOT ENTRY POINT
E303 C3D8E4 WBOOTE JMP WBOOT ;WARM BOOT ENTRY POINT
E306 C33BE3 JMP CONST ;CONSOLE STATUS ROUTINE
    
```

```

E309 C347E3      JMP      CONIN      ;CONSOLE INPUT
E30C C35CE3      JMP      CONOUT     ;CONSOLE OUTPUT
E30F C37CE3      JMP      LIST       ;LIST DEVICE OUTPUT
E312 C371E3      JMP      PUNCH      ;PUNCH DEVICE OUTPUT
E315 C367E3      JMP      READER     ;READER DEVICE INPUT
E318 C323E5      JMP      HOME       ;HOME DRIVE
E31B C365E5      JMP      SETDRV     ;SELECT DISK
E31E C325E5      JMP      SETTRK     ;SET TRACK
E321 C317E5      JMP      SETSEC     ;SET SECTOR
E324 C31DE5      JMP      SETDMA     ;SET DMA ADDRESS
E327 C3A4E6      JMP      READ       ;READ THE DISK
E32A C39DE6      JMP      WRITE      ;WRITE THE DISK
E32D C387E3      JMP      LISTST     ;LIST DEVICE STATUS
E330 C32AE5      JMP      SECTAN     ;SECTOR TRANSLATION
E333 C31BF8      DJDRV   JMP      DJSEL      ;HOOKUP FOR SINGLE.COM PROGRAM
    
```

```

E336 0600      DEFCON  DW      6      ;CONSOLE BAUD RATE
    
```

```

E338 0C00      DEFLST  DW     12     ;PRINTER BAUD RATE
    
```

```

E33A 00        GROUP   DB      0      ;GROUP BYTE
    
```

```

*****
*
* 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.
*
*****
    
```

```

F803 =          CITYY   EQU      DJCIN      ;INPUT FROM THE DISK JOCKEY 2D
F806 =          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.
*
*****
    
```

```

E33B 21B5E3      CONST   LXI      H,CSTBLE      ;BEGINNING OF JUMP TABLE
E33E C34DE3      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.
*
*****
    
```

```

*****
E341 21BDE3 CSREADR LXI H,CSRTBLE ;BEGINNING OF READER STATUS TABLE
E344 C36AE3 JMP READERA

```

```

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

```

```

E347 CD17E7 CONIN CALL FLUSH ;FLUSH THE DISK BUFFER
E34A 218DE3 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.
*

```

```

E34D 3A0300 CONIN1 LDA IOBYTE
E350 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.
*

```

```

E351 E606 SELDEV ANI 6H ;STRIP OFF UNWANTED BITS
E353 1600 MVI D,0 ;FORM OFFSET
E355 5F MOV E,A
E356 19 DAD D ;ADD OFFSET
E357 7E MOV A,M ;PICK UP HIGH BYTE
E358 23 INX H
E359 66 MOV H,M ;PICK UP LOW BYTE
E35A 6F MOV L,A ;FORM ADDRESS
E35B E9 PCHL ;GO THERE !

```

```

*****
*
* CONOUT: TAKE THE PROPER BRANCH ADDRESS BASED ON THE TWO LEAST *
* SIGNIFICANT BITS OF IOBYTE. *
*
*****

```

```

E35C C5 CONOUT PUSH B ;SAVE THE CHARACTER
E35D CD17E7 CALL FLUSH ;FLUSH THE DISK BUFFER
E360 C1 POP B ;RESTORE THE CHARACTER
E361 2195E3 LXI H,COTBLE ;BEGINNING OF THE CHARACTER OUT TABLE
E364 C34DE3 JMP CONIN1 ;DO THE DECODE

```

```

*****
*
* READER: SELECT THE CORRECT READER DEVICE FOR INPUT. THE *
* READER IS SELECTED FROM BITS 2 AND 3 OF IOBYTE. *

```

*

E367 21ADE3 READER LXI H,RTBLE ;BEGINNING OF READER INPUT TABLE

*
* ENTRY AT READERA WILL DECODE BITS 2 & 3 OF IOBYTE, USED
* BY CSREADER.
*

E36A 3A0300 READERA LDA IOBYTE

*
* ENTRY AT READER1 WILL SHIFT THE BITS INTO POSITION, USED
* BY LIST AND PUNCH.
*

E36D 1F READR1 RAR
E36E C351E3 JMP SELDEV

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

E371 21A5E3 PUNCH LXI H,PTBLE ;BEGINNING OF PUNCH TABLE
E374 3A0300 LDA IOBYTE

*
* ENTRY AT PNCH1 ROTATES BITS A LITTLE MORE IN PREP FOR
* SELDEV, USED BY LIST.
*

E377 1F PNCH1 RAR
E378 1F RAR
E379 C36DE3 JMP READR1

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

E37C 219DE3 LIST LXI H,LTBLE ;BEGINNING OF THE LIST DEVICE ROUTINES
E37F 3A0300 LIST1 LDA IOBYTE
E382 1F RAR
E383 1F RAR
E384 C377E3 JMP PNCH1

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

E387 21C5E3 LISTST LXI H,LSTBLE ;BEGINNING OF THE LIST DEVICE STATUS
E38A C37FE3 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
*

CBIO5H4

E38D 4EE4 CITBLE DW MIOIN ;INPUT FROM I/O MASTER
; ;
E38F 0CE4 DW CICRT ;INPUT FROM CRT (CURRENTLY SWITCHBOARD
; SERIAL PORT 1)
E391 F7E3 DW CIUCL ;INPUT FROM SWBD PARALLEL PORT 4
; ;
E393 03F8 DW CITY ;INPUT FROM TTY (CURRENTLY INPUT FROM
; DISK JOCKEY 2D)

← *CICRT*

*
* CONSOLE OUTPUT TABLE
*

E395 61E4 COTBLE DW MIOOUT ;OUTPUT TO I/O MASTER
; ;
E397 31E4 DW COCRT ;OUTPUT TO CRT
; ;
E399 31E4 DW COCRT ;OUTPUT TO CRT
; ;
E39B 06F8 DW COTTY ;OUTPUT TO TTY (CURRENTLY OUTPUT TO
; DISK JOCKEY 2D)

← *MIOOUT*
← *COCRT*
← *COCRT*
← *MIOCRT*

*
* LIST DEVICE TABLE
*

E39D 06F8 LTBLE DW COTTY ;OUTPUT TO TTY (CURRENTLY ASSIGNED
; BY INTIOBY,OUTPUT TO 2D)
E39F 3CE4 DW COPTR ;OUTPUT TO PRINTER
; ;
E3A1 CDE3 DW COLPT ;OUTPUT TO LINE PRINTER (CURRENTLY
; SWITCHBOARD SERIAL PORT 1)
E3A3 D8E3 DW COUL1 ;OUTPUT TO USER LINE PRINTER 1 (CURRENTLY
; SWITCHBOARD SERIAL PORT 1)

— *OKI*

table

*
* PUNCH DEVICE TABLE
*

E3A5 06F8 PTBLE DW COTTY ;OUTPUT TO THE TTY (CURRENTLY ASSIGNED


```

;          BY INTIOBY,OUTPUT TO 2D)
E3A7 3CE4   DW      COPTR   ;OUTPUT TO PRINTER
;
E3A9 CDE3   DW      COUP1   ;OUTPUT TO USER PUNCH 1 (CURRENTLY
;          SWITCHBOARD SERIAL PORT 1)
E3AB CDE3   DW      COUP2   ;OUTPUT TO USER PUNCH 2 (CURRNTLLY
;          SWITCHBOARD SERIAL PORT 1)

```

```

*
* READER DEVICE INPUT TABLE
*

```

```

E3AD 03F8   RTBLE  DW      CITY    ;INPUT FROM TTY (CURRENTLY ASSIGNED
;          BY INTIOBY, INPUT FROM 2D)
E3AF 0CE4   DW      CIPTR   ;INPUT FROM PAPER TAPE READER (CURRENTLY
;          SWITCHBOARD SERIAL PORT 1)
E3B1 0CE4   DW      CIUR1   ;INPUT FROM USER READER 1 (CURRENTLY
;          SWITCHBOARD SERIAL PORT 1)
E3B3 0CE4   DW      CIUR2   ;INPUT FROM USER READER 2 (CURRENTLY
;          SWITCHBOARD SERIAL PORT 1)

```

```

*
* CONSOLE STATUS TABLE
*

```

```

E3B5 73E4   CSTBLE DW      MIOSTAT 3 ;STATUS FROM I/O MASTER
;
E3B7 20E4   DW      CSCRT 1 ;STATUS FROM CRT (CURRENTLY SWITCHBOARD
;          SERIAL PORT 1)
E3B9 03E4   DW      CSUC1 2 ;STATUS FROM SWBD PARALLEL PORT 4, AS
;          READ FROM ATTN BIT 0
E3BB 18E4   DW      CSTTY 4 ;STATUS OF TTY (CURRENTLY STSTUS FROM
;          DISK JOCKEY 2D)

```

CBIOSHY
← CSRT

```

*
* STATUS FROM READER DEVICE
*

```

```

E3BD 18E4   CSRTBLE DW      CSTTY   ;STATUS FROM TTY (CURRENTLY ASSIGNED
;          BY INTIOBY, STATUS OF 2D)
E3BF 20E4   DW      CSPTR   ;STATUS FROM PAPER TAPE READER (CURRENTLY
;          SWITCHBOARD SERIAL PORT 1)
E3C1 20E4   DW      CSUR1   ;STATUS FROM USER READER 1 (CURRENTLY
;          SWITCHBOARD SERIAL PORT 1)
E3C3 20E4   DW      CSUR2   ;STATUS OF USER READER 2 (CURRENTLY
;          SWITCHBOARD SERIAL PORT 1)

```

```

*
* STATUS FROM LIST DEVICE
*

```

```

E3C5 2EE4   LSTBLE DW      READY   ;CONSOLE ALWAYS READY
E3C7 2EE4   DW      READY   ;GET LIST STATUS
E3C9 29E4   DW      LSLPT
E3CB 29E4   DW      LSLPT

```

*
 * THE FOLLOWING EQUATES SET OUTPUT DEVICE TO OUTPUT TO THE
 * SWITCHBOARD SERIAL PORT 1.
 *

```

E3CD = COPTP EQU $ ;OUTPUT FROM PAPER TAPE PUNCH
E3CD = COUP1 EQU $ ;OUTPUT FROM USER PUNCH 1
E3CD = COUP2 EQU $ ;OUTPUT FROM USER PUNCH 2
E3CD DB02 COLPT IN 2 ;OUTPUT FROM LINE PRINTER,GET STATUS
E3CF E680 ANI 80H ;WAIT UNTIL OK TO SEND
E3D1 CACDE3 JZ COLPT
E3D4 79 MOV A,C ;OUTPUT THE CHARACTER
E3D5 D301 OUT 1
E3D7 C9 RET
    
```

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

```

E3D8 CDCDE3 COUL1 CALL COLPT ;OUTPUT THE CHARACTER
E3DB 3AF6E3 LDA COUNT
E3DE 3D DCR A
E3DF 32F6E3 STA COUNT
E3E2 C0 RNZ
E3E3 3E4E MVI A,78
E3E5 32F6E3 STA COUNT
E3E8 0E03 MVI C,AETX
E3EA CDCDE3 CALL COLPT
E3ED CD0CE4 PWAIT CALL CIPTR
E3F0 FE06 CPI AACK
E3F2 C2EDE3 JNZ PWAIT
E3F5 C9 RET

E3F6 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.
 *

```

E3F7 DB03 CIUC1 IN 3 ;GET ATTENTION BYTE
E3F9 E601 ANI 1 ;GET BIT 0 ONLY
E3FB CAF7E3 JZ CIUC1 ;WAIT FOR CHARACTER
E3FE DB04 IN 4 ;GET CHARACTER
E400 E67F ANI 7FH ;STRIP OFF THE PARITY
E402 C9 RET

E403 DB03 CSUC1 IN 3 ;GET ATTENTION BYTE
E405 E601 ANI 1 ;GET BIT 0 ONLY
E407 EE01 XRI 1 ;CHANGE POLARITY
    
```

E409 C31BE4 JMP STAT ;RETURN PROPER INDICATION

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

E40C = CICRT EQU \$;INPUT FROM CRT
 E40C = CIUR1 EQU \$;INPUT FROM USER READER 1
 E40C = CIUR2 EQU \$;INPUT FROM USER READER 2
 E40C DB02 CIPTR IN 2 ;INPUT FROM PAPER TAPE READER, GET STATUS
 E40E E640 ANI 40H ;WAIT FOR CHARACTER
 E410 CA0CE4 JZ CIPTR
 E413 DB01 IN 1
 E415 E67F ANI 7FH ;STRIP OFF THE PARITY
 E417 C9 RET

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

E418 CD21F8 CSTTY CALL DJTSTAT ;STATUS FROM DISK JOCKEY 2D
 E41B 3E00 STAT MVI A,0 ;PREP FOR ZERO RETURN
 E41D C0 RNZ ;NOTHING FOUND
 E41E 3D DCR A ;RETURN WITH 0FFH
 E41F C9 RET

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

E420 = CSUR1 EQU \$;STATUS OF USER READER 1
 E420 = CSUR2 EQU \$;STATUS OF USER READER 2
 E420 = CSPTR EQU \$;STATUS OF PAPER TAPE READER
 E420 DB02 CSCRT IN 2 ;STATUS FROM CRT, GET STATUS
 E422 E640 ANI 40H ;STRIP OF DATA READY BIT
 E424 EE40 XRI 40H ;MAKE CORRECT POLARITY
 E426 C31BE4 JMP STAT ;RETURN PROPER INDICATION

 *
 * LIST DEVICE STATUS ROUTINES. *
 *

E429 DB02 LSLPT IN 2 ;ALL OTHER DEVICES WAIT
 E42B E680 ANI 80H
 E42D C8 RZ
 E42E 3EFF READY MVI A,0FFH

E430 C9 RET

 *
 * VIO-X VIDEO DRIVER
 *

 MIOCKT CALL MIOOUT

 CBIOS #4

E431 DB09 COCRT IN 9 ;READ STATUS PORT
 E433 E601 ANI 1 ;MASK TXRDY BIT
 E435 CA31E4 JZ COCRT ;WAIT FOR READY
 E438 79 MOV A,C ;GET CHAR
 E439 D308 OUT 8 ;OUTPUT IT
 E43B C9 RET ;ALL DONE

 *
 * ROUTINE FOR OKIDATA PRINTER
 * PRINTER IS ON PORT 0 WITH PRINTER READY ON PORT 5 BIT 1
 *

E43C DB02 COPTR IN 2 ;INPUT FROM PORT 2
 E43E E608 ANI 8 ;WAIT UNTIL OK TO SEND
 E440 CA3CE4 JZ COPTR
 E443 DB05 COPTR1 IN 5 ;BUFFER FULL?
 E445 E601 ANI 1
 E447 CA43E4 JZ COPTR1 ;WAIT UNTIL PRINTER READY
 E44A 79 MOV A,C ;OUTPUT THE CHARACTER
 E44B D300 OUT 0
 E44D C9 RET

 *
 * TERMINAL ROUTINES FOR MULTI- I/O BOARD FOR USE AS CONSOLE
 *

E44E 3A3AE3 MIOIN LDA GROUP ;GET GROUP BYTE
 E451 F601 ORI CONGRP ;SELECT CONSOLE
 E453 D34F OUT GRPSEL
 E455 DB4D CONINA IN LSR ;READ STATUS REGISTER
 E457 E601 ANI DR ;WAIT TILL CHARACTER READY
 E459 CA55E4 JZ CONINA
 E45C DB48 IN RBR ;READ CHARACTER
 E45E E67F ANI 7FH ;STRIP PARITY
 E460 C9 RET

E461 3A3AE3 MIOOUT LDA GROUP ;GET GROUP BYTE
 E464 F601 ORI CONGRP ;SELECT CONSOLE
 E466 D34F OUT GRPSEL
 E468 DB4D CONOUT1 IN LSR ;READ STATUS
 E46A E620 ANI THRE ;WAIT TILL TRANSMITTER BUFFER EMPTY
 E46C CA68E4 JZ CONOUT1
 E46F 79 MOV A,C ;CHARACTER IS IN (C)

```

E470 D348      OUT      THR          ;OUTPUT TO TRANSMITTER BUFFER
E472 C9        RET
E473 3A3AE3    MIOSTAT LDA      GROUP      ;GET GROUP BYTE
E476 F601      ORI      CONGRP     ;SELECT CONSOLE
E478 D34F      OUT      GRPSEL
E47A DB4D      IN       LSR        ;READ STATUS REGISTER
E47C E601      ANI      DR
E47E C8        RZ
E47F 3EFF      MVI      A,0FFH     ;CHARACTER READY
E481 C9        RET
    
```

```

*****
*
* GOCPM IS THE ENTRY POINT FROM COLD BOOTS, AND WARM BOOTS. IT
* INITIALIZES SOME OF THE LOCATIONS IN PAGE 0, AND SETS UP THE
* INITIAL DMA ADDRESS (80H).
*
*****
    
```



```

E482 218000    GOCPM   LXI      H,BUFF      ;SET UP INITIAL DMA ADDRESS
E485 CD1DE5    CALL     SETDMA
E488 3EC3      MVI      A,(JMP)    ;INITIALIZE JUMP TO WARM BOOT
E48A 320000    STA      WBOT
E48D 320500    STA      ENTRY     ;INITIALIZE JUMP TO BDOS
E490 2103E3    LXI      H,WBOOTE   ;ADDRESS IN WARM BOOT JUMP
E493 220100    SHLD    WBOT+1
E496 2106D5    LXI      H,BDOS+6  ;ADDRESS IN BDOS JUMP
E499 220600    SHLD    ENTRY+1
E49C AF        XRA      A          ;A ← 0
E49D 32F4EE    STA      BUFSEC    ;DISK JOCKEY BUFFER EMPTY
E4A0 3218E7    STA      BUFWRTN   ;SET BUFFER NOT DIRTY FLAG
E4A3 3A0400    LDA      CDISK     ;JUMP TO CP/M WITH CURRENTLY SELECTED DISK IN C
E4A6 4F        MOV      C,A
E4A7 3AD4E4    LDA      CWFLG
E4AA B7        ORA      A
E4AB 11D6E4    LXI      D,COLDBEG ;BEGINNING OF INITIAL COMMAND
E4AE 3E01      MVI      A,COLDEND-COLDBEG+1 ;LENGTH OF COMMAND
E4B0 CAB8E4    JZ       CLDCMND
E4B3 11D7E4    LXI      D,WARBEG
E4B6 3E01      MVI      A,WARMEND-WARBEG+1
E4B8 2108CD    CLDCMND LXI      H,CCP+8 ;COMMAND BUFFER
E4BB 3207CD    STA      CCP+7
E4BE 47        MOV      B,A
E4BF CDE0E7    CALL    MOVLOP
E4C2 3AD4E4    LDA      CWFLG
E4C5 B7        ORA      A
E4C6 3AD5E4    LDA      AUTOFLG
E4C9 CACDE4    JZ       CLDBOT
E4CC 1F        RAR
E4CD 1F        CLDBOT RAR
E4CE DA00CD    JC       CCP
E4D1 C303CD    JMP      CCP+3     ;ENTER CP/M

E4D4 00        CWFLG  DB      0          ;COLD/WARM BOOT FLAG
    
```

```
*****
*
* THE FOLLOWING BYTE DETERMINES IF AN INITIAL COMMAND IS TO BE
* GIVEN TO CP/M ON WARM OR COLD BOOTS. THE VALUE OF THE BYTE IS
* USED TO GIVE THE COMMAND TO CP/M:
*
* 0 = 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.
*****
```

E4D5 00 AUTOFLG DB 0 ;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 0
*
* WILL EXECUTE MICROSOFT BASIC, AND MBASIC WILL EXECUTE THE
* "MYPROG" BASIC PROGRAM.
*****
```

E4D6 00 COLDBEG DB '' ;COLD BOOT COMMAND GOES HERE
 COLDEND DB 0
 E4D7 00 WARBEG DB '' ;WARM BOOT COMMAND GOES HERE
 WARMEND DB 0

```
*****
*
* 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.
*****
```

E4D8 310001	WBOOT	LXI	SP,TPA	;SET UP STACK POINTER
E4DB 3E01		MVI	A,1	
E4DD 32D4E4		STA	CWFLG	;SET COLD/WARM BOOT FLAG
E4E0 AF		XRA	A	
E4E1 4F		MOV	C,A	
E4E2 2100CB		LXI	H,CCP-200H	;INITIAL DMA ADDRESS
E4E5 E5		PUSH	H	
E4E6 3223E9		STA	HEAD	
E4E9 3E01		MVI	A,1	
E4EB F5		PUSH	PSW	;SAVE FIRST SECTOR - 1
E4EC CDF5E7		CALL	HDDRV	;SELECT DRIVE A
E4EF 0E00		MVI	C,0	
E4F1 CD14E8		CALL	HDTRK	;HOME THE DRIVE
E4F4 F1	WARMLOD	POP	PSW	;RESTORE SECTOR

