

```

* 830116 PUT IN OKIDATA AND DIABLO ROUTINES
*
* 830116 FIRST ATTEMPT AT RECONFIGURING- MINOR EQUATE CHANGES ONLY. JJO
*
*****
*
* MORROW DESIGNS CBIOS FOR CP/M VERSION 2.2.
*
* THIS CBIOS CAN BE CONFIGURED TO RUN WITH THE FOLLOWING DEVICES.
* THE DISKS MAY BE CONFIGURED TO RUN WITH ANY OR ALL OF THE DISK
* SYSTEMS. THE LOGICAL ORDER OF THE DISKS CAN BE SET TO ANY ORDER.
*
* DISK SYSTEMS:
*   HDCA 10, 20 AND 26 MEGABYTE HARD DISKS.
*   HDDMA 5, 10, 16, MEGABYTE HARD DISK SYSTEMS.
*   DJDMA FLOPPY DISK CONTROLLER WITH 8 AND 5 1/4 INCH DISKS.
*   DJ 2D/B FLOPPY DISK CONTROLLER WITH 8 INCH DISKS.
*
* CONSOLE I/O:
*   DISK JOCKEY 2D/B SERIAL.
*   DISK JOCKEY DMA SERIAL.
*   MULTI I/O SERIAL.
*   DECISION I SERIAL.
*
* PRINTER I/O:
*   MULTI I/O SERIAL WITH HANDSHAKING.
*   MULTI I/O DIABLO 1620 SIMULATOR FOR THE HYTYPE II.
*
* NOTE: FLOPPY SYSTEMS DISKETTE (DRIVE A:) HAS TO HAVE 1024 BYTE
* SECTORS IN ORDER FOR THE COLD AND WARM BOOT LOADERS TO
* WORK. BE SURE TO FORMAT ALL NEW SYSTEM DISKETTES WITH
* 1024 BYTE SECTORS. THE SYSTEM DISKETTE CAN BE EITHER
* SINGLE OR DOUBLE SIDED. THE SECTOR SIZE ON NORMAL (NON
* A: DRIVE) DISKETTES IS NOT RESTRICTED. THUS IF YOU HAVE
* A DISKETTE WITH SOFTWARE THAT IS SUPPOSED TO RUN ON THE
* A: DRIVE THEN YOU SHOULD MOUNT THE DISKETTE IN THE B:
* DRIVE AND THEN PIP IT OVER TO A 1024 BYTE SECTOR
* SYSTEM DISKETTE.
*
* WRITTEN BY LES KENT AND MARC KUPPER          3/4/82
*
* DATE      PROGRAMMER  DESCRIPTION
*
**11 20 82 MARC      PUBLIC RELEASE OF REVISION E.31
* 11 19 82 MARC      CHANGED HDC3 EQUATE TO HDCA
* 11 19 82 MARC      CHANGED BLANK IO ROUTINES FROM RET TO JMP $
* 11 19 82 MARC      CONVERTED BIOSLN TO A BYTE VALUE
* 11 9 82 MARC       REDUCED BAD MAP SIZE TO 1 FOR NON MW SYSTEMS
* 11 8 82 MARC       DELETED BAUD RATE TEST FROM MULTIO DRIVERS
* 11 4 82 MARC       ADDED INITIAL IOBYTE TO IOCONF
* 11 3 82 MARC       ADDED THE NORTH STAR CHARACTER I/O SYSTEM
* 11 2 82 MARC       ADDED CHARACTER REDIRECTION CODE FOR THE IOBYTE
* 11 1 82 MARC       CHANGED SERIAL I/O ENTRY NAMES TO IOBYTE NAMES
* 10 18 82 MARC      FIXED SETHIGH FOR 2 SIDED DJDMA 8 INCH DISKS
* 10 18 82 MARC      DELETED THE HYTYPE DRIVERS
**10 1 82 MARC      PUBLIC RELEASE OF REVISION E.3

```

DJ DMA

```
* 9 29 82 MARC      40H NOW POINTS TO THE HDDMA COMMAND CHANNEL      *
* 9 28 82 MARC      MW'S NOW HAVE 1024 DIRECTORY ENTRIES          *
* 9 28 82 MARC      DELETED THE CENTRONICS DRIVERS              *
* 9 27 82 MARC      CHANGED LOGIN MESSAGE TO LOOK LIKE A LABEL  *
* 9 27 82 MARC      CHANGED THE LOGIN MESSAGES TO SAY M5, M10, ... *
* 9 27 82 MARC      REDEFINED THE DPARAM TABLE STRUCTURE      *
* 9 22 82 MARC      ADDED A SERIAL CONSOLE FOR THE SWITCHBOARD  *
* 9 22 82 MARC      ADDED INITIALIZATION CODE FOR SERIAL GROUP 2 *
* 9 22 82 MARC      ADDED SECTOR SIZE BYTE TO THE HDCA DPB'S    *
* 9 22 82 MARC      ADDED SECTOR SIZE PARAMETER TO DPBGEN      *
* 9 9 82 MARC       FIXED SYSTEM LENGTH CHECKS FOR 64K SYSTEMS  *
* 9 9 82 MARC       SETHIGH WAS BOTCHING 2 SIDED DPB POINTERS  *
* 8 31 82 MARC      CHANGED TRACKS IN HD DRIVER TO HDTRAK     *
* 8 27 82 MARC      ADDED CODE/SYSTEM LENGTH CHECKER          *
* 8 27 82 MARC      MWRESET SAVE/RESTORES THE TRACK NUMBER    *
* 8 26 82 MARC      MWRESET NOW SETS *STEP AND *DIR FOR CMI    *
* 8 20 82 MARC      ADDED 'EQU'ED HANDSHAKING TO THE SERIAL LST: *
* 8 19 82 MARC      REMOVED CLOCK SWITCHING CODE FROM HDCA DRIVER *
* 8 18 82 MARC      ADDED HANDSHAKE CONFIGURATION CODE        *
* 8 18 82 MARC      ADDED HANDSHAKE CONFIGURATION BYTES       *
* 8 18 82 MARC      REMOVED 'EQU'ED HANDSHAKING FROM LST:     *
* 8 12 82 MARC      ADDED CONFIGURATION ENTRIES FOR A0 & D0   *
* 8 11 82 MARC      ADDED THE AUTOSTART COMMAND STRUCTURE     *
* 8 11 82 MARC      REDEFINED THE CONFIGURATION TABLE        *
* 8 11 82 MARC      ADDED DJDMA DRIVE PARAMETER TABLE        *
* 8 9 82 MARC       ADDED CLOCK SWITCHING TO HDCA CODE         *
* 8 9 82 MARC       ADDED SEEK COMPLETE CLEARING IN HDCA      *
* 8 6 82 MARC       ADDED BUFFER DISABLE ON HOME              *
* 8 6 82 MARC       FIXED 8250 UART INITIALIZATION SEQUENCE   *
* 8 6 82 MARC       STRIP PARITY ON CONOUT TO CLEAR UP GLITCHES *
* 8 6 82 MARC       FIXED THE 8 INCH DPB256SS DPB'S EXM      *
* 8 6 82 MARC       INCREASED THE HD CAPACITIES SLIGHTLY     *
* 8 6 82 MARC       DELETED ALL NON-SUPPORTED MW DRIVES      *
* 8 6 82 MARC       DELETED CALL TO FLUSH IN CONOUT          *
* 8 6 82 MARC       MOVED PRINTER BACK TO PORT 3              *
* 7 28 82 MARC      MOVED CONIN FLUSH CALL TO CONOUT          *
* 7 27 82 MARC      FIXED DOUBLE SIDED HEAD SETTLE TIME      *
* 7 14 82 MARC      OPTIMIZED MWISSUE                          *
* 7 14 82 MARC      CLEAN UP LOGIN MESSAGE FOR HD A BIT      *
* 6 30 82 MARC      FIXED MF MULTI DENSITY PROBLEMS          *
* 6 29 82 MARC      ADDED OLIVETTI HD561/1 HD561/2 DRIVES    *
* 6 28 82 MARC      ADDED A MW ERROR REPORTER                 *
* 6 18 82 MARC      ADDED NONSTANDARD SYSTEM MODE FLAG       *
* 6 17 82 MARC      ADDED A BUFFER ERROR FLAG                 *
* 6 17 82 MARC      ADDED SAVE/RESTORE OF 50-52 TO MW DRIVER  *
* 6 17 82 MARC      FIXED CENTRONICS DRIVERS                  *
* 6 7 82 MARC       FIXED ALLOCATION MAP SIZES                  *
* 6 7 82 MARC       FIXED MW PARTITIONING                      *
* 6 7 82 MARC       FIXED HD PARTITIONING (AGAIN)             *
* 5 13 82 MARC      FIXED ILLEGAL MAC LABELS                  *
* 5 11 82 MARC      FIXED NORTH STAR DRIVE CONFIGURATIONS    *
* 4 30 82 MARC      FIXED QUANTUM Q2040 TRACKS TO 512       *
* 4 29 82 MARC      FIXED ST412 STEP CONSTANT TO 0           *
* 4 26 82 MARC      ADDED UNALLOCATED WRITING                 *
* 4 22 82 MARC      FIXED HD PARTITION OVERLAP                *
* 4 20 82 MARC      STARTED TESTING AND DEBUGGING OF E.3     *
```

```

* 4 19 82 MARC      ADDED 1 SECTOR TO HD WARM BOOT LOADER      *
* 4 19 82 MARC      ADDED MOD. NUMBER TO CBIOS REV. NUMBER    *
* 4 19 82 MARC      CLEAN UP LOGIN MESSAGE 'IF'S              *
* 4 15 82 MARC      FIXED MCR INITIALIZATION FOR LST:          *
* 4 15 82 MARC      ADDED SEAGATE ST412 DRIVE                  *
* 4 6 82 MARC       MOVED SERIAL LST: DEVICE TO PORT 2        *
* 4 1 82 MARC       ADDED COMMON GROUP SELECT ROUTINES        *
* 4 1 82 MARC       FIXED DIABLO HYTYPE II INITIALIZATION     *
* 4 1 82 MARC       FIXED LISTST FOR PROM DRIVER              *
* 3 16 82 MARC      ADDED TANDON TM602 AND TM603 DRIVES        *
* 3 16 82 MARC      USE 'PART NUMBER' EQUATES FOR MW DRIVES   *
* 3 15 82 MARC      DROPPED HDREV AND MWREV EQUATES           *
* 3 15 82 MARC      SEAGATE ST506 HEAD SETTLE IS 0 MS.       *
* 3 15 82 MARC      ADDED MINISCRIBE 1006 AND 1012 DRIVES     *
* *3 1 82 MARC      PUBLIC RELEASE OF REVISION E.2           *
* 2 -- 82 MARC      PRE-RELEASE TESTING AND DEBUGGING         *
* 2 1 82 LES + MARC INITIAL CODING OF REVISION E              *
*

```

\*\*\*\*\*

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

```

0035 = REVNUM EQU 53 ;CBIOS REVISION NUMBER 5.X = E
0016 = CPMREV EQU 22 ;CP/M REVISION NUMBER 2.2

```

\*\*\*\*\*

```

*
* THE FOLLOWING FLAGS SET A 'NON-STANDARD' SYSTEM MODE AND AN
* ASSEMBLY TIME DEBUGGER.
*
* IF THIS CBIOS IS USED WITH THE CP/M 2.2 SYSTEM THAT IS SHIPPED ON
* A MORROW DESIGNS DISKETTE THEN NOSTAND CAN BE SET TO 1. THIS
* WILL ALLOW THE CBIOS TO USE VARIOUS DATA AREAS FOUND INSIDE OF
* THE CP/M 2.2 BDOS. IF THE CBIOS IS USED WITH A DIFFERENT
* OPERATING SYSTEM THEN NOSTAND SHOULD BE SET TO 0.
*
* THE DEBUG FLAG MERELY CAUSES VARIOUS INTERNAL VALUES AND
* ADDRESSES TO BE PRINTED DURING THE ASSEMBLY PROCESS. THIS
* PRINTING IS FORCED VIA ASSEMBLY ERRORS AND THUS SHOULD NOT
* AFFECT THE RESULTING CODE IN ANY WAY.
*

```

\*\*\*\*\*

```

0001 = NOSTAND EQU 1 ;SET TO 1 FOR NON-STANDARD MODE
0000 = DEBUG EQU 0 ;SET TO 1 FOR DEBUGGING MODE

```

\*\*\*\*\*

```

*
* THE FOLLOWING IS SET TO THE MEMORY SIZE OF THE CP/M THE CBIOS IS
* BEING CREATED FOR.
*

```

\*\*\*\*\*

```

0040 = MSIZE EQU 64 ;MEMORY SIZE OF TARGET CP/M
0016 = BIOSLN EQU 16H ;BIOS LENGTH. ALSO IN ABOOT&.ASM

```

```
*****
*
* THE FOLLOWING EQUATES SET UP THE DISK SYSTEMS TO BE INCLUDED
* ALONG WITH THE TYPES OF DRIVES AND THE LOGICAL ORDER OF THE
* DRIVES.
*
*****
```

```
0001 = MAXHD EQU 1 ;SET TO NUMBER OF HDCA HARD DISK DRIVES
0000 = MAXMW EQU 0 ;SET TO NUMBER OF HDDMA HARD DISKS
0000 = MAXFD EQU 0 ;SET TO NUMBER OF 2D/B FLOPPIES
0001 = MAXDM EQU 1 ;SET TO NUMBER OF DJ DMA FLOPPIES 8 INCH
0000 = MAXMF EQU 0 ;SET TO NUMBER OF DJ DMA FLOPPIES 5 1/4 INCH

0001 = HDORDER EQU 1 ;SET THE ORDER OF LOGICAL DRIVES ELSE 0 IF
0000 = MWORDER EQU 0 ; NOT INCLUDED.
0000 = FDORDER EQU 0
0002 = DMORDER EQU 2
0000 = MFORDER EQU 0

;HDCA CONTROLLER DISK DRIVES. SET ONLY ONE
0000 = M10F EQU 0 ;FUJITSU M2301B
0001 = M20 EQU 1 ;FUJITSU M2302B
0000 = M26 EQU 0 ;SHUGART SA4000
0000 = M10M EQU 0 ;MEMOREX

;HDDMA CONTROLLER DISK DRIVES. SET ONLY ONE
0000 = MWQUIET EQU 0 ;SET FOR NO NAMES PRINTED ON LOGIN
0000 = ST506 EQU 0 ;SEAGATE ST-506
0000 = ST412 EQU 0 ;SEAGATE ST-412
0000 = CM5619 EQU 0 ;CMI CM-5619

0000 = WMDRIVE EQU 0 ;DEVICE TO WARM BOOT FROM. THIS IS THE
; CP/M LOGICAL DRIVE NUMBER.

BADSIZ IF MAXMW NE 0 ;ONLY HDDMA DRIVES USE THE BAD MAP
EQU 32 ;NUMBER OF BADMAP ENTRIES
ELSE
0001 = BADSIZ EQU 1 ;LEAVE ONE ENTRY AS FILLER
ENDIF
```

```
*****
*
* SINCE MOST HARD DISK DRIVES HOLD MORE THAN 8 MEGABYTES WE
* PARTITION THE DRIVE. WE PARTITION OUR DRIVES USING TWO DIFFERENT
* FORMULAS.
*
* ONE IS THE SO CALLED 'STANDARD PARTITIONING' WHERE WE TRY TO
* CREATE AS MANY 8 MEGABYTE PARTITIONS AS POSSIBLE PLUS A SMALL
* PARTITION TO TAKE UP THE SLACK ON THE END OF THE DRIVE.
*
* ANOTHER WAY THE DRIVES ARE PARTITIONED IS THE SO CALLED 'EVEN
* PARTITION' FORMULA. THIS MEANS THAT THE DRIVE IS SPLIT INTO
* EQUALE SIZED PARTITIONS WITH THE ONLY RESTRICTION BEING THAT NO
* PARTITION BE OVER 8 MEGABYTES IN LENGTH.
*
```



\* ALL HARD DISK DRIVES SHIPPED FROM MORROW DESIGNS ARE PARTITIONED \*
\* USING THE STANDARD PARTITION FORMULA. IF THE USER WISHES TO \*
\* IMPLEMENT EVEN PARTITIONING THEN HE/SHE MUST SET HDPART OR MWPART \*
\* TO THE NUMBER OF PARTITIONS DESIRED. \*

0000 = HDPART EQU 0 ;SET TO NUMBER OF NON STANDARD PARTITIONS
0000 = MWPART EQU 0 ;SET TO NUMBER OF NON STANDARD PARTITIONS

\* THE FOLLOWING EQUATES DEFINE THE CONSOLE AND PRINTER ENVIRONMENTS. \*

\* DEFINE THE CONSOLE DRIVER TO BE USED. \*

\* CONTYP IS: 0 NOTHING, USED FOR PATCHING TO PROM'S.
1 PROVIDE FOR 128 BYTES OF PATCH SPACE.
2 MULTI I/O OR DECISION I DRIVER.
3 2D/B DRIVER.
4 DJDMA SERIAL PORT
5 SWITCHBOARD SERIAL PORT
6 NORTH STAR MOTHERBOARD (2 SERIAL + 1 PARALLEL)

\* SET CBAUD TO THE DIVISOR LATCH VALUE FOR THE CONSOLE. FOR AN
\* EXPLANATION OF THE VALUES LOOK AT THE DEFCON TABLE. \*

0002 = CONTYP EQU 2
0006 = CBAUD EQU 6

\* DEFINE THE PRINTER DRIVER TO BE USED. \*

\* LSTTYP IS: 0 NOTHING, USED FOR PATCHING TO PROM'S.
1 PROVIDE FOR 128 BYTES OF PATCH SPACE.
2 MULTIO SERIAL, NO PROTOCOL.
3 MULTIO SERIAL, CLEAR TO SEND PROTOCOL.
4 MULTIO SERIAL, DATA SET READY PROTOCOL.
5 MULTIO SERIAL, XON/XOFF PROTOCOL.

\* NOTE: THE DECISION BOARD IS FUNCTIONALLY IDENTICAL TO THE MULTI
\* I/O BOARD FOR SERIAL PRINTER I/O. SELECTIONS 2 TO 5 WILL
\* WORK ON THE WUNDERBUSS I/O BOARD. TO USE DRIVERS 6 OR 7
\* THE MULTR3 EQUATE WILL HAVE TO BE SET. \*

\* SET PBAUD TO THE DIVISOR LATCH VALUE FOR THE PRINTER. FOR AN
\* EXPLANATION OF THE VALUES SEE THE DEFLST TABLE. \*

0003 = LSTTYP EQU 3  
 0060 = LBAUD EQU 96

```
*****
*
* THE NEXT EQUATE DETERMINES IF YOU HAVE A MULTI I/O REV 3 OR A
* DECISION I MOTHER BOARD FOR PARALLEL I/O. IF ARE NOT USING
* EITHER OF THESE BOARDS THEN YOU NEED NOT WORRY ABOUT THIS EQUATE.
* IF YOU ARE USING A MULTI I/O REV. OTHER THAN 3.X OR 4.X THEN YOU
* SHOULD SET MULTR3 TO 0.
*
*****
```

0000 = MULTR3 EQU 0 ;0 = DECISION, 1 = MULTI I/O REV. 3 OR 4  
 0001 = CONGRP EQU 1 ;COSOLE PORT (1 = P1, 2 = P2, 3 = P3)  
 0003 = LSTGRP EQU 3 ;PRINTER PORT (1 = P1, 2 = P2, 3 = P3)

```
*****
*
* THE FOLLOWING EQUATES ARE INTERNAL TO THE CBIOS.
*
*****
```

0000 = M10 EQU M10F OR M10M  
 HDLOG IF HDPART NE 0 ;USE NON STANDARD PARTITIONS  
 EQU HDPART  
 0003 = HDLOG EQU M10\*2+M20\*3+M26\*3 ;LOGICAL DISKS PER DRIVE FOR HDCA  
 ENDIF  
 MWLOG IF MWPART NE 0 ;USE NON STANDARD PARTITIONS  
 EQU MWPART  
 0000 # MWLOG SET ST506+ST412\*2++CM5619\*2 ;LOGICAL DISKS PER DRIVE FOR HDDMA  
 ENDIF  
 0001 = HDCA EQU M26 OR M20 OR M10 ;HDCA CONTROLLER  
 0001 = FUJITSU EQU M20 OR M10F  
 0015 = HDSPT EQU 32\*M26+21\*M20+21\*M10 ;SECTORS PER TRACK  
 0000 # HDMA SET ST506 OR ST412 OR CM5619 ;HD DMA CONTROLLER  
 0009 = MWSPT EQU 9 ;SECTORS PER TRACK  
 0004 = MAXLOG EQU (MAXHD\*HDLOG)+(MAXMW\*MWLOG)+MAXFD+MAXDM+MAXMF

```
*****
*
* CP/M SYSTEM EQUATES.
*
*****
```

0800 = CCPLN EQU 800H

```

0E00 =      BDOSLN EQU      0E00H

0000 =      SIZE EQU      (MSIZE*1024)
D400 =      CCP EQU      SIZE-(BIOSLN*100H+CCPLN+BDOSLN)
DC00 =      BDOS EQU      CCP+CCPLN
EA00 =      BIOS EQU      CCP+CCPLN+BDOSLN

3700 =      OFFSETC EQU    2100H-BIOS      ;OFFSET FOR SYSGEN

          IF      DEBUG
DBGTMP SET  OFFSETC      ;DDT OFFSET      ! <DEBUG>
DBGTMP SET  CCP          ;CCP ADDRESS     ! <DEBUG>
DBGTMP SET  BDOS        ;BDOS ADDRESS    ! <DEBUG>
DBGTMP SET  BIOS        ;CBIOS ADDRESS   ! <DEBUG>
          ENDIF

0004 =      CDISK EQU      4              ;ADDRESS OF LAST LOGGED DISK
0080 =      BUFF EQU      80H            ;DEFAULT BUFFER ADDRESS
0100 =      TPA EQU      100H           ;TRANSIENT MEMORY
0003 =      IOBYTE EQU    3              ;IOBYTE LOCATION
0000 =      WBOT EQU      0              ;WARM BOOT JUMP ADDRESS
0005 =      ENTRY EQU     5              ;BDOS ENTRY JUMP ADDRESS

          IF      NOSTAND NE 0
E9E7 =      CBLOCK EQU    BIOS-19H      ;CURRENT ACTUAL BLOCK# * BLKMSK
          ;USED FOR UNALLOCATED WRITTING

          ENDIF
    
```

```

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

```

000A =      RETRIES EQU    10            ;MAX RETRIES ON DISK I/O BEFORE ERROR
001A =      CLEAR EQU     'Z'-64        ;CLEAR SCREEN ON AN ADM 3

0000 =      ANUL EQU      0              ;NULL
0003 =      AETX EQU     'C'-64        ;ETX CHARACTER
0006 =      AACK EQU     'F'-64        ;ACK CHARACTER
0007 =      ABEL EQU     'G'-64        ;BELL
0008 =      ABS EQU      'H'-64        ;BACK SPACE
0009 =      AHT EQU     'I'-64        ;HORIZONTAL TAB
000A =      ALF EQU     'J'-64        ;LINE FEED
000B =      AVT EQU     'K'-64        ;VERTICAL TAB
000C =      AFF EQU     'L'-64        ;FORM FEED
000D =      ACR EQU     'M'-64        ;CARRIAGE RETURN
0011 =      XON EQU     'Q'-64        ;XON CHARACTER
0013 =      XOFF EQU    'S'-64        ;XOFF CHARACTER
001B =      AESC EQU     1BH           ;ESCAPE CHARACTER
001E =      ARS EQU     1EH           ;RS CHARACTER
001F =      AUS EQU     1FH           ;US CHARACTER
0020 =      ASP EQU     ' '           ;SPACE
007F =      ADEL EQU     7FH           ;DELETE
    
```

```

*****
    
```

\*
\* THE FOLLOWING ARE THE MACROS USED IN GENERATING THE DPH, DPB AND
\* ALLOCATION TABLES.
\*
\*\*\*\*\*

DPBGEN MACRO NAM, LOG, DSPT, DBSH, DBLM, DEXM, DDSM, DDRM, DAL0, DAL1, DCKS, DOFF, SSIZ
DPB&NAM&LOG EQU \$
DW DSPT
DB DBSH
DB DBLM
DB DEXM
DW DDSM
DW DDRM
DB DAL0
DB DAL1
DW DCKS
DW DOFF
DB SSIZ
ENDM

DPHGEN MACRO NAM, LOG, DPB1, DPB2
DPH&NAM&LOG EQU \$
DW 0
DW 0, 0, 0
DW DIRBUF
DW &DPB1&DPB2
DW CSV&NAM&LOG
DW ALV&NAM&LOG
ENDM

ALLOC MACRO NAM, LOG, AL, CS
CSV&NAM&LOG: DS CS
ALV&NAM&LOG: DS AL
ENDM

\*\*\*\*\*
\*
\* THE FOLLOWING MARCO IS USED IN GENERATING THE LOGICAL ORDER OF THE
\* CP/M DRIVES.
\*
\*\*\*\*\*

ORDER MACRO NUM
IF NUM EQ HDORDER
DW HDDST
ENDIF
IF NUM EQ MWORDER
DW MWDST
ENDIF
IF NUM EQ FDORDER
DW FDDST
ENDIF

```
IF NUM EQ DMORDER
DW DMDST
ENDIF
```

```
IF NUM EQ MFORDER
DW MFDST
ENDIF
ENDM
```

```
*****
*
* THE FOLLOING ARE OFFSET NUMBERS OF DEVICE SPECIFICATION TABLES.
*
*****
```

```
0000 = D$WBOOT EQU 0 ;WARM BOOT
0001 = D$STRAN EQU 1 ;SECTOR TRANSLATION
0002 = D$SEL1 EQU 2 ;DRIVE SELECT, RETURN DPH
0003 = D$SEL2 EQU 3 ;DRIVE SELECT
0004 = D$HOME EQU 4 ;HOME DRIVE
0005 = D$STRK EQU 5 ;SET TRACK
0006 = D$SSEC EQU 6 ;SET SECTOR
0007 = D$SDMA EQU 7 ;SET DMA ADDRESS
0008 = D$READ EQU 8 ;READ A PHYSICAL SECTOR
0009 = D$WRITE EQU 9 ;WRITE A PHYSICAL SECTOR
000A = D$BAD EQU 10 ;RETURN POINTER TO BAD SECTOR INFO
```

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

```
EA00 ORG BIOS ;CBIOS STARTING ADDRESS

EA00 C3B9F5 JMP CBOOT ;COLD BOOT ENTRY POINT
EA03 C3DCB8 WBOOT: JMP WBOOT ;WARM BOOT ENTRY POINT

IF CONTYP NE 0
EA06 C3C2EA CONST: JMP CONIST ;CONSOLE STATUS ROUTINE
EA09 C3A5EA CIN: JMP CONIN ;CONSOLE INPUT
EA0C C39EEA COUT: JMP COSTRP ;CONSOLE OUTPUT
ELSE
CONST: JMP $ ;CONSOLE STATUS ROUTINE PROM POINTER
CIN: JMP $ ;CONSOLE INPUT PROM POINTER
COUT: JMP $ ;CONSOLE OUTPUT PROM POINTER
ENDIF

EA0F C30CEB POUT: IF (LSTTYP NE 0) OR (CONTYP EQ 6)
JMP LSTOUT ;LIST DEVICE OUTPUT
ELSE
POUT: JMP COUT ;LIST DEVICE OUTPUT
ENDIF

IF CONTYP EQ 6 ;NORTH STAR DRIVERS HAVE PUNCH ENTRY POINTS
```

```

EA12 C30CEA      JMP     PUNOUT      ;PUNCH DEVICE OUTPUT
                 ELSE
                 JMP     COUT        ;USE CONSOLE I/O
                 ENDIF

EA15 C309EA      IF     CONTYP EQ 6      ;NORTH STAR DRIVERS HAVE READER ENTRY POINTS
                 JMP     RDRIN      ;READER DEVICE INPUT
                 ELSE
                 JMP     CIN        ;USE CONSOLE I/O
                 ENDIF

EA18 C3FFEB      JMP     HOME        ;HOME DRIVE
EA1B C31CEC      JMP     SETDRV      ;SELECT DISK
EA1E C30DEC      JMP     SETTRK     ;SET TRACK
EA21 C3F3EB      JMP     SETSEC     ;SET SECTOR
EA24 C3F9EB      JMP     SETDMA     ;SET DMA ADDRESS
EA27 C31AED      JMP     READ       ;READ THE DISK
EA2A C311ED      JMP     WRITE      ;WRITE THE DISK

EA2D C319EB      IF     LSTTYP NE 0
                 JMP     LSTOST     ;LIST DEVICE STATUS
                 ELSE
                 JMP     DONOP      ;LIST DEVICE STATUS
                 ENDIF

EA30 C313EC      JMP     SECTRAN     ;SECTOR TRANSLATION

```

;
; THE FOLLOWING JUMPS ARE EXTENDED BIOS CALLS DEFINED BY MORROW DESIGNS
;

```

EA33 C3F8EB      IF     MAXFD NE 0
                 JMP     FDSEL      ;HOOKUP FOR SINGLE.COM PROGRAM
                 ELSE
                 JMP     DONOP
                 ENDIF

EA36 C30000      JMP     0          ;END OF THE JUMP TABLE

```

\*\*\*\*\*
\*
\* DRIVE CONFIGURATION TABLE.
\*
\*\*\*\*\*

```

EA39 00          DRCONF: DB     0          ;REVISION 0 STRUCTURE
EA3A 20          DB     32          ;32 BYTES LONG NOW

```

\*\*\*\*\*
\*
\* THE FOLLOWING IS THE TABLE OF POINTERS TO THE DEVICE
\* SPECIFICATION TABLES. THE ORDER OF THIS TABLE DEFINES THE
\* LOGICAL ORDER OF THE CP/M DRIVES.
\*
\*\*\*\*\*

```
EA3B =          DSTTAB: EQU      $
0001 #          DN              SET      1
                                REPT    16
                                ORDER   %DN
                                DN       DN+1
                                ENDM
EA3B+12EF      DW              HDDST
EA3D+4DF2      DW              DMDST
```

```
*****
*
* I/O CONFIGURATION TABLE.
*
* AT THIS CBIOS REVISION 11 BYTES ARE DEFINED FOR THIS TABLE.
* SEVERAL EXTENSIVE CHANGES ARE PLANNED FOR THE TABLE. FUTURE
* REVISION OF THE IOCONF TABLE WILL HAVE INDEPENDANT ENTRIES FOR
* THREE SERIAL PORTS AND WILL BE USED BY SEVERAL CHARACTER DRIVERS.
* ALSO THE IOBYTE WILL BE IMPLEMENTED FOR ALL THE CHARACTER
* DRIVERS. I MIGHT EVEN WRITE AN EXTERNAL PROGRAM TO EDIT THIS
* TABLE.
*
* THE FIRST TWO BYTES SHOW THE I/O CONFIGURATION THAT THE CBIOS WAS
* ASSEMBLED WITH. THESE BYTES ARE USED BY EXTERNAL SOFTWARE TO
* DETERMINE THE CONFIGURATION OPTIONS THAT ARE AVAILABLE.
*
* THE NEXT BYTE IS THE INITIAL IOBYTE VALUE. THIS VALUE IS WRITTEN
* TO LOCATION 3 ON COLD BOOTS. SEE THE CP/M 2 ALTERNATION GUIDE
* FOR A DESCRIPTION OF THE IOBYTE.
*
* THE NEXT BYTE IS TO MAKE SURE THAT THE GROUP SELECT BYTE ON THE
* MULT I/O OR DECISION I STAYS CONSISTANT THROUGHOUT THE CBIOS.
* ONLY THE GROUP BITS THEMSELVES (BITS 0 AND 1) SHOULD BE CHANGED
* AS YOU OUTPUT TO THE GROUP PORT. IF YOU MODIFY ONE OF THE OTHER
* BITS (SUCH AS DRIVER-ENABLE) THEN YOU SHOULD MODIFY THE SAME BIT
* IN THIS BYTE. FOR EXAMPLE:
*
*                                ;SELECT CONSOLE GROUP
*      LDA      GROUP            ;GET GROUP BYTE
*      ORI      CONGRP          ;SELECT THE CONSOLE PORT
*      OUT     GRPSEL           ;SELECT THE GROUP
*
*                                ;MODIFY A BIT IN THE GROUP BYTE
*      LDA      GROUP            ;GET GROUP BYTE
*      ORI      BANK            ;SET THE BANK BIT
*      STA     GROUP            ;SAVE NEW GROUP SETTING
*      ORI     GROUP2          ;SELECT SECOND SERIAL PORT
*      OUT    GRPSEL           ;SELECT THE DESIRED GROUP
*
* NOTE: YOU SHOULD NOT SET THE GROUP BITS THEMSELVES IN THE
*       GROUP BYTE.
*
* THE FOLLOWING TWO WORDS DEFINE THE DEFAULT BAUD RATES FOR THE
* CONSOLE AND THE LIST DEVICES. THESE WORDS ARE PROVIDED SO THAT
* THE USER CAN EASILY MODIFY THEM AND THAT THEY WILL ALSO BE USED
```

```

* IN THE FUTURE BY MORROW DESIGNS SOFTWARE.
*
* THE FOLLOWING IS A LIST OF POSSIBLE BAUD RATES AND THE DECIMAL
* VALUE NEEDED FOR THE DEFCON OR DEFLST WORDS.
*
* BAUD RATE      DEFCON/DEFLST  BAUD RATE      DEFCON/DEFLST
*      50        2304          2000          58
*      75        1536          2400          48
*     110        1047          3600          32
*    134.5       857           4800          24
*     150        768           7200          16
*     300        384           9600          12
*     600        192          19200           6
*    1200         96          38400           3
*    1800         64          56000           2
*
* THE NEXT TWO BYTES ARE USED TO CONFIGURE THE HARDWARE HANDSHAKING
* PROTOCOL USED BY THE SERIAL LIST DRIVERS WITH THE MULTIO OR
* WUNDERBUSS I/O BOARDS. THE FIRST OF THESE TWO BYTES IS A MASK.
* THIS MASK IS AND'ED WITH THE 8250'S MODEM STATUS REGISTER TO STRIP
* OUT THE DESIRED HANDSHAKE LINES. NEXT THE RESULT OF THE ANDING
* IS XOR'ED WITH THE SECOND OF THE TWO BYTES. THIS XORING ALLOWS
* THE HANDSHAKE LINES TO BE INVERTED. COMMON BYTE VALUES ARE
* SHOWN BELOW.
*
* CTS   EQU      10H          ;CLEAR TO SEND STATUS MASK
*
*       DB       CTS          ;MORROW DESIGNS 'CLEAR TO SEND'
*       DB       0
*
*       DB       CTS          ;INVERTED CLEAR TO SEND
*       DB       CTS
*
*       DB       0            ;NO HANDSHAKING
*       DB       0FFH
*
* THE LAST BYTE IN THE REVISION ONE STRUCTURE IS THE LAST CHARACTER
* THAT WAS RECEIVED FROM THE PRINTER. THIS BYTE IS USED TO
* IMPLEMENT XON/XOFF SOFTWARE HANDSHAKING. THIS HANDSHAKING
* PROTOCOL SHOULD NOT BOTHER PRINTERS THAT HAVE NOT IMPLEMENTED
* XON/XOFF PROTOCOL SO THIS DRIVER IS ENABLED ALL THE TIME.
*
*****

```

```

EA3F 02      IOCONF: DB      2          ;REVISION 2 STRUCTURE
EA40 0B      DB      11          ;11 BYTES LONG NOW
EA41 02      DB      CONTYP       ;CONSOLE DEVICE DRIVER NUMBER
EA42 03      DB      LSTTYP       ;LIST DEVICE DRIVE NUMBER

EA43 =      IOBYT  EQU      $          ;THE INITIAL IOBYTE IS KEPT HERE
EA43 C0      DB      11$00$00$00B ;ALL DEVICES GO TO CON:

EA44 00      GROUP: DB      0          ;GROUP BYTE
EA45 0600    DEFCON: DW      CBAUD     ;CONSOLE BAUD RATE DIVISOR VALUE

```



```
EA47 6000  DEFLST: DW      LBAUD          ;PRINTER BAUD RATE DIVISOR VALUE

           IF      (LSTTYP NE 3) AND (LSTTYP NE 4) ;XON/XOFF PROTOCOL
LSTAND: DB      0          ;SERIAL LIST HANDSHAKE MASK
LSTXOR: DB      0FFH      ;SERIAL LIST INVERSION FLAG
           ENDIF

           IF      LSTTYP EQ 3      ;CLEAR TO SEND PROTOCOL
EA49 10    LSTAND: DB      CTS        ;SERIAL LIST HANDSHAKE MASK
EA4A 00    LSTXOR: DB      0          ;SERIAL LIST INVERSION FLAG
           ENDIF

           IF      LSTTYP EQ 4      ;DATA SET READY PROTOCOL
LSTAND: DB      DSR        ;SERIAL LIST HANDSHAKE MASK
LSTXOR: DB      0          ;SERIAL LIST INVERSION FLAG
           ENDIF

EA4B 11    LASTCH: DB      XON        ;LAST CHARACTER RECIEVED FROM THE PRINTER
```

```
*****
*
* THE FOLLOWING TABLE ARE DRIVE PARAMETERS FOR DRIVES CONNECTED TO
* THE DJDMA FLOPPY DISK CONTROLLER.  THERE IS ONE ENTRY FOR EACH OF
* THE THE EIGHT DRIVE THAT THE CONTROLLER CAN ADDRESS.  THE FIRST
* FOUR ENTRIES ARE FOR THE 8 INCH DRIVES AND THE LAST FOUR ARE FOR
* THE 5 1/4 INCH DRIVES.  USERS WITH FAST STEPPING 8 INCH DRIVES
* (SA850/1) OR SLOW 5 1/4 INCH DRIVES (SA400) SHOULD ADJUST THIS
* TABLE FOR OPTIMAL DEVICE PERFORMACE.
*
* EACH TABLE ENTRY CONTAINS FOUR FIXED LENGTH FIELDS.  THE FIELDS
* ARE DEFINED AS FOLLOWS:
*
* TRACKS THIS BYTE CONTAINS THE NUMBER OF TRACKS ON THE
* DRIVE.  MOST 8 INCH DRIVES HAVE 77 TRACKS AND
* MOST 5 1/4 INCH DRIVES HAVE 35 OR 40 TRACKS.
*
* CONFIG THIS A A FLAG BYTE THAT INDICATES AS TO WHETHER
* OR NOT THIS DRIVE HAS BEEN CONFIGURED.  SET TO
* 0 TO FORCE RECONFIGURATION.
*
* STEP THIS WORD CONTAINS THE STEPPING RATE CONSTANT.
* THE DJDMA'S DELAY ROUTINES TICK 34.1 TIMES PER
* MILLISECOND.  THUS THE STEP CONSTANT WOULD BE THE
* DRIVE MANUFACTORS RECOMENDED STEPPING DELAY TIMES
* 34.1.  EXAMPLE.  SHUGART SA 850'S STEP AT 3
* MILLISEOND INTERVALS.  THE STEP CONSTANT WOULD BE
* 3 * 43.1 OR 102.
*
* RFU THE NEXT TWO WORDS ARE RESERVED FOR FUTURE USE.
* THEY MUST BE ZERO.
*
* SETTLE THIS WORD IS SIMILAR TO THE PREVIOUSLY DEFINED
* STEP WORD.  THIS SPECIFIES THE HEAD SETTLE TIMING
* AFTER THE HEADS HAVE BEEN STEPPED.  EXAMPLE,
* SHUGART'S SA 850 HEAD SETTLE TIME IS 15
* MILLISECONDS.  THE SETTLE CONSTANT WOULD BE 15 *
```

```

*           34.1 OR 512.
*
* AN ASSEMBLER MACRO (DCONF) HAS BEEN PROVIDED TO ASSIST IN
* GENERATING THE DPARAM TABLE. THIS MACROS PARAMETERS ARE THE
* NUMBER OF TRACKS, THE STEP RATE IN MILLISECONDS, AND THE HEAD
* SETTLE TIME IN MILLISECONDS. FOR EXAMPLE:
*
*           ;SHUGART SA 850
* DCONF 77, 3, 15 ;77 TRACKS, 3 MS STEP, 15 MS SETTLE
*
*           ;SHUGART SA 400
* DCONF 35, 40, 10 ;35 TRACKS, 40 MS STEP, 10 MS SETTLE
*
* NOTE: CAUTION SHOULD BE USED WHEN DEFINING THE DRIVE PARAMETERS.
* INCORRECT DEFINATIONS MAY DAMAGE THE FLOPPY DISK DRIVE. MORROW
* DESIGNS TAKES NO RESPONSIBILITY FOR DAMAGE THAT OCCURES THROUGH
* THE MISUSE OF THIS MACRO.
*
*****

```

```

IF (MAXDM NE 0) OR (MAXMF NE 0) ;DJDMA PRESENT?

```

```

DCONF MACRO TRACKS, STEP, SETTLE
DB TRACKS ;NUMBER OF TRACKS
DB 0 ;RESET THE CALIBRATED FLAG
DW STEP*341/10 ;STEP TIME
DW 0 ;RESERVED FOR FUTURE USE, MUST BE ZERO
DW 0 ;RESERVED FOR FUTURE USE, MUST BE ZERO
DW SETTLE*341/10 ;HEAD SETTLE TIME
ENDM

```

```

EA4C 0050 DMARAP: DB 0, 10*8 ;REVISION 0, LENGTH 80 BYTES

```

```

EA4E = DPARAM: EQU $ ;DRIVE PARAMETER TABLE

```

```

*****
*
* DEFINE 8 INCH DRIVE PARAMETERS
* USE SA800 PARAMETERS: 77 TRACKS, 8 MS STEP, 8 MS SETTLE
*
*****

```

```

EA4E+4D DCONF 77, 8, 8 ;DRIVE 0
EA4F+00 DB 77 ;NUMBER OF TRACKS
EA50+1001 DB 0 ;RESET THE CALIBRATED FLAG
EA52+0000 DW 8*341/10 ;STEP TIME
EA54+0000 DW 0 ;RESERVED FOR FUTURE USE, MUST BE ZERO
EA56+1001 DW 0 ;RESERVED FOR FUTURE USE, MUST BE ZERO
EA58+4D DCONF 77, 8, 8 ;DRIVE 1
EA59+00 DB 77 ;NUMBER OF TRACKS
EA5A+1001 DB 0 ;RESET THE CALIBRATED FLAG
EA5C+0000 DW 8*341/10 ;STEP TIME
EA5E+0000 DW 0 ;RESERVED FOR FUTURE USE, MUST BE ZERO
EA60+1001 DW 0 ;RESERVED FOR FUTURE USE, MUST BE ZERO
EA60+1001 DW 8*341/10 ;HEAD SETTLE TIME

```

```

EA62+4D      DCONF  77, 8, 8      ;DRIVE 2
EA63+00      DB       77      ;NUMBER OF TRACKS
EA64+1001    DW       8*341/10 ;RESET THE CALIBRATED FLAG
EA66+0000    DW       0       ;STEP TIME
EA68+0000    DW       0       ;RESERVED FOR FUTURE USE, MUST BE ZERO
EA6A+1001    DW       8*341/10 ;RESERVED FOR FUTURE USE, MUST BE ZERO
EA6C+4D      DCONF  77, 8, 8      ;HEAD SETTLE TIME
EA6D+00      DB       77      ;DRIVE 3
EA6E+1001    DW       8*341/10 ;NUMBER OF TRACKS
EA70+0000    DW       0       ;RESET THE CALIBRATED FLAG
EA72+0000    DW       0       ;STEP TIME
EA74+1001    DW       8*341/10 ;RESERVED FOR FUTURE USE, MUST BE ZERO

```

```

*****
*
* DEFINE 5 1/4 INCH DRIVE PARAMETERS
* USE TANDON PARAMETERS: 40 TRACKS, 5 MS STEP, 15 MS SETTLE
*
*****

```

```

EA76+28      DCONF  40, 5, 15    ;DRIVE 0
EA77+00      DB       40      ;NUMBER OF TRACKS
EA78+AA00    DW       5*341/10 ;RESET THE CALIBRATED FLAG
EA7A+0000    DW       0       ;STEP TIME
EA7C+0000    DW       0       ;RESERVED FOR FUTURE USE, MUST BE ZERO
EA7E+FF01    DW       15*341/10 ;RESERVED FOR FUTURE USE, MUST BE ZERO
EA80+28      DCONF  40, 5, 15    ;HEAD SETTLE TIME
EA81+00      DB       40      ;DRIVE 1
EA82+AA00    DW       5*341/10 ;NUMBER OF TRACKS
EA84+0000    DW       0       ;RESET THE CALIBRATED FLAG
EA86+0000    DW       0       ;STEP TIME
EA88+FF01    DW       15*341/10 ;RESERVED FOR FUTURE USE, MUST BE ZERO
EA8A+28      DCONF  40, 5, 15    ;RESERVED FOR FUTURE USE, MUST BE ZERO
EA8B+00      DB       40      ;HEAD SETTLE TIME
EA8C+AA00    DW       5*341/10 ;DRIVE 2
EA8E+0000    DW       0       ;NUMBER OF TRACKS
EA90+0000    DW       0       ;RESET THE CALIBRATED FLAG
EA92+FF01    DW       15*341/10 ;STEP TIME
EA94+28      DCONF  40, 5, 15    ;RESERVED FOR FUTURE USE, MUST BE ZERO
EA95+00      DB       40      ;RESERVED FOR FUTURE USE, MUST BE ZERO
EA96+AA00    DW       5*341/10 ;HEAD SETTLE TIME
EA98+0000    DW       0       ;DRIVE 3
EA9A+0000    DW       0       ;NUMBER OF TRACKS
EA9C+FF01    DW       15*341/10 ;RESET THE CALIBRATED FLAG

```

ENDIF

```

*****
*
* CONSOLE DRIVER ROUTINES.
*
*****

```

```

*
* ROUTINE USED DEPENDS ON THE VALUE OF CONTYP. POSSIBLE CONTYP
* VALUES ARE LISTED AS FOLLOWS:
*
* CONTYP IS: 0 NOTHING, USED FOR PATCHING TO PROM'S
*            1 PROVIDE FOR 128 BYTES OF PATCH SPACE
*            2 MULTI I/O OR DECISION I DRIVER
*            3 2D/B DRIVER
*            4 DJDMA SERIAL PORT
*            5 SWITCHBOARD SERIAL PORT
*            6 NORTH STAR MOTHERBOARD (2 SERIAL + 1 PARALLEL)
*
*****

```

```

*****
*
* THIS ROUTINE IS AN EXPERIMENT TO REDUCE MISSED AND GARBLED
* CHARACTERS ON CONSOLE OUTPUT.
*
*****

```

```

                IF      CONTYP NE 0

EA9E 79          COSTRP: MOV     A,C           ;STRIP PARITY ON CONOUT
EA9F E67F        ANI      7FH
EAA1 4F          MOV     C,A
EAA2 C3B4EA      JMP      CONOUT

                ENDIF

```

```

*****
*
* THE FOLOWING EQUATES WILL DEFINE THE DECISION I MOTHER
* BOARD I/O OR THE MULTI I/O ENVIRONMENTS IF NEEDED.
*
*****

```

```

FFFF =          MULTIO EQU      (CONTYP EQ 2) OR (LSTTYP GE 2) ;MULTI I/O BOARD USED?

                IF      MULTIO          ;DEFINE MULTI I/O ENVIRONMENT
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
004C =          MCR EQU        MBASE+4
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

IF MULTR3 EQ 0 ;DAISY OUTPUT PORTS ARE DIFFERENT
0048 = DAISIO EQU MBASE ; FOR DECISION I AND MULTI I/O.
0049 = DAISI1 EQU MBASE+1 ;THESE TWO ARE THE DECISION I PORTS
ELSE
DAISIO EQU MBASE+1 ; AND THESE ARE THE MULTI I/O'S.
DAISI1 EQU MBASE
ENDIF

```

```
; DEFINE DAISY 0 STATUS INPUT BITS
```

```

0001 = RIBBON EQU 01H ;END OF RIBBON
0002 = PAPER EQU 02H ;PAPER OUT
0004 = COVER EQU 04H ;COVER OPEN
0008 = PFRDY EQU 08H ;PAPER FEED READY
0010 = CRRDY EQU 10H ;CARRIAGE READY
0020 = PWRDY EQU 20H ;PRINT WHEEL READY
0040 = CHECK EQU 40H ;PRINTER CHECK (ERROR)
0080 = READY EQU 80H ;PRINTER READY

```

```
; DEFINE DAISY 0 STATUS INPUT BITS FOR DIABLO HYTYPE II DRIVER
```

```

1020 = CRSTRD EQU 1020H ;CARRIAGE READY
0810 = PFSTRD EQU 810H ;PAPER FEED READY
2040 = PWSTRD EQU 2040H ;PRINT WHEEL READY

```

```
; DEFINE DAISY 0 OUTPUT BITS
```

```

0001 = D9 EQU 01H ;DATA BIT 9
0002 = D10 EQU 02H ;DATA BIT 10
0004 = D11 EQU 04H ;DATA BIT 11
0008 = D12 EQU 08H ;DATA BIT 12

0010 = PFSTB EQU 10H ;PAPER FEED STROBE
0020 = CRSTB EQU 20H ;CARRIAGE STROBE
0040 = PWSTB EQU 40H ;PRINT WHEEL STROBE
0080 = REST EQU 80H ;PRINTER RESTORE (RIBBON LIFT ON MULTI I/O)

```

```
; DEFINE CLOCK SELECT BITS
```

```

0040 = RLIFT EQU 40H ;RIBBON LIFT
0080 = PSELECT EQU 80H ;SELECT (NOT USED BY DIABLO)

```

```
; DEFINE MODEM CONTROL REGISTER BITS
```

0001 = DTRENB EQU 1 ;DTR ENABLE
0002 = RTSENB EQU 2 ;RTS ENABLE

; 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

; DEFINE SPECIAL CONSTANTS FOR THE HYTYP II DRIVER

000A = CPERI EQU 10 ;DEFAULT TO 10 CHARACTERS PER INCH
0006 = LPERI EQU 6 ;DEFAULT LINES PER INCH
0078 = HINC EQU 120 ;HORIZONTAL INCREMENTS PER INCH
0030 = VINC EQU 48 ;VERTICAL INCREMENTS PER INCH
00A0 = NUMTABS EQU 160 ;NUMBER OF HORIZONTAL TABS
0400 = MAXCHRS EQU 1024 ;MAXIMUM NUMBER OF PRINTER CHARACTERS TO QUEUE
0630 = MAXRGT EQU 1584 ;MAXIMUM CARRIAGE POSITION
006E = DFRMLN EQU 110 ;FORMS LENGTH TIMES 10
0000 = AUTOLF EQU 0 ;DEFAULT TO NOIAUTO LINE FEED

ENDIF

\*\*\*\*\*
\*
\* CONTYP: 2 MULTI I/O OR DECISION I CONSOLE DRIVER
\*
\*\*\*\*\*

IF CONTYP EQ 2

\*\*\*\*\*
\*
\* THIS DRIVER ON COLD BOOT WILL INSPECT BITS 1-3 OF THE SENSE
\* SWITCHES. IF THE VALUE FOUND IS IN THE RANGE 0-6 THEN THE
\* CONSOLE BAUD RATE WILL BE TAKEN FROM THE RATE TABLE. OTHERWISE
\* THE BAUD RATE WILL BE SET FROM THE DEFCON WORD WHICH IS FOUND
\* JUST BELOW THE REGULAR CBIOS JUMP TABLE. THE STANDARD DIVISOR
\* TABLE IS GIVEN BELOW.
\*
\* SENSE SWITCH: 123 (0 = OFF, 1 = ON)
\* 000 = 110
\* 001 = 300
\* 010 = 1200
\* 011 = 2400
\* 100 = 4800
\* 101 = 9600
\* 110 = 19200
\* DEFCON = 9600
\*
\* NOTE: IF YOU ARE USING A MULTIO THEN THE SWITCHES WILL NOT BE
\* AVAILABLE SO THE BAUD RATE WILL BE TAKEN FROM DEFCON.
\*

```

*
*****
*
* DUE TO ITS LENGTH, THE TTYSET ROUTINE DRIVER IS BELOW THE CBOOT
* CBOOT ROUTINE.
*
*****

```

```

*****
*
* READ A CHARACTER FROM THE SERIAL PORT.
*
*****

```

```

EAA5 CDD3EA CONIN: CALL SELCON ;SELECT CONSOLE

EAA8 DB4D CONIN1: IN LSR ;READ STATUS REGISTER
EAAA E601 ANI DR ;WAIT TILL CHARACTER READY
EAAC CA8EA JZ CONIN1
EAAF DB48 IN RBR ;READ CHARACTER
EAB1 E67F ANI 7FH ;STRIP PARITY
EAB3 C9 RET

```

```

*****
*
* OUTPUT A CHARACTER TO SERIAL PORT.
*
*****

```

```

EAB4 CDD3EA CONOUT: CALL SELCON ;SELECT CONSOLE

EAB7 DB4D CONOUT1: IN LSR ;READ STATUS
EAB9 E620 ANI THRE ;WAIT TILL TRANSMITTER BUFFER EMPTY
EABB CAB7EA JZ CONOUT1
EABE 79 MOV A,C ;CHARACTER IS IN (C)
EABF D348 OUT THR ;OUTPUT TO TRANSMITTER BUFFER
EAC1 C9 RET

```

```

*****
*
* RETURN SERIAL PORT STATUS. RETURNS ZERO IF CHARACTER IS NOT
* READY TO BE READ. ELSE RETURNS 255 IF READY.
*
*****

```

```

EAC2 CDD3EA CONIST: CALL SELCON ;SELECT CONSOLE

EAC5 DB4D IN LSR ;READ STATUS REGISTER
EAC7 E601 ANI DR
EAC9 C8 RZ ;NO CHARACTER READY
EACA 3EFF MVI A,0FFH ;CHARACTER READY
EACC C9 RET

ENDIF ;MULTI I/O OR DECISION I

```

\*\*\*\*\*  
\*  
\* CONTYP: 3 2DB CONSOLE DRIVER \*  
\*  
\*\*\*\*\*

```
IF CONTYP EQ 3
CONOUT: JMP FDCOUT ;CONSOLE OUTPUT
CONIN: JMP FDCIN ;CONSOLE INPUT
CONIST: CALL FDTSTAT ;CONSOLE STATUS
MVI A,0FFH
RZ
INR A
RET
ENDIF ;2DB
```

\*\*\*\*\*  
\*  
\* CONTYP: 4 DJDMA CONSOLE DRIVER \*  
\*  
\*\*\*\*\*

```
IF CONTYP EQ 4
CONOUT: LXI H,DMCHAN ;COMMAND FOR SERIAL OUTPUT
MVI M,SEROUT
INX H
MOV M,C
JMP DOCMD
CONIN: LXI H,SERIN+1 ;SERIAL INPUT STATUS
XRA A
CI2: CMP M ;WAIT TILL 40H DEPOSITED AT 3FH
JZ CI2
MOV M,A ;CLEAR STATUS
DCX H ;POINT TO INPUT DATA
MVI A,7FH ;FOR MASKING OUT PARITY
ANA M
RET
CONIST: LDA SERIN+1 ;PICK UP SERIAL INPUT STATUS
ORA A
RZ ;IF ZERO THEN NO CHARACTER READY
MVI A,0FFH ;SET CHARACTER READY
RET
ENDIF
```

\*\*\*\*\*  
\*  
\* CONTYP: 5 SWITCHBOARD AS SERIAL CONSOLE \*  
\*  
\*\*\*\*\*



```

IF CONTYP EQ 5
SWBASE EQU 0 ;BASE OF THE SWITCHBOARD
CONIST: IN SWBASE+2 ;GET THE FIRST PORTS STATUS
ANI 4 ;MASK THE DATA READY BITS
RZ ;RETURN CONSOLE NOT READY
MVI A,0FFH
TTYSET: RET ;NULL TERMINAL INITIALIZATION
CONIN: IN SWBASE+2 ;GET SWITCHBOARD STATUS
ANI 4 ;TEST FOR DATA READY
JZ CONIN
IN SWBASE ;GET A CHARACTER
ANI 7FH ;STRIP OFF PARITY
RET
CONOUT: IN SWBASE+2 ;CHECK STATUS
ANI 8 ;WAIT TILL OUTPUT BUFFER EMPTY
JZ CONOUT
MOV A,C ;WRITE A CHARACTER
OUT SWBASE
RET

```

ENDIF

```

*****
*
* MULTIO/WUNDERBUSS GROUP SELECT ROUTINES
*
*****

```

IF (CONTYP EQ 2) OR (LSTTYP GE 2) ;NEED GROUP SELECT ROUTINES?

```

EACD 3A44EA SELG0: LDA GROUP ;SELECT GROUP ZERO
EAD0 D34F OUT GRPSEL
EAD2 C9 RET
EAD3 3A44EA SELCON: LDA GROUP ;SELECT CONSOLE GROUP
EAD6 F601 ORI CONGRP
EAD8 D34F OUT GRPSEL
EADA C9 RET
EADB 3A44EA SELRDR: LDA GROUP ;SELECT READER/PUNCH GROUP
EADE F602 ORI 5-LSTGRP ;USE 'OTHER' SERIAL PORT
EAE0 D34F OUT GRPSEL
EAE2 C9 RET
EAE3 3A44EA SELLST: LDA GROUP ;SELECT PRINTER GROUP
EAE6 F603 ORI LSTGRP
EAE8 D34F OUT GRPSEL
EAEA C9 RET

```

ENDIF

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

```
EAEB 00      AUTOST: DB      0          ;REVISION 0 STRUCTURE
EAEC 14      DB      100H - (LOW $) ;THE REST OF THE PAGE IS USED FOR THIS STUFF

EAED 00      AUTOFLG:DB      0          ;AUTO COMMAND FEATURE ENABLE FLAG

EAEE F2EA    COLDMES:DW      COLDCM      ;POINTER TO THE COLD START COMMAND
EAF0 F3EA    WARMES: DW      WARMCM      ;POINTER TO THE WARM START COMMAND
```

```
*****
*
* IF THERE IS A COMMAND INSERTED HERE, IT WILL BE PASSED TO THE
* CCP IF THE AUTO FEATURE IS ENABLED. FOR EXAMPLE:
*
*      COLDCM: DB      COLDEND-COLDCM
*              DB      'MBASIC MYPROG'
*      COLDEND EQU      $
*
* WILL EXECUTE MICROSOFT BASIC, AND MBASIC WILL EXECUTE THE
* "MYPROG" BASIC PROGRAM. NOTE: THE COMMAND LINE MUST BE IN
* UPPER CASE FOR MOST COMMANDS.
*
*****
```

```
EAF2 01      COLDCM: DB      COLDEND-COLDCM      ;LENGTH OF COLD BOOT COMMAND
              DB      ' '                      ;COLD BOOT COMMAND GOES HERE
EAF3 =      COLDEND EQU      $

EAF3 01      WARMCM: DB      WARMEND-WARMCM      ;LENGTH OF WARM BOOT COMMAND
              DB      ' '                      ;WARM BOOT COMMAND GOES HERE
EAF4 =      WARMEND EQU      $
```

```
*****
*
* AT THE FIRST PAGE BOUNDARY FOLLOWING THE CBIOS WE HAVE A SERIES OF
* POINTERS THAT POINT TO VARIOUS INTERNAL TABLES. AT THE START OF
* EACH OF THESE TABLES WE HAVE A REVISION BYTE AND A LENGTH BYTE.
* THE REVISION BYTE IS THE CURRENT REVISION NUMBER FOR THAT
* PARTICULAR STRUCTURE AND THE LENGTH BYTE IS THE LENGTH OF THAT
* STRUCTURE. THIS LENGTH DOES NOT INCLUDE THE REVISION BYTE NOR
* THE LENGTH BYTE ITSELF.
*
*      REVISION      DESCRIPTION
*      E.0          1 AND 2 DEFINED
*
*****
```

```

*      E.3      THIS TABLE IS MOVED TO A PAGE BOUNDRY
*      E.3      0, 3 AND 4 DEFINED
*
* THE POINTERS DEFINED SO FAR ARE AS FOLLOWS:
*
* 0)  HIGH BYTE IS THE PAGE NUMBER OF THE CBIOS.  LOW BYTE IS
*     THE CBIOS REVISION NUMBER.  USED TO DETERMINE POINTER
*     STRUCTURE.
*
* 1)  THIS POINTS TO THE DRIVE CONFIGURATION TABLE.
*
* 2)  THIS POINTS TO THE I/O CONFIGURATION BYTES FOR THE SERIAL
*     DRIVERS.  EG, THE CONSOLE, PRINTER, READER, AND PUNCH
*     DEVICES.
*
* 3)  THIS POINTS TO THE DRIVE PARAMETER TABLE FOR DJDMA FLOPPY
*     DISK DRIVES.  IF NO DJDMA IS PRESENT THEN THIS POINTER IS
*     NULL (0).
*
* 4)  THIS POINTS TO THE AUTOSTART COMMAND STRUCTURES.  USED TO
*     AUTOMATICALLY INVOKE A COMMAND ON COLD OR WARM BOOT
*
* 5)  THIS WILL BE A NULL (0) POINTER.  IT MARKS THE END OF THE
*     TABLE.
*
*****

```

```

IF      $ GT BIOS+256      ;TEST FOR CODE OVERLAP
'Fatal error, pointer table placement.'
ELSE
EAF4    DS      BIOS+256-$      ;START AT A PAGE BOUNDRY
        ENDIF

EB00 EA    DB      HIGH ($-1)      ;CBIOS PAGE NUMBER
EB01 35    DB      REVNUM          ;CBIOS REVISION NUMBER
EB02 39EA  DW      DRCONF          ;DRIVE CONFIGURATION TABLE POINTER
EB04 3FEA  DW      IOCONF          ;I/O CONFIGURATION TABLE POINTER
EB06 4CEA  IF      (MAXDM NE 0) OR (MAXMF NE 0) ;DJDMA PRESENT?
        DW      DMARAP          ;DRIVE PARAMETER TABLE POINTER
        ELSE
        DW      0
        ENDIF

EB08 EBEA  DW      AUTOST          ;AUTO COMMAND STRUCTURE POINTER
EB0A 0000  DW      0              ;END OF TABLE MARKER

```

```

*****
*
* THE FOLLOWING CODE PERFORMS THE MAPPING OF LOGICAL TO PHYSICAL
* SERIAL I/O DEVICES.  THE PHYSICAL ENTRY POINTS ARE CONIN, CONOUT,
* CONIST, RDRIN, PUNOUT, LSTOUT, AND LSTOST.  THESE ENTRY POINTS
* ARE MAPPED VIA THE INTEL STANDARD I/O BYTE (IOBYTE) AT LOCATION 3
* IN THE BASE PAGE TO THE LOW LEVEL DEVICE DRIVERS.
*
* NOTE:  A NAMING CONVENTION HAS BEEN CHOSEN TO REDUCE LABEL
* COLLISIONS.  THE FIRST THREE CHARACTERS OF A NAME INDICATE THE
* DEVICE DRIVERS NAME, THE FOLLOWING THREE CHARACTERS INDICATED THE

```

\* FUNCTION PERFORMED BY THAT PARTICULAR DEVICE ROUTINE. THE DEVICE  
 \* NAMES ARE DEFINED AND DESCRIBED IN THE "AN INTRODUCTION TO CP/M  
 \* FEATURES AND FACILITIES" MANUAL IN THE SECTION ON THE STAT  
 \* COMMAND AND IN THE "CP/M INTERFACE GUIDE" IN THE IOBYTE SECTION.  
 \* THE DEVICE FUNCTION POSTFIXES ARE AS FOLLOWS.

- \* DEVSET INITIAL DEVICE SETUP AND INITIALZATION
- \* DEVIN READ ONE CHARACTER FROM THE DEVICE
- \* DEVOUT WRITE ONE CHARACTER TO THE DEVICE
- \* DEVIST RETURN THE DEVICE CHARACTER INPUT READY STATUS
- \* DEVOST RETURN THE DEVICE CHARACTER OUTPUT READY STATUS

\* THE SETUP ROUTINE INITIALIZES THE DEVICE AND RETURNS. THE INPUT  
 \* ROUTINE RETURNS ONE CHARACTER IN THE A REGISTER (PARITY RESET).  
 \* THE OUTPUT ROUTINE WRITE ONE CHARACTER FROM THE C REGISTER. THE  
 \* INPUT STATUS ROUTINE RETURNS IN THE A REGISTER A 0 IF THE DEVICE  
 \* DOES NOT HAVE A CHARACTER READY FOR INPUT FOR 0FFH IF A CHARACTER  
 \* IS READY FOR INPUT. THE OUTPUT STATUS ROUTINE RETURNS IN THE A  
 \* REGISTER A 0 IF THE DEVICE IS NOT READY ACCEPT A CHARACTER AND A  
 \* 0FFH IF THE DEVICE IS READY. THE INPUT AND OUTPUT ROUTINES  
 \* SHOULD WAIT UNTILL THE DEVICE IS READY FOR THE DESIRED OPERATION  
 \* BEFORE THE DOING THE OPERATION AND RETURNING.

\* NOT ALL OF THESE FUNCTIONS NEED TO BE IMPLEMENTED FOR ALL THE  
 \* DEVICES. THE FOLLOWING IS A TABLE OF THE ENTRY POINTS NEEDED FOR  
 \* EACH DEVICE HANDLER.

DEVICE NAME	SETUP	INPUT	OUTPUT	INPUT STATUS	OUTPUT STATUS
CON:		CONIN	CONOUT	CONIST	
RDR:		RDRIN		RDRIST	
PUN:			PUNOUT		
LST:			LSTOUT		LSTOST
TTY:	TTYSET	TTYIN	TTYOUT	TTYIST	TTYOST
CRT:	CRTSET	CRTIN	CRTOUT	CRTIST	CRTOST
UC1:	UC1SET	UC1IN	UC1OUT	UC1IST	
PTR:	PTRSET	PTRIN		PTRIST	
UR1:	UR1SET	UR1IN		UR1IST	
UR2:	UR2SET	UR2IN		UR2IST	
PTP:	PTPSET		PTPOUT		
UP1:	UP1SET		UP1OUT		
UP2:	UP2SET		UP2OUT		
LPT:	LPTSET		LPTOUT		LPTOST
UL1:	UL1SET		UL1OUT		UL1OST

\* THE CONIN, CONOUT, CONIST, RDRIN, RDRIST, PUNOUT, LSTOUT, AND  
 \* LSTOST ROUTINES ARE THE LOGICAL DEVICE DRIVER ENTRY POINTS  
 \* PROVIDED BY THIS DEVICE MAPPER. THE OTHER ENTRY NAMES MUST BE  
 \* PROVIDED BY THE PHYSICAL DEVICE DRIVERS.

\*\*\*\*\*

IF CONTYP EQ 6

;/I/O BYTE IMPLEMENTED FOR NORTH STAR  
; DRIVERS. OTHER DRIVERS TO FOLLOW

CONIN: MVI E,1

CALL REDIR

TTYIN CRTIN

DM DM

DM DM

CONOUT: MVI E,1

CALL REDIR

TTYOUT CRTOUT

DM DM

DM DM

CONIST: MVI E,1

CALL REDIR

TTYIST CRTIST

DM DM

DM DM

RDRIN: MVI E,7

CALL REDIR

TTYIN PTRIN

DM DM

DM DM

RDRIST: MVI E,7

CALL REDIR

TTYIST PTRIST

DM DM

DM DM

PUNOUT: MVI E,5

CALL REDIR

TTYOUT PTPOUT

DM DM

DM DM

ENDIE

LSTOST: MVI E,3

CALL REDIR

DIOUT DIOUT

DM DM

DM DM

EB0C 1E03

EB0E CD26EB

EB11 42EB

EB13 42EB

EB15 5FEB

EB17 5FEB

EB19 1E03

CONSOLE INPUT

IOBYTE: 76543210

CON: = TTY: XXXXXX00

CON: = CRT: XXXXXX01

CON: = BAT: XXXXXX10

CON: = UCI: XXXXXX11

CONSOLE OUTPUT

IOBYTE: 76543210

CON: = TTY: XXXXXX00

CON: = CRT: XXXXXX01

CON: = BAT: XXXXXX10

CON: = UCI: XXXXXX11

CONSOLE INPUT

IOBYTE: 76543210

CON: = TTY: XXXXXX00

CON: = CRT: XXXXXX01

CON: = BAT: XXXXXX10

CON: = UCI: XXXXXX11

CONSOLE INPUT

IOBYTE: 76543210

CON: = TTY: XXXXXX00

CON: = CRT: XXXXXX01

CON: = BAT: XXXXXX10

CON: = UCI: XXXXXX11

CONSOLE OUTPUT

IOBYTE: 76543210

CON: = TTY: XXXXXX00

CONSOLE INPUT

IOBYTE: 76543210

CON: = TTY: XXXXXX00

CON: = CRT: XXXXXX01

CON: = BAT: XXXXXX10

CON: = UCI: XXXXXX11

CONSOLE OUTPUT

IOBYTE: 76543210

CON: = TTY: XXXXXX00

```

EB1B CD26EB      CALL      REDIR      ;          IOBYTE: 76543210
EB1E 3FEB        DW         OKIOST    ;LST: = TTY:   00XXXXXX
EB20 3FEB        DW         OKIOST    ;LST: = CRT:   01XXXXXX
EB22 3AEB        DW         DIAOST    ;LST: = LPT:   10XXXXXX
EB24 3AEB        DW         DIAOST    ;LST: = UL1:   11XXXXXX

EB26 3A0300      REDIR:  LDA      IOBYTE      ;GET THE INTEL STANDARD IOBYTE
EB29 07          REDIR0: RLC                      ;SHIFT THE NEXT FIELD IN
EB2A 1D          DCR         E          ;BUMP THE SHIFT COUNT
EB2B C229EB      JNZ        REDIR0

EB2E E606        REDIR1: ANI      110B      ;MASK THE REDIRECTION FIELD
EB30 5F          MOV         E,A      ;MAKE THE WORD TABLE OFFSET
EB31 1600        MVI         D,0
EB33 E1          POP         H          ;GET THE TABLE BASE
EB34 19          DAD         D          ;OFFSET INTO OUR TABLE
EB35 7E          MOV         A,M      ;LOAD THE LOW LEVEL I/O ROUTINE POINTER
EB36 23          INX         H
EB37 66          MOV         H,M
EB38 6F          MOV         L,A
EB39 E9          PCHL

EB3A DB02        DIAOST: IN        2          ;WAIT UNTIL READY
EB3C E680        ANI         80H
EB3E C8          RZ
EB3F 3EFF        OKIOST: MVI      A,0FFH    ;OK NOW
EB41 C9          RET

EB42 DB02        OKOUT:  IN        2          ;WAIT UNTIL READY
EB44 E608        ANI         8
EB46 CA42EB      JZ         OKOUT
EB49 DB05        OKOUT1: IN       5          ;BUFFER FULL?
EB4B E601        ANI         1
EB4D CA49EB      JZ         OKOUT1
EB50 79          MOV         A,C          ;OUTPUT THE CHARACTER
EB51 D300        OUT         0
EB53 C9          RET

EB54 DB02        DIOUTA: IN       2          ;OUTPUT FROM PTR-GET STATUS
EB56 E680        ANI         80H
EB58 CA54EB      JZ         DIOUTA
EB5B 79          MOV         A,C          ;OUTPUT THE CHARACTER
EB5C D301        OUT         1
EB5E C9          RET

EB5F CD54EB      DIOUT:  CALL     DIOUTA      ;OUTPUT THE CHARACTER
EB62 3A85EB      LDA      COUNT
EB65 3D          DCR         A
EB66 3285EB      STA      COUNT
EB69 C0          RNZ
EB6A 3E4E        MVI         A,78
EB6C 3285EB      STA      COUNT
EB6F 0E03        MVI         C,AETX
EB71 CD54EB      CALL     DIOUTA
EB74 DB02        DIOUTB: IN       2          ;INPUT FROM DIABLO
EB76 E640        ANI         40H

```

EB78 CA74EB  
EB7B DB01  
EB7D E67F  
EB7F FE06  
EB81 C274EB  
EB84 C9

JZ DIOUTB  
IN 1  
ANI 7FH  
CPI AACK  
JNZ DIOUTB  
RET

;STRIP PARITY

EB85 32

COUNT: DB 50

```

*****
*
* CONTYP: 1          BLANK SPACE FOR CONSOLE DRIVER
*
* THE DRIVER ENTRIES CONOUT, CONIN, CONIST ARE DEFINED IN THE CP/M
* ALTERNATION GUIDE.  EG.  INPUT PARAMETERS ARE IN REGISTER C AND
* RESULTS ARE RETURNED IN REGISTER A.  THE TTYSET ROUTINE IS USED
* FOR INITIALIZATION CODE.  IT SHOULD EXECUTE A RET WHEN COMPLETE.
*
* THE TTYSET ROUTINE COULD BE PLACED JUST BELOW THE CBOOT ROUTINE.
* THIS SPACE (BELOW CBOOT) IS RECYLED FOR USE AS A DISK BUFFER
* AFTER CBOOT IS DONE.
*
*****

```

```

          IF          CONTYP EQ 1          ;USER DEFINED IO AREA
TTYSET EQU          $          ;CONSOLE INITIALIZATION
CONOUT EQU          $          ;CONSOLE OUTPUT
CONIN EQU           $          ;CONSOLE INPUT
CONIST EQU          $          ;CONSOLE INPUT STATUS
          JMP        $
          DS         125
          ENDIF
                                     ;USER IO

```

```

*****
*
* CONYTP: 6          NORTH STAR
*
* THE FOLLOWING CODE IMPLEMENTS THE NORTH STAR CONSOLE I/O SYSTEM.
* THIS SYSTEM IS FOR USERS WHO PURCHASE A MORROW DESIGNS DISK
* SYSTEM TO REPLACE THEIR NORTH STAR DISK SYSTEM.  THE MAPPING OF
* THE LOGICAL TO PHYSICAL ENTRY POINTS IS PERFORMED AS FOLLOWS:
*
* DEVICE NAME          LEFT SERIAL  RIGHT SERIAL  PARALLEL PORT
*
* CONSOLE              CON: =  TTY:    CRT:    UC1:
* READER               RDR: =  TTY:    PTR:    UR1:
* PUNCH                PUN: =  TTY:    PTP:    UP1:
* LIST                 LST: =  TTY:    CRT:    UL1:
*
* FOR EXAMPLE, TO USE A PRINTER CONNECTED TO THE RIGHT SERIAL PORT,
* USE THE CP/M COMMAND:
*
*       STAT LST:=CRT:
*
* LIKewise, THE CP/M COMMAND "STAT LST:=UL1:" IS USED IF YOU HAVE A

```

\* PRINTER CONNECTED TO THE PARALLEL PORT.  
\*

\*\*\*\*\*

```

                IF          CONTYP EQ 6                ;USE NORTH STAR I/O?
NSLDAT EQU      2
NSLSTA EQU      3                ;LEFT SERIAL PORT DATA PORT
                                ;LEFT SERIAL PORT STATUS PORT
NSRDAT EQU      4
NSRSTA EQU      5                ;RIGHT SERIAL PORT DATA PORT
                                ;RIGHT SERIAL PORT STATUS PORT
NSSTBE EQU      1
NSSRBR EQU      2                ;TRANSMITTER BUFFER EMPTY STATUS BIT
                                ;RECIEVER BUFFER READY STATUS BIT

                                ;SEE THE 8251 DATA SHEETS FOR MORE
                                ; CONFIGURATION INFORMATION.

NSLIN1 EQU      0CEH
NSRIN1 EQU      0CEH                ;LEFT SERIAL PORT INITIALIZATION # 1
                                ;RIGHT SERIAL PORT INITIALIZATION # 1
                                ;76543210 BIT DEFINATIONS
                                ;11001110 DEFAULT CONFIGURATION
                                ;XXXXXX00 SYNCHRONOUS MODE
                                ;XXXXXX01 1X CLOCK RATE
                                ;XXXXXX10 16X CLOCK RATE
                                ;XXXXXX11 64X CLOCK RATE
                                ;XXXX00XX 5 BIT CHARACTERS
                                ;XXXX01XX 6 BIT CHARACTERS
                                ;XXXX10XX 7 BIT CHARACTERS
                                ;XXXX11XX 8 BIT CHARACTERS
                                ;XXX0XXXX PARITY DISBABLE
                                ;XXX1XXXX PARITY ENABLE
                                ;XX0XXXXX ODD PARITY GENERATION/CHECK
                                ;XX1XXXXX EVEN PARITY GENERATION/CHECK
                                ;00XXXXXX INVALID
                                ;01XXXXXX 1 STOP BIT
                                ;10XXXXXX 1.5 STOP BITS
                                ;11XXXXXX 2 STOP BITS

NSLIN2 EQU      37H
NSRIN2 EQU      37H                ;LEFT SERIAL PORT INITIALIZATION # 2
                                ;RIGHT SERIAL PORT INITIALIZATION # 2
                                ;76543210 BIT DEFINATIONS
                                ;00110111 DEFAULT CONFIGURATION
                                ;XXXXXXXX1 ENABLE TRANSMITTER
                                ;XXXXXXXX1X ASSERT DTR*
                                ;XXXXX1XX ENABLE RECIEVER
                                ;XXXX1XXX SEND BREAK CHARACTER, TXD LOW
                                ;XXX1XXXX RESET PE, OE, FE ERROR FLAGS
                                ;XX1XXXXX ASSERT RTS*
                                ;X1XXXXXX INTERNAL RESET
                                ;1XXXXXXX ENTER HUNT MODE (FOR SYNC)

NSPDAT EQU      0
NSPSTA EQU      6                ;PARALLEL DATA PORT
                                ;PARALLEL STATUS PORT

NSPRBR EQU      1                ;RECIEVER BUFFER READY STATUS BIT
    
```



```

NSPTBE EQU 2 ;TRANSMITTER BUFFER EMPTY STATUS BIT
NSRAM EQU 0C0H ;NORTH STAR MEMORY PARITY PORT,
; SET TO 0 FOR NO NORTH STAR RAM

```

```

*****
*
* LEFT SERIAL PORT ROUTINES. USE TTY: DEVICE.
*
*****

```

```

TTYIN: ;READ A CHARACTER
    IN     NSLSTA
    ANI    NSSRBR
    JZ     TTYIN ;WAIT TILL A CHARACTER IS READY
    IN     NSLDAT ;GET THE CHARACTER
    ANI    7FH ;STRIP PARITY
    RET

```

```

TTYOUT: ;WRITE A CHARACTER
    IN     NSLSTA
    ANI    NSSTBE
    JZ     TTYOUT ;WAIT TILL THE BUFFER IS EMPTY
    MOV    A,C ;WRITE THE CHARACTER
    OUT   NSLDAT
    RET

```

```

TTYIST: ;RETURN INPUT BUFFER STATUS
    IN     NSLSTA
    ANI    NSSRBR
    RZ     ;RETURN NOT READY
    MVI    A,0FFH
    RET ;THERE IS A CHARACTER READY

```

```

TTYOST: ;RETURN OUTPUT BUFFER STATUS
    IN     NSLSTA
    ANI    NSSTBE
    RZ     ;RETURN NOT READY
    MVI    A,0FFH
    RET ;RETURN READY

```

```

*****
*
* RIGHT SERIAL PORT ROUTINES. USE CRT:, PTR:, AND PTP: DEVICES.
*
*****

```

```

CRTIN: ;READ A CHARACTER
PTRIN:
    IN     NSRSTA
    ANI    NSSRBR
    JZ     CRTIN ;WAIT TILL A CHARACTER IS READY
    IN     NSRDAT ;GET THE CHARACTER
    ANI    7FH ;STRIP PARITY
    RET

```

```

CRTOUT:                ;WRITE A CHARACTER
PTPOUT:
    IN      NSRSTA
    ANI     NSSTBE
    JZ      CRTOUT      ;WAIT TILL THE BUFFER IS EMPTY
    MOV     A,C          ;WRITE THE CHARACTER
    OUT     NSRDAT
    RET

```

```

CRTIST:                ;RETURN INPUT BUFFER STATUS
PTRIST:
    IN      NSRSTA
    ANI     NSSRBR
    RZ      ;RETURN NOT READY
    MVI     A,0FFH
    RET      ;THERE IS A CHARACTER READY

```

```

CRTOST:                ;RETURN OUTPUT BUFFER STATUS
    IN      NSRSTA
    ANI     NSSTBE
    RZ      ;RETURN NOT READY
    MVI     A,0FFH
    RET      ;RETURN READY

```

```

*****
*
* PARALLEL PORT ROUTINES.  USE UC1:, UR1:, UR2:, UP1:, UP2:, LPT:,
* AND UL1: DEVICES.
*
*****

```

```

UC1IN:                ;READ A CHARACTER
UR1IN:
UR2IN:
    IN      NSPSTA
    ANI     NSPRBR
    JZ      UC1IN      ;WAIT TILL A CHARACTER IS READY
    IN      NSPDAT     ;GET THE CHARACTER
    PUSH   PSW
    MVI     A,30H      ;RESET THE PARALLEL INPUT FLAG
    OUT     NSPSTA
    POP    PSW
    ANI     7FH        ;STRIP PARITY
    RET

```

```

UC1OUT:                ;WRITE A CHARACTER
UP1OUT:
UP2OUT:
LPTOUT:
UL1OUT:
    IN      NSPSTA
    ANI     NSPTBE
    JZ      UC1OUT     ;WAIT TILL THE BUFFER IS EMPTY
    MVI     A,20H      ;RESET THE PARALLEL OUTPUT FLAG
    OUT     NSPSTA
    MOV     A,C        ;WRITE THE CHARACTER, STROBE BIT 7

```

```

NSPOUT: ORI    80H
        OUT    NSPDAT
        ANI    7FH
        OUT    NSPDAT
        ORI    80H
        OUT    NSPDAT
        RET
    
```

```

UC1IST:                                ;RETURN INPUT BUFFER STATUS
UR1IST:
UR2IST:
        IN     NSPSTA
        ANI    NSPRBR
        RZ
        MVI    A,0FFH
        RET
    
```

```

LPTOST:                                ;RETURN OUTPUT BUFFER STATUS
UL1OST:
        IN     NSPSTA
        ANI    NSPTBE
        RZ
        MVI    A,0FFH
        RET
    
```

```

        ENDIF                                ;NORTH STAR I/O CONFIGURATION
    
```

\*\*\*\*\*

```

*
* LST: DEVICE DRIVER ROUTINES.
*
* ROUTINE USED DEPENDS ON THE VALUE OF LSTTYP. POSSIBLE LSTTYP
* VALUES ARE LISTED AS FOLLOWS:
*
* LSTTYP IS:  0      NOTHING, USED FOR PATCHING TO PROM'S
*             1      PROVIDE FOR 128 BYTES OF PATCH SPACE
*             2      MULTIO SERIAL, NO PROTOCOL
*             3      MULTIO SERIAL, CLEAR TO SEND PROTOCOL
*             4      MULTIO SERIAL, DATA SET READY PROTOCOL
*             5      MULTIO SERIAL, XON/XOFF PROTOCOL
*
    
```

\*\*\*\*\*

\*\*\*\*\*

```

*
* LSTTYP: 1      BLANK SPACE FOR PRINTER DRIVER
*
* THE DRIVER ENTRIES LSTOUT AND LSTOST ARE DEFINED IN THE CP/M
* ALTERNATION GUIDE. EG. INPUT PARAMETERS ARE IN REGISTER C AND
* RESULTS ARE RETURNED IN REGISTER A. THE LSTSET ROUTINE IS USED
* FOR INITIALIZATION CODE. IT SHOULD EXECUTE A RET WHEN COMPLETE.
*
* THE LSTSET ROUTINE COULD BE PLACED JUST BELOW THE CBOOT ROUTINE.
* THIS SPACE (BELOW CBOOT) IS RECYLED FOR USE AS A DISK BUFFER
* AFTER CBOOT IS DONE.
*
    
```

\*\*\*\*\*

\*\*\*\*\*

```

IF          LSTTYP EQ 1
LSTSET EQU $ ;PRINTER INITIALIZATION
LSTOUT EQU $ ;PRINTER OUTPUT
LSTOST EQU $ ;PRINTER OUTPUT STATUS
JMP $
DS 125
ENDIF ;USER IO

```

\*\*\*\*\*

```

*
* LSTTYP: 2, 3, 4, OR 5 SERIAL PRINTER, MULTI PROTOCOL
*
*****

```

\* CODE REMOVED

\*\*\*\*\*

```

*
* 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).
*
*****

```

```

EB86 218000 GOCPM: LXI H,BUFF ;SET UP INITIAL DMA ADDRESS
EB89 CDF9EB CALL SETDMA
EB8C 3EC3 MVI A,(JMP) ;INITIALIZE JUMP TO WARM BOOT
EB8E 320000 STA WBOT
EB91 320500 STA ENTRY ;INITIALIZE JUMP TO BDOS
EB94 2103EA LXI H,WBOOTE ;SET UP LOW MEMORY ENTRY TO CBIOS WARM BOOT
EB97 220100 SHLD WBOT+1
EB9A 2106DC LXI H,BDOS+6 ;SET UP LOW MEMORY ENTRY TO BDOS
EB9D 220600 SHLD ENTRY+1
EBA0 AF XRA A ;A <- 0
EBA1 32F6F4 STA BUFSEC ;SET BUFFER TO UNKNOWN STATE
EBA4 3290ED STA BUFWRN ;SET BUFFER NOT DIRTY FLAG
EBA7 32F2F4 STA ERROR ;CLEAR BUFFER ERROR FLAG

EBAA 3ADBEB LDA CWFLG ;GET COLD/WARM BOOT FLAG
EBAD B7 ORA A
EBAE 21EEEE LXI H,COLDMES ;POINTER TO INITIAL COLD COMMAND
EBB1 CAB7EB JZ CLDCMND
EBB4 21F0EA LXI H,WARMES ;POINTER TO INITIAL WARM COMMAND
EBB7 5E CLDCMND:MOV E,M ;DO ONE LEVEL OF INDIRECTION
EBB8 23 INX H
EBB9 56 MOV D,M
EBBA 1A LDAX D ;GET COMMAND LENGTH
EBBB 3C INR A ;BUMP LENGTH TO INCLUDE LENGTH BYTE ITSELF
EBBC 2107D4 LXI H,CCP+7 ;COMMAND BUFFER (INCLUDES LENGTH BYTE)
EBBF 4F MOV C,A ;SET UP FOR BLOCK MOVE
EBC0 0600 MVI B,0
EBC2 CDE6EE CALL MOVBYT ;MOVE COMMAND TO INTERNAL CCP BUFFER
EBC5 3ADBEB LDA CWFLG ;FIGURE OUT WHETHER OR NOT TO SEND MESSAGE
EBC8 B7 ORA A

```

```

EBC9 3AEDEA      LDA      AUTOFLG
EBCC CAD0EB      JZ        CLDBOT
EBCF 1F          RAR
EBD0 1F          CLDBOT: RAR
EBD1 3A0400      LDA      CDISK      ;JUMP TO CP/M WITH CURRENTLY SELECTED DISK IN C
EBD4 4F          MOV      C,A
EBD5 DA00D4      JC        CCP        ;ENTER CP/M, SEND MESSAGE
EBD8 C303D4      JMP      CCP+3      ;ENTER CP/M, NO MESSAGE

EBDB 00          CWFLG:  DB      0          ;COLD/WARM BOOT FLAG
    
```

```

*****
*
* WBOOT LOADS IN ALL OF CP/M EXCEPT THE CBIOS, THEN INITIALIZES *
* SYSTEM PARAMETERS AS IN COLD BOOT. SEE THE COLD BOOT LOADER *
* LISTING FOR EXACTLY WHAT HAPPENS DURING WARM AND COLD BOOTS. *
*
*****
    
```

```

EBDC 310001      WBOOT:  LXI      SP,TPA      ;SET UP STACK POINTER
EBDF 3E01        MVI      A,1
EBE1 32DBEB      STA      CWFLG      ;SET COLD/WARM BOOT FLAG

EBE4 2600        MVI      H,WMDRIVE      ;MOVE DRIVE TO WARM BOOT OFF OF INTO (H)
EBE6 2E00        MVI      L,D$WBOOT      ;PEFORM WARM BOOT OPERATION
EBE8 CD60EE      CALL     JUMPER
EBEB D286EB      JNC      GOCPM        ;NO ERROR

EBEE 76          HLT
EBEF 00          DB      0          ;HALT COMPUTER

EBF0 C3DCEB      JMP      WBOOT      ;IN CASE USER RESTARTS THE COMPUTER
    
```

```

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

```

EBF3 60          SETSEC: MOV      H,B          ;ENTER WITH SECTOR NUMBER IN (BC)
EBF4 69          MOV      L,C
EBF5 22EBF4      SHLD     CPMSEC
EBF8 C9          DONOP:  RET
    
```

```

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

```

EBF9 60          SETDMA: MOV      H,B          ;ENTER WITH DMA ADDRESS IN (BC)
EBFA 69          MOV      L,C
EBFB 2271ED      SHLD     CPMDMA      ;CP/M DMA ADDRESS
EBFE C9          RET
    
```

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

```
EBFF 3A90ED HOME: LDA BUFWRTN ;TEST BUFFER DIRTY FLAG
EC02 B7 ORA A
EC03 C20AEC JNZ DOHOME ;SKIP BUFFER DISABLE IF BUFFER DIRTY
EC06 AF XRA A ;INVALIDATE BUFFER ON HOME CALL
EC07 32F6F4 STA BUFSEC
EC0A 010000 DOHOME: LXI B,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.
*
*****
```

```
EC0D 60 SETTRK: MOV H,B ;ENTER WITH TRACK NUMBER IN (BC)
EC0E 69 MOV L,C
EC0F 22EEF4 SHLD CPMTRK
EC12 C9 RET
```

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

```
EC13 3AEDF4 SECTRAN:LDA CPMDRV ;GET THE DRIVE NUMBER
EC16 67 MOV H,A ;DRIVE IN (H)
EC17 2E01 MVI L,D$STRAN
EC19 C360EE JMP JUMPER ;SEE DEVICE LEVEL SECTOR TRANSLATION ROUTINES
```

```
*****
*
* SETDRV SELECTS THE NEXT DRIVE TO BE USED IN READ/WRITE
* OPERATIONS. IF THE DRIVE HAS NEVER BEEN SELECTED IT CALLS
* A LOW LEVEL DRIVE SELECT ROUTINE THAT SHOULD PERFORM SOME
* SORT OF CHECK IF THE DEVICE IS WORKING. IF NOT WORKING THEN
* IT SHOULD REPORT AN ERROR. IF THE LOGICAL DRIVE HAS BEEN
* SELECTED BEFORE THEN SETDRV JUST RETURNS THE DPH WITHOUT
* CHECKING THE DRIVE.
*
*****
```

```
EC1C 79 SETDRV: MOV A,C ;SAVE THE LOGICAL DRIVE NUMBER
EC1D 32EDF4 STA CPMDRV
EC20 FE04 CPI MAXLOG ;CHECK FOR A VALID DRIVE NUMBER
EC22 D281EC JNC ZRET ;ILLEGAL DRIVE

EC25 7B MOV A,E ;CHECK IF BIT 0 OF (E) = 1
EC26 E601 ANI 1
```

```

EC28 C260EC      JNZ      SETD3      ;DRIVE HAS ALLREADY BEEN ACCESSED

EC2B 61          MOV      H,C          ;MOVE LOGICAL DRIVE INTO (H)
EC2C 2E02        MVI      L,D$SEL1      ;
EC2E CD60EE      CALL     JUMPER      ;CALL LOW LEVEL DRIVE SELECT
EC31 7C          MOV      A,H          ;CHECK IF THE LOW LEVEL DRIVE SELECT RETURNED
EC32 B5          ORA      L          ; ZERO TO INDICATE AN ERROR
EC33 CA81EC      JZ       ZRET         ;YES, AN ERROR SO REPORT TO CP/M

EC36 E5          PUSH     H          ;SAVE DPH ADDRESS
EC37 CD74EC      CALL     GDPH        ;GET ENTRY IF DPH SAVE TABLE
EC3A D1          POP      D          ;DPH -> (DE)
EC3B 73          MOV      M,E          ;PUT ADDRESS OF DPH IN TABLE
EC3C 23          INX     H          ;
EC3D 72          MOV      M,D          ;
EC3E 23          INX     H          ;
EC3F 71          MOV      M,C          ;PUT SECTOR SIZE IN TABLE
EC40 23          INX     H          ;
EC41 7E          MOV      A,M          ;CHECK IF BAD MAP HAS EVER BEEN READ FOR THIS
EC42 B7          ORA      A          ; DRIVE
EC43 CCA2EC      CZ       GETBAD      ;NEVER BEEN READ SO READ IN BAD MAP
EC46 EB          XCHG                    ;DPH -> (HL)

EC47 79          SETD0:  MOV     A,C          ;MOVE SECTOR SIZE CODE INTO (A)
EC48 3222ED      STA     SECSIZ      ;SAVE SECTOR SIZE
EC4B AF          XRA     A          ;
EC4C 0D          SETD1:  DCR     C          ;CREATE NUMBER OF (128 BYTES/PHYSICAL SECTOR)-1
EC4D CA56EC      JZ     SETD2      ;
EC50 07          RLC                    ;
EC51 F601        ORI     1          ;
EC53 C34CEC      JMP     SETD1      ;
EC56 3260ED      SETD2:  STA     SECPSEC  ;SAVE FOR DEBLOCKING
EC59 3AEDF4      LDA     CPMDRV     ;SAVE CURRENT DRIVE AS OLD DRIVE
EC5C 32FCF4      STA     LASTDRV    ; IN CASE OF SELECT ERRORS
EC5F C9          RET                    ;

EC60 D5          SETD3:  PUSH     D          ;SAVE DPH ADDRESS
EC61 61          MOV     H,C          ;DRIVE IN (H)
EC62 2E03        MVI     L,D$SEL2    ;SELECT DRIVE
EC64 CD60EE      CALL     JUMPER     ;
EC67 CD74EC      CALL     GDPH      ;QUICK SELECT
EC6A D1          POP     D          ;
EC6B 5E          MOV     E,M          ;DPH -> (DE)
EC6C 23          INX     H          ;
EC6D 56          MOV     D,M          ;
EC6E 23          INX     H          ;
EC6F 4E          MOV     C,M          ;SECTOR SIZE -> (C)
EC70 EB          XCHG                    ;DPH -> (HL)
EC71 C347EC      JMP     SETD0      ;

EC74 3AEDF4      GDPH:  LDA     CPMDRV    ;RETURN POINTER TO DPH SAVE AREA
EC77 07          RLC                    ;EACH ENTRY IS 4 BYTES LONG
EC78 07          RLC                    ;
EC79 5F          MOV     E,A          ;
EC7A 1600        MVI     D,0         ;
EC7C 2192EC      LXI     H,DPHTAB   ;DPH SAVE AREA TABLE

```

```

EC7F 19      DAD      D      ;ADD OFFSET
EC80 C9      RET      ;(HL) = DPH SAVE AREA FOR CURRENT DRIVE

EC81 210000  ZRET:   LXI      H,0      ;SELDRV ERROR EXIT
EC84 3AFCF4  LDA      LASTDRV ;GET LAST SELECTED DRIVE
EC87 4F      MOV      C,A
EC88 3A0400  LDA      CDISK    ;PICK UP USER/DRIVE
EC8B E6F0    ANI      0F0H   ;SAVE USER NUMBER
EC8D B1      ORA      C      ;PUT TOGETHER WITH OLD DRIVE
EC8E 320400  STA      CDISK
EC91 C9      RET
    
```

```

*****
*
* DPH SAVE AREA. EACH ENTRY IS 4 BYTES LONG:
* 0 - LSB OF DPH ADDRESS
* 1 - MSB OF DPH ADDRESS
* 2 - SECTOR SIZE CODE (1 = 128, 2 = 256, 3 = 512...)
* 3 - BAD MAP HAS BEEN INITIALIZED (0 = UNINITIALIZED)
*
*****
    
```

```

DPHTAB: REPT      MAXLOG*4
        DB      0
        ENDM

EC92+00  DB      0
EC93+00  DB      0
EC94+00  DB      0
EC95+00  DB      0
EC96+00  DB      0
EC97+00  DB      0
EC98+00  DB      0
EC99+00  DB      0
EC9A+00  DB      0
EC9B+00  DB      0
EC9C+00  DB      0
EC9D+00  DB      0
EC9E+00  DB      0
EC9F+00  DB      0
ECA0+00  DB      0
ECA1+00  DB      0
    
```

```

*****
*
* GETBAD - CHECK IF A DEVICE HAS A BAD MAP. IF THE DEVICE HAS
* A BAD SECTOR MAP THEN APPEND BAD ENTRIES TO END OF BADMAP
* TABLE.
*
*****
    
```

```

ECA2 3601  GETBAD: MVI      M,1      ;SET DRIVE INITIALIZED
ECA4 C5    PUSH     B
ECA5 D5    PUSH     D
ECA6 3AEDF4 LDA      CPMDRV ;PICK UP CURRENT DRIVE
ECA9 67    MOV      H,A      ;CALL DRIVE ROUTINE TO RETURN A POINTER TO
ECAA 2E0A  MVI      L,D$BAD ;THE TRACK AND SECTOR OF THE BAD MAP
    
```



```

ECAC CD60EE          CALL    JUMPER

ECAF 7C             MOV     A,H           ;IF ROUTINE RETURNS 0 THEN THE DEVICE HAS
ECB0 B5             ORA     L           ; NO BAD SECTOR MAP
ECB1 CAE9EC        JZ      BADRET

ECB4 5E             MOV     E,M           ;PICK UP TRACK NUMBER OF BAD SECTOR MAP -> (DE)
ECB5 23             INX     H
ECB6 56             MOV     D,M
ECB7 23             INX     H
ECB8 EB            XCHG
ECB9 22EEF4        SHLD   CPMTRK
ECBC EB            XCHG
ECBD 7E             MOV     A,M           ;PICK UP SECTOR NUMBER OF OF BAD SECTOR MAP
ECBE 23             INX     H
ECBF 66             MOV     H,M
ECC0 6F            MOV     L,A
ECC1 22F0F4        SHLD   TRUESEC
ECC4 CDF0ED        CALL   FILL           ;READ IN BAD SECTOR MAP INTO THE BUFFER
ECC7 D8            RC

ECC8 2A0FED        LHLD   BADPTR           ;PICK UP BAD MAP POINTER
ECCB 116DF5        LXI    D,BUFFER       ;START AT BEGINNING OF BUFFER
ECCE 1A            BADL: LDAX   D           ;PICK UP AN ENTRY FROM THE BUFFER
ECCF B7            ORA     A
ECD0 CAE6EC        JZ      BADE           ;ALL DONE
ECD3 7E             MOV     A,M           ;PICK UP ENTRY FROM BAD MAP TABLE
ECD4 3C            INR     A
ECD5 CAECEC        JZ      OVERFLO        ;BAD MAP OVERFLOW
ECD8 3AEDF4        LDA    CPMDRV        ;PUT DRIVE IN TABLE
ECDB 77            MOV     M,A
ECDC 23             INX     H
ECDD 010800        LXI    B,8
ECE0 CDE6EE        CALL   MOVBYT        ;MOVE THE REST OF INFORMATION INTO THE TABLE
ECE3 C3CEEC        JMP    BADL

ECE6 220FED        BADE:  SHLD   BADPTR           ;RESTORE NEW BAD MAP POINTER
ECE9 D1            BADRET: POP    D
ECEA C1            POP    B
ECEB C9            RET

ECEC 21F5EC        OVERFLO: LXI   H,OMES
ECEB CD05EF        CALL   MESSAGE
ECF2 C3E9EC        JMP    BADRET

ECF5 0D0A424144OMES: DB    0DH, 0AH, 'BAD MAP OVERFLOW!', 0DH, 0AH, 0

ED0B 210000        NOBAD: LXI   H,0           ;USED BY DEVICE DRIVES TO INDICATE NO BAD
ED0E C9            RET           ; SECTOR MAP

ED0F 6DF9        BADPTR: DW    BADMAP           ;POINTER TO NEXT AVAILABLE BAD MAP ENTRY

```

```

*****
*
* WRITE ROUTINE MOVES DATA FROM MEMORY INTO THE BUFFER. IF THE
* DESIRED CP/M SECTOR IS NOT CONTAINED IN THE DISK BUFFER, THE
*
```

```

* BUFFER IS FIRST FLUSHED TO THE DISK IF IT HAS EVER BEEN
* WRITTEN INTO, THEN A READ IS PERFORMED INTO THE BUFFER TO GET
* THE DESIRED SECTOR. ONCE THE CORRECT SECTOR IS IN MEMORY, THE
* BUFFER WRITTEN INDICATOR IS SET, SO THE BUFFER WILL BE
* FLUSHED, THEN THE DATA IS TRANSFERRED INTO THE BUFFER.
*
*****

```

```

ED11 79      WRITE:  MOV     A,C           ;SAVE WRITE COMMAND TYPE
ED12 328AED  STA     WRITTYP
ED15 3E01    MVI     A,1           ;SET WRITE COMMAND
ED17 C31EED  JMP     RWENT

```

```

*****
*
* READ ROUTINE TO BUFFER DATA FROM THE DISK. IF THE SECTOR
* REQUESTED FROM CP/M IS IN THE BUFFER, THEN THE DATA IS SIMPLY
* TRANSFERRED FROM THE BUFFER TO THE DESIRED DMA ADDRESS. IF
* THE BUFFER DOES NOT CONTAIN THE DESIRED SECTOR, THE BUFFER IS
* FLUSHED TO THE DISK IF IT HAS EVER BEEN WRITTEN INTO, THEN
* FILLED WITH THE SECTOR FROM THE DISK THAT CONTAINS THE
* DESIRED CP/M SECTOR.
*
*****

```

```

ED1A AF      READ:   XRA     A           ;SET THE COMMAND TYPE TO READ
                IF     NOSTAND NE 0
ED1B 32E7F4  STA     UNALOC        ;CLEAR UNALLOCATED WRITE FLAG
                ENDF
ED1E 3274ED  RWENT:  STA     RDWR        ;SAVE COMMAND TYPE

```

```

*****
*
* REDWRT CALCULATES THE PHYSICAL SECTOR ON THE DISK THAT
* CONTAINS THE DESIRED CP/M SECTOR, THEN CHECKS IF IT IS THE
* SECTOR CURRENTLY IN THE BUFFER. IF NO MATCH IS MADE, THE
* BUFFER IS FLUSHED IF NECESSARY AND THE CORRECT SECTOR READ
* FROM THE DISK.
*
*****

```

```

ED21 0600    REDWRT: MVI     B,0           ;THE 0 IS MODIFIED TO CONTAIN THE LOG2
ED22 =      SECSIZ EQU     $-1           ; OF THE PHYSICAL SECTOR SIZE/128
                ; ON THE CURRENTLY SELECTED DISK
ED23 2AEBF4  LHLD    CPMSEC        ;GET THE DESIRED CP/M SECTOR #
ED26 7C      MOV     A,H
ED27 E680    ANI     80H           ;SAVE ONLY THE SIDE BIT
ED29 4F      MOV     C,A           ;REMEMBER THE SIDE
ED2A 7C      MOV     A,H
ED2B E67F    ANI     7FH           ;FORGET THE SIDE BIT
ED2D 67      MOV     H,A
ED2E 2B      DCX     H           ;TEMPORARY ADJUSTMENT
ED2F 05      DIVLOOP:DCR    B           ;UPDATE REPEAT COUNT
ED30 CA3DED  JZ     DIVDONE
ED33 B7      ORA     A
ED34 7C      MOV     A,H

```

```

ED35 1F          RAR
ED36 67          MOV      H,A
ED37 7D          MOV      A,L
ED38 1F          RAR          ;DIVIDE THE CP/M SECTOR # BY THE SIZE
                                ; OF THE PHYSICAL SECTORS

ED39 6F          MOV      L,A
ED3A C32FED      JMP      DIVLOOP
ED3D 23          DIVDONE:INX H
ED3E 7C          MOV      A,H
ED3F B1          ORA      C          ;RESTORE THE SIDE BIT
ED40 67          MOV      H,A
ED41 22F0F4      SHLD     TRUESEC      ;SAVE THE PHYSICAL SECTOR NUMBER
ED44 21EDF4      LXI      H,CPMDRV    ;POINTER TO DESIRED DRIVE,TRACK, AND SECTOR
ED47 11F3F4      LXI      D,BUFDRV    ;POINTER TO BUFFER DRIVE,TRACK, AND SECTOR
ED4A 0606        MVI      B,6          ;COUNT LOOP
ED4C 05          DTSLOP:DCR     B      ;TEST IF DONE WITH COMPARE
ED4D CA5BED      JZ       MOVE      ;YES, MATCH. GO MOVE THE DATA
ED50 1A          LDAX    D          ;GET A BYTE TO COMPARE
ED51 BE          CMP      M          ;TEST FOR MATCH
ED52 23          INX      H          ;BUMP POINTERS TO NEXT DATA ITEM
ED53 13          INX      D
ED54 CA4CED      JZ       DTSLOP     ;MATCH, CONTINUE TESTING
    
```

```

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

```

ED57 CDF0ED      CALL     FILL      ;FILL THE BUFFER WITH CORRECT PHYSICAL SECTOR
ED5A D8          RC          ;NO GOOD, RETURN WITH ERROR INDICATION
    
```

```

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

```

ED5B 3AEBF4      MOVE:   LDA      CPMSEC      ;GET THE CP/M SECTOR TO TRANSFER
ED5E 3D          DCR      A          ;ADJUST TO PROPER SECTOR IN BUFFER
ED5F E600        ANI      0          ;STRIP OFF HIGH ORDERED BITS
ED60 =          SECPSEC EQU  $-1      ;THE 0 IS MODIFIED TO REPRESENT THE # OF
                                ; CP/M SECTORS PER PHYSICAL SECTORS
ED61 6F          MOV      L,A          ;PUT INTO HL
ED62 2600        MVI      H,0
ED64 29          DAD      H          ;FORM OFFSET INTO BUFFER
ED65 29          DAD      H
ED66 29          DAD      H
ED67 29          DAD      H
ED68 29          DAD      H
ED69 29          DAD      H
ED6A 29          DAD      H
ED6B 116DF5      LXI      D,BUFFER    ;BEGINNING ADDRESS OF BUFFER
ED6E 19          DAD      D          ;FORM BEGINNING ADDRESS OF SECTGR TO TRANSFER
    
```

```

ED6F EB          XCHG          ;DE = ADDRESS IN BUFFER
ED70 210000      LXI          H,0 ;GET DMA ADDRESS, THE 0 IS MODIFIED T/
                                ;   CONTAIN THE DMA ADDRESS

ED71 =          CPMDMA EQU    $-2
ED73 3E00        MVI          A,0 ;THE ZERO GETS MODIFIED TO CONTAIN
                                ;   A ZERO IF A READ, OR A 1 IF WRITE

ED74 =          RDWR   EQU    $-1
ED75 A7          ANA          A    ;TEST WHICH KIND OF OPERATION
ED76 C280ED      JNZ          INTO ;TRANSFER DATA INTO THE BUFFER
ED79 CDE3EE      OUTOF: CALL    MOV128
ED7C 3AF2F4      LDA          ERROR ;GET THE BUFFER ERROR FLAG
ED7F C9          RET

ED80 EB          INTO: XCHG          ;
ED81 CDE3EE      CALL    MOV128    ;MOVE THE DATA, HL = DESTINATION
                                ;   DE = SOURCE

ED84 3E01        MVI          A,1
ED86 3290ED      STA          BUFWRN ;SET BUFFER WRITTEN INTO FLAG
ED89 3E00        MVI          A,0 ;CHECK FOR DIRECTORY WRITE
ED8A =          WRITTYP EQU    $-1
ED8B 3D          DCR          A    ;TEST FOR A DIRECTORY WRITE
ED8C 3E00        MVI          A,0
ED8E C0          RNZ          ;NO ERROR EXIT
    
```

```

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

```

ED8F 3E00      FLUSH: MVI          A,0 ;THE 0 IS MODIFIED TO REFLECT IF
                                ;   THE BUFFER HAS BEEN WRITTEN INTO

ED90 =          BUFWRN EQU    $-1
ED91 B7        ORA          A    ;TEST IF WRITTEN INTO
ED92 C8        RZ          ;NOT WRITTEN, ALL DONE
ED93 3E09      MVI          A,D$WRITE
ED95 32D3ED    STA          RWOP+1
ED98 CD9FED    CALL    PREP      ;DO THE PHYSICAL WRITE
ED9B 32F2F4    STA          ERROR ;SET UP THE ERROR FLAG
ED9E C9        RET
    
```

```

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

```

ED9F CD83EE    PREP:  CALL    ALT      ;CHECK FOR ALTERNATE SECTORS
EDA2 F3        DI          ;RESET INTERRUPTS
EDA3 AF        XRA          A    ;RESET BUFFER WRITTEN FLAG
EDA4 3290ED    STA          BUFWRN

EDA7 060A      MVI          B,RETRIES ;MAXIMUM NUMBER OF RETRIES TO ATTEMPT
    
```

```

EDA9 C5      RETRYLP:PUSH  B           ;SAVE THE RETRY COUNT
EDAA 2E03    MVI          L,D$SEL2   ;SELECT DRIVE
EDAC CD5CEE  CALL          JUMPBUF
EDAF 2AF8F4  LHL D         ALTTRK       ;TRACK NUMBER -> (HL)
EDB2 7C      MOV          A,H         ;TEST FOR TRACK ZERO
EDB3 B5      ORA          L
EDB4 E5      PUSH         H           ;SAVE TRACK NUMBER
EDB5 2E04    MVI          L,D$HOME
EDB7 CC5CEE  CZ           JUMPBUF
EDBA C1      POP          B           ;RESTORE TRACK #
EDBB 2E05    MVI          L,D$STRK
EDBD CD5CEE  CALL          JUMPBUF
EDC0 2AFAF4  LHL D         ALTSEC       ;SECTOR -> (HL)
EDC3 44      MOV          B,H
EDC4 4D      MOV          C,L
EDC5 2E06    MVI          L,D$SSEC
EDC7 CD5CEE  CALL          JUMPBUF
EDCA 016DF5  LXI          B,BUFFER    ;SET THE DMA ADDRESS
EDCD 2E07    MVI          L,D$SDMA
EDCF CD5CEE  CALL          JUMPBUF
EDD2 2E00    RWOP: MVI          L,0     ;GET OPERATION ADDRESS
EDD4 CD5CEE  CALL          JUMPBUF
EDD7 C1      POP          B           ;RESTORE THE RETRY COUNTER
EDD8 3E00    MVI          A,0         ;NO ERROR EXIT STATUS
EDDA D0      RNC           ;RETURN NO ERROR
EDDB 05      DCR          B           ;UPDATE THE RETRY COUNTER
EDDC 37      STC           ;ASSUME RETRY COUNT EXPIRED
EDDD 3EFF    MVI          A,0FFH     ;ERROR RETURN
EDDF C8      RZ           ;RETURN SAD NEWS
EDE0 78      MOV          A,B
EDE1 FE05    CPI          RETRIES/2
EDE3 C2A9ED  JNZ          RETRYLP       ;TRY AGAIN
EDE6 C5      PUSH         B           ;SAVE RETRY COUNT
EDE7 2E04    MVI          L,D$HOME     ;HOME DRIVE AFTER (RETRIES/2) ERRORS
EDE9 CD5CEE  CALL          JUMPBUF
EDEC C1      POP          B
EDED C3A9ED  JMP          RETRYLP       ;TRY AGAIN

```

```

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

```

```

EDF0 CD8FED  FILL: CALL    FLUSH          ;FLUSH BUFFER FIRST
EDF3 D8      RC           ;CHECK FOR ERROR

```

```

EDF4 11EDF4      LXI      D,CPMDRV      ;UPDATE THE DRIVE, TRACK, AND SECTOR
EDF7 21F3F4      LXI      H,BUFDRV
EDFA 010500      LXI      B,5              ;NUMBER OF BYTES TO MOVE
EDFD CDE6EE      CALL     MOVBYT          ;COPY THE DATA

EE00 3A74ED      LDA      RDWR            ;TEST READ WRITE FLAG
EE03 B7          ORA      A
EE04 CA50EE      JZ       FREAD          ;SKIP WRITE TYPE CHECK IF READING
EE07 3A8AED      LDA      WRITYP         ;0 = ALLOC, 1 = DIR, 2 = UNALLOC

IF NOSTAND NE 0 ;DO NON STANDARD (BUT QUICK AND DIRTY) CHECK
EE0A B7          ORA      A
EE0B C231EE      JNZ     FNALOC          ;SKIP IF NOT AN ALLOCATED WRITE

EE0E 3AE7F4      LDA      UNALOC         ;CHECK UNALLOCATED WRITE IN PROGRESS FLAG
EE11 B7          ORA      A
EE12 CA4BEE      JZ       FWITIN         ;WE ARE DOING AN ALLOCATED WRITE
EE15 2AE7E9      LHL D    CBLOCK         ;GET CURRENT BLOCK ADDRESS
EE18 EB          XCHG
EE19 2AE8F4      LHL D    OBLOCK        ; AND OLD BLOCK ADDRESS
EE1C 7A          MOV     A,D             ;COMPARE OLD VERSUS NEW
EE1D BC          CMP     H
EE1E C247EE      JNZ     AWITIN         ;DIFFERENT, CLEAR UNALLOCATED WRITING MODE
EE21 7B          MOV     A,E
EE22 BD          CMP     L
EE23 C247EE      JNZ     AWITIN
EE26 21EDF4      LXI     H,CPMDRV       ;TEST FOR DIFFERENT DRIVE
EE29 3AEAF4      LDA     UNADRV
EE2C BE          CMP     M
EE2D C247EE      JNZ     AWITIN         ;DRIVE IS DIFFERENT, CLEAR UNALLOCATED MODE
EE30 C9          RET          ;UNALLOCATED WRITE, DO NOTHING...

EE31 3D          FNALOC: DCR     A
EE32 CA47EE      JZ      AWITIN         ;DO A DIRECTORY WRITE
                        ;WE ARE NOW DOING AN UNALLOCATED WRITE
                        ;SAVE CURRENT BLOCK NUMBER
EE35 2AE7E9      LHL D    CBLOCK
EE38 22E8F4      SHLD   OBLOCK
EE3B 3AEDF4      LDA     CPMDRV        ;SAVE DRIVE THAT THIS BLOCK BELONGS TO
EE3E 32EAF4      STA     UNADRV
EE41 3E01        MVI     A,1            ;SET UNALLOCATED WRITE FLAG
EE43 32E7F4      STA     UNALOC
EE46 C9          RET          ; AND WE DO NOTHING ABOUT THE WRITE

EE47 AF          AWITIN: XRA     A      ;CLEAR UNALLOCATED WRITING MODE
EE48 32E7F4      STA     UNALOC

ELSE ;DO STANDARD UNALLOCATED TEST
SUI 2 ;TEST FOR AN UNALLOCATED WRITE
RZ

ENDIF

EE4B 3A22ED      FWITIN: LDA     SECSIZ   ;CHECK FOR 128 BYTE SECTORS
EE4E 3D          DCR     A
EE4F C8          RZ          ;NO DEBLOCKING NEEDED

```

```

EE50 3E08    FREAD: MVI    A,D$READ
EE52 32D3ED  STA    RWOP+1
EE55 CD9FED  CALL   PREP      ;READ THE PHYSICAL SECTOR THE BUFFER
EE58 32F2F4  STA    ERROR     ;SET THE ERROR STATUS
EE5B C9      RET
    
```

```

*****
*
* JUMPBUF, JUMPER ARE USED TO DISPATCH TO A LOW LEVEL DEVICE
* SUBROUTINE. JUMPER IS CALLED WITH THE DRIVE IN (H) AND THE
* ROUTINE NUMBER (SEE DESCRIPTION ABOVE) IN (L). IT PASSES
* ALONG THE (BC) AND (DE) REGISTERS UNALTERED. JUMPBUF IS
* A CALL TO JUMPER WITH THE DRIVE NUMBER FROM BUFDRV.
*
*****
    
```

```

EE5C 3AF3F4  JUMPBUF:LDA   BUFDRV      ;DISPATCH WITH BUFDRV FOR DRIVE
EE5F 67      MOV    H,A

EE60 D5      JUMPER: PUSH   D
EE61 C5      PUSH   B
EE62 E5      PUSH   H
EE63 7C      MOV    A,H      ;LOGICAL DRIVE INTO (A)
EE64 113BEA  LXI    D,DSTTAB  ;DRIVE SPECIFICATION POINTER TABLE
EE67 4F      JUMPL: MOV    C,A      ;SAVE LOGICAL IN (C)
EE68 1A      LDAX  D
EE69 6F      MOV    L,A
EE6A 13      INX   D
EE6B 1A      LDAX  D
EE6C 67      MOV    H,A      ;GET A DST POINTER IN (HL)
EE6D 13      INX   D
EE6E 79      MOV    A,C      ;LOGICAL IN (A)
EE6F 96      SUB    M      ;SUBTRACT FROM FIRST ENTRY IN DST
EE70 D267EE  JNC    JUMPL  ;KEEP SCANNING TABLE TILL CORRECT DRIVER FOUND

EE73 23      INX   H      ;BUMP (HL) TO POINT TO START OF DISPATCH TABLE
EE74 D1      POP   D      ;REAL (HL) -> (DE)
EE75 7B      MOV   A,E      ;MOVE OFFSET NUMBER INTO (A)
EE76 07      RLC                ;EACH ENTRY IS 2 BYTES
EE77 5F      MOV   E,A      ;MAKE AN OFFSET
EE78 1600    MVI   D,0
EE7A 19      DAD   D      ;(HL) = **ROUTINE
EE7B 7E      MOV   A,M      ;PICK UP ADDRESS OF HANDLER FOR SELECTED
EE7C 23      INX   H      ; FUNCTION
EE7D 66      MOV   H,M
EE7E 6F      MOV   L,A      ;(HL) = *ROUTINE
EE7F 79      MOV   A,C      ;LOGICAL IN (A)
EE80 C1      POP   B      ;RESTORE SAVED REGISTERS
EE81 D1      POP   D
EE82 E9      PCHL
    
```

```

*****
*
* CHECK FOR ALTERNATE SECTORS IN BAD SECTOR TABLE. IF AN
* ALTERNATE SECTOR IS FOUND REPLACE ALTRK AND ALTSEC WITH
*
*****
    
```

\* NEW SECTOR NUMBER ELSE PASS ALONG UNALTERED. \*

\*

\*

\*\*\*\*\*

```

EE83 216DF9  ALT:  LXI    H,BADMAP    ;ADDRESS OF BAD MAP -> (HL)
EE86 3AF3F4  LDA    BUFDRV    ;PICK UP DRIVE NUMBER CURRENTLY WORKING ON
EE89 4F      MOV    C,A      ;MOVE DRIVE INTO (C) FOR SPEED IN SEARCH
EE8A EB      ALL:  XCHG
EE8B 2A0FED  LHLD   BADPTR    ;GET BAD MAP POINTER
EE8E EB      XCHG    ; -> (DE)
EE8F 7A      MOV    A,D      ;CHECK IF AT END OF BAD MAP TABLE
EE90 BC      CMP    H
EE91 C2A6EE  JNZ    ALT2      ;STILL MORE
EE94 7B      MOV    A,E
EE95 BD      CMP    L
EE96 C2A6EE  JNZ    ALT2      ;STILL MORE
EE99 2AF4F4  LHLD   BUFTRK   ;NO ALTERNATE SECTOR SO USE SELECTED SECTOR
EE9C 22F8F4  SHLD   ALTTRK
EE9F 2AF6F4  LHLD   BUFSEC
EEA2 22FAF4  SHLD   ALTSEC
EEA5 C9      RET

EEA6 E5      ALT2:  PUSH   H      ;SAVE CURRENT BAD MAP ENTRY ADDRESS
EEA7 79      MOV    A,C      ;MOVE DRIVE INTO (A)
EEA8 BE      CMP    M      ;CHECK IF DRIVE IN TABLE MATCHES
EEA9 C2DBEE  JNZ    ALTMIS   ;DOES NOT MATCH SKIP THIS ENTRY
EEAC 23      INX    H      ;POINT TO LSB OF ALTERNATE TRACK
EEAD 3AF4F4  LDA    BUFTRK   ;PICK UP LSB OF BUFFER TRACK
EEB0 BE      CMP    M
EEB1 C2DBEE  JNZ    ALTMIS
EEB4 23      INX    H      ;POINT TO MSB ALTERNATE TRACK
EEB5 3AF5F4  LDA    BUFTRK+1 ;PICK UP MSB OF BUFFER TRACK
EEB8 BE      CMP    M
EEB9 C2DBEE  JNZ    ALTMIS
EEBC 23      INX    H      ;POINT TO LSB OF ALTERNATE SECTOR
EEBD 3AF6F4  LDA    BUFSEC   ;PICK UP LSB OF BUFFER SECTOR
EEC0 BE      CMP    M
EEC1 C2DBEE  JNZ    ALTMIS
EEC4 23      INX    H      ;POINT TO MSB OF ALTERNATE SECTOR
EEC5 3AF7F4  LDA    BUFSEC+1 ;PICK UP MSB OF BUFFER SECTOR
EEC8 BE      CMP    M
EEC9 C2DBEE  JNZ    ALTMIS   ;FOUND AN ALTERNATE SECTOR
EECC 23      INX    H      ;POINT TO REAL INFO ON THE ALTERNATE SECTOR
EECD 11F8F4  LXI    D,ALTTRK
EED0 EB      XCHG    ;MOVLOP (DE) = SOURCE, (HL) = DEST
EED1 C5      PUSH   B
EED2 010400  LXI    B,4
EED5 CDE6EE  CALL   MOVBYT   ;MOVE ALTERNATE SECTOR INFO IN CORRECT PLACE
EED8 C1      POP    B
EED9 E1      POP    H
EEDA C9      RET

EEDB E1      ALTMIS: POP    H      ;CURRENT ALTERNATE DID NOT MATCH
EEDC 110900  LXI    D,9      ;BUMP POINTER BY THE LENGTH OF AN ENTRY
EEDF 19      DAD    D
EEE0 C38AEE  JMP    ALL      ;LOOP FOR MORE

```



```
*****
*
* MOVER MOVES 128 BYTES OF DATA. SOURCE POINTER IN DE, DEST
* POINTER IN HL.
*
*****
```

```
EEE3 018000 MOV128: LXI      B,128          ;LENGTH OF TRANSFER
EEE6 AF      MOVBYT: XRA      A          ;CHECK IF HOST PROCESSOR IS A Z80
EEE7 C603    ADI      3
EEE9 E2F7EE JPO      Z80MOV        ;YES, ITS A Z80 SO USE BLOCK MOVE
```

```
EEEC 1A      M8080: LDAX   D          ;GET A BYTE OF SOURCE
EED 77      MOV    M,A          ;MOVE IT
EEEE 13     INX   D          ;BUMP POINTERS
EEEF 23     INX   H
EEF0 0B     DCX   B          ;UPDATE COUNTER
EEF1 78     MOV   A,B          ;TEST FOR END
EEF2 B1     ORA   C
EEF3 C2ECE E JNZ   M8080
EEF6 C9     RET
```

```
EEF7 EB      Z80MOV: XCHG          ;SOURCE IN (HL), DESTINATION IN (DE)
EEF8 EDB0    DW     0B0EDH      ;LDIR
EEFA EB      XCHG
EEFB C9      RET
```

```
*****
*
* RETURN DPH POINTER. ENTER WITH (DE) WITH DPH BASE ADDRESS
* AND (A) WITH LOGICAL DRIVE NUMBER. RETURNS WITH DPH ADDRESS
* IN (HL).
*
*****
```

```
EEFC 6F      RETDPH MOV    L,A          ;MOVE LOGICAL DRIVE INTO (L)
EEFD 2600    MVI    H,0
EEFF 29      DAD    H          ;MULTIPLY BY 16 (SIZE OF DPH)
EF00 29      DAD    H
EF01 29      DAD    H
EF02 29      DAD    H
EF03 19      DAD    D          ;(HL) = POINTER TO DPH
EF04 C9      RET
```

```
*****
*
* UTILITY ROUTINE TO OUTPUT THE MESSAGE POINTED AT BY (HL)
* TERMINATED WITH A NULL.
*
*****
```

```
EF05 7E      MESSAGE:MOV   A,M          ;GET A CHARACTER OF THE MESSAGE
EF06 23      INX    H          ;BUMP TEXT POINTER
EF07 B7      ORA    A          ;TEST FOR END
EF08 C8      RZ              ;RETURN IF DONE
```

```

EF09 E5      PUSH      H          ;SAVE POINTER TO TEXT
EF0A 4F      MOV        C,A        ;OUTPUT CHARACTER IN C
EF0B CD0CEA  CALL      COUT       ;OUTPUT THE CHARACTER
EF0E E1      POP        H          ;RESTORE THE POINTER
EF0F C305EF  JMP         MESSAGE      ;CONTINUE UNTIL NULL REACHED

```

```

*****
*
* THE FOLLOWING CODE IS FOR THE DISKUS HARD DISK
*
*****

```

```

; WANT HDC3 OR 4 CONTROLLER INCLUDED ?
0050 =      HDORG      EQU      50H      ;HARD DISK CONTROLLER ORIGIN

0050 =      HDSTAT     EQU      HDORG     ;DISK STATUS
0050 =      HDCNTL     EQU      HDORG     ;DISK CONTROL
0051 =      HDRESLT    EQU      HDORG+1   ;DISK RESULTS
0051 =      HDCMND     EQU      HDORG+1   ;DISK COMMANDS
0052 =      HDSKOMP    EQU      HDORG+2   ;SEEK COMPLETE CLEAR PORT (ON HDC4)
0052 =      HDFUNC     EQU      HDORG+2   ;FUNCTION PORT
0053 =      HDDATA     EQU      HDORG+3   ;DATA PORT

; STATUS PORT (50)

0001 =      TKZERO     EQU      01H      ;TRACK ZERO
0002 =      OPDONE     EQU      02H      ;OPERATION DONE
0004 =      COMPLT     EQU      04H      ;SEEK COMPLETE
0008 =      TMOUT      EQU      08H      ;TIME OUT
0010 =      WFAULT     EQU      10H      ;WRITE FAULT
0020 =      DRVRDY     EQU      20H      ;DRIVE READY
0040 =      INDEX      EQU      40H      ;DELTA INDEX

; CONTROL PORT (50)

0001 =      HDFREN     EQU      01H      ;ENABLE EXTERNAL DRIVERS
0002 =      HDRUN      EQU      02H      ;ENABLE CONTROLLERS STATE MACHINE
0004 =      HDCLOK     EQU      04H      ;CLOCK SOURCE CONTROL BIT, HIGH = DISK
0008 =      HDWPRT     EQU      08H      ;WRITE PROTECT A DRIVE

; RESULT PORT (51)

0002 =      RETRY      EQU      02H      ;RETRY FLAG

; COMMAND PORT (51)

0000 =      IDBUFF     EQU      0        ;INITIALIZE DATA BUFFER POINTER
0001 =      RSECT      EQU      1        ;READ SECTOR
0005 =      WSECT      EQU      5        ;WRITE SECTOR
0008 =      ISBUFF     EQU      8        ;INITIALIZE HEADER BUFFER POINTER

; FUNCTION PORT (52)

0004 =      PSTEP      EQU      04H      ;STEP BIT
00FB =      NSTEP      EQU      0FFH-PSTEP ;STEP BIT MASK

```

00FC = NULL EQU 0FCH ;NULL COMMAND

; MISC CONSTANTS

0004 = HDRLEN EQU 4 ;SECTOR HEADER LENGTH  
 0200 = SECLEN EQU 512 ;SECTOR DATA LENGTH

\*\*\*\*\*  
 \*  
 \* DEVICE SPECIFICATION TABLE FOR HDCA CONTROLLER DRIVER \*  
 \*  
 \*\*\*\*\*

EF12 03	HDDST:	DB	MAXHD*HDLOG	;NUMBER OF LOGICAL DRIVES
EF13 29EF		DW	HDWARM	;WARM BOOT
EF15 5FEF		DW	HDTRAN	;SECTOR TRANSLATION
EF17 63EF		DW	HDLDRV	;FIRST TIME SELECT
EF19 ABEF		DW	HDDR	;GENERAL SELECT
EF1B BEEF		DW	HDHOME	;HOME CURRENT SELECTED DRIVE
EF1D CCEF		DW	HDSEEK	;SEEK TO SELECTED TRACK
EF1F 00F0		DW	HDSEC	;SELECT SECTOR
EF21 F0EF		DW	HDDMA	;SET DMA ADDRESS
EF23 1EF0		DW	HDREAD	;READ A SECTOR
EF25 53F0		DW	HDWRITE	;WRITE A SECTOR
EF27 0BED		DW	NOBAD	;NO BAD SECTOR MAP
EF29 CDA2EF	HDWARM:	CALL	DIVLOG	;GET PHYSICAL DRIVE NUMBER IN (C)
EF2C AF		XRA	A	
EF2D 2100D2		LXI	H,CCP-200H	;INITIAL DMA ADDRESS
EF30 E5		PUSH	H	
EF31 32EFF0		STA	HEAD	;SELECT HEAD ZERO
EF34 3C		INR	A	; 1 -> (A)
EF35 F5		PUSH	PSW	;SAVE FIRST SECTOR - 1
EF36 CDB1EF		CALL	HDD2	;SELECT DRIVE
EF39 0E00		MVI	C,0	
EF3B CDBEEF		CALL	HDHOME	;HOME THE DRIVE
EF3E F1	HDWRLD:	POP	PSW	;RESTORE SECTOR
EF3F E1		POP	H	;RESTORE DMA ADDRESS
EF40 3C		INR	A	
EF41 32F0F0		STA	HDSECT	
EF44 FE0D		CPI	13	;PAST BDOS ?
EF46 C8		RZ		;YES, ALL DONE
EF47 24		INR	H	;UPDATE DMA ADDRESS
EF48 24		INR	H	
EF49 22ECF0		SHLD	HDADD	
EF4C E5		PUSH	H	
EF4D F5		PUSH	PSW	
EF4E 01000A	HDWRRD:	LXI	B,RETRIES*100H+0	;RETRY COUNTER
EF51 C5	HDWR:	PUSH	B	;SAVE THE RETRY COUNT
EF52 CD1EF0		CALL	HDREAD	;READ THE SECTOR
EF55 C1		POP	B	
EF56 D23EEF		JNC	HDWRLD	;TEST FOR ERROR
EF59 05		DCR	B	;UPDATE THE ERROR COUNT
EF5A C251EF		JNZ	HDWR	;KEEP TRYING IF NOT TOO MANY ERRORS
EF5D 37		STC		;ERROR FLAG
EF5E C9		RET		

```

EF5F 60      HDTRAN: MOV      H,B      ;SECTOR TRANSLATION IS HANDLED VIA
EF60 69      MOV      L,C      ; PHYSICAL SECTOR HEADER SKEWING
EF61 23      INX      H
EF62 C9      RET

EF63 32EBF0   HDLDRV: STA      HDCUR      ;SAVE LOGICAL DISK
EF66 CDA2EF   CALL     DIVLOG     ;DIVIDE BY LOGICAL DISKS PER DRIVE
EF69 79      MOV      A,C
EF6A 32EEF0   STA      HDDISK     ;SAVE NEW PHYSICAL DRIVE
EF6D CDD2F0   CALL     HDPTR      ;GET TRACK POINTERS
EF70 7E      MOV      A,M      ;GET CURRENT TRACK
EF71 3C      INR      A          ;CHECK IF -1
EF72 C295EF   JNZ      HDL2      ;NOPE, ALLREADY ACCESSED
EF75 F6FC     ORI      NULL      ;SELECT DRIVE
EF77 D352     OUT     HDFUNC
EF79 3E05     MVI     A,HDFREN+HDCLOK ;ENABLE DRIVERS
EF7B D350     OUT     HDCNTL
EF7D 0EEF     MVI     C,239      ;WAIT 2 MINUTES FOR DISK READY
EF7F 210000   LXI     H,0
EF82 2B      HDTDEL: DCX     H
EF83 7C      MOV     A,H
EF84 B5      ORA     L
EF85 CCA0EF   CZ      DCRC
EF88 CA81EC   JZ      ZRET      ;DRIVE NOT READY ERROR
EF8B DB50     IN     HDSTAT   ;TEST IF READY YET
EF8D E620     ANI     DRVRDY
EF8F C282EF   JNZ     HDTDEL

                IF      NOT FUJITSU
                LXI     H,0      ;TIME ONE REVOLUTION OF THE DRIVE
                MVI     C,INDEX
                IN     HDSTAT
                ANA     C
                MOV     B,A      ;SAVE CURRENT INDEX LEVEL IN B
HDINXD1: IN    HDSTAT
                ANA     C
                CMP     B      ;LOOP UNTILL INDEX LEVEL CHANGES
                JZ      HDINXD1
HDINDX2: INX  H
                IN     HDSTAT   ;START COUNTING UNTILL INDEX RETURNS TO
                ANA     C      ; PREVIOUS STATE
                CMP     B
                JNZ     HDINDX2

                IF      M10
                DAD     H      ;MEMOREX M10'S HAVE 40 MS HEAD SETTLE
                ENDIF      ;HL*2

                IF      M26
                XRA     A      ;SHUGART M26'S HAVE 30 MS HEAD SETTLE
                MOV     A,H    ;HL/2 + HL (SAME AS HL*1.5)
                RAR
                MOV     D,A
                MOV     A,L
                RAR

```

```

MOV     E,A
DAD     D
ENDIF

```

```

SHLD   SETTLE           ;SAVE THE COUNT FOR TIMEOUT DELAY
ENDIF

```

```

EF92 CDBEEF           CALL   HDHOME

```

```

EF95 3AEBF0   HDL2:  LDA   HDCUR           ;LOAD LOGICAL DRIVE
EF98 11FDF4   LXI   D,DPHHD0           ;START OF HARD DISK DPH'S
EF9B 0E03     MVI   C,3             ;HARD DISK SECTOR SIZE EQUALS 512 BYTES
EF9D C3FCEE   JMP   RETDPH

```

```

EFA0 0D       DCRC:  DCR   C           ;CONDITIONAL DECREMENT C ROUTINE
EFA1 C9       RET

```

```

EFA2 0E00     DIVLOG: MVI   C,0
EFA4 D603     DIVLX:  SUI   HDLOG
EFA6 D8       RC
EFA7 0C       INR   C
EFA8 C3A4EF   JMP   DIVLX

```

```

EFAB 32EBF0   HDDRV: STA   HDCUR
EFAE CDA2EF   CALL  DIVLOG           ;GET THE PHYSICAL DRIVE #
EFB1 79       HDD2:  MOV   A,C
EFB2 32EEF0   STA   HDDISK           ;SELECT THE DRIVE
EFB5 F6FC     ORI   NULL
EFB7 D352     OUT  HDFUNC
EFB9 3E0F     MVI   A,HDFREN+HDRUN+HDCLOK+HDWPRT ;WRITE PROTECT
EFBB D350     OUT  HDCNTL
EFBD C9       RET

```

```

EFBE CDD2F0   HDHOME: CALL  HDPTR           ;GET TRACK POINTER
EFC1 3600     MVI   M,0             ;SET TRACK TO ZERO
EFC3 DB50     IN    HDSTAT        ;TEST STATUS
EFC5 E601     ANI   TKZERO         ;AT TRACK ZERO ?
EFC7 C8       RZ          ;YES

```

```

IF NOT FUJITSU
HDSTEPO: IN   HDSTAT           ;TEST STATUS
ANI   TKZERO         ;AT TRACK ZERO ?
JZ   HDDELAY
MVI   A,1
STC
CALL  ACCOK           ;TAKE ONE STEP OUT
JMP  HDSTEPO

```

```
ELSE
```

```

EFC8 AF       XRA   A
EFC9 C3DDEF   JMP   ACCOK
ENDIF

```

```

IF NOT FUJITSU
HDDELAY: LHLD  SETTLE           ;GET HDDELAY

```

```

DELOOP: DCX      H                ;WAIT 20MS
         MOV      A,H
         ORA      L
         INX      H
         DCX      H
         JNZ      DELOOP
         RET
         ENDF

EFCC CDD2F0      HDSEEK: CALL     HDPTR      ;GET POINTER TO CURRENT TRACK
EFCF 5E          MOV      E,M            ;GET CURRENT TRACK
EFD0 71          MOV      M,C            ;UPDATE THE TRACK
EFD1 7B          MOV      A,E            ;NEED TO SEEK AT ALL ?
EFD2 91          SUB      C
EFD3 C8          RZ
EFD4 3F          CMC                    ;GET CARRY INTO DIRECTION
EFD5 DADAEF     JC      HDTRK2
EFD8 2F          CMA
EFD9 3C          INR      A

EFDA C3DDEF     HDTRK2: JMP      ACCOK
                   ELSE
HDTRK2: CALL     ACCOK
                   JMP      HDDELAY
                   ENDF

EFDD 47          ACCOK:  MOV      B,A      ;PREP FOR BUILD
EFDE CDDDF0     CALL     BUILD
EFE1 E6FB       SLOOP:  ANI      NSTEP    ;GET STEP PULSE LOW
EFE3 D352       OUT      HDFUNC        ;OUTPUT LOW STEP LINE
EFE5 F604       ORI      PSTEP        ;SET STEP LINE HIGH
EFE7 D352       OUT      HDFUNC        ;OUTPUT HIGH STEP LINE
EFE9 05         DCR      B              ;UPDATE REPEAT COUNT
EFEA C2E1EF     JNZ      SLOOP          ;KEEP GOING THE REQUIRED # OF TRACKS
EFED C3F6EF     JMP      WSDONE

EFF0 60         HDDMA:  MOV      H,B      ;SAVE THE DMA ADDRESS
EFF1 69         MOV      L,C
EFF2 22ECF0     SHLD     HDADD
EFF5 C9         RET

EFF6 DB50       WSDONE: IN      HDSTAT    ;WAIT FOR SEEK COMPLETE TO FINISH
EFF8 E604       ANI      COMPLT
EFAA CAF6EF     JZ      WSDONE
EFFD DB52       IN      HDSKOMP        ;CLEAR SDONE BIT ON AN HDCA4
EFFF C9         RET

HDSEC: IF      M26
MVI      A,01FH ;FOR COMPATIBILITY WITH CBIOS REVS.
; 2.3 AND 2.4
ANA      C      ;MASK IN SECTOR NUMBER (0-31)
CZ      GETSPT  ;TRANSLATE SECTOR 0 TO SECTOR 32
STA      HDSECT ;SAVE TRANSLATED SECTOR NUMBER (1-32)
MVI      A,0E0H ;GET THE HEAD NUMBER
ANA      C
RLC

```

```

                RLC
                RLC
                STA     HEAD           ;SAVE THE HEAD NUMBER
GETSPT: MVI     A,HDSPT
                RET

                ELSE

F000 79      HDSEC:  MOV     A,C
F001 CD15F0  CALL    DIVSPT
F004 C615    ADI     HDSPT
F006 A7      ANA     A
F007 CC11F0  CZ      GETSPT
F00A 32F0F0  STA     HDSECT
F00D 79      MOV     A,C
F00E 32EFF0  STA     HEAD
F011 3E15    GETSPT: MVI     A,HDSPT
F013 0D      DCR     C
F014 C9      RET

F015 0E00    DIVSPT: MVI     C,0
F017 D615    DIVSX:  SUI     HDSPT
F019 D8      RC
F01A 0C      INR     C
F01B C317F0  JMP     DIVSX
                ENDIF

F01E CD9CF0  HDREAD: CALL    HDPREP
F021 D8      RC
F022 AF      XRA     A
F023 D351    OUT     HDCMND
F025 2F      CMA
F026 D353    OUT     HDDATA
F028 D353    OUT     HDDATA
F02A 3E01    MVI     A,RSECT           ;READ SECTOR COMMAND
F02C D351    OUT     HDCMND
F02E CD82F0  CALL    PROCESS
F031 D8      RC
F032 AF      XRA     A
F033 D351    OUT     HDCMND
F035 0680    MVI     B,SECLN/4
F037 2AECF0  LHLD    HDADD
F03A DB53    IN      HDDATA
F03C DB53    IN      HDDATA
F03E DB53    RTLOOP: IN     HDDATA           ;MOVE FOUR BYTES
F040 77      MOV     M,A
F041 23      INX    H
F042 DB53    IN      HDDATA
F044 77      MOV     M,A
F045 23      INX    H
F046 DB53    IN      HDDATA
F048 77      MOV     M,A
F049 23      INX    H
F04A DB53    IN      HDDATA
F04C 77      MOV     M,A
F04D 23      INX    H

```

```

F04E 05          DCR      B
F04F C23EF0     JNZ      RTLOOP
F052 C9         RET

F053 CD9CF0     HDWRITE:CALL  HDPREP          ;PREPARE HEADER
F056 D8         RC
F057 AF         XRA      A
F058 D351       OUT      HDCMND
F05A 2AECF0     LHLD     HDADD
F05D 0680       MVI      B, SECLN/4
F05F 7E         WTLOOP: MOV     A, M          ;MOVE 4 BYTES
F060 D353       OUT      HDDATA
F062 23         INX      H
F063 7E         MOV      A, M
F064 D353       OUT      HDDATA
F066 23         INX      H
F067 7E         MOV      A, M
F068 D353       OUT      HDDATA
F06A 23         INX      H
F06B 7E         MOV      A, M
F06C D353       OUT      HDDATA
F06E 23         INX      H
F06F 05         DCR      B
F070 C25FF0     JNZ      WTLOOP
F073 3E05       MVI      A, WSECT          ;ISSUE WRITE SECTOR COMMAND
F075 D351       OUT      HDCMND
F077 CD82F0     CALL     PROCESS
F07A D8         RC
F07B 3E10       MVI      A, WFAULT
F07D A0         ANA      B
F07E 37         STC
F07F C8         RZ
F080 AF         XRA      A
F081 C9         RET

F082 DB50       PROCESS: IN    HDSTAT          ;WAIT FOR COMMAND TO FINISH
F084 47         MOV      B, A
F085 E602       ANI      OPDONE
F087 CA82F0     JZ      PROCESS
F08A 3E07       MVI      A, HDFREN+HDRUN+HDCLOK ;WRITE PROTECT
F08C D350       OUT      HDCNTL
F08E DB50       IN      HDSTAT
F090 E608       ANI      TMOUT          ;TIMED OUT ?
F092 37         STC
F093 C0         RNZ
F094 DB51       IN      HDRESLT
F096 E602       ANI      RETRY          ;ANY RETRIES ?
F098 37         STC
F099 C0         RNZ
F09A AF         XRA      A
F09B C9         RET

F09C DB50       HDPREP: IN    HDSTAT
F09E E620       ANI      DRVRDY
F0A0 37         STC
F0A1 C0         RNZ

```



```

F0A2 3E08      MVI      A,ISBUFF      ;INITIALIZE POINTER
F0A4 D351      OUT      HDCMND
F0A6 CDDDF0    CALL     BUILD
F0A9 F60C      ORI      0CH
F0AB D352      OUT      HDFUNC
F0AD 3AEFF0    LDA      HEAD
F0B0 D353      OUT      HDDATA      ;FORM HEAD BYTE
F0B2 CDD2F0    CALL     HDPTR      ;GET POINTER TO CURRENT DRIVES TRACK
F0B5 7E        MOV      A,M          ;FORM TRACK BYTE
F0B6 D353      OUT      HDDATA
F0B8 A7        ANA      A
F0B9 0680      MVI      B,80H
F0BB CAC0F0    JZ       ZKEY
F0BE 0600      MVI      B,0
F0C0 3AF0F0    LDA      HDSECT      ;FORM SECTOR BYTE
F0C3 D353      OUT      HDDATA
F0C5 78        MOV      A,B
F0C6 D353      OUT      HDDATA
F0C8 3E07      MVI      A,HDFREN+HDRUN+HDCLOK ;WRITE PROTECT
F0CA D350      OUT      HDCNTL
F0CC 3E0F      MVI      A,HDFREN+HDRUN+HDCLOK+HDWPRT ;WRITE PROTECT
F0CE D350      OUT      HDCNTL
F0D0 AF        XRA      A
F0D1 C9        RET

F0D2 2AEF0    HDPTR:  LHLD     HDDISK      ;GET A POINTER TO THE CURRENT DRIVES
F0D5 2600      MVI      H,0          ; TRACK POSITION
F0D7 EB        XCHG
F0D8 21F1F0    LXI      H,HDTRAK
F0DB 19        DAD      D
F0DC C9        RET

F0DD 3AEFF0    BUILD:  LDA      HEAD      ;BUILD A CONTROLLER COMMAND BYTE
F0E0 17        RAL
F0E1 17        RAL
F0E2 17        RAL
F0E3 17        RAL
F0E4 21EEF0    LXI      H,HDDISK
F0E7 B6        ORA      M
F0E8 EEF0      XRI      0FH
F0EA C9        RET

F0EB 00        HDCUR:  DB      0          ;CURRENT LOGICAL DISK
F0EC 0000      HDADD:  DW      0          ;DMA ADDRESS
F0EE 00        HDDISK: DB      0          ;CURRENT PHYSICAL DISK NUMBER
F0EF 00        HEAD:   DB      0          ;CURRENT PHYSICAL HEAD NUMBER
F0F0 00        HDSECT: DB      0          ;CURRENT PHYSICAL SECTOR NUMBER

F0F1 FF        HDTRAK: DB      0FFH      ;TRACK POINTER FOR EACH DRIVE
F0F2 FF        DB      0FFH      ;ALL DRIVE DEFAULT TO AN UNCALIBRATED
F0F3 FF        DB      0FFH      ; STATE (FF)
F0F4 FF        DB      0FFH

F0F5 0000      SETTLE: DW      0          ;TIME DELAY CONSTANT FOR HEAD SETTLE

```

```

ENDIF

```

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

```

IF      MAXFD NE 0      ;INCLUDE DISCUS 2D ?
FDORIG EQU 0F800H      ;ORIGIN OF DISK JOCKEY PROM
FDBOOT EQU FDORIG+00H  ;DISK JOCKEY 2D INITIALIZATION
FDCIN  EQU FDORIG+03H  ;DISK JOCKEY 2D CHARACTER INPUT ROUTINE
FDCOUT EQU FDORIG+06H  ;DISK JOCKEY 2D CHARACTER OUTPUT ROUTINE
FDHOME EQU FDORIG+09H  ;DISK JOCKEY 2D TRACK ZERO SEEK
FDSEEK EQU FDORIG+0CH  ;DISK JOCKEY 2D TRACK SEEK ROUTINE
FDSEC  EQU FDORIG+0FH  ;DISK JOCKEY 2D SET SECTOR ROUTINE
FDDMA  EQU FDORIG+12H  ;DISK JOCKEY 2D SET DMA ADDRESS
FDREAD EQU FDORIG+15H  ;DISK JOCKEY 2D READ ROUTINE
FDWRITE EQU FDORIG+18H ;DISK JOCKEY 2D WRITE ROUTINE
FDSEL  EQU FDORIG+1BH  ;DISK JOCKEY 2D SELECT DRIVE ROUTINE
FDTSTAT EQU FDORIG+21H ;DISK JOCKEY 2D TERMINAL