

```

*****
*
* 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, 1024 BYTES DOUBLE
* DENSITY. THERE ARE CONDITIONAL ASSEMBLIES FOR DISKUS HARD
* DISK CONTROLLER.
*
* WRITTEN BY BOBBY DALE GIFFORD.
* 12/8/80
*
* CUSTOMIZED BY JAY O'BRIEN
* 1/16/82
*
* DISK MAP OF SECTORS USED BY COLD BOOT, WARM BOOT, FIRMWARE,
* AND CP/M:
*
* TRK 0 SEC 1 = FIRST SECTOR OF COLD BOOT.          E700H
*           2 = COLD BOOT 256.                      80H
*           3 = COLD BOOT 512.                      80H
*           4 = COLD BOOT 1024.                     80H
*           5 = WARM BOOT 256.                      80H
*           6 = WARM BOOT 512.                      80H
*           7 = WARM BOOT 1024.                     80H
*           8 = COLD/WARM BOOT.                     2C00H
*           9 = FIRMWARE.                            E400H
*           10 = FIRMWARE+80H.                       E480H
*           11 = FIRMWARE+100H.                      E500H
*           12 = FIRMWARE+180H.                      E580H
*           13 = FIRMWARE+200H.                      E600H
*           14 = FIRMWARE+280H.                      E680H
*           15 = FIRMWARE+300H.                      E700H
*           16 = FIRMWARE+380H.                      E780H
*           17 = CCP.                                 2700H
*           18 = CCP+80H.                             2780H
*           19 = CCP+100H.                            2800H
*           20 = CCP+180H.                            2880H
*           21 = CCP+200H.                            2900H
*           22 = CCP+280H.                            2980H
*           23 = CCP+300H.                            2A00H
*           24 = CCP+380H.                            2A80H
*           25 = CCP+400H.                            2B00H
*           26 = CCP+480H.                            2B80H
*           = REST OF CP/M.                          2C00H-4FFFH
*
*****

```

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

```

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

```

CBIOSG.PRN

3/26/82

60K DJTAD

H/D A.E.C

FLOPPY DIS

V10X

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. *
 *

0001 = FIRST EQU 1 ;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)

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 ;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
      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
0008 =      VIOX     EQU      8H         ;BASE PORT FOR VIO-X
    
```

```

*****
*
* 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 C3ABE5    JMP      CBOOT            ;COLD BOOT ENTRY POINT
DD03 C3AEDE    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 C304DF    JMP      HOME            ;HOME DRIVE
DD1B C346DF    JMP      SETDRV         ;SELECT DISK
DD1E C306DF    JMP      SETTRK        ;SET TRACK
DD21 C3F8DE    JMP      SETSEC         ;SET SECTOR
DD24 C3FEDE    JMP      SETDMA         ;SET DMA ADDRESS
DD27 C36AE0    JMP      READ            ;READ THE DISK
DD2A C363E0    JMP      WRITE           ;WRITE THE DISK
DD2D C382DD    JMP      LISTST         ;LIST DEVICE STATUS
DD30 C30BDF    JMP      SECTRAN        ;SECTOR TRANSLATION

DD33 C31BF4    DJDRV  IF      MAXFLOP NE 0
                DJSEL  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 = CITY 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 CONINI ;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 CDDDE0 CONIN CALL FLUSH ;FLUSH THE DISK BUFFER
DD45 2188DD LXI H,CITBLE ;BEGINNING OF CHARACTER INPUT TABLE
```

```
*
* ENTRY AT CONINI 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 CDDDE0 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 08DE      DW      CICRT      ;INPUT FROM CRT (CURRENTLY SWITCHBOARD
;          SERIAL PORT 1)
DD8C 62DD      DW      READER     ;INPUT FROM READER (DEPENDS ON READER
;          SELECTION)
DD8E 03F0      DW      CITYY     ;INPUT FROM TTY (CURRENTLY INPUT FROM
;          DISK JOCKEY 2D)

```

*
* CONSOLE OUTPUT TABLE
*

```

DD90 25DE      COTBLE DW      COCRT ;OUTPUT TO CRT
;
DD92 25DE      DW      COCRT      ;OUTPUT TO CRT
;
DD94 77DD      DW      LIST       ;OUTPUT TO LIST DEVICE (DEPENDS ON
;          BITS 6&7 OF IOBYTE)
DD96 06F0      DW      COTTY     ;OUTPUT TO TTY (CURRENTLY OUTPUT TO
;          DISK JOCKEY 2D)

```

*
* LIST DEVICE TABLE
*

```

DD98 06F0      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
*

```

DDA0 06F0      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 03F0      RTBLE  DW      CITYY ;INPUT FROM TTY (CURRENTLY ASSIGNED
;          BY INTIOBY, INPUT FROM 2D)
DDAA 08DE      DW      CIPTR     ;INPUT FROM PAPER TAPE READER (CURRENTLY
;          SWITCHBOARD SERIAL PORT 1)
DDAC 08DE      DW      CIURI     ;INPUT FROM USER READER 1 (CURRENTLY
;          SWITCHBOARD SERIAL PORT 1)

```


DDAE 08DE DW CIUR2 ;INPUT FROM USER READER 2 (CURRENTLY SWITCHBOARD SERIAL PORT 1)

*
* CONSOLE STATUS TABLE
*

DDB0 FFDD CSTBLE DW CSUC1 ;STATUS FROM SWBD PARALLEL PORT 4, AS READ FROM ATTN BIT 0)
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
*

DDC0 35DE LSTBLE DW READY ;CONSOLE ALWAYS READY
DDC2 35DE DW READY ;GET LIST STATUS
DDC4 30DE DW LSLPT
DDC6 30DE DW LSLPT

*
* ROUTINES FOR MY SYSTEM. J. J. O'BRIEN
*

DDC8 00

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 DB02 COLPT IN 2 ;OUTPUT FROM LINE PRINTER,GET STATUS
DDCB E680 ANI 80H ;WAIT UNTIL OK TO SEND
DDCD CAC9DD JZ COLPT

```

DDD0 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
DE0A E640      ANI      40H         ;WAIT FOR CHARACTER
DE0C CA08DE    JZ       CIPTR       ;
DE0F DB01      IN      1          ;
DE11 E67F      ANI      7FH         ;STRIP OFF THE PARITY
DE13 C9        RET
    
```

```

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

```

DE14 CD21F0    CSTTY  CALL    DJTSTAT    ;STATUS FROM DISK JOCKEY 2D
DE17 3E00      STAT   MVI     A,0         ;PREP FOR ZERO RETURN
DE19 C0        RNZ                     ;NOTHING FOUND
DE1A 3D        DCR     A              ;RETURN WITH 0FFH
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 DB02      CSCRT  IN      2          ;STATUS FROM CRT, GET STATUS
DE1E E640      ANI      40H         ;STRIP OF DATA READY BIT
DE20 EE40      XRI      40H         ;MAKE CORRECT POLARITY
DE22 C317DE    JMP     STAT         ;RETURN PROPER INDICATION
    
```

```

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

```

DE25 DB09      COCRT  IN      VIOX+1    ;READ STATUS PORT
DE27 E601      ANI      1           ;MASK TXRDY BIT
DE29 CA25DE    JZ       COCRT       ;WAIT FOR READY
DE2C 79        MOV     A,C          ;GET CHAR
DE2D D308      OUT     VIOX         ;OUTPUT CHAR
DE2F C9        RET                 ;ALL DONE
    
```

```

*****
*
*  LIST DEVICE STATUS ROUTINES.
*
*****
    
```

```

DE30 DB02      LSLPT  IN      2          ;ALL OTHER DEVICES WAIT
DE32 E680      ANI      80H
DE34 C8        RZ
    
```

```
DE35 3EFF    READY  MVI    A,0FFH
DE37 C9      RET
```

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

```
DE38 0E1A    TINIT  MVI    C,CLEAR    ;INITIALIZE THE TERMINAL ROUTINE
DE3A DB07    IN      7            ;GET KEYBOARD INTERLOCK BYTE
DE3C E601    ANI    1            ;GET BIT 1 ONLY
DE3E C6C0    ADI    INTIOBY    ;ADD INTIOBY TO KEYBOARD BIT
DE40 320300  STA    IOBYTE    ;INITIALIZE IOBYTE
DE43 C30CDD  JMP    COUT
```

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

```
DE46 DB02    COPTR  IN      2            ;INPUT FROM PORT 2
DE48 E608    ANI    8            ;WAIT UNTIL OK TO SEND
DE4A CA46DE  JZ     COPTR
DE4D DB05    COPTR1 IN     5            ;BUFFER FULL?
DE4F E601    ANI    1
DE51 CA4DDE  JZ     COPTR1    ;WAIT UNTIL PRINTER READY
DE54 79      MOV    A,C            ;OUTPUT THE CHARACTER
DE55 D300    OUT    0
DE57 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).
*
*****
```

```
DE58 218000  GOCPM  LXI    H,BUFF    ;SET UP INITIAL DMA ADDRESS
DE5B CDFEDE  CALL   SETDMA
DE5E 3EC3    MVI    A,(JMP)    ;INITIALIZE JUMP TO WARM BOOT
DE60 320000  STA    WBOT
DE63 320500  STA    ENTRY    ;INITIALIZE JUMP TO BDOS
DE66 2103DD  LXI    H,WBOOTE    ;ADDRESS IN WARM BOOT JUMP
DE69 220100  SHLD   WBOT+1
DE6C 2106CF  LXI    H,BDOS+6    ;ADDRESS IN BDOS JUMP
DE6F 220600  SHLD   ENTRY+1
DE72 AF     XRA    A            ;A <- 0
```

```

DE73 32D6E4      STA      BUFSEC          ;DISK JOCKEY BUFFER EMPTY
DE76 32DEE0      STA      BUFWRN       ;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 3AAADE      LDA      CWFLG
DE80 A7          ANA      A
DE81 11ACDE      LXI      D,COLDBEG     ;BEGINNING OF INITIAL COMMAND
DE84 3E01        MVI      A,COLDEND-COLDBEG+1 ;LENGTH OF COMMAND
DE86 CA8EDE      JZ       CLDCMND
DE89 11ADDE      LXI      D,WARMBEG
DE8C 3E01        MVI      A,WARMEND-WARMBEG+1
DE8E 2108C7      CLDCMND LXI      H,CCP+8     ;COMMAND BUFFER
DE91 3207C7      STA      CCP+7
DE94 47          MOV      B,A
DE95 CDA5E1      CALL    MOVLOP
DE98 3AAADE      LDA      CWFLG
DE9B A7          ANA      A
DE9C 3AABDE      LDA      AUTOFLG
DE9F CAA3DE      JZ       CLDBOT
DEA2 1F          RAR
DEA3 1F          CLDBOT RAR
DEA4 DA00C7      JC       CCP
DEA7 C303C7      JMP      CCP+3          ;ENTER CP/M

DEAA 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.
*
*****
    
```

```

DEAB 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.
*
*****
    
```

```

COLDBEG DB      ''          ;COLD BOOT COMMAND
    
```

```

DEAC 00      COLDEND DB      0
              WARMBEG DB      ' '          ;WARM BOOT COMMAND GOES HERE
DEAD 00      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. *
*
*****
    
```

```

DEAE 310001  WBOOT  LXI      SP,TPA          ;SET UP STACK POINTER
DEB1 3E01    MVI      A,1
DEB2 =      WFLG   EQU      $-1            ;TEST IF BEGINNING OR
DEB3 A7      ANA      A                    ;      ENDING A WARM BOOT
DEB4 3E01    MVI      A,1
DEB6 32B2DE  STA      WFLG
DEB9 32AADE  STA      CWFLG              ;SET COLD/WARM BOOT FLAG
DEBC CA58DE  JZ       GOCPM
DEBF AF      XRA      A
DEC0 32B2DE  STA      WFLG
DEC3 4F      MOV      C,A

                IF      (MAXHD NE 0) AND FIRST ;SUPPLY WARM BOOT FROM HARD DISK ?
DEC4 2100C5  LXI      H,CCP-200H          ;INITIAL DMA ADDRESS
DEC7 E5      PUSH     H
DEC8 32F5E2  STA      HEAD
DECB 3E04    MVI      A,4
DECD F5      PUSH     PSW                ;SAVE FIRST SECTOR
DECE CDBAE1  CALL     HDDRV                ;SELECT DRIVE A
DED1 0E00    MVI      C,0
DED3 CDECE1  CALL     HDTRK                ;HOME THE DRIVE
DED6 F1      WARMLOD POP     PSW          ;RESTORE SECTOR
DED7 E1      POP      H                ;RESTORE DMA ADDRESS
DED8 3C      INR      A
DED9 32D9E2  STA      HDSECTR
DEDC FE10    CPI      16                ;PAST BDOS ?
DEDE CAAEDE  JZ       WBOOT            ;YES, ALL DONE
DEE1 24      INR      H                ;UPDATE DMA ADDRESS
DEE2 24      INR      H
DEE3 2250E2  SHLD    HDADD
DEE6 E5      PUSH     H
DEE7 F5      PUSH     PSW
DEE8 01000A  WARMRD  LXI      B,RETRIES*100H+0 ;RETRY COUNTER
DEEB C5      WRMREAD PUSH     B          ;SAVE THE RETRY COUNT
DEEC CD36E2  CALL     HDREAD                ;READ THE SECTOR
DEEF C1      POP      B
DEF0 D2D6DE  JNC     WARMLOD                ;TEST FOR ERROR
DEF3 05      DCR      B                ;UPDATE THE ERROR COUNT
DEF4 C2EBDE  JNZ     WRMREAD                ;KEEP TRYING IF NOT TO MANY ERRORS
DEF7 76      HLT
                ENDIF

                IF      (MAXFLOP NE 0) AND NOT FIRST ;SUPPLY WARM BOOT FROM 2D ?
                CALL    DJDRV          ;SELECT DRIVE A
    
```

```

MVI C,0 ;SELECT SINGLE DENSITY
CALL DJDEN
MVI C,0 ;SELECT SIDE 0
CALL DJSIDE
MVI A,15 ;INITIALIZE THE SECTOR TO READ
STA NEWSEC
LXI H,CCP-100H ;AND THE DMA ADDRESS
SHLD NEWDMA
CALL WARMLOD ;READ IN CP/M
LXI B,CCP+500H ;LOAD ADDRESS FOR REST OF WARM BOOT
CALL DJDMA
MVI C,8
CALL DJSEC
CALL WARMRD
JMP CCP+503H

WARMLOD MVI A,15 ;PREVIOUS SECTOR
NEWSEC EQU $-1
INR A ;UPDATE THE PREVIOUS SECTOR
INR A
CPI 27 ;WAS IT THE LAST ?
JC NOWRAP
SUI 9 ;YES
CPI 19
RZ
LHLD NEWDMA
LXI D,-480H
DAD D
SHLD NEWDMA
NOWRAP STA NEWSEC ;SAVE THE NEW SECTOR TO READ
MOV C,A
CALL DJSEC
LXI H,CCP-100H ;GET THE PREVIOUS DMA ADDRESS
NEWDMA EQU $-2
LXI D,100H ;UPDATE THE DMA ADDRESS
DAD D
SHLD NEWDMA ;SAVE THE DMA ADDRESS
MOV B,H
MOV C,L
CALL DJDMA ;SET THE DMA ADDRESS
CALL WARMRD
JMP WARMLOD

WARMRD LXI B,RETRIES*100H+0;MAXIMUM # OF ERRORS
WRMREAD PUSH B
CALL DJTRK ;SET THE TRACK
CALL DJREAD ;READ THE SECTOR
POP B
RNC ;CONTINUE IF SUCCESSFUL
DCR B
JNZ WRMREAD ;KEEP TRYING
JMP DJERR
ENDIF

```

*

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

```
DEF8 60 SETSEC MOV H,B
DEF9 69 MOV L,C
DEFA 22CEE4 SHLD CPMSEC
DEFD C9 DONOP RET
```

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

```
DEFE 60 SETDMA MOV H,B ;HL <- BC
DEFF 69 MOV L,C
DF00 22BEE0 SHLD CPMDMA ;CP/M DMA ADDRESS
DF03 C9 RET
```

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

```
DF04 0E00 HOME MVI C,0 ;TRACK TO SEEK TO
```

 * *
 * SETTRK SAVES THE TRACK # TO SEEK TO. NOTHING IS DONE AT THIS *
 * POINT, EVERYTHING IS DEFERRED UNTIL A READ OR WRITE. *
 * *

```
DF06 79 SETTRK MOV A,C ;A <- TRACK #
DF07 32D1E4 STA CPMTRK ;CP/M TRACK #
DF0A C9 RET
```

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

```
DF0B 3AD0E4 SECTRAN LDA (MAXHD NE 0) AND (MAXFLOP NE 0) ;BOTH TYPES ?
CPMDRV ;GET THE DRIVE NUMBER
```

```
DF0E FE03 IF FIRST
DF10 DA42DF CPI MAXHD*LOGDSK ;OVER THE # OF HARD DISKS ?
JC TRANHD
ELSE
CPI MAXFLOP ;OVER THE # OF FLOPPIES ?
JNC TRANHD
```


ENDIF
ENDIF

IF (MAXHD EQ 0) OR (MAXFLOP EQ 0) ;JUST ONE TYPE ?

SECTRAN EQU \$
ENDIF

```

DF13 03      TRANFP  IF      MAXFLOP NE 0      ;FLOPPY TRANSLATION
DF14 D5      TRANFP  INX      B
DF15 C5      TRANFP  PUSH     D                ;SAVE TABLE ADDRESS
DF16 CD42E0  TRANFP  PUSH     B                ;SAVE SECTOR #
DF19 7E      TRANFP  CALL     GETDPB          ;GET DPB ADDRESS INTO HL
DF1A B7      TRANFP  MOV      A,M           ;GET # OF CP/M SECTORS/TRACK
DF1B 1F      TRANFP  ORA      A                ;CLEAR CARY
DF1C 91      TRANFP  RAR      A                ;DIVIDE BY TWO
DF1D F5      TRANFP  SUB      C
DF1E FA2ADF  TRANFP  PUSH     PSW            ;SAVE ADJUSTED SECTOR
DF21 F1      SIDEA   POP      PSW            ;DISCARD ADJUSTED SECTOR
DF22 C1      SIDEA   POP      B                ;RESTORE SECTOR REQUESTED
DF23 D1      SIDEA   POP      D                ;RESTOR ADDRESS OF XLT TABLE
DF24 EB      SIDEONE XCHG    B                ;HL <- &(TRANSLATION TABLE)
DF25 09      SIDEONE DAD      B                ;BC = OFFSET INTO TABLE
DF26 6E      SIDEONE MOV      L,M           ;HL <- PHYSICAL SECTOR
DF27 2600    SIDEONE MVI     H,0
DF29 C9      SIDEONE RET
DF2A 010F00  SIDETWO LXI     B,15            ;OFFSET TO SIDE BIT
DF2D 09      SIDETWO DAD      B
DF2E 7E      SIDETWO MOV      A,M
DF2F E608    SIDETWO ANI     8                ;TEST FOR DOUBLE SIDED
DF31 CA21DF  SIDETWO JZ      SIDEA          ;MEDIA IS ONLY SINGLE SIDED
DF34 F1      SIDETWO POP      PSW            ;RETRIEVE ADJUSTED SECTOR
DF35 C1      SIDETWO POP      B
DF36 2F      SIDETWO CMA
DF37 3C      SIDETWO INR      A                ;MAKE SECTOR REQUEST POSITIVE
DF38 4F      SIDETWO MOV      C,A           ;MAKE NEW SECTOR THE REQUESTED SECTOR
DF39 D1      SIDETWO POP      D
DF3A CD24DF  SIDETWO CALL    SIDEONE
DF3D 3E80    SIDETWO MVI     A,80H          ;SIDE TWO BIT
DF3F B4      SIDETWO ORA      H                ; AND SECTOR
DF40 67      SIDETWO MOV      H,A
DF41 C9      SIDETWO RET
DF42 60      TRANHD  IF      MAXHD NE 0      ;HARD DISK TRANSLATION ROUTINE
DF43 69      TRANHD  MOV      H,B
DF44 23      TRANHD  MOV      L,C
DF45 C9      TRANHD  INX      H
                TRANHD  RET
                TRANHD  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.
*
*****

```

```

DF46 79      SETDRV  MOV     A,C           ;SAVE THE DRIVE #
DF47 32D0E4  STA     CPMDRV
DF4A FE05    CPI     MAXFLOP+(MAXHD*LOGDSK) ;CHECK FOR A VALID DRIVE #
DF4C D233E0  JNC     ZRET           ;ILLEGAL DRIVE #
DF4F 7B      MOV     A,E           ;TEST IF DRIVE EVER LOGGED IN BEFORE
DF50 E601    ANI     1
DF52 C21AE0  JNZ     SETDRV1        ;BIT 0 OF E = 0 -> NEVER SELECTED BEFORE

                IF     (MAXHD NE 0) AND (MAXFLOP NE 0) ;BOTH TYPES ?
DF55 3AD0E4  LDA     CPMDRV        ;GET THE DRIVE NUMBER

                IF     FIRST
DF58 FE03    CPI     MAXHD*LOGDSK ;OVER THE # OF HARD DISKS ?
DF5A DAD2DF  JC      DRVHD
DF5D D603    SUI     MAXHD*LOGDSK
                ELSE
                CPI     MAXFLOP ;OVER THE # OF FLOPPIES ?
                JNC     SUBFP
                ENDIF
                ENDIF

                IF     (MAXFLOP NE 0) AND FIRST
DF5F 4F      MOV     C,A           ;SAVE DRIVE #
DF60 3E00    MVI     A,0           ;HAVE THE FLOPPIES BEEN ACCESSED YET ?
DF61 =      FLOPFLG EQU    $-1
DF62 A7      ANA     A
DF63 C27DDF  JNZ     FLOPOK
DF66 0611    MVI     B,17           ;FLOPPIES HAVN'T BEEN ACCESSED
DF68 2100F4  LXI     H,DJBOOT        ;CHECK IF 2D CONTROLLER IS INSTALLED
DF6B 3EC3    MVI     A,(JMP)
DF6D BE      CLOPP  CMP     M
DF6E C233E0  JNZ     ZRET
DF71 05      DCR     B
DF72 C26DDF  JNZ     CLOPP
DF75 CD00F4  CALL    DJBOOT          ;INITIALIZE THE CONTROLLER
DF78 3E01    MVI     A,1           ;SAVE 2D INITIALIZED FLAG
DF7A 3261DF  STA     FLOPFLG
                ENDIF

                IF     MAXFLOP NE 0
DF7D 210100  FLOPOK LXI     H,1           ;SELECT SECTOR 1 OF TRACK 1
DF80 22D2E4  SHLD   TRUESEC
DF83 3E01    MVI     A,1
DF85 32D1E4  STA     CPMTRK
DF88 CD6FE1  CALL    FILL           ;FLUSH BUFFER AND REFILL
DF8B DA33E0  JC      ZRET           ;TEST FOR ERROR RETURN

```

```

DF8E CD27F4      CALL    DJSTAT      ;GET STATUS ON CURRENT DRIVE
DF91 E60C        ANI     0CH          ;STRIP OFF UNWANTED BITS
DF93 F5          PUSH    PSW          ;USED TO SELECT A DPB
DF94 1F          RAR
DF95 215BE0      LXI     H,XLTS        ;TABLE OF XLT ADDRESSES
DF98 5F          MOV     E,A
DF99 1600        MVI     D,0
DF9B 19          DAD     D
DF9C E5          PUSH    H          ;SAVE POINTER TO PROPER XLT
DF9D CD42E0      CALL    GETDPB       ;GET DPH POINTER INTO DE
DFA0 EB          XCHG
DFA1 D1          POP     D
DFA2 0602        MVI     B,2          ;NUMBER OF BYTES TO MOVE
DFA4 CDA5E1      CALL    MOVLOP       ;MOVE THE ADDRESS OF XLT
DFA7 110800      LXI     D,8          ;OFFSET TO DPB POINTER
DFAA 19          DAD     D          ;HL <- &DPH.DPB
DFAB E5          PUSH    H
DFAC 2A07F0      LHL    ORIGIN+7     ;GET ADDRESS OF DJ TERMINAL OUT ROUTINE
DFAF 23          INX     H          ;BUMP TO LOOK AT ADDRESS OF
                                ;      UART STATUS LOCATION

DFB0 7E          MOV     A,M
DFB1 EE03        XRI     3          ;ADJUST FOR PROPER REV DJ
DFB3 6F          MOV     L,A
DFB4 26F3        MVI     H,(ORIGIN+300H)/100H
DFB6 7E          MOV     A,M
DFB7 E608        ANI     DBLSID       ;CHECK DOUBLE SIDED BIT
DFB9 11CEE3      LXI     D,DPB128S   ;BASE FOR SINGLE SIDED DPB'S
DFBC C2C2DF      JNZ    SIDEOK
DFBF 110EE4      LXI     D,DPB128D   ;BASE OF DOUBLE SIDED DPB'S
DFC2 EB          XCHG       ;HL <- DBP BASE, DE <- &DPH.DPB
DFC3 D1          POP     D          ;RESTORE DE (POINTER INTO DPH)
DFC4 F1          POP     PSW        ;OFFSET TO CORRECT DPB
DFC5 17          RAL
DFC6 17          RAL
DFC7 4F          MOV     C,A
DFC8 0600        MVI     B,0
DFCA 09          DAD     B
DFCB EB          XCHG       ;PUT DPB ADDRESS IN DPH
DFCC 73          MOV     M,E
DFCD 23          INX     H
DFCE 72          MOV     M,D
                                ENDIF

DFCF C31AE0      IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
                                JMP    SETDRV1      ;SKIP OVER THE HARD DISK SELECT
                                IF      NOT FIRST
SUBFP          SUI     MAXFLOP      ;ADJUST THE DRIVE #
                                ENDIF
                                ENDIF

DFD2 CD39E0      DRVHD  IF      MAXHD NE 0
DFD5 79          CALL    DIVLOG      ;DIVIDE BY LOGICAL DISKS PER DRIVE
DFD6 32FBE2      MOV     A,C
DFD9 CDE9E2      STA    HDDISK
DFDC 7E          CALL    DRVPTR
                                MOV     A,M

```

```

CP/M MACRO ASSEM 2.0      #020      *** Cbios For CP/M Ver. 2.2 ***

DFDD 3C                   INR       A
DFDE C21AE0              JNZ       SETDRV1
DFE1 F6FC                ORI       NULL           ;SELECT DRIVE
DFE3 D352                OUT      HDFUNC
DFE5 3E05                MVI     A,SCENBL       ;ENABLE THE CONTROLLER
DFE7 D350                OUT      HDCNTL
DFE9 0EEF                MVI     C,239         ;WAIT APPROX 2 MINUTES FOR DISK TO READY
DFEB 210000              LXI     H,0
DFEE 2B                   TDELAY DCX      H
DFEF 7C                   DFEF   MOV      A,H
DFF0 B5                   DFF0   ORA     L
DFF1 CC37E0              DFF1   CZ      DCRC
DFF4 C8                   DFF4   RZ
DFF5 DB50                DFF5   IN      HDSTAT       ;TEST IF READY YET
DFF7 E620                DFF7   ANI     DRVRDY
DFF9 C2EEDF              DFF9   JNZ     TDELAY

                                IF      SDELAY
DFFC 210000              DFC   LXI     H,0           ;TIME ONE REVOLUTION OF THE DRIVE
DFFF 0E40                DFFF   MVI     C,INDEX
E001 DB50                E001   IN      HDSTAT
E003 A1                  E003   ANA     C
E004 47                   E004   MOV     B,A           ;SAVE CURRENT INDEX LEVEL IN B
E005 DB50                E005   IN      HDSTAT
E007 A1                  E007   ANA     C
E008 B8                   E008   CMP     B           ;LOOP UTIL INDEX LEVEL CHANGES
E009 CA05E0              E009   JZ      INDX1
E00C 23                   E00C   INX     H
E00D DB50                E00D   IN      HDSTAT       ;START COUNTING UNTIL INDEX RETURNS TO
E00F A1                  E00F   ANA     C           ;      PREVIOUS STATE
E010 B8                   E010   CMP     B
E011 C20CE0              E011   JNZ     INDX2
                                IF      M10
                                DAD     H
                                ENDF
E014 22E1E1              E014   SHLD   SETTLE       ;SAVE THE COUNT FOR TIMEOUT DELAY
                                ENDF
E017 CDCBE1              E017   CALL   HDHOME
                                ENDF

SETDRV1 CALL GETDPB       ;GET ADDRESS OF DPB IN HL
E01A CD42E0              E01A   LXI     B,15       ;OFFSET TO SECTOR SIZE
E01D 010F00              E01D   DAD     B
E020 09                   E020   MOV     A,M           ;GET SECTOR SIZE
E021 7E                   E021   ANI     7H
E022 E607                E022   STA     SECSIZ
E024 326FE0              E024   MOV     A,M
E027 7E                   E027   RAR
E028 1F                   E028   RAR
E029 1F                   E029   RAR
E02A 1F                   E02A   RAR
E02B 1F                   E02B   RAR
E02C E60F                E02C   ANI     0FH
E02E 32ADE0              E02E   STA     SECPSEC
E031 EB                   E031   XCHG       ;HL <- DPH
E032 C9                   E032   RET

```

E033 210000 ZRET LXI H,0 ;SELDRV ERROR EXIT
E036 C9 RET

E037 0D DCRC IF MAXHD NE 0
E038 C9 DCR C ;CONDITIONAL DECREMENT C ROUTINE
RET

E039 0E00 DIVLOG MVI C,0
E03B D603 DIVLOGX SUI LOGDSK
E03D D8 RC
E03E 0C INR C
E03F C33BE0 JMP DIVLOGX
ENDIF

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

E042 3AD0E4 GETDPB LDA CPMDRV
E045 6F MOV L,A ;FORM OFFSET
E046 2600 MVI H,0
E048 29 DAD H
E049 29 DAD H
E04A 29 DAD H
E04B 29 DAD H
E04C 117EE4 LXI D,DPBASE ;BASE OF DPH'S
E04F 19 DAD D
E050 E5 PUSH H ;SAVE ADDRESS OF DPH
E051 110A00 LXI D,10 ;OFFSET TO DPB
E054 19 DAD D
E055 7E MOV A,M ;GET LOW BYTE OF DPB ADDRESS
E056 23 INX H
E057 66 MOV H,M ;GET LOW BYTE OF DPB
E058 6F MOV L,A
E059 D1 POP D
E05A C9 RET

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

E05B 00E3 XLTS IF MAXFLOP NE 0
E05D 1BE3 DW XLT128 ;XLT FOR 128 BYTE SECTORS
E05F 50E3 DW XLT256 ;XLT FOR 256 BYTE SECTORS
E061 8DE3 DW XLT512 ;XLT FOR 512 BYTE SECTORS
DW XLT124 ;XLT FOR 1024 BYTE SECTORS
ENDIF

*

```

* WRITE ROUTINE MOVES DATA FROM MEMORY INTO THE BUFFER. IF THE *
* DESIRED CP/M SECTOR IS NOT CONTAINED IN THE DISK BUFFER, THE *
* BUFFER IS FIRST FLUSHED TO THE DISK IF IT HAS EVER BEEN *
* WRITTEN INTO, THEN A READ IS PERFORMED INTO THE BUFFER TO GET *
* THE DESIRED SECTOR. ONCE THE CORRECT SECTOR IS IN MEMORY, THE *
* BUFFER WRITTEN INDICATOR IS SET, SO THE BUFFER WILL BE *
* FLUSHED, THEN THE DATA IS TRANSFERRED INTO THE BUFFER. *
*
*****

```

```

E063 79      WRITE  MOV     A,C           ;SAVE WRITE COMMAND TYPE
E064 32D5E0  STA     WRITTYP
E067 3E01    MVI     A,1           ;SET WRITE COMMAND
E069 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. *
*
*****

```

```

E06A AF      READ   XRA     A           ;SET THE COMMAND TYPE TO READ
E06B 32C1E0  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. *
*
*****

```

```

E06E 0600    REDWRT MVI     B,0           ;THE 0 IS MODIFIED TO CONTAIN THE LOG2
E06F =       SECSIZ EQU     $-1          ; OF THE PHYSICAL SECTOR SIZE/128
                                           ; ON THE CURRENTLY SELECTED DISK.
E070 2ACEE4  LHL   CPMSEC        ;GET THE DESIRED CP/M SECTOR #
E073 7C      MOV     A,H
E074 E680    ANI     80H           ;SAVE ONLY THE SIDE BIT
E076 4F      MOV     C,A           ;REMEMBER THE SIDE
E077 7C      MOV     A,H
E078 E67F    ANI     7FH           ;FORGET THE SIDE BIT
E07A 67      MOV     H,A
E07B 2B      DCX     H           ;TEMPORARY ADJUSTMENT
E07C 05      DIVLOOP DCR     B           ;UPDATE REPEAT COUNT
E07D CABAE0  JZ      DIVDONE
E080 B7      ORA     A

```

```

E081 7C      MOV      A,H
E082 1F      RAR
E083 67      MOV      H,A
E084 7D      MOV      A,L
E085 1F      RAR                      ;DIVIDE THE CP/M SECTOR # BY THE SIZE
                                           ; OF THE PHYSICAL SECTORS

E086 6F      MOV      L,A
E087 C37CE0  JMP      DIVLOOP
E08A 23      DIVDONE INX      H
E08B 7C      MOV      A,H
E08C B1      ORA      C                      ;RESTORE THE SIDE BIT
E08D 67      MOV      H,A
E08E 22D2E4  SHLD   TRUESEC          ;SAVE THE PHYSICAL SECTOR NUMBER
E091 21D0E4  LXI    H,CPMDRV         ;POINTER TO DESIRED DRIVE,TRACK, AND SECTOR
E094 11D4E4  LXI    D,BUFDRV        ;POINTER TO BUFFER DRIVE,TRACK, AND SECTOR
E097 0605    MVI    B,5              ;COUNT LOOP
E099 05      DTSLOP DCR      B          ;TEST IF DONE WITH COMPARE
E09A CAASE0  JZ      MOVE           ;YES, MATCH. GO MOVE THE DATA
E09D 1A      LDAX   D          ;GET A BYTE TO COMPARE
E09E BE      CMP      M          ;TEST FOR MATCH
E09F 23      INX      H          ;BUMP POINTERS TO NEXT DATA ITEM
E0A0 13      INX      D
E0A1 CA99E0  JZ      DTSLOP        ;MATCH, CONTINUE TESTING
    
```

```

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

```

E0A4 CD6FE1  CALL   FILL           ;FILL THE BUFFER WITH CORRECT PHYSICAL SECTOR
E0A7 D8      RC          ;NO GOOD, RETURN WITH ERROR INDICATION
    
```

```

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

```

E0A8 3ACEE4  MOVE   LDA      CPMSEC          ;GET THE CP/M SECTOR TO TRANSFER
E0AB 3D      DCR      A          ;ADJUST TO PROPER SECTOR IN BUFFER
E0AC E600    ANI      0              ;STRIP OFF HIGH ORDERED BITS
E0AD =      SECPSEC EQU    $-1    ;THE 0 IS MODIFIED TO REPRESENT THE # OF
                                           ; CP/M SECTORS PER PHYSICAL SECTORS

E0AE 6F      MOV      L,A          ;PUT INTO HL
E0AF 2600    MVI      H,0
E0B1 29      DAD      H          ;FORM OFFSET INTO BUFFER
E0B2 29      DAD      H
E0B3 29      DAD      H
E0B4 29      DAD      H
E0B5 29      DAD      H
E0B6 29      DAD      H
E0B7 29      DAD      H
E0B8 11D8E4  LXI    D,BUFFER          ;BEGINNING ADDRESS OF BUFFER
    
```

```

CP/M MACRO ASSEM 2.0      #024      *** Cbios For CP/M Ver. 2.2 ***

E0BB 19                   DAD      D                ;FORM BEGINNING ADDRESS OF SECTOR TO TRANSFER
E0BC EB                   XCHG     ;DE = ADDRESS IN BUFFER
E0BD 210000               LXI     H,0            ;GET DMA ADDRESS, THE 0 IS MODIFIED TO
                                ;          CONTAIN THE DMA ADDRESS

E0BE =                    CPMDMA EQU     $-2
E0C0 3E00                 MVI     A,0            ;THE ZERO GETS MODIFIED TO CONTAIN
                                ;          A ZERO IF A READ, OR A 1 IF WRITE

E0C1 =                    RDWR    EQU     $-1
E0C2 A7                   ANA     A                ;TEST WHICH KIND OF OPERATION
E0C3 C2CBE0               JNZ     INTO           ;TRANSFER DATA INTO THE BUFFER
E0C6 CDA3E1               OUTOP  CALL    MOVER
E0C9 AF                   XRA     A
E0CA C9                   RET

E0CB EB                   INTO    XCHG           ;
E0CC CDA3E1               CALL    MOVER         ;MOVE THE DATA, HL = DESTINATION
                                ;          DE = SOURCE

E0CF 3E01                 MVI     A,1
E0D1 32DEE0               STA     BUFWRN        ;SET BUFFER WRITTEN INTO FLAG
E0D4 3E00                 MVI     A,0            ;CHECK FOR DIRECTORY WRITE
E0D5 =                    WRITTYP EQU  $-1
E0D6 3D                   DCR     A
E0D7 3E00                 MVI     A,0
E0D9 32D5E0               STA     WRITTYP       ;SET NO DIRECTORY WRITE
E0DC C0                   RNZ


```

```

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

```

```

E0DD 3E00                 FLUSH  MVI     A,0            ;THE 0 IS MODIFIED TO REFLECT IF
                                ;          THE BUFFER HAS BEEN WRITTEN INTO

E0DE =                    BUFWRN EQU  $-1
E0DF A7                   ANA     A                ;TEST IF WRITTEN INTO
E0E0 C8                   RZ

                                ;NOT WRITTEN, ALL DONE

                                IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E0E1 2118F4               LXI     H,DJWRITE        ;WRITE OPERATION FOR DISK JOCKEY
E0E4 116BE2               LXI     D,HDWRITE        ;WRITE OPERATION FOR HARD DISK
E0E7 CDB2E1               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. *
 * *

```

E0EA AF      PREP      XRA      A              ;RESET BUFFER WRITTEN FLAG
E0EB 32DEE0  STA      BUFWRN
E0EE 2250E1  SHLD     RETRYOP      ;SET UP THE READ/WRITE OPERATION
E0F1 060A    MVI      B,RETRIES    ;MAXIMUM NUMBER OF RETRIES TO ATTEMPT
E0F3 C5      RETRYLP  PUSH     B              ;SAVE THE RETRY COUNT
E0F4 3AD4E4  LDA      BUFDRV      ;GET DRIVE NUMBER INVOLVED IN THE OPERATION
  
```

```

                                IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
                                IF      FIRST
E0F7 FE03    CPI      MAXHD*LOGDSK
E0F9 DAFEE0  JC       NOADJST
E0FC D603    SUI      MAXHD*LOGDSK
                                ELSE
                                CPI      MAXFLOP
                                JC       NOADJST
                                SUI      MAXFLOP
                                ENDIF
  
```

```

E0FE 4F      NOADJST  MOV      C,A
E0FF 2133DD  LXI      H,DJDRV      ;SELECT DRIVE
E102 11BAE1  LXI      D,HDDRV
E105 CDAEE1  CALL     DECIDGO
                                ELSE
                                MOV      C,A
                                IF      MAXHD NE 0
                                CALL     HDDRV
                                ENDIF
                                IF      MAXFLOP NE 0
                                CALL     DJDRV      ;SELECT THE DRIVE
                                ENDIF
                                ENDIF
  
```

```

E108 3AD5E4  LDA      BUFTRK
E10B A7      ANA      A              ;TEST FOR TRACK ZERO
E10C 4F      MOV      C,A
E10D C5      PUSH     B
  
```

```

                                IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E10E 2109F4  LXI      H,DJHOME
E111 11CBE1  LXI      D,HDHOME
E114 CCAEE1  CZ       DECIDGO
                                ELSE
                                IF      MAXHD NE 0
                                CZ       HDHOME
                                ENDIF
                                IF      MAXFLOP NE 0
                                CZ       DJHOME      ;HOME THE DRIVE IF TRACK 0
                                ENDIF
                                ENDIF
  
```

```

E117 C1      POP      B              ;RESTORE TRACK #
  
```

```

E118 210CF4      IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E11B 11ECE1      LXI      H,DJTRK
E11E CDAEE1      LXI      D,HDTRK
                  CALL     DECIDGO
                  ELSE
                  IF      MAXHD NE 0
                  CALL     HDTRK
                  ENDIF
                  IF      MAXFLOP NE 0
                  CALL     DJTRK          ;SEEK TO PROPER TRACK
                  ENDIF
                  ENDIF

E121 2AD6E4      LHLD     BUFSEC
E124 7C          MOV      A,H          ;GET SECTOR INVOLVED IN OPERATION
E125 07          RLC          ;BIT 0 OF A EQUALS SIDE #
E126 E601        ANI      1          ;STRIP OFF UNNECESSARY BITS
E128 4F          MOV      C,A          ;C <- SIDE #

                  IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E129 2130F4      LXI      H,DJSIDE
E12C 1118E2      LXI      D,HDSIDE
E12F CDAEE1      CALL     DECIDGO
                  ELSE
                  IF      MAXHD NE 0
                  CALL     HDSIDE
                  ENDIF
                  IF      MAXFLOP NE 0
                  CALL     DJSIDE        ;SELECT THE SIDE
                  ENDIF
                  ENDIF

E132 2AD6E4      LHLD     BUFSEC
E135 7C          MOV      A,H
E136 E67F        ANI      7FH          ;STRIP OFF SIDE BIT
E138 47          MOV      B,A          ;C <- SECTOR #
E139 4D          MOV      C,L

                  IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E13A 210FF4      LXI      H,DJSEC
E13D 1121E2      LXI      D,HDSEC
E140 CDAEE1      CALL     DECIDGO
                  ELSE
                  IF      MAXHD NE 0
                  CALL     HDSEC
                  ENDIF
                  IF      MAXFLOP NE 0
                  CALL     DJSEC        ;SELECT THE SIDE
                  ENDIF
                  ENDIF

E143 01D8E4      LXI      B,BUFFER      ;SET THE DMA ADDRESS

E146 2112F4      IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E149 1113E2      LXI      H,DJDMA
                  LXI      D,HDDMA

```

```

E14C CDAEE1      CALL    DECIDGO
                ELSE
                IF      MAXHD NE 0
                CALL    HDDMA
                ENDIF
                IF      MAXFLOP NE 0
                CALL    DJDMA          ;SELECT THE SIDE
                ENDIF
                ENDIF

E14F CD0000      CALL    0          ;GET OPERATION ADDRESS
E150 =          RETRYOP EQU    $-2
E152 C1         POP     B          ;RESTORE THE RETRY COUNTER
E153 3E00       MVI    A,0        ;NO ERROR EXIT STATUS
E155 D0         RNC
E156 05         DCR     B          ;UPDATE THE RETRY COUNTER
E157 37         STC
E158 3EFF       MVI    A,0FFH     ;ERROR RETURN
E15A C8         RZ
E15B 78         MOV     A,B
E15C FE05       CPI    RETRIES/2
E15E C2F3E0     JNZ    RETRYLP     ;TRY AGAIN

E161 C5         PUSH    B
                IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E162 2109F4     LXI    H,DJHOME
E165 11CBE1     LXI    D,HDHOME
E168 CDAEE1     CALL    DECIDGO
                ELSE
                IF      MAXHD NE 0
                CALL    HDHOME
                ENDIF
                IF      MAXFLOP NE 0
                CALL    DJHOME     ;HOME THE DRIVE IF TRACK 0
                ENDIF
                ENDIF

E16B C1         POP     B
E16C C3F3E0     JMP    RETRYLP
    
```

```

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

```

E16F CDDDE0     FILL   CALL    FLUSH          ;FLUSH BUFFER FIRST
E172 D8         RC
E173 11D0E4     LXI    D,CPMDRV        ;CHECK FOR ERROR
                ;UP
    
```

```

dir
A: PIP          COM : -MY-BIOS DOC : MAC      COM : MOVCPMH  COM
A: DDT          COM : STAT      COM : SYSGEN  COM : WSMSGS  OVR
A: WSOVLY1     OVR
A>pip lst:=cbios6.prn [s#002]^z]q#003]
    
```

CP/M MACRO ASSEM 2.0

*
* 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, 1024 BYTES DOUBLE
* DENSITY. THERE ARE CONDITIONAL ASSEMBLIES FOR DISKUS HARD
* DISK CONTROLLER.
*

* WRITTEN BY BOBBY DALE GIFFORD.
* 12/8/80

* CUSTOMIZED BY JAY O'BRIEN
* 1/16/82

*
*
*
*
*
*
*

*

*
*
*

```

*
* DISK MAP OF SECTORS USED BY COLD BOOT, WARM BOOT, FIRMWARE,
* AND CP/M:
*
* TRK 0 SEC 1 = FIRST SECTOR OF COLD BOOT.          E700H
*          2 = COLD BOOT 256.                        80H
*          3 = COLD BOOT 512.                        80H
*          4 = COLD BOOT 1024.                       80H
*          5 = WARM BOOT 256.                        80H
*          6 = WARM BOOT 512.                        80H
*          7 = WARM BOOT 1024.                       80H
*          8 = COLD/WARM BOOT.                       2C00H
*          9 = FIRMWARE.                             E400H
*         10 = FIRMWARE+80H.                         E480H
*         11 = FIRMWARE+100H.                       E500H
*         12 = FIRMWARE+180H.                       E580H
*         13 = FIRMWARE+200H.                       E600H
*         14 = FIRMWARE+280H.                       E680H
*         15 = FIRMWARE+300H.                       E700H
*         16 = FIRMWARE+380H.                       E780H
*         17 = CCP.                                  2700H
*         18 = CCP+80H.                              2780H
*         20 = CCP+100H.                             2800H
*         22 = CCP+180H.                             2880H
*         24 = CCP+200H.                             2900H
*         26 = CCP+280H.                             2980H
*         28 = CCP+300H.                             2A00H
*         30 = CCP+380H.                             2A80H
*         32 = CCP+400H.                             2B00H
*         34 = CCP+480H.                             2B80H
*         36 = REST OF CP/M.                        2C00H-4FFFH
*
*****

```

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

ABORTED: CBIOS6.PRN
#002

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

```

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

```

```

*****
*
* THE FOLLOWING EQUATES SET UP THE RELATIONSHIP BETWEEN THE
*

```

```
E118 210CF4      IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E11B 11ECE1      LXI      H,DJTRK
E11E CDAEE1      LXI      D,HDTRK
                  CALL     DECIDGO
                  ELSE
                  IF      MAXHD NE 0
                  CALL     HDTRK
                  ENDIF
                  IF      MAXFLOP NE 0
                  CALL     DJTRK          ;SEEK TO PROPER TRACK
                  ENDIF
                  ENDIF

E121 2AD6E4      LHL D   BUFSEC
E124 7C          MOV     A,H          ;GET SECTOR INVOLVED IN OPERATION
E125 07          RLC          ;BIT 0 OF A EQUALS SIDE #
E126 E601       ANI     1          ;STRIP OFF UNNECESSARY BITS
E128 4F         MOV     C,A          ;C <- SIDE #

E129 2130F4      IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E12C 1118E2      LXI      H,DJSIDE
E12F CDAEE1      LXI      D,HDSIDE
                  CALL     DECIDGO
                  ELSE
                  IF      MAXHD NE 0
                  CALL     HDSIDE
                  ENDIF
                  IF      MAXFLOP NE 0
                  CALL     DJSIDE        ;SELECT THE SIDE
                  ENDIF
                  ENDIF

E132 2AD6E4      LHL D   BUFSEC
E135 7C          MOV     A,H
E136 E67F       ANI     7FH          ;STRIP OFF SIDE BIT
E138 47         MOV     B,A          ;C <- SECTOR #
E139 4D         MOV     C,L

E13A 210FF4      IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E13D 1121E2      LXI      H,DJSEC
E140 CDAEE1      LXI      D,HDSEC
                  CALL     DECIDGO
                  ELSE
                  IF      MAXHD NE 0
                  CALL     HDSEC
                  ENDIF
                  IF      MAXFLOP NE 0
                  CALL     DJSEC        ;SELECT THE SIDE
                  ENDIF
                  ENDIF

E143 01D8E4      LXI      B,BUFFER          ;SET THE DMA ADDRESS

E146 2112F4      IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E149 1113E2      LXI      H,DJDMA
                  LXI      D,HDDMA
```

```

E14C CDAEE1      CALL    DECIDGO
                  ELSE
                  IF      MAXHD NE 0
                  CALL    HDDMA
                  ENDIF
                  IF      MAXFLOP NE 0
                  CALL    DJDMA          ;SELECT THE SIDE
                  ENDIF
                  ENDIF

E14F CD0000      CALL    0          ;GET OPERATION ADDRESS
E150 =          RETRYOP EQU    $-2
E152 C1          POP     B          ;RESTORE THE RETRY COUNTER
E153 3E00        MVI     A,0        ;NO ERROR EXIT STATUS
E155 D0          RNC          ;RETURN NO ERROR
E156 05          DCR     B          ;UPDATE THE RETRY COUNTER
E157 37          STC          ;ASSUME RETRY COUNT EXPIRED
E158 3EFF        MVI     A,0FFH     ;ERROR RETURN
E15A C8          RZ
E15B 78          MOV     A,B
E15C FE05        CPI     RETRIES/2
E15E C2F3E0      JNZ     RETRYLP     ;TRY AGAIN

E161 C5          PUSH    B
                  IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E162 2109F4      LXI     H,DJHOME
E165 11CBE1      LXI     D,HDHOME
E168 CDAEE1      CALL    DECIDGO
                  ELSE
                  IF      MAXHD NE 0
                  CALL    HDHOME
                  ENDIF
                  IF      MAXFLOP NE 0
                  CALL    DJHOME     ;HOME THE DRIVE IF TRACK 0
                  ENDIF
                  ENDIF

E16B C1          POP     B
E16C C3F3E0      JMP     RETRYLP
    
```

```

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

```

E16F CDDDE0      FILL    CALL    FLUSH          ;FLUSH BUFFER FIRST
E172 D8          RC          ;CHECK FOR ERROR
E173 11D0E4      LXI     D,CPMDRV        ;UPDATE THE DRIVE, TRACK, AND SECTOR
E176 21D4E4      LXI     H,BUFDRV
E179 0604        MVI     B,4          ;NUMBER OF BYTES TO MOVE
E17B CDA5E1      CALL    MOVLOP        ;COPY THE DATA

E17E 3AC1E0      LDA     RDWR
E181 A7          ANA     A
E182 CA97E1      JZ      FREAD
    
```

```

E185 3AD5E0 LDA WRITTYP
E188 3D DCR A
E189 3D DCR A
E18A C8 RZ
E18B CD42E0 CALL GETDPB
E18E 110F00 LXI D,15
E191 19 DAD D
E192 7E MOV A,M
E193 E603 ANI 3
E195 3D DCR A
E196 C8 RZ

```

```

E197 = FREAD EQU $
IF (MAXHD NE 0) AND (MAXFLOP NE 0)
E197 2115F4 LXI H,DJREAD
E19A 1136E2 LXI D,HDREAD
E19D CDB2E1 CALL DECIDE
ELSE
IF MAXHD NE 0
LXI H,HDREAD
ENDIF
IF MAXFLOP NE 0
LXI H,DJREAD ;SELECT THE SIDE
ENDIF
ENDIF
E1A0 C3EAE0 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.
*
*****

```

```

E1A3 0680 MOVER MVI B,128 ;LENGTH OF TRANSFER
E1A5 1A MOVLOP LDAX D ;GET A BTE OF SOURCE
E1A6 77 MOV M,A ;MOVE IT
E1A7 13 INX D ;BUMP POINTERS
E1A8 23 INX H
E1A9 05 DCR B ;UPDATE COUNTER
E1AA C2A5E1 JNZ MOVLOP ;CONTINUE MOVING UNTIL DONE
E1AD C9 RET

```

```

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

```

```

E1AE CDB2E1 DECIDGO IF (MAXHD NE 0) AND (MAXFLOP NE 0)
E1B1 E9 CALL DECIDE ;WHICH CONTROLLER ?
PCHL
ENDIF
IF (MAXHD NE 0) AND (MAXFLOP NE 0)

```



```

E1B2 3AD4E4    DECIDE LDA    BUFDRV          ;GET PROPER ROUTINE INTO H&L, BASED
                IF      FIRST           ; ON CURRENTLY SELECTED DRIVE
E1B5 FE03      CPI      MAXHD*LOGDSK
E1B7 D0        RNC
                ELSE
                CPI      MAXFLOP
                RC
                ENDIF
E1B8 EB       XCHG
E1B9 C9       RET
                ENDIF
    
```

```

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

```

E1BA 79      HDDRV  IF      MAXHD NE 0
E1BB CD39E0  MOV      A,C          ;SELECT HARD DISK DRIVE
E1BE 79      CALL    DIVLOG       ;GET THE PHYSICAL DRIVE #
E1BF 32FBE2  MOV      A,C
E1C2 F6FC    STA      HDDISK      ;SELECT THE DRIVE
E1C4 D352    ORI      NULL
E1C6 3E0F    OUT     HDFUNC
E1C8 D350    MVI     A,WENABL
E1CA C9      OUT     HDCNTL
                RET
    
```

```

E1CB CDE9E2  HDHOME CALL    DRVPTR
E1CE 3600    MVI     M,0          ;SET TRACK TO ZERO
    
```

```

E1D0 DB50    STEPO  IF      SDELAY
E1D2 E601    IN      HDSTAT      ;TEST STATUS
E1D4 CAE0E1  ANI     TKZERO      ;AT TRACK ZERO ?
E1D7 3E01    JZ      DELAY
E1D9 37      MVI     A,1
E1DA CD00E2  STC
E1DD C3D0E1  CALL    ACCOK      ;TAKE ONE STEP OUT
                JMP     STEPO
    
```

ELSE

```

IN      HDSTAT
ANI     TKZERO
RZ
XRA    A
JMP    ACCOK
ENDIF
    
```

```

E1E0 210000  DELAY  IF      SDELAY
E1E1 =       SETTLE LXI     H,0          ;GET DELAY
E1E3 2B     DELOOP EQU     $-2
E1E4 7C     MOV     H          ;WAIT 20MS
                MOV     A,H
    
```

```

E1E5 B5      ORA      L
E1E6 23      INX      H
E1E7 2B      DCX      H
E1E8 C2E3E1  JNZ      DELOOP
E1EB C9      RET
                ENDIF

E1EC CDE9E2  HDTRK  CALL      DRVPTR      ;GET POINTER TO CURRENT TRACK
E1EF 5E      MOV      E,M          ;GET CURRENT TRACK
E1F0 71      MOV      M,C          ;UPDATE THE TRACK
E1F1 7B      MOV      A,E          ;NEED TO SEEK AT ALL ?
E1F2 91      SUB      C
E1F3 C8      RZ
E1F4 3F      CMC          ;GET CARRY INTO DIRECTION
E1F5 DAFAE1  JC      HDTRK2
E1F8 2F      CMA
E1F9 3C      INR      A
                IF      NOT SDELAY
                HDTRK2  JMP      ACCOK
                ELSE
E1FA CD00E2  HDTRK2  CALL      ACCOK
E1FD C3E0E1  JMP      DELAY
                ENDIF

E200 47      ACCOK  MOV      B,A          ;PREP FOR BUILD
E201 CDF4E2  CALL      BUILD
E204 E6FB      SLOOP  ANI      NSTEP      ;GET STEP PULSE LOW
E206 D352      OUT     HDFUNC      ;OUTPUT LOW STEP LINE
E208 F604      ORI     PSTEP      ;SET STEP LINE HIGH
E20A D352      OUT     HDFUNC      ;OUTPUT HIGH STEP LINE
E20C 05      DCR     B          ;UPDATE REPEAT COUNT
E20D C204E2  JNZ     SLOOP      ;KEEP GOING THE REQUIRED # OF TRACKS
E210 C319E2  JMP     WSDONE

E213 60      HDDMA  MOV     H,B          ;SAVE THE DMA ADDRESS
E214 69      MOV     L,C
E215 2250E2  SHLD   HDADD
E218 =      HDSECT EQU     $
E218 C9      RET

E219 DB50      WSDONE  IN      HDSTAT      ;WAIT FOR SEEK COMPLETE TO FINISH
E21B E604      ANI     COMPLT
E21D CA19E2  JZ      WSDONE
E220 C9      RET

                IF     M26
E221 3E1F      HDSECT  MVI     A,01FH      ;FOR COMPATIBILITY WITH CBIOS REV 2.3, 2.4
E223 A1      ANA     C
E224 CC33E2  CZ      GETSPT
E227 32D9E2  STA     HDSECTR
E22A 3EE0      MVI     A,0E0H
E22C A1      ANA     C
E22D 07      RLC
E22E 07      RLC
E22F 07      RLC
E230 32F5E2  STA     HEAD

```

```

E233 3E20      GETSPT MVI      A,HDSPT
E235 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,0
                DIVSPTX SUI     HDSPT
                RC
                INR    C
                JMP    DIVSPTX
                ENDF

E236 CDB4E2    HDREAD  CALL   HDPREP
E239 D8        RC
E23A AF        XRA     A
E23B D351      OUT     HDCMND
E23D 2F        CMA
E23E D353      OUT     HDDATA
E240 D353      OUT     HDDATA
E242 3E01      MVI     A,RSECT      ;READ SECTOR COMMAND
E244 D351      OUT     HDCMND
E246 CD9AE2    CALL   PROCESS
E249 D8        RC
E24A AF        XRA     A
E24B D351      OUT     HDCMND
E24D 0680      MVI     B,SECLN/4
E24F 210000    LXI     H,0
E250 =         HDADD  EQU     $-2
E252 DB53      IN      HDDATA
E254 DB53      IN      HDDATA
E256 DB53      RTLOOP  IN      HDDATA      ;MOVE FOUR BYTES
E258 77        MOV     M,A
E259 23        INX    H
E25A DB53      IN      HDDATA
E25C 77        MOV     M,A
E25D 23        INX    H
E25E DB53      IN      HDDATA
E260 77        MOV     M,A
E261 23        INX    H
E262 DB53      IN      HDDATA
E264 77        MOV     M,A
E265 23        INX    H
E266 05        DCR    B
E267 C256E2    JNZ    RTLOOP
    
```

```

E26A C9          RET

E26B CDB4E2     HDWRITE CALL  HDPREP          ;PREPARE HEADER
E26E D8         RC
E26F AF         XRA      A
E270 D351       OUT      HDCMND
E272 2A50E2     LHLD     HDADD
E275 0680       MVI      B,SECLN/4
E277 7E         WTLOOP  MOV      A,M          ;MOVE 4 BYTES
E278 D353       OUT      HDDATA
E27A 23         INX      H
E27B 7E         MOV      A,M
E27C D353       OUT      HDDATA
E27E 23         INX      H
E27F 7E         MOV      A,M
E280 D353       OUT      HDDATA
E282 23         INX      H
E283 7E         MOV      A,M
E284 D353       OUT      HDDATA
E286 23         INX      H
E287 05         DCR      B
E288 C277E2     JNZ      WTLOOP
E28B 3E05       MVI      A,WSECT          ;ISSUE WRITE SECTOR COMMAND
E28D D351       OUT      HDCMND
E28F CD9AE2     CALL     PROCESS
E292 D8         RC
E293 3E10       MVI      A,WFAULT
E295 A0         ANA      B
E296 37         STC
E297 C8         RZ
E298 AF         XRA      A
E299 C9         RET

E29A DB50       PROCESS IN   HDSTAT          ;WAIT FOR COMMAND TO FINISH
E29C 47         MOV      B,A
E29D E602       ANI      OPDONE
E29F CA9AE2     JZ      PROCESS
E2A2 3E07       MVI      A,DSKCLK
E2A4 D350       OUT      HDCNTL
E2A6 DB50       IN       HDSTAT
E2A8 E608       ANI      TMOUT          ;TIMED OUT ?
E2AA 37         STC
E2AB C0         RNZ
E2AC DB51       IN       HDRESLT
E2AE E602       ANI      RETRY          ;ANY RETRIES ?
E2B0 37         STC
E2B1 C0         RNZ
E2B2 AF         XRA      A
E2B3 C9         RET

E2B4 DB50       HDPREP  IN   HDSTAT
E2B6 E620       ANI      DRVRDY
E2B8 37         STC
E2B9 C0         RNZ
E2BA 3E08       MVI      A,ISBUFF          ;INITIALIZE POINTER
E2BC D351       OUT      HDCMND

```

```

E2BE CDF4E2      CALL    BUILD
E2C1 F60C        ORI     0CH
E2C3 D352        OUT     HDFUNC
E2C5 3AF5E2      LDA     HEAD
E2C8 D353        OUT     HDDATA          ;FORM HEAD BYTE
E2CA CDE9E2      CALL    DRVPTR
E2CD 7E          MOV     A,M          ;FORM TRACK BYTE
E2CE D353        OUT     HDDATA
E2D0 A7          ANA     A
E2D1 0680        MVI     B,80H
E2D3 CAD8E2      JZ     ZKEY
E2D6 0600        MVI     B,0
E2D8 3E00        MVI     A,0          ;FORM SECTOR BYTE
E2D9 =          HDSECTR EQU    $-1
E2DA D353        OUT     HDDATA
E2DC 78          MOV     A,B
E2DD D353        OUT     HDDATA
E2DF 3E07        MVI     A,DSKCLK
E2E1 D350        OUT     HDCNTL
E2E3 3E0F        MVI     A,WENABL
E2E5 D350        OUT     HDCNTL
E2E7 AF          XRA     A
E2E8 C9          RET

E2E9 2AFBE2      DRVPTR LHL    HDDISK
E2EC EB          XCHG
E2ED 1600        MVI     D,0
E2EF 21FFE2      LXI     H,DRIVES
E2F2 19          DAD     D
E2F3 C9          RET

E2F4 3E00        BUILD  MVI     A,0
E2F5 =          HEAD   EQU    $-1
E2F6 17          RAL
E2F7 17          RAL
E2F8 17          RAL
E2F9 17          RAL
E2FA F600        HDDISK ORI     0
E2FB =          EQU    $-1
E2FC EEF0        XRI     0FH
E2FE C9          RET

E2FF =          DRIVES EQU    $
                      REPT  MAXHD
                      DB     0FFH
                      ENDM

E2FF+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
E300 00	XLT128	DB	0
E301 01070D1319		DB	1,7,13,19,25
E306 050B1117		DB	5,11,17,23
E30A 03090F15		DB	3,9,15,21
E30E 02080E141A		DB	2,8,14,20,26
E313 060C1218		DB	6,12,18,24
E317 040A1016		DB	4,10,16,22
E31B 00	XLT256	DB	0
E31C 0102131425		DB	1,2,19,20,37,38
E322 0304151627		DB	3,4,21,22,39,40
E328 0506171829		DB	5,6,23,24,41,42
E32E 0708191A2B		DB	7,8,25,26,43,44
E334 090A1B1C2D		DB	9,10,27,28,45,46
E33A 0B0C1D1E2F		DB	11,12,29,30,47,48
E340 0D0E1F2031		DB	13,14,31,32,49,50
E346 0F10212233		DB	15,16,33,34,51,52
E34C 11122324		DB	17,18,35,36
E350 00	XLT512	DB	0
E351 0102030411		DB	1,2,3,4,17,18,19,20
E359 2122232431		DB	33,34,35,36,49,50,51,52
E361 0506070815		DB	5,6,7,8,21,22,23,24
E369 2526272835		DB	37,38,39,40,53,54,55,56
E371 090A0B0C19		DB	9,10,11,12,25,26,27,28
E379 292A2B2C39		DB	41,42,43,44,57,58,59,60
E381 0D0E0F101D		DB	13,14,15,16,29,30,31,32
E389 2D2E2F30		DB	45,46,47,48
E38D 00	XLT124	DB	0
E38E 0102030405		DB	1,2,3,4,5,6,7,8
E396 191A1B1C1D		DB	25,26,27,28,29,30,31,32
E39E 3132333435		DB	49,50,51,52,53,54,55,56
E3A6 090A0B0C0D		DB	9,10,11,12,13,14,15,16
E3AE 2122232425		DB	33,34,35,36,37,38,39,40
E3B6 393A3B3C3D		DB	57,58,59,60,61,62,63,64
E3BE 1112131415		DB	17,18,19,20,21,22,23,24
E3C6 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. *
 * * *

*

```

E3CE 1A00 DPB128S DW 26 ;CP/M SECTORS/TRACK
E3D0 03 DB 3 ;BSH
E3D1 07 DB 7 ;BLM
E3D2 00 DB 0 ;EXM
E3D3 F200 DW 242 ;DSM
E3D5 3F00 DW 63 ;DRM
E3D7 C0 DB 0C0H ;AL0
E3D8 00 DB 0 ;AL1
E3D9 1000 DW 16 ;CKS
E3DB 0200 DW 2 ;OFF
E3DD 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.
*

```

E3DE 3400 DPB256S DW 52 ;CP/M SECTORS/TRACK
E3E0 04 DB 4 ;BSH
E3E1 0F DB 15 ;BLM
E3E2 00 DB 0 ;EXM
E3E3 F200 DW 242 ;DSM
E3E5 7F00 DW 127 ;DRM
E3E7 C0 DB 0C0H ;AL0
E3E8 00 DB 0 ;AL1
E3E9 2000 DW 32 ;CKS
E3EB 0200 DW 2 ;OFF
E3ED 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.
*

```

E3EE 3C00 DPB512S DW 60 ;CP/M SECTORS/TRACK
E3F0 04 DB 4 ;BSH
E3F1 0F DB 15 ;BLM
E3F2 00 DB 0 ;EXM
E3F3 1801 DW 280 ;DSM
E3F5 7F00 DW 127 ;DRM
E3F7 C0 DB 0C0H ;AL0
E3F8 00 DB 0 ;AL1
E3F9 2000 DW 32 ;CKS
E3FB 0200 DW 2 ;OFF
E3FD 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.
*

```

E3FE 4000 DP1024S DW      64      ;CP/M SECTORS/TRACK
E400 04    DB      4          ;BSH
E401 0F    DB     15          ;BLM
E402 00    DB      0          ;EXM
E403 2B01  DW     299         ;DSM
E405 7F00  DW     127         ;DRM
E407 C0    DB    0C0H        ;AL0
E408 00    DB      0          ;AL1
E409 2000  DW     32         ;CKS
E40B 0200  DW      2         ;OFF
E40D 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.
*

```

E40E 3400 DPB128D DW      52      ;CP/M SECTORS/TRACK
E410 04    DB      4          ;BSH
E411 0F    DB     15          ;BLM
E412 01    DB      1          ;EXM
E413 F200  DW     242         ;DSM
E415 7F00  DW     127         ;DRM
E417 C0    DB    0C0H        ;AL0
E418 00    DB      0          ;AL1
E419 2000  DW     32         ;CKS
E41B 0200  DW      2         ;OFF
E41D 09    DB      9H

```

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

```

E41E 6800 DPB256D DW     104      ;CP/M SECTORS/TRACK
E420 04    DB      4          ;BSH
E421 0F    DB     15          ;BLM
E422 00    DB      0          ;EXM
E423 E601  DW     486         ;DSM
E425 FF00  DW     255         ;DRM

```



```

E427 F0      DB      0F0H      ;AL0
E428 00      DB      0        ;AL1
E429 4000    DW      64        ;CKS
E42B 0200    DW      2         ;OFF
E42D 1A      DB      1AH

```

```

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

```

```

E42E 7800    DPB512D DW      120      ;CP/M SECTORS/TRACK
E430 04      DB      4          ;BSH
E431 0F      DB      15         ;BLM
E432 00      DB      0          ;EXM
E433 3102    DW      561        ;DSM
E435 FF00    DW      255        ;DRM
E437 F0      DB      0F0H      ;AL0
E438 00      DB      0          ;AL1
E439 4000    DW      64        ;CKS
E43B 0200    DW      2         ;OFF
E43D 3B      DB      3BH

```

```

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

```

```

E43E 8000    DP1024D DW      128      ;CP/M SECTORS/TRACK
E440 04      DB      4          ;BSH
E441 0F      DB      15         ;BLM
E442 00      DB      0          ;EXM
E443 5702    DW      599        ;DSM
E445 FF00    DW      255        ;DRM
E447 F0      DB      0F0H      ;AL0
E448 00      DB      0          ;AL1
E449 4000    DW      64        ;CKS
E44B 0200    DW      2         ;OFF
E44D 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
E44E 0004    DPBHD1 DW      1024     ;CP/M SECTORS/TRACK
E450 05      DB      5          ;BSH

```

```

E451 1F          DB          31          ;BLM
E452 01          DB          1          ;EXM
E453 B507       DW          1973       ;DSM
E455 FF01       DW          511        ;DRM
E457 FF         DB          0FFH      ;AL0
E458 FF         DB          0FFH      ;AL1
E459 0000       DW          0          ;CKS
E45B 0100       DW          1          ;OFF
E45D 33         DB          33H        ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
;LOG2(#BYTES PER SECTOR/128) + 1 +
;8 IF DOUBLE SIDED.

E45E 0004       DPBHD2  DW          1024 ;CP/M SECTORS/TRACK
E460 05         DB          5          ;BSH
E461 1F         DB          31         ;BLM
E462 01         DB          1          ;EXM
E463 B507       DW          1973       ;DSM
E465 FF01       DW          511        ;DRM
E467 FF         DB          0FFH      ;AL0
E468 FF         DB          0FFH      ;AL1
E469 0000       DW          0          ;CKS
E46B 4000       DW          64         ;OFF
E46D 33         DB          33H        ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
;LOG2(#BYTES PER SECTOR/128) + 1 +
;8 IF DOUBLE SIDED.

E46E 0004       DPBHD3  DW          1024 ;CP/M SECTORS/TRACK
E470 05         DB          5          ;BSH
E471 1F         DB          31         ;BLM
E472 01         DB          1          ;EXM
E473 B507       DW          1973       ;DSM
E475 FF01       DW          511        ;DRM
E477 FF         DB          0FFH      ;AL0
E478 FF         DB          0FFH      ;AL1
E479 0000       DW          0          ;CKS
E47B 7F00       DW          127        ;OFF
E47D 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      0FFH    ;AL0
        DB      0FFH    ;AL1
        DW      0       ;CKS
        DW      98      ;OFF
        DB      33H     ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
                  ;LOG2(#BYTES PER SECTOR/128) + 1 +
                  ;8 IF DOUBLE SIDED.

DPBHD3  DW      672       ;CP/M SECTORS/TRACK
        DB      5        ;BSH
        DB      31       ;BLM
        DB      1        ;EXM
        DW      1028    ;DSM
        DW      511     ;DRM
        DB      0FFH    ;AL0
        DB      0FFH    ;AL1
        DW      0       ;CKS
        DW      195     ;OFF
        DB      33H     ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
                  ;LOG2(#BYTES PER SECTOR/128) + 1 +
                  ;8 IF DOUBLE SIDED.

      ENDIF
      ENDIF

```

```

*
* CP/M DISK PARAMETER HEADERS, UNINITIALIZED.
*
*****

```

```

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

```

```

E47E = DPBASE EQU $
0000 # DN SET 0
IF FIRST
REPT MAXHD ;GENERATE HARD DISK DPH'S FOLLOWED
HEADER %DN,DPBHD1 ; BY FLOPPY DPH'S
DN SET DN+1
HEADER %DN,DPBHD2
DN SET DN+1
IF (M26 NE 0) OR (M20 NE 0)
HEADER %DN,DPBHD3
DN SET DN+1
ENDIF
ENDM

```

```

E47E+0000 DW 0 ;TRANSLATION TABLE FILLED IN LATER
E480+0000000000 DW 0,0,0 ;SCRATCH
E486+D8E8 DW DIRBUF ;DIRECTORY BUFFER
E488+4EE4 DW DPBHD1 ;DPB FILLED IN LATER
E48A+4FEA DW CSV0 ;DIRECTORY CHECK VECTOR
E48C+58E9 DW ALV0 ;ALLOCATION VECTOR
E48E+0000 DW 0 ;TRANSLATION TABLE FILLED IN LATER
E490+0000000000 DW 0,0,0 ;SCRATCH
E496+D8E8 DW DIRBUF ;DIRECTORY BUFFER
E498+5EE4 DW DPBHD2 ;DPB FILLED IN LATER
E49A+46EB DW CSV1 ;DIRECTORY CHECK VECTOR
E49C+4FEA DW ALV1 ;ALLOCATION VECTOR
E49E+0000 DW 0 ;TRANSLATION TABLE FILLED IN LATER
E4A0+0000000000 DW 0,0,0 ;SCRATCH
E4A6+D8E8 DW DIRBUF ;DIRECTORY BUFFER
E4A8+6EE4 DW DPBHD3 ;DPB FILLED IN LATER
E4AA+3DEC DW CSV2 ;DIRECTORY CHECK VECTOR
E4AC+46EB DW ALV2 ;ALLOCATION VECTOR
REPT MAXFLOP
HEADER %DN,0
DN SET DN+1
ENDM

```

```

E4AE+0000 DW 0 ;TRANSLATION TABLE FILLED IN LATER
E4B0+0000000000 DW 0,0,0 ;SCRATCH
E4B6+D8E8 DW DIRBUF ;DIRECTORY BUFFER
E4B8+0000 DW 0 ;DPB FILLED IN LATER
E4BA+88EC DW CSV3 ;DIRECTORY CHECK VECTOR
E4BC+3DEC DW ALV3 ;ALLOCATION VECTOR
E4BE+0000 DW 0 ;TRANSLATION TABLE FILLED IN LATER

```

```

E4C0+000000000000 DW 0,0,0 ;SCRATCH
E4C6+D8E8 DW DIRBUF ;DIRECTORY BUFFER
E4C8+0000 DW 0 ;DPB FILLED IN LATER
E4CA+13ED DW CSV4 ;DIRECTORY CHECK VECTOR
E4CC+C8EC DW ALV4 ;ALLOCATION VECTOR
ELSE
REPT MAXFLOP ;GENERATE FLOPPY DPH'S FOLLOWED BY
HEADER %DN,0 ; HARD DISK DPH'S
DN SET DN+1
ENDM
REPT MAXHD
HEADER %DN,DPBHD1
DN SET DN+1
HEADER %DN,DPBHD2
DN SET DN+1
IF (M26 NE 0) OR (M20 NE 0)
HEADER %DN,DPBHD3
DN SET DN+1
ENDIF
ENDM
ENDIF
    
```

```

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

```

E4CE 0000 CPMSEC DW 0 ;CP/M SECTOR #
E4D0 00 CPMDRV DB 0 ;CP/M DRIVE #
E4D1 00 CPMTRK DB 0 ;CP/M TRACK #
E4D2 0000 TRUESEC DW 0 ;DISK JOCKEY SECTOR THAT CONTAINS CP/M SECTOR
E4D4 00 BUFDRV DB 0 ;DRIVE THAT BUFFER BELONGS TO
E4D5 00 BUFTRK DB 0 ;TRACK THAT BUFFER BELONGS TO
E4D6 0000 BUFSEC DW 0 ;SECTOR THAT BUFFER BELONGS TO
E4D8 = BUFFER EQU $
    
```

```

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

```

HEXNUM MACRO NUM
IF (NUM/16) > 9
DB (NUM/16 AND 0FH) + 'A' - 10
ELSE
DB (NUM/16 AND 0FH) + '0'
ENDIF
IF (NUM AND 0FH) > 9
DB (NUM AND 0FH) + 'A' - 10
ELSE
DB (NUM AND 0FH) + '0'
ENDIF
ENDM
    
```

```

CP/M MACRO ASSEM 2.0 #042 *** Cbios For CP/M Ver. 2.2 ***

E4D8 0D0A0A PROMPT DB ACR,ALF,ALF
E4DB 4D6F72726F DB 'Morrow Designs '
E4EA 36 DB '0'+MSIZE/10 ;CP/M MEMORY SIZE
E4EB 30 DB '0'+(MSIZE MOD 10)
E4EC 4B2043502F DB 'K CP/M ' ;CP/M VERSION NUMBER
E4F3 32 DB CPMREV/10+'0'
E4F4 2E DB '.'
E4F5 32 DB (CPMREV MOD 10)+'0'
E4F6 2C20436269 DB ', Cbios rev '
E502 322E DB REVNUM/10+'0', '.' ;CBIOS REVISION NUMBER
E504 38 DB REVNUM MOD 10+'0'
IF MAXHD NE 0
E505 2E DB '.'
E506 32 DB MREV/10+'0'
E507 36 DB MREV MOD 10+'0'
IF (M10 OR M20) AND SDELAY
DB 'M'
ENDIF
IF (M10 OR M20) AND NOT SDELAY
DB 'F'
ENDIF
ENDIF

E508 0D0A DB ACR,ALF
E50A 466F7220 DB 'For '

IF MAXFLOP NE 0
E50E 6120446973 DB 'a Disk Jockey 2D @ '
HEXNUM %(ORIGIN/256)
E521+46 DB (240/16 AND 0FH) + 'A' - 10
E522+30 DB (240 AND 0FH) + '0'
E523 30304820 DB '00H '
ENDIF

IF (MAXHD NE 0) AND (MAXFLOP NE 0)
E527 616E6420 DB 'and '
ENDIF

IF MAXHD NE 0
IF MAXHD EQ 1
E52B 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

```

```

E52E 4D323620      IF      MREV EQ 26
                   DB      'M26 '
                   ENDIF
E532 6861726420    DB      'hard disk'
                   IF      MAXHD NE 1
                   DB      's'
                   ENDIF
E53B 204020        DB      ' @ '
                   HEXNUM  %HDORG
E53E+35           DB      (80/16 AND 0FH) + '0'
E53F+30           DB      (80 AND 0FH) + '0'
E540 482E         DB      'H.'
                   ENDIF
E542 0D0A         DB      ACR,ALF
E544 0D0A         DB      ACR,ALF
E546 2020202020   DB      ' THE W6GO/K6HHD LIST'
E560 0D0A         DB      ACR,ALF
E562 2020202020   DB      ' Electronics Enterprises'
E57E 0D0A         DB      ACR,ALF
E580 2020202020   DB      ' Rio Linda, California'
E59B 0D0A         DB      ACR,ALF
E59D 00           DB      0
    
```

```

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

```

E59E 7E          MESSAGE MOV      A,M          ;GET A CHARACTER OF THE MESSAGE
E59F 23          INX           H              ;BUMP TEXT POINTER
E5A0 A7          ANA           A              ;TEST FOR END
E5A1 C8          RZ            ;RETURN IF DONE
E5A2 E5          PUSH          H              ;SAVE POINTER TO TEXT
E5A3 4F          MOV           C,A           ;OUTPUT CHARACTER IN C
E5A4 CD0CDD     CALL          COUT          ;OUTPUT THE CHARACTER
E5A7 E1          POP           H              ;RESTORE THE POINTER
E5A8 C39EE5     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.
*
*****
    
```

```

E5AB 310001     CBOOT  LXI      SP,TPA        ;SET UP STACK
E5AE 3EC0       MVI      A,INTIOBY
E5B0 320300     STA      IOBYTE
E5B3 CD38DE     CALL     TINIT          ;INITIALIZE THE TERMINAL
E5B6 21D8E4     LXI      H,PROMPT      ;PREP FOR SENDING SIGNON MESSAGE
E5B9 CD9EE5     CALL     MESSAGE       ;SEND THE PROMPT
E5BC AF        XRA      A              ;SELECT DISK A
    
```



```

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,252,0
DN SET DN+1
ALLOC %DN,129,0
DN SET DN+1
ENDIF
ENDM
E958+ ALV0 DS 247
EA4F+ CSV0 DS 0
EA4F+ ALV1 DS 247
EB46+ CSV1 DS 0
EB46+ ALV2 DS 247
EC3D+ CSV2 DS 0
REPT MAXFLOP
ALLOC %DN,75,64
DN SET DN+1
ENDM
EC3D+ ALV3 DS 75
EC88+ CSV3 DS 64
ECC8+ ALV4 DS 75
ED13+ CSV4 DS 64
ENDIF
ED53 END

```

0006 AACK	E200 ACCOK	000D ACR	0003 AETX	000A ALF
E958 ALV0	EA4F ALV1	EB46 ALV2	EC3D ALV3	ECC8 ALV4
DEAB AUTOFLG	CF00 BDOS	A000 BIAS	DD00 BIOS	E4D4 BUFDRV
0080 BUFF	E4D8 BUFFER	E4D6 BUFSEC	E4D5 BUFTRK	E0DE BUFWRN
E2F4 BUILD	E5AB CBOOT	C700 CCP	0004 CDISK	DE08 CICRT
DE08 CIPTR	DD88 CITBLE	F003 CTTY	DDF3 CIUC1	DE08 CIUR1
DE08 CIUR2	DEA3 CLDBOT	DE8E CLDCMND	001A CLEAR	DF6D CLOPP
DE25 COCRT	DEAC COLDBEG	DEAC 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	E0BE CPMDMA
E4D0 CPMDRV	0016 CPMREV	E4CE CPMSEC	E4D1 CPMTRK	DE1C CSCRT
DE1C CSPTR	DD3C CSREADR	DDB8 CSRTBLE	DDB0 CSTBLE	DE14 CSTTY
DDFF CSUC1	DE1C CSUR1	DE1C CSUR2	EA4F CSV0	EB46 CSV1
EC3D CSV2	EC88 CSV3	ED13 CSV4	DEAA CWFLG	0008 DBLSID
E037 DCRC	E1B2 DECIDE	E1AE DECIDGO	E1E0 DELAY	E1E3 DELOOP
E8D8 DIRBUF	E08A DIVDONE	E039 DIVLOG	E03B DIVLOGX	E07C 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	DEFD DONOP	E43E DP1024D	E3FE DP1024S
E40E DPB128D	E3CE DPB128S	E41E DPB256D	E3DE DPB256S	E42E DPB512D
E3EE DPB512S	E47E DPBASE	E44E DPBHD1	E45E DPBHD2	E46E DPBHD3
E2FF DRIVES	DFD2 DRVHD	E2E9 DRVPTTR	0020 DRVRDY	0007 DSKCLK
E099 DTSLOP	0005 ENTRY	E16F FILL	0001 FIRST	DF61 FLOPFLG
DF7D FLOPOK	E0DD FLUSH	E197 FREAD	E042 GETDPB	E233 GETSPT
DE58 GOCPM	E250 HDADD	0051 HDCMND	0050 HDCNTL	0053 HDDATA
E2FB HDDISK	E213 HDDMA	E1BA HDDRV	0052 HDFUNC	E1CB HDHOME
0050 HDORG	E2B4 HDPREP	E236 HDREAD	0051 HDRESLT	0004 HDRLN
E221 HDSEC	E2D9 HDSECTR	E218 HDSIDE	0020 HDSPT	0050 HDSTAT
E1EC HDTRK	E1FA HDTRK2	E26B HDWRITE	E2F5 HEAD	DF04 HOME
0000 IDBUFF	0040 INDEX	E005 INDX1	E00C INDX2	00C0 INTIOBY
E0CB INTO	0003 IOBYTE	0008 ISBUFF	DD77 LIST	DD7A LIST1
DD82 LISTST	0003 LOGDSK	DE30 LSLPT	DDC0 LSTBLE	DD98 LTBLE
0030 M10	0000 M20	0001 M26	0002 MAXFLOP	0001 MAXHD
00F7 MDIR	E59E MESSAGE	E0A8 MOVE	E1A3 MOVER	E1A5 MOVLOP
001A MREV	003C MSIZE	E0FE NOADJST	00FB NSTEP	00FC NULL
4A00 OFFSETC	0002 OPDONE	F000 ORIGIN	E0C6 OUTOF	DD72 PNCH1
E0EA PREP	E29A PROCESS	E4D8 PROMPT	0004 PSTEP	DDA0 PTBLE
DD6C PUNCH	DDE9 PWAIT	E0C1 RDWR	DD62 READER	E06A READ
DD65 READERA	DD68 READR1	DE35 READY	E06E REDWRT	000A RETRIES
0002 RETRY	E0F3 RETRYLP	E150 RETRYOP	001C REVNUM	0001 RSECT
DDA8 RTBLE	E256 RTLOOP	0005 SCENBL	0001 SDELAY	0200 SECLN
E0AD SECPSEC	E06F SECSIZ	DF0B SECTRAN	DD4C SELDEV	DEFE SETDMA
DF46 SETDRV	E01A SETDRV1	DEF8 SETSEC	E1E1 SETTLE	DF06 SETTRK
DF21 SIDEA	DFC2 SIDEOK	DF24 SIDEONE	DF2A SIDETWO	E204 SLOOP
DE17 STAT	E1D0 STEPO	DFEE TDELAY	DE38 TINIT	0001 TKZERO
0008 TMOUT	0100 TPA	DF13 TRANFP	DF42 TRANHD	E4D2 TRUESEC
0038 VIOX	DEAD WARMBEG	DEAD WARMEND	DED6 WARMLOD	DEE8 WARMRD
DD03 WBOOTE	DEAE WBOOT	0000 WBOT	000F WENABL	0010 WFAULT
DEB2 WFLG	000B WRESET	E063 WRITE	E0D5 WRITTYP	DEEB WRMREAD
E219 WSDONE	0005 WSECT	E277 WTLOOP	E38D XLT124	E300 XLT128
E31B XLT256	E350 XLT512	E05B XLTS	E2D8 ZKEY	E033 ZRET