```

E4F5 E1      POP      H          ;RESTORE DMA ADDRESS
E4F6 3C      INR      A
E4F7 3207E9  STA      HDSECTR
E4FA FE0C    CPI      12         ;PAST BDOS ?
E4FC CA82E4  JZ       GOCPM          ;YES, ALL DONE
E4FF 24      INR      H          ;UPDATE DMA ADDRESS
E500 24      INR      H
E501 227EE8  SHLD     HDADD
E504 E5      PUSH     H
E505 F5      PUSH     PSW
E506 01000A  WARMRD  LXI      B,RETRIES*100H+0 ;RETRY COUNTER
E509 C5      WRMREAD PUSH     B          ;SAVE THE RETRY COUNT
E50A CD64E8  CALL     HDREAD        ;READ THE SECTOR
E50D C1      POP      B
E50E D2F4E4  JNC     WARMLOD        ;TEST FOR ERROR
E511 05      DCR      B          ;UPDATE THE ERROR COUNT
E512 C209E5  JNZ     WRMREAD        ;KEEP TRYING IF NOT TO MANY ERRORS
E515 76      HLT
E516 00      DB       0          ;TRY NOT TO SCREW UP DECISION CPU'S
    
```

```

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

```

E517 60      SETSEC  MOV      H,B
E518 69      MOV      L,C
E519 22ECEE  SHLD     CPMSEC
E51C C9      DONOP  RET          ;NULL SINGLE.COM HOOKUP FOR NO FLOPPIES
    
```

```

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

```

E51D 60      SETDMA  MOV      H,B          ;HL <- BC
E51E 69      MOV      L,C
E51F 22F8E6  SHLD     CPMDMA        ;CP/M DMA ADDRESS
E522 C9      RET
    
```

```

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

```

E523 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.
*
*****
    
```

```

E525 79      SETTRK  MOV     A,C           ;A <- TRACK #
E526 32EFEE  STA     CPMTRK        ;CP/M TRACK #
E529 C9      RET
    
```

```

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

```

E52A 3AE5EE  SECTRAN LDA     CPMDRV          ;GET THE DRIVE NUMBER
E52D FE03    CPI     MAXHD*LOGDSK ;OVER THE # OF HARD DISKS ?
E52F DA61E5  JC      TRANHD
    
```

```

E532 03      TRANFP  INX     B
E533 D5      PUSH   D           ;SAVE TABLE ADDRESS
E534 C5      PUSH   B           ;SAVE SECTOR #
E535 CD7CE6  CALL   GETDPB          ;GET DPB ADDRESS INTO HL
E538 7E      MOV    A,M        ;GET # OF CP/M SECTORS/TRACK
E539 B7      ORA    A           ;CLEAR CARY
E53A 1F      RAR    A           ;DIVIDE BY TWO
E53B 91      SUB    C
E53C F5      PUSH   PSW        ;SAVE ADJUSTED SECTOR
E53D FA49E5  JM     SIDETWO
E540 F1      SIDEA  POP    PSW   ;DISCARD ADJUSTED SECTOR
E541 C1      POP    B           ;RESTORE SECTOR REQUESTED
E542 D1      POP    D           ;RESTOR ADDRESS OF XLT TABLE
E543 EB      SIDEONE XCHG   ;HL <- &(TRANSLATION TABLE)
E544 09      DAD    B           ;BC = OFFSET INTO TABLE
E545 6E      MOV    L,M        ;HL <- PHYSICAL SECTOR
E546 2600    MVI    H,0
E548 C9      RET
    
```

```

E549 010F00  SIDETWO LXI    B,15         ;OFFSET TO SIDE BIT
E54C 09      DAD    B
E54D 7E      MOV    A,M
E54E E608    ANI    8           ;TEST FOR DOUBLE SIDED
E550 CA40E5  JZ     SIDEA          ;MEDIA IS ONLY SINGLE SIDED
E553 F1      POP    PSW        ;RETRIEVE ADJUSTED SECTOR
E554 C1      POP    B
E555 2F      CMA          ;MAKE SECTOR REQUEST POSITIVE
E556 3C      INR    A
E557 4F      MOV    C,A        ;MAKE NEW SECTOR THE REQUESTED SECTOR
E558 D1      POP    D
E559 CD43E5  CALL   SIDEONE
E55C 3E80    MVI    A,80H         ;SIDE TWO BIT
E55E B4      ORA    H           ;
E55F 67      MOV    H,A        ;
E560 C9      RET
    
```

```

E561 60      TRANHD  MOV    H,B
E562 69      MOV    L,C
E563 23      INX    H
    
```


E564 C9 RET

```
*****
*
* 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.
*****
```

```
E565 79      SETDRV  MOV    A,C          ;SAVE THE DRIVE #
E566 32EEEE  STA    CPMDRV
E569 FE04    CPI    MAXFLOP+(MAXHD*LOGDSK) ;CHECK FOR A VALID DRIVE #
E56B D26DE6  JNC    ZRET          ;ILLEGAL DRIVE #
E56E 7B      MOV    A,E          ;TEST IF DRIVE EVER LOGGED IN BEFORE
E56F E601    ANI    1
E571 C254E6  JNZ    SETDRV1       ;BIT 0 OF E = 0 -> NEVER SELECTED BEFORE
E574 3AE5EE  LDA    CPMDRV       ;GET THE DRIVE NUMBER
E577 FE03    CPI    MAXHD*LOGDSK ;OVER THE # OF HARD DISKS ?
E579 DA27E6  JC     DRVHD
E57C D603    SUI    MAXHD*LOGDSK
E57E 4F      MOV    C,A          ;SAVE DRIVE #
E57F 3E00    MVI    A,0          ;HAVE THE FLOPPIES BEEN ACCESSED YET ?
E580 =      FLOPFLG EQU    $-1
E581 A7      ANA    A
E582 C2D2E5  JNZ    FLOPOK
E585 0611    MVI    B,17          ;FLOPPIES HAVN'T BEEN ACCESSED
E587 2100F8  LXI    H,DJBOOT      ;CHECK IF 2D CONTROLLER IS INSTALLED
E58A 3EC3    MVI    A,(JMP)
E58C BE      CLOPP  CMP    M
E58D C26DE6  JNZ    ZRET
E590 23      INX    H
E591 23      INX    H
E592 23      INX    H
E593 05      DCR    B
E594 C28CE5  JNZ    CLOPP
E597 11AFE5  LXI    D,DJINIT      ;INITIALIZATION SEQUENCE
E59A 21E2FF  LXI    H,ORIGIN+7E2H ;LOAD ADDRESS
E59D 061E    MVI    B,30          ;BYTE COUNT
E59F CDE0E7  CALL   MOVLOP
E5A2 3EFF    MVI    A,0FFH        ;START 1791
E5A4 32F9FB  STA    DREG
E5A7 3ED0    MVI    A,CLRCMD     ;1791 RESET
E5A9 32FCFB  STA    CMDREG
E5AC C3CDE5  JMP    DJNEXT

E5AF 0000001800DJINIT DB    0, 0, 0, 18H, 0, 0, 8, 0, 7EH, 0, 8, 0, 9, 0FFH, 9, 0FFH
E5BF 09FF09FF09      DB    9, 0FFH, 9, 0FFH, 9, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0

E5CD 3E01    DJNEXT  MVI    A,1          ;SAVE 2D INITIALIZED FLAG
```

```

E5CF 3280E5          STA      FLOPFLG
E5D2 210100          FLOPOK LXI      H,1          ;SELECT SECTOR 1 OF TRACK 1
E5D5 22F0EE          SHLD    TRUESEC
E5D8 3E01            MVI     A,1
E5DA 32EFEE          STA     CPMTRK
E5DD CDAAE7          CALL    FILL          ;FLUSH BUFFER AND REFILL
E5E0 DA6DE6          JC      ;TEST FOR ERROR RETURN
E5E3 CD27F8          CALL    DJSTAT        ;GET STATUS ON CURRENT DRIVE
E5E6 E60C            ANI     0CH           ;STRIP OFF UNWANTED BITS
E5E8 F5              PUSH    PSW           ;USED TO SELECT A DPB
E5E9 1F              RAR
E5EA 2195E6          LXI     H,XLTS        ;TABLE OF XLT ADDRESSES
E5ED 5F              MOV     E,A
E5EE 1600            MVI     D,0
E5F0 19              DAD     D
E5F1 E5              PUSH    H              ;SAVE POINTER TO PROPER XLT
E5F2 CD7CE6          CALL    GETDPB        ;GET DPH POINTER INTO DE
E5F5 EB              XCHG
E5F6 D1              POP     D
E5F7 0602            MVI     B,2           ;NUMBER OF BYTES TO MOVE
E5F9 CDE0E7          CALL    MOVLOP        ;MOVE THE ADDRESS OF XLT
E5FC 110800          LXI     D,8           ;OFFSET TO DPB POINTER
E5FF 19              DAD     D              ;HL <- &DPH.DPB
E600 E5              PUSH    H
E601 2A07F8          LHLD   ORIGIN+7      ;GET ADDRESS OF DJ TERMINAL OUT ROUTINE
E604 23              INX     H              ;BUMP TO LOOK AT ADDRESS OF
                          ;      UART STATUS LOCATION

E605 7E              MOV     A,M
E606 EE03            XRI     3              ;ADJUST FOR PROPER REV DJ
E608 6F              MOV     L,A
E609 26FB            MVI     H,(ORIGIN+300H)/100H
E60B 7E              MOV     A,M
E60C E608            ANI     DBLSID        ;CHECK DOUBLE SIDED BIT
E60E 11FCE9          LXI     D,DPB128S    ;BASE FOR SINGLE SIDED DPB'S
E611 C217E6          JNZ     SIDEOK
E614 113CEA          LXI     D,DPB128D    ;BASE OF DOUBLE SIDED DPB'S
E617 EB              SIDEOK XCHG           ;HL <- DBP BASE, DE <- &DPH.DPB
E618 D1              POP     D              ;RESTORE DE (POINTER INTO DPH)
E619 F1              POP     PSW           ;OFFSET TO CORRECT DPB
E61A 17              RAL
E61B 17              RAL
E61C 4F              MOV     C,A
E61D 0600            MVI     B,0
E61F 09              DAD     B
E620 EB              XCHG           ;PUT DPB ADDRESS IN DPH
E621 73              MOV     M,E
E622 23              INX     H
E623 72              MOV     M,D
E624 C354E6          JMP     SETDRV1      ;SKIP OVER THE HARD DISK SELECT

E627 CD73E6          DRVHD CALL    DIVLOG        ;DIVIDE BY LOGICAL DISKS PER DRIVE
E62A 79              MOV     A,C
E62B 3229E9          STA     HDDISK
E62E CD17E9          CALL    DRVPTR
E631 7E              MOV     A,M
E632 3C              INR     A

```

```

E633 C254E6      JNZ      SETDRV1
E636 F6FC        ORI      NULL          ;SELECT DRIVE
E638 D352        OUT      HDFUNC
E63A 3E05        MVI      A,SCENBL      ;ENABLE THE CONTROLLER
E63C D350        OUT      HDCNTL
E63E 0EEF        MVI      C,239          ;WAIT APPROX 2 MINUTES FOR DISK TO READY
E640 210000      LXI      H,0
E643 2B          TDELAY DCX      H
E644 7C          MOV      A,H
E645 B5          ORA      L
E646 CC71E6      CZ       DCRC
E649 C8          RZ
E64A DB50        IN       HDSTAT      ;TEST IF READY YET
E64C E620        ANI      DRVRDY
E64E C243E6      JNZ      TDELAY
E651 CD06E8      CALL     HDHOME

```

```

E654 CD7CE6      SETDRV1 CALL    GETDPB      ;GET ADDRESS OF DPB IN HL
E657 010F00      LXI      B,15          ;OFFSET TO SECTOR SIZE
E65A 09          DAD      B
E65B 7E          MOV      A,M          ;GET SECTOR SIZE
E65C E607        ANI      7H
E65E 32A9E6      STA      SECSIZ
E661 7E          MOV      A,M
E662 1F          RAR
E663 1F          RAR
E664 1F          RAR
E665 1F          RAR
E666 E60F        ANI      0FH
E668 32E7E6      STA      SECPSEC
E66B EB          XCHG      ;HL <- DPH
E66C C9          RET

```

```

E66D 210000      ZRET     LXI      H,0          ;SELDRV ERROR EXIT
E670 C9          RET

```

```

E671 0D          DCRC     DCR      C          ;CONDITIONAL DECREMENT C ROUTINE
E672 C9          RET

```

```

E673 0E00      DIVLOG MVI      C,0
E675 D603      DIVLOGX SUI     LOGDSK
E677 D8        RC
E678 0C        INR      C
E679 C375E6    JMP      DIVLOGX

```

```

*****
*
* GETDPB RETURNS HL POINTING TO THE DPB OF THE CURRENTLY
* SELECTED DRIVE, DE POINTING TO DPH.
*
*****

```

```

E67C 3AE000      GETDPB LDA      CPMDRV
E67F 6F          MOV      L,A          ;FORM OFFSET
E680 2600      MVI      H,0
E682 29          DAD      H

```

```

E683 29      DAD      H
E684 29      DAD      H
E685 29      DAD      H
E686 11ACEA  LXI      D,DPBASE      ;BASE OF DPH'S
E689 19      DAD      D
E68A E5      PUSH     H      ;SAVE ADDRESS OF DPH
E68B 110A00  LXI      D,10      ;OFFSET TO DPB
E68E 19      DAD      D
E68F 7E      MOV      A,M      ;GET LOW BYTE OF DPB ADDRESS
E690 23      INX      H
E691 66      MOV      H,M      ;GET LOW BYTE OF DPB
E692 6F      MOV      L,A
E693 D1      POP      D
E694 C9      RET
    
```

```

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

```

E695 2EE9  XLTS    DW      XLT128      ;XLT FOR 128 BYTE SECTORS
E697 49E9          DW      XLT256      ;XLT FOR 256 BYTE SECTORS
E699 7EE9          DW      XLT512      ;XLT FOR 512 BYTE SECTORS
E69B BBE9          DW      XLT124      ;XLT FOR 1024 BYTE SECTORS
    
```

```

*****
*
* 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.
*
*****
    
```

```

E69D 79      WRITE   MOV      A,C      ;SAVE WRITE COMMAND TYPE
E69E 320FE7  STA      WRITYP
E6A1 3E01    MVI      A,1      ;SET WRITE COMMAND
E6A3 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.
*
*****
    
```

```

*****
E6A4 AF      READ  XRA   A           ;SET THE COMMAND TYPE TO READ
E6A5 32FBE6  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.
*
*****

E6A8 0600    REDWRT MVI     B,0           ;THE 0 IS MODIFIED TO CONTAIN THE LOG2
E6A9 =        SECSIZ EQU    $-1          ;      OF THE PHYSICAL SECTOR SIZE/128
                                           ;      ON THE CURRENTLY SELECTED DISK.

E6AA 2AECEE          LHLD   CPMSEC        ;GET THE DESIRED CP/M SECTOR #
E6AD 7C            MOV    A,H
E6AE E680          ANI    80H           ;SAVE ONLY THE SIDE BIT
E6B0 4F            MOV    C,A          ;REMEMBER THE SIDE
E6B1 7C            MOV    A,H
E6B2 E67F          ANI    7FH           ;FORGET THE SIDE BIT
E6B4 67            MOV    H,A
E6B5 2B            DCX    H            ;TEMPORARY ADJUSTMENT
E6B6 05            DIVLOOP DCR    B      ;UPDATE REPEAT COUNT
E6B7 CAC4E6        JZ     DIVDONE
E6BA B7            ORA    A
E6BB 7C            MOV    A,H
E6BC 1F            RAR
E6BD 67            MOV    H,A
E6BE 7D            MOV    A,L
E6BF 1F            RAR                ;DIVIDE THE CP/M SECTOR # BY THE SIZE
                                           ;      OF THE PHYSICAL SECTORS

E6C0 6F            MOV    L,A
E6C1 C3B6E6        JMP    DIVLOOP
E6C4 23            DIVDONE INX    H
E6C5 7C            MOV    A,H
E6C6 B1            ORA    C            ;RESTORE THE SIDE BIT
E6C7 67            MOV    H,A
E6C8 22F0EE        SHLD   TRUESEC        ;SAVE THE PHYSICAL SECTOR NUMBER
E6CB 21EEEE        LXI    H,CPMDRV      ;POINTER TO DESIRED DRIVE,TRACK, AND SECTOR
E6CE 11F2EE        LXI    D,BUFDRV      ;POINTER TO BUFFER DRIVE,TRACK, AND SECTOR
E6D1 0605          MVI    B,5           ;COUNT LOOP
E6D3 05            DTSLOP DCR    B      ;TEST IF DONE WITH COMPARE
E6D4 CAE2E6        JZ     MOVE           ;YES, MATCH. GO MOVE THE DATA
E6D7 1A            LDAX  D            ;GET A BYTE TO COMPARE
E6D8 BE            CMP    M            ;TEST FOR MATCH
E6D9 23            INX    H            ;BUMP POINTERS TO NEXT DATA ITEM
E6DA 13            INX    D
E6DB CAD3E6        JZ     DTSLOP        ;MATCH, CONTINUE TESTING

```

```

*****
*
* DRIVE, TRACK, AND SECTOR DON'T MATCH, FLUSH THE BUFFER IF
*
*****

```

* NECESSARY AND THEN REFILL. *

*

E6DE CDAAE7 CALL FILL ;FILL THE BUFFER WITH CORRECT PHYSICAL SECTOR
 E6E1 D8 RC ;NO GOOD, RETURN WITH ERROR INDICATION

*

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

*

E6E2 3AECEE MOVE LDA CPMSEC ;GET THE CP/M SECTOR TO TRANSFER
 E6E5 3D DCR A ;ADJUST TO PROPER SECTOR IN BUFFER
 E6E6 E600 ANI 0 ;STRIP OFF HIGH ORDERED BITS
 E6E7 = SECPSEC EQU \$-1 ;THE 0 IS MODIFIED TO REPRESENT THE # OF
 ; CP/M SECTORS PER PHYSICAL SECTORS
 ; PUT INTO HL
 E6E8 6F MOV L,A
 E6E9 2600 MVI H,0
 E6EB 29 DAD H ;FORM OFFSET INTO BUFFER
 E6EC 29 DAD H
 E6ED 29 DAD H
 E6EE 29 DAD H
 E6EF 29 DAD H
 E6F0 29 DAD H
 E6F1 29 DAD H
 E6F2 11ECEA LXI D,BUFFER ;BEGINNING ADDRESS OF BUFFER
 E6F5 19 DAD D ;FORM BEGINNING ADDRESS OF SECTGR TO TRANSFER
 E6F6 EB XCHG ;DE = ADDRESS IN BUFFER
 E6F7 210000 LXI H,0 ;GET DMA ADDRESS, THE 0 IS MODIFIED T/
 ; CONTAIN THE DMA ADDRESS
 E6F8 = CPMDMA EQU \$-2
 E6FA 3E00 MVI A,0 ;THE ZERO GETS MODIFIED TO CONTAIN
 ; A ZERO IF A READ, OR A 1 IF WRITE
 E6FB = RDWR EQU \$-1
 E6FC A7 ANA A ;TEST WHICH KIND OF OPERATION
 E6FD C205E7 JNZ INTO ;TRANSFER DATA INTO THE BUFFER
 E700 CDDEE7 OUTOF CALL MOVER
 E703 AF XRA A
 E704 C9 RET
 E705 EB INTO XCHG ;
 E706 CDDEE7 CALL MOVER ;MOVE THE DATA, HL = DESTINATION
 ; DE = SOURCE
 E709 3E01 MVI A,1
 E70B 3218E7 STA BUFWRTN ;SET BUFFER WRITTEN INTO FLAG
 E70E 3E00 MVI A,0 ;CHECK FOR DIRECTORY WRITE
 E70F = WRITYP EQU \$-1
 E710 3D DCR A
 E711 3E00 MVI A,0
 E713 320FE7 STA WRITYP ;SET NO DIRECTORY WRITE
 E716 C0 RNZ ;NO ERROR EXIT

```

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

```

```

E717 3E00      FLUSH   MVI     A,0           ;THE 0 IS MODIFIED TO REFLECT IF
;              THE BUFFER HAS BEEN WRITTEN INTO
E718 =        BUFWRN EQU     $-1
E719 A7        ANA     A           ;TEST IF WRITTEN INTO
E71A C8        RZ              ;NOT WRITTEN, ALL DONE

E71B 2118F8    LXI     H,DJWRITE   ;WRITE OPERATION FOR DISK JOCKEY
E71E 1199E8    LXI     D,HDWRITE   ;WRITE OPERATION FOR HARD DISK
E721 CDEDE7    CALL    DECIDE

```

```

*****
*
* PREP PREPARES TO READ/WRITE THE DISK. RETRIES ARE ATTEMPTED.
* UPON ENTRY, H&L MUST CONTAIN THE READ OR WRITE OPERATION
* ADDRESS.
*
*****

```

```

E724 F3       PREP    DI           ;RESET INTERRUPTS
E725 AF       XRA     A           ;RESET BUFFER WRITTEN FLAG
E726 3218E7   STA     BUFWRN
E729 228BE7   SHLD    RETRYOP     ;SET UP THE READ/WRITE OPERATION
E72C 060A     MVI     B,RETRIES   ;MAXIMUM NUMBER OF RETRIES TO ATTEMPT
E72E C5       RETRYLP PUSH    B       ;SAVE THE RETRY COUNT
E72F 3AF2EE   LDA     BUFDRV     ;GET DRIVE NUMBER INVOLVED IN THE OPERATION
E732 FE03     CPI     MAXHD*LOGDSK
E734 DA39E7   JC      NOADJST
E737 D603     SUI     MAXHD*LOGDSK

E739 4F       NOADJST MOV    C,A
E73A 2133E3   LXI     H,DJDRV     ;SELECT DRIVE
E73D 11F5E7   LXI     D,HDRV
E740 CDE9E7   CALL    DECIDGO
E743 3AF3EE   LDA     BUFTRK
E746 A7       ANA     A           ;TEST FOR TRACK ZERO
E747 4F       MOV    C,A
E748 C5       PUSH   B
E749 2109F8   LXI     H,DJHOME
E74C 1106E8   LXI     D,HDHOME
E74F CCE9E7   CZ      DECIDGO
E752 C1       POP    B           ;RESTORE TRACK #
E753 210CF8   LXI     H,DJTRK
E756 1114E8   LXI     D,HDTRK
E759 CDE9E7   CALL    DECIDGO
E75C 2AF4EE   LHLD   BUFSEC
E75F 7C       MOV    A,H           ;GET SECTOR INVOLVED IN OPERATION
E760 07       RLC           ;BIT 0 OF A EQUALS SIDE #
E761 E601     ANI    1           ;STRIP OFF UNNECESSARY BITS
E763 4F       MOV    C,A           ;C <- SIDE #
E764 2130F8   LXI     H,DJSIDE

```

```

E767 113DE8      LXI      D,HDSIDE
E76A CDE9E7      CALL     DECIDGO
E76D 2AF4EE      LHL     BUFSEC
E770 7C          MOV      A,H
E771 E67F        ANI      7FH          ;STRIP OFF SIDE BIT
E773 47          MOV      B,A          ;C <- SECTOR #
E774 4D          MOV      C,L
E775 210FF8      LXI      H,DJSEC
E778 1146E8      LXI      D,HDSEC
E77B CDE9E7      CALL     DECIDGO
E77E 01ECEA      LXI      B,BUFFER     ;SET THE DMA ADDRESS
E781 2112F8      LXI      H,DJDMA
E784 1138E8      LXI      D,HDDMA
E787 CDE9E7      CALL     DECIDGO
E78A CD0000      CALL     0             ;GET OPERATION ADDRESS
E78B =          RETRYOP EQU    $-2
E78D C1          POP      B             ;RESTORE THE RETRY COUNTER
E78E 3E00        MVI      A,0           ;NO ERROR EXIT STATUS
E790 D0          RNC
E791 05          DCR      B             ;UPDATE THE RETRY COUNTER
E792 37          STC
E793 3EFF        MVI      A,0FFH       ;ASSUME RETRY COUNT EXPIRED
E795 C8          RZ                    ;ERROR RETURN
E796 78          MOV      A,B           ;RETURN SAD NEWS
E797 FE05        CPI      RETRIES/2     ;RESEEK AFTER HALF RETRIES DONE
E799 C22EE7      JNZ      RETRYLP       ;TRY AGAIN
E79C C5          PUSH     B
E79D 2109F8      LXI      H,DJHOME
E7A0 1106E8      LXI      D,HDHOME
E7A3 CCE9E7      CZ       DECIDGO
E7A6 C1          POP      B
E7A7 C32EE7      JMP      RETRYLP       ;TRY AGAIN

```

```

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

```

```

E7AA CD17E7      FILL     CALL     FLUSH          ;FLUSH BUFFER FIRST
E7AD D8          RC              ;CHECK FOR ERROR
E7AE 11EEEE      LXI      D,CPMDRV      ;UPDATE THE DRIVE, TRACK, AND SECTOR
E7B1 21F2EE      LXI      H,BUFDRV
E7B4 0604        MVI      B,4           ;NUMBER OF BYTES TO MOVE
E7B6 CDE0E7      CALL     MOVLOP        ;COPY THE DATA
E7B9 3AFBE6      LDA      RDWR
E7BC A7          ANA      A
E7BD CAD2E7      JZ       FREAD
E7C0 3A0FE7      LDA      WRITYP
E7C3 3D          DCR      A
E7C4 3D          DCR      A
E7C5 C8          RZ
E7C6 CD7CE6      CALL     GETDPB
E7C9 110F00      LXI      D,15
E7CC 19          DAD      D
E7CD 7E          MOV      A,M

```


E7CE E603 ANI 3
 E7D0 3D DCR A
 E7D1 C8 RZ

E7D2 = FREAD EQU \$
 E7D2 2115F8 LXI H,DJREAD
 E7D5 1164E8 LXI D,HDREAD
 E7D8 CDEDE7 CALL DECIDE
 E7DB C324E7 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.
 *

E7DE 0680 MOVER MVI B,128 ;LENGTH OF TRANSFER
 E7E0 1A MOVLOP LDAX D ;GET A BTE OF SOURCE
 E7E1 77 MOV M,A ;MOVE IT
 E7E2 13 INX D ;BUMP POINTERS
 E7E3 23 INX H
 E7E4 05 DCR B ;UPDATE COUNTER
 E7E5 C2E0E7 JNZ MOVLOP ;CONTINUE MOVING UNTIL DONE
 E7E8 C9 RET

 *
 * ROUTINES TO DECIDE WHICH CONTROLLER TO USE.
 *

E7E9 CDEDE7 DECIDGO CALL DECIDE ;WHICH CONTROLLER ?
 E7EC E9 PCHL

E7ED 3AF2EE DECIDE LDA BUFDRV ;GET PROPER ROUTINE INTO H&L, BASED
 E7F0 FE03 CPI MAXHD*LOGDSK
 E7F2 D0 RNC
 E7F3 EB XCHG
 E7F4 C9 RET

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

E7F5 79 HDDRV MOV A,C ;SELECT HARD DISK DRIVE
 E7F6 CD73E6 CALL DIVLOG ;GET THE PHYSICAL DRIVE #
 E7F9 79 MOV A,C
 E7FA 3229E9 STA HDDISK ;SELECT THE DRIVE
 E7FD F6FC ORI NULL
 E7FF D352 OUT HDFUNC
 E801 3E0F MVI A,WENABL

```

E803 D350      OUT      HDCNTL
E805 C9        RET

E806 CD17E9    HDHOME   CALL      DRVPTN
E809 3600      MVI      M,0      ;SET TRACK TO ZERO
E80B DB50      IN        HDSTAT  ;TEST STATUS
E80D E601      ANI      TKZERO  ;AT TRACK ZERO ?
E80F C8        RZ
E810 AF        XRA      A      ;YES
E811 C325E8    JMP      ACCOK

E814 CD17E9    HDTRK    CALL      DRVPTN  ;GET POINTER TO CURRENT TRACK
E817 5E        MOV      E,M      ;GET CURRENT TRACK
E818 71        MOV      M,C      ;UPDATE THE TRACK
E819 7B        MOV      A,E      ;NEED TO SEEK AT ALL ?
E81A 91        SUB      C
E81B C8        RZ
E81C 3F        CMC      ;GET CARRY INTO DIRECTION
E81D DA22E8    JC        HDTRK2
E820 2F        CMA
E821 3C        INR      A
E822 C325E8    HDTRK2   JMP      ACCOK
E825 47        ACCOK    MOV      B,A      ;PREP FOR BUILD
E826 CD22E9    CALL     BUILD
E829 E6FB      SLOOP    ANI      NSTEP  ;GET STEP PULSE LOW
E82B D352      OUT     HDFUNC  ;OUTPUT LOW STEP LINE
E82D F604      ORI     PSTEP  ;SET STEP LINE HIGH
E82F D352      OUT     HDFUNC  ;OUTPUT HIGH STEP LINE
E831 05        DCR     B      ;UPDATE REPEAT COUNT
E832 C229E8    JNZ     SLOOP  ;KEEP GOING THE REQUIRED # OF TRACKS
E835 C33EE8    JMP     WSDONE

E838 60        HDDMA   MOV     H,B      ;SAVE THE DMA ADDRESS
E839 69        MOV     L,C
E83A 227EE8    SHLD   HDADD
E83D =        HDSIDE  EQU     $
E83D C9        RET

E83E DB50      WSDONE   IN     HDSTAT  ;WAIT FOR SEEK COMPLETE TO FINISH
E840 E604      ANI     COMPLT
E842 CA3EE8    JZ     WSDONE
E845 C9        RET

E846 79        HDSEC   MOV     A,C
E847 CD5BE8    CALL   DIVSPT
E84A C615      ADI     HDSPT
E84C A7        ANA     A
E84D CC57E8    CZ     GETSPT
E850 3207E9    STA     HDSECTR
E853 79        MOV     A,C
E854 3223E9    STA     HEAD
E857 3E15      GETSPT  MVI     A,HDSPT
E859 0D        DCR     C
E85A C9        RET

E85B 0E00      DIVSPT  MVI     C,0

```

```

E85D D615      DIVSPTX SUI      HDSPT
E85F D8        RC
E860 0C        INR          C
E861 C35DE8    JMP          DIVSPTX

E864 CDE2E8    HDREAD  CALL      HDPREP
E867 D8        RC
E868 AF        XRA          A
E869 D351      OUT          HDCMND
E86B 2F        CMA
E86C D353      OUT          HDDATA
E86E D353      OUT          HDDATA
E870 3E01      MVI          A, RSECT      ; READ SECTOR COMMAND
E872 D351      OUT          HDCMND
E874 CDC8E8    CALL      PROCESS
E877 D8        RC
E878 AF        XRA          A
E879 D351      OUT          HDCMND
E87B 0680      MVI          B, SECLN/4
E87D 210000    LXI          H, 0
E87E =         HDADD  EQU          $-2
E880 DB53      IN          HDDATA
E882 DB53      IN          HDDATA
E884 DB53      RTLOOP  IN          HDDATA      ; MOVE FOUR BYTES
E886 77        MOV          M, A
E887 23        INX          H
E888 DB53      IN          HDDATA
E88A 77        MOV          M, A
E88B 23        INX          H
E88C DB53      IN          HDDATA
E88E 77        MOV          M, A
E88F 23        INX          H
E890 DB53      IN          HDDATA
E892 77        MOV          M, A
E893 23        INX          H
E894 05        DCR          B
E895 C284E8    JNZ          RTLOOP
E898 C9        RET

E899 CDE2E8    HDWRITE CALL      HDPREP      ; PREPARE HEADER
E89C D8        RC
E89D AF        XRA          A
E89E D351      OUT          HDCMND
E8A0 2A7EE8    LHL          HDADD
E8A3 0680      MVI          B, SECLN/4
E8A5 7E        WTLOOP  MOV          A, M      ; MOVE 4 BYTES
E8A6 D353      OUT          HDDATA
E8A8 23        INX          H
E8A9 7E        MOV          A, M
E8AA D353      OUT          HDDATA
E8AC 23        INX          H
E8AD 7E        MOV          A, M
E8AE D353      OUT          HDDATA
E8B0 23        INX          H
E8B1 7E        MOV          A, M
E8B2 D353      OUT          HDDATA

```

```

E8B4 23      INX      H
E8B5 05      DCR      B
E8B6 C2A5E8  JNZ      WTLOOP
E8B9 3E05    MVI      A,WSECT      ;ISSUE WRITE SECTOR COMMAND
E8BB D351    OUT      HDCMND
E8BD CDC8E8  CALL     PROCESS
E8C0 D8      RC
E8C1 3E10    MVI      A,WFAULT
E8C3 A0      ANA      B
E8C4 37      STC
E8C5 C8      RZ
E8C6 AF      XRA      A
E8C7 C9      RET

E8C8 DB50    PROCESS IN      HDSTAT      ;WAIT FOR COMMAND TO FINISH
E8CA 47      MOV      B,A
E8CB E602    ANI      OPDONE
E8CD CAC8E8  JZ       PROCESS
E8D0 3E07    MVI      A,DSKCLK
E8D2 D350    OUT      HDCNTL
E8D4 DB50    IN       HDSTAT
E8D6 E608    ANI      TMOUT      ;TIMED OUT ?
E8D8 37      STC
E8D9 C0      RNZ
E8DA DB51    IN       HDRESLT
E8DC E602    ANI      RETRY      ;ANY RETRIES ?
E8DE 37      STC
E8DF C0      RNZ
E8E0 AF      XRA      A
E8E1 C9      RET

E8E2 DB50    HDPREP  IN      HDSTAT
E8E4 E620    ANI      DRVRDY
E8E6 37      STC
E8E7 C0      RNZ
E8E8 3E08    MVI      A,ISBUFF    ;INITIALIZE POINTER
E8EA D351    OUT      HDCMND
E8EC CD22E9  CALL     BUILD
E8EF F60C    ORI      0CH
E8F1 D352    OUT      HDFUNC
E8F3 3A23E9  LDA      HEAD
E8F6 D353    OUT      HDDATA      ;FORM HEAD BYTE
E8F8 CD17E9  CALL     DRVPTR
E8FB 7E      MOV      A,M          ;FORM TRACK BYTE
E8FC D353    OUT      HDDATA
E8FE A7      ANA      A
E8FF 0680    MVI      B,80H
E901 CA06E9  JZ       ZKEY
E904 0600    MVI      B,0
E906 3E00    ZKEY    MVI      A,0      ;FORM SECTOR BYTE
E907 =      HDSECTR EQU     $-1
E908 D353    OUT      HDDATA
E90A 78      MOV      A,B
E90B D353    OUT      HDDATA
E90D 3E07    MVI      A,DSKCLK
E90F D350    OUT      HDCNTL

```

E911 3E0F MVI A,WENABL
 E913 D350 OUT HDCNTL
 E915 AF XRA A
 E916 C9 RET

E917 2A29E9 DRVPTR LHL D HDDISK
 E91A EB XCHG
 E91B 1600 MVI D,0
 E91D 212DE9 LXI H,DRIVES
 E920 19 DAD D
 E921 C9 RET

E922 3E00 BUILD MVI A,0
 E923 = HEAD EQU \$-1
 E924 17 RAL
 E925 17 RAL
 E926 17 RAL
 E927 17 RAL
 E928 F600 ORI 0
 E929 = HDDISK EQU \$-1
 E92A EEF0 XRI 0FF0H
 E92C C9 RET

E92D = DRIVES EQU \$
 REPT MAXHD
 DB 0FFH
 ENDM
 E92D+FF DB 0FFH

 *
 * 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. *
 *

E92E 00 XLT128 DB 0
 E92F 01070D1319 DB 1,7,13,19,25
 E934 050B1117 DB 5,11,17,23
 E938 03090F15 DB 3,9,15,21
 E93C 02080E141A DB 2,8,14,20,26
 E941 060C1218 DB 6,12,18,24
 E945 040A1016 DB 4,10,16,22

E949 00 XLT256 DB 0
 E94A 0102131425 DB 1,2,19,20,37,38
 E950 0304151627 DB 3,4,21,22,39,40
 E956 0506171829 DB 5,6,23,24,41,42
 E95C 0708191A2B DB 7,8,25,26,43,44
 E962 090A1B1C2D DB 9,10,27,28,45,46
 E968 0B0C1D1E2F DB 11,12,29,30,47,48
 E96E 0D0E1F2031 DB 13,14,31,32,49,50

```

E974 0F10212233 DB 15,16,33,34,51,52
E97A 11122324 DB 17,18,35,36

E97E 00 XLT512 DB 0
E97F 0102030411 DB 1,2,3,4,17,18,19,20
E987 2122232431 DB 33,34,35,36,49,50,51,52
E98F 0506070815 DB 5,6,7,8,21,22,23,24
E997 2526272835 DB 37,38,39,40,53,54,55,56
E99F 090A0B0C19 DB 9,10,11,12,25,26,27,28
E9A7 292A2B2C39 DB 41,42,43,44,57,58,59,60
E9AF 0D0E0F101D DB 13,14,15,16,29,30,31,32
E9B7 2D2E2F30 DB 45,46,47,48
    
```

```

E9BB 00 XLT124 DB 0
E9BC 0102030405 DB 1,2,3,4,5,6,7,8
E9C4 191A1B1C1D DB 25,26,27,28,29,30,31,32
E9CC 3132333435 DB 49,50,51,52,53,54,55,56
E9D4 090A0B0C0D DB 9,10,11,12,13,14,15,16
E9DC 2122232425 DB 33,34,35,36,37,38,39,40
E9E4 393A3B3C3D DB 57,58,59,60,61,62,63,64
E9EC 1112131415 DB 17,18,19,20,21,22,23,24
E9F4 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.
*
*****
    
```

```

E9FC 1A00 DPB128S DW 26 ;CP/M SECTORS/TRACK
E9FE 03 DB 3 ;BSH
E9FF 07 DB 7 ;BLM
EA00 00 DB 0 ;EXM
EA01 F200 DW 242 ;DSM
EA03 3F00 DW 63 ;DRM
EA05 C0 DB 0C0H ;AL0
EA06 00 DB 0 ;AL1
EA07 1000 DW 16 ;CKS
EA09 0200 DW 2 ;OFF
EA0B 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.
*
*****
    
```


;8 IF DOUBLE SIDED.

```
*****
*
* THE FOLLOWING DPB DEFINES A DISKETTE FOR 128 BYTE SECTORS,
* SINGLE DENSITY, AND DOUBLE SIDED.
*
```

```
EA3C 3400 DPB128D DW 52 ;CP/M SECTORS/TRACK
EA3E 04 DB 4 ;BSH
EA3F 0F DB 15 ;BLM
EA40 01 DB 1 ;EXM
EA41 F200 DW 242 ;DSM
EA43 7F00 DW 127 ;DRM
EA45 C0 DB 0C0H ;AL0
EA46 00 DB 0 ;AL1
EA47 2000 DW 32 ;CKS
EA49 0200 DW 2 ;OFF
EA4B 09 DB 9H
```

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

```
EA4C 6800 DPB256D DW 104 ;CP/M SECTORS/TRACK
EA4E 04 DB 4 ;BSH
EA4F 0F DB 15 ;BLM
EA50 00 DB 0 ;EXM
EA51 E601 DW 486 ;DSM
EA53 FF00 DW 255 ;DRM
EA55 F0 DB 0F0H ;AL0
EA56 00 DB 0 ;AL1
EA57 4000 DW 64 ;CKS
EA59 0200 DW 2 ;OFF
EA5B 1A DB 1AH
```

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

```
EA5C 7800 DPB512D DW 120 ;CP/M SECTORS/TRACK
EA5E 04 DB 4 ;BSH
EA5F 0F DB 15 ;BLM
EA60 00 DB 0 ;EXM
EA61 3102 DW 561 ;DSM
EA63 FF00 DW 255 ;DRM
EA65 F0 DB 0F0H ;AL0
EA66 00 DB 0 ;AL1
EA67 4000 DW 64 ;CKS
```


EA69 0200 DW 2 ;OFF
 EA6B 3B DB 3BH

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

EA6C 8000 DP1024D DW 128 ;CP/M SECTORS/TRACK
 EA6E 04 DB 4 ;BSH
 EA6F 0F DB 15 ;BLM
 EA70 00 DB 0 ;EXM
 EA71 5702 DW 599 ;DSM
 EA73 FF00 DW 255 ;DRM
 EA75 F0 DB 0F0H ;AL0
 EA76 00 DB 0 ;AL1
 EA77 4000 DW 64 ;CKS
 EA79 0200 DW 2 ;OFF
 EA7B 7C DB 7CH

 *
 * THE FOLLOWING DPB'S ARE FOR THE STANDARD FORMAT TO BE
 * COMPATABLE WITH OLDER VERSIONS OF THE CBIOS.
 *

EA7C A002 DPBHD1 DW 672 ;CP/M SECTORS/TRACK
 EA7E 05 DB 5 ;BSH
 EA7F 1F DB 31 ;BLM
 EA80 01 DB 1 ;EXM
 EA81 DF07 DW 2015 ;DSM
 EA83 FF01 DW 511 ;DRM
 EA85 FF DB 0FFH ;AL0
 EA86 FF DB 0FFH ;AL1
 EA87 0000 DW 0 ;CKS
 EA89 0100 DW 1 ;OFF
 EA8B 33 DB 33H ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
 ;LOG2(#BYTES PER SECTOR/128) + 1 +
 ;8 IF DOUBLE SIDED.

EA8C A002 DPBHD2 DW 672 ;CP/M SECTORS/TRACK
 EA8E 05 DB 5 ;BSH
 EA8F 1F DB 31 ;BLM
 EA90 01 DB 1 ;EXM
 EA91 DF07 DW 2015 ;DSM
 EA93 FF01 DW 511 ;DRM
 EA95 FF DB 0FFH ;AL0
 EA96 FF DB 0FFH ;AL1
 EA97 0000 DW 0 ;CKS
 EA99 6200 DW 98 ;OFF
 EA9B 33 DB 33H ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
 ;LOG2(#BYTES PER SECTOR/128) + 1 +
 ;8 IF DOUBLE SIDED.

```
EA9C A002      DPBHD3 DW      672      ;CP/M SECTORS/TRACK
EA9E 05        DB        5          ;BSH
EA9F 1F        DB        31          ;BLM
EAA0 01        DB        1          ;EXM
EAA1 0404      DW      1028         ;DSM
EAA3 FF01      DW      511          ;DRM
EAA5 FF        DB      0FFH         ;AL0
EAA6 FF        DB      0FFH         ;AL1
EAA7 0000      DW        0          ;CKS
EAA9 C300      DW      195          ;OFF
EAAB 33        DB      33H          ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
;LOG2(#BYTES PER SECTOR/128) + 1 +
;8 IF DOUBLE SIDED.
```

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

```
HEADER MACRO ND,DPB
      DW      0          ;TRANSLATION TABLE FILLED IN LATER
      DW      0,0,0      ;SCRATCH
      DW      DIRBUF     ;DIRECTORY BUFFER
      DW      DPB        ;DPB FILLED IN LATER
      DW      CSV&ND     ;DIRECTORY CHECK VECTOR
      DW      ALV&ND     ;ALLOCATION VECTOR
      ENDM
```

```
EAAC = DPBASE EQU $
0000 # DN SET 0
      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
      HEADER %DN,DPBHD3
DN SET DN+1
      ENDM
EAAC+0000 DW 0 ;TRANSLATION TABLE FILLED IN LATER
EAAE+000000000000 DW 0,0,0 ;SCRATCH
EAB4+F6EE DW DIRBUF ;DIRECTORY BUFFER
EAB6+7CEA DW DPBHD1 ;DPB FILLED IN LATER
EAB8+72F0 DW CSV0 ;DIRECTORY CHECK VECTOR
EABA+76EF DW ALV0 ;ALLOCATION VECTOR
EABC+0000 DW 0 ;TRANSLATION TABLE FILLED IN LATER
EABE+000000000000 DW 0,0,0 ;SCRATCH
EAC4+F6EE DW DIRBUF ;DIRECTORY BUFFER
EAC6+8CEA DW DPBHD2 ;DPB FILLED IN LATER
EAC8+6EF1 DW CSV1 ;DIRECTORY CHECK VECTOR
EACA+72F0 DW ALV1 ;ALLOCATION VECTOR
EACC+0000 DW 0 ;TRANSLATION TABLE FILLED IN LATER
EACE+000000000000 DW 0,0,0 ;SCRATCH
EAD4+F6EE DW DIRBUF ;DIRECTORY BUFFER
EAD6+9CEA DW DPBHD3 ;DPB FILLED IN LATER
```

```

EAD8+EFF1      DW      CSV2          ;DIRECTORY CHECK VECTOR
EADA+6EF1      DW      ALV2          ;ALLOCATION VECTOR
               REPT    MAXFLOP
               HEADER  %DN,0
               DN      SET    DN+1
               ENDM
EADC+0000      DW      0             ;TRANSLATION TABLE FILLED IN LATER
EADE+00000000 DW      0,0,0         ;SCRATCH
EAE4+F6EE      DW      DIRBUF       ;DIRECTORY BUFFER
EAE6+0000      DW      0             ;DPB FILLED IN LATER
EAES+3AF2      DW      CSV3         ;DIRECTORY CHECK VECTOR
EAEA+EFF1      DW      ALV3         ;ALLOCATION VECTOR
    
```

```
EAEC =          BUFFER EQU    $
```

```

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

```

EAEC 801A      PROMPT DB      80H, CLEAR          ;CLEAN BUFFER AND SCREEN
EAEE 0A0D0A0D0A DB      ACR,ALF,ACR,ALF,ACR,ALF
EAF4 4D6F72726F DB      'Morrow Designs '
EB03 36        DB      '0'+MSIZE/10             ;CP/M MEMORY SIZE
EB04 32        DB      '0'+(MSIZE MOD 10)
EB05 4B2043502F DB      'K CP/M '              ;CP/M VERSION NUMBER
EB0C 32        DB      CPMREV/10+'0'
EB0D 2E        DB      '.'
EB0E 32        DB      (CPMREV MOD 10)+'0'
EB0F 2C20436269 DB      ', Cbios rev '
EB1B 322E      DB      REVNUM/10+'0', '.'       ;CBIOS REVISION NUMBER
EB1D 39        DB      REVNUM MOD 10+'0'
EB1E 2E        DB      '.'
EB1F 32        DB      MREV/10+'0'
EB20 30        DB      MREV MOD 10+'0'
EB21 0A0D      DB      ACR,ALF
EB23 466F7220  DB      'For '
EB27 6120446973 DB      'a Disk Jockey 2D/B'
EB39 20616E6420 DB      ' and '
EB3E 6120      DB      'a '
EB40 46756A6974 DB      'Fujitsu M20 '
EB4C 6861726420 DB      'hard disk'
EB55 2E        DB      '.'
EB56 0A0D0A0D  DB      ACR,ALF,ACR,ALF
EB5A 2020202020 DB      ' THE W6GO/K6HHD LIST'
EB74 0A0D      DB      ACR,ALF
EB76 2020202020 DB      ' Electronics Enterprises'
EB92 0A0D      DB      ACR,ALF
EB94 2020202020 DB      ' Rio Linda, California'
EBAF 0A0D      DB      ACR,ALF
EBB1 00        DB      0             ;END OF MESSAGE
    
```

```

*****
*
* UTILITY ROUTINE TO OUTPUT THE MESSAGE POINTED AT BY H&L,
*
*****
    
```

```

* TERMINATED WITH A NULL.
*
*****
EBB2 7E MESSAGE MOV A,M ;GET A CHARACTER OF THE MESSAGE
EBB3 23 INX H ;BUMP TEXT POINTER
EBB4 B7 ORA A ;TEST FOR END
EBB5 C8 RZ ;RETURN IF DONE
EBB6 E5 PUSH H ;SAVE POINTER TO TEXT
EBB7 4F MOV C,A ;OUTPUT CHARACTER IN C
EBB8 CD5CE3 CALL CONOUT ;OUTPUT THE CHARACTER
EBBB E1 POP H ;RESTORE THE POINTER
EBBC C3B2EB 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.
*
*****

```

```

EBBF 310001 CBOOT LXI SP,TPA ;SET UP STACK

EBC2 AF XRA A ;CLEAR COLD BOOT FLAG
EBC3 32D4E4 STA CWFLG
EBC6 323AE3 STA GROUP ;CLEAR GROUP SELECT BYTE
EBC9 2100FC LXI H,DJRAM ; OF THE JUMP TABLE.
EBCC 1100F8 LXI D,ORIGIN
EBCF 0633 MVI B,33H ;SIZE OF JUMP TABLE
EBD1 CDE0E7 CALL MOVLOP ;COPY TABLE
EBD4 3EC0 MVI A,INTIOBY
EBD6 320300 STA IOBYTE
EBD9 CDFFE8 CALL TINIT ;INITIALIZE THE TERMINAL
EBDC CD34EC CALL LINIT ;INITIALIZE THE LIST DEVICE
EBDF 21ECEA LXI H,PROMPT ;PREP FOR SENDING SIGNON MESSAGE
EBE2 CDB2EB CALL MESSAGE ;SEND THE PROMPT
EBE5 AF XRA A ;SELECT DISK A
EBE6 32EEEE STA CPMDRV
EBE9 320400 STA CDISK
EBEC 3280E5 STA FLOPFLG
EBEF 2103E3 LXI H,BIOS+3 ;PATCH COLD BOOT TO WARM CODE
EBF2 2201E3 SHLD BIOS+1
EBF5 C382E4 JMP GOCPM

```

```

*****
*
* TERMINAL INITIALIZATION ROUTINE
*
* THIS INITIALIZING ROUTINE SAMPLES BIT 0 OF SWBD PORT 7 TO
* DETERMINE IF THE KEYBOARD IS PLUGGED IN. IF THE KEYBOARD IS
* PLUGGED IN, THE LSB RETURNS A 0. 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.
*
*****

```



```
EBF8 1B7B3F TV950 DB 1BH,7BH,3FH ;TELEVIDEO COMMAND SEQUENCE
EBFB 31373100 DB '171',0
```

```
EBFF 21F8EB TINIT LXI H,TV950 ;SET TELEVIDEO 950 TO 19.2KB
EC02 CDB2EB CALL MESSAGE
```

```
EC05 2AFFFF LHL D0FFFF ;WAIT FOR TERMINAL
EC08 2B WAITC DCX H
EC09 7C MOV A,H
EC0A B5 ORA L
EC0B C208EC JNZ WAITC
EC0E 3A3AE3 LDA GROUP 0 ;GET GROUP BYTE
EC11 F601 ORI CONGRP 1 ;SELECT CONSOLE DEVICE
EC13 D34F (+7) OUT GRPSEL (+7)
EC15 DB48 IN RBR Base ;CLEAR RECIEVER BUFFERS
EC17 DB48 IN RBR Base
```

```
EC19 AF XRA A
EC1A D34D OUT LSR +5 ;CLEAR STATUS LINE STATUS REGISTER
EC1C D349 OUT IER +1 ;SET NO INTERRUPTS INTERRUPT ENABLE REGISTER
EC1E 2A36E3 LHL DEFCON (6) ;GET DEFAULT BAUD RATE
EC21 EB XCHG
EC22 3E87 MVI A,DLAB+WLS1+WLS0+STB = 87H ;ENABLE DIVISOR ACCESS LATCH
EC24 D34B OUT LCR +3 LINE CONTROL REGISTER ;SET THE BAUD RATE IN (DE)
EC26 7A MOV A,D
EC27 D349 OUT DLM Divisor (MSB) + 1 ;SET UPPER DIVISOR
EC29 7B MOV A,E
EC2A D348 OUT DLL Divisor (LSB) 0 ;SET LOWER DIVISOR
EC2C 3E07 MVI A,WLS1+WLS0+STB = 7H
EC2E D34B OUT LCR 2 1 4 LINE CONTROL REGISTER
```

DIVISOR LATCH ACCESS BIT 80H
WLS1 WORD LENGTH SELECT BIT 1 FOR 8 BIT WORD
WLS0 WORD LENGTH SELECT BIT 0
STB STOP BIT COUNT - 2 STOP BITS

```
EC30 AF DONE XRA A ;CLEAR STATUS REGISTER
EC31 D34D OUT LSR
; IN 7 ;GET KEYBOARD INTERLOCK BYTE
; ANI 1 ;GET BIT 1 ONLY
; ADI INTIOBY ;ADD INTIOBY TO KEYBOARD BIT
; STA IOBYTE ;INITIALIZE IOBYTE
```

~~LDA~~ ORI

```
EC34 3A3AE3 LINIT LDA GROUP 0 ;GET GROUP BYTE
EC37 F603 ORI LSTGRP ;SELECT LIST DEVICE
EC39 D34F OUT GRPSEL
EC3B 3E80 MVI A,DLAB ;ACCESS DIVISOR LATCH
EC3D D34B OUT LCR
EC3F 2A38E3 LHL DEFCON (12) ;GET LST: BAUD RATE DIVISOR
EC42 7C MOV A,H
EC43 D349 OUT DLM ;SET UPPER BAUD RATE
EC45 7D MOV A,L
EC46 D348 OUT DLL
EC48 3E07 MVI A,STB+WLS0+WLS1
EC4A D34B OUT LCR
EC4C DB48 IN RBR ;CLEAR INPUT BUFFER
EC4E AF XRA A
EC4F D349 OUT IER ;NO INTERUPTS
```

```

EC51 C9          RET
EC52 00FF00     DB      0,0FFH,0
EC55           DS      512-(S-BUFFER) ;MAXIMUM SIZE BUFFER FOR 512 BYTE SECTORS
ECEC           DS      512          ;ADDITIONAL SPACE FOR FLOPPIES 1K SECTORS
    
```

```

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

```

EEEC 0000     CPMSEC  DW      0          ;CP/M SECTOR #
EEEE 00       CPMDRV  DB      0          ;CP/M DRIVE #
EEEF 00       CPMTRK  DB      0          ;CP/M TRACK #
EEF0 0000     TRUESEC DW      0          ;DISK JOCKEY SECTOR THAT CONTAINS CP/M SECTOR
EEF2 00       BUFDRV  DB      0          ;DRIVE THAT BUFFER BELONGS TO
EEF3 00       BUFTRK  DB      0          ;TRACK THAT BUFFER BELONGS TO
EEF4 0000     BUFSEC  DW      0          ;SECTOR THAT BUFFER BELONGS TO
EEF6         DIRBUF  DS      128         ;DIRECTORY BUFFER
          ALLOC  MACRO  ND,AL,CS
          ALV&ND DS      AL
          CSV&ND DS      CS
          ENDM
    
```

```

0000 #        DN      SET      0
          REPT  MAXHD
          ALLOC  %DN,252,0
          DN      SET      DN+1
          ALLOC  %DN,252,0
          DN      SET      DN+1
          ALLOC  %DN,129,0
          DN      SET      DN+1
          ENDM
    
```

```

EF76+        ALV0     DS      252
F072+        CSV0     DS      0
F072+        ALV1     DS      252
F16E+        CSV1     DS      0
F16E+        ALV2     DS      129
F1EF+        CSV2     DS      0
          REPT  MAXFLOP
          ALLOC  %DN,75,64
          DN      SET      DN+1
          ENDM
    
```

```

F1EF+        ALV3     DS      75
F23A+        CSV3     DS      64
    
```

```

F27A         END
    
```

0006	AACK	E825	ACCOK	000A	ACR	0003	AETX	000D	ALF
EF76	ALV0	F072	ALV1	F16E	ALV2	F1EF	ALV3	E4D5	AUTOFLG
0004	BANK	D500	BDOS	A800	BIAS	E300	BIOS	EEF2	BUFDRV
0080	BUFF	EAEC	BUFFER	EEF4	BUFSEC	EEF3	BUFTRK	E718	BUFWRN
E922	BUILD	EBBF	CBOOT	CD00	CCP	0004	CDISK	E40C	CICRT
E40C	CIPTR	E38D	CITBLE	F803	CITTY	E3F7	CIUC1	E40C	CIUR1
E40C	CIUR2	E4CD	CLDBOT	E4B8	CLDCMND	001A	CLEAR	004A	CLK
E58C	CLOPP	00D0	CLRCMD	FBFC	CMDREG	E431	COCRT	E4D6	COLDBEG
E4D6	COLDEND	E3CD	COLPT	0004	COMPLT	0001	CONGRP	E347	CONIN
E34D	CONIN1	E455	CONINA	E35C	CONOUT	E468	CONOUT1	E33B	CONST
E3CD	COPTP	E43C	COPTR	E443	COPTR1	E395	COTBLE	F806	COTTY
E3D8	COUL1	E3F6	COUNT	E3CD	COUP1	E3CD	COUP2	E6F8	CPMDMA
EEEE	CPMDRV	0016	CPMREV	EEEC	CPMSEC	EEEF	CPMTRK	E420	CSCRT
E420	CSPTR	E341	CSREADR	E3BD	CSRTBLE	E3B5	CSTBLE	E418	CSTTY
E403	CSUC1	E420	CSUR1	E420	CSUR2	F072	CSV0	F16E	CSV1
F1EF	CSV2	F23A	CSV3	0010	CTS	E4D4	CWFLG	0048	DAISI0
0049	DAISI1	0048	DAISY0	0049	DAISY1	0008	DBLSID	E671	DCRC
E7ED	DECIDE	E7E9	DECIDGO	E336	DEFCON	E338	DEFLST	0020	DENABLE
EEF6	DIRBUF	E6C4	DIVDONE	E673	DIVLOG	E675	DIVLOGX	E6B6	DIVLOOP
E85B	DIVSPT	E85D	DIVSPTX	F800	DJBOOT	F803	DJCIN	F806	DJCOUT
F82D	DJDEN	F812	DJDMA	E333	DJDRV	F82A	DJERR	F809	DJHOME
E5AF	DJINIT	E5CD	DJNEXT	FC00	DJRAM	F815	DJREAD	F80F	DJSEC
F81B	DJSEL	F830	DJSIDE	F827	DJSTAT	F80C	DJTRK	F821	DJTSTAT
F818	DJWRITE	0080	DLAB	0048	DLL	0049	DLM	EC30	DONE
E51C	DONOP	EA6C	DP1024D	EA2C	DP1024S	EA3C	DPB128D	E9FC	DPB128S
EA4C	DPB256D	EA0C	DPB256S	EA5C	DPB512D	EA1C	DPB512S	EAAC	DPBASE
EA7C	DPBHD1	EA8C	DPBHD2	EA9C	DPBHD3	FBF9	DREG	0001	DR
E92D	DRIVES	E627	DRVHD	E917	DRVPTR	0020	DRVRDY	0007	DSKCLK
0020	DSR	E6D3	DTSLOP	0008	ENINT	0005	ENTRY	E7AA	FILL
E580	FLOPFLG	E5D2	FLOPOK	E717	FLUSH	E7D2	FREAD	E67C	GETDPB
E857	GETSPT	E482	GOCPM	E33A	GROUP	004F	GRPSEL	0000	GZERO
E87E	HDADD	0051	HDCMND	0050	HDCNTL	0053	HDDATA	E929	HDDISK
E838	HDDMA	E7F5	HDDRV	0052	HDFUNC	E806	HDHOME	0050	HDORG
E8E2	HDPREP	E864	HDREAD	0051	HDRESLT	0004	HDRLEN	E846	HDSEC
E907	HDSECTR	E83D	HDSIDE	0015	HDSPT	0050	HDSTAT	E814	HDTRK
E822	HDTRK2	E899	HDWRITE	E923	HEAD	E523	HOME	0000	IDBUFF
0049	IER	0040	INDEX	00C0	INTIOBY	E705	INTO	FBF8	IO
0003	IOBYTE	0008	ISBUFF	004B	LCR	EC34	LINIT	E37C	LIST
E37F	LIST1	E387	LISTST	0003	LOGDSK	E429	LSLPT	004D	LSR
E3C5	LSTBLE	0003	LSTGRP	E39D	LTBLE	0001	MAXFLOP	0001	MAXHD
0048	MBASE	00F7	MDIR	EBB2	MESSAGE	E44E	MIOIN	E461	MIOOUT
E473	MIOSTAT	E6E2	MOVE	E7DE	MOVER	E7E0	MOVLOP	0014	MREV
003E	MSIZE	004E	MSR	E739	NOADJST	00FB	NSTEP	00FC	NULL
3E00	OFFSETC	0002	OPDONE	F800	ORIGIN	E700	OUTOF	E377	PNCH1
E724	PREP	E8C8	PROCESS	EAEC	PROMPT	0004	PSTEP	E3A5	PTBLE
E371	PUNCH	E3ED	PWAIT	0048	RBR	E6FB	RDWR	E367	READER
E6A4	READ	E36A	READERA	E36D	READR1	E42E	READY	E6A8	REDWRT
0010	RESTOR	000A	RETRIES	0002	RETRY	E72E	RETRYLP	E78B	RETRYOP
001D	REVNUM	0001	RSECT	E3AD	RTBLE	E884	RTLOOP	0001	S0
0002	S1	0005	SCENBL	0200	SECLN	E6E7	SECPSEC	E6A9	SECSIZ
E52A	SECTRAN	E351	SELDEV	0049	SENSESW	E51D	SETDMA	E565	SETDRV
E654	SETDRV1	E517	SETSEC	E525	SETTRK	E540	SIDEA	E617	SIDEOK
E543	SIDEONE	E549	SIDETWO	E829	SLOOP	0003	SMASK	E41B	STAT
0004	STB	E643	TDELAY	0048	THR	0020	THRE	EBFF	TINIT
0001	TKZERO	0008	TMOUT	0100	TPA	E532	TRANFP	E561	TRANHD
EEF0	TRUESEC	EBF8	TV950	EC08	WAITC	E4D7	WARMBEG	E4D7	WARMEND

E4F4 WARMLOD	E506 WARMRD	E303 WBOOTE	E4D8 WBOOT	0000 WBOT
000F WENABL	0010 WFAULT	0001 WLS0	0002 WLS1	000B WRESET
E69D WRITE	E70F WRITYP	E509 WRMREAD	E83E WSDONE	0005 WSECT
E8A5 WTLOOP	E9BB XLT124	E92E XLT128	E949 XLT256	E97E XLT512
E695 XLTS	E906 ZKEY	E66D ZRET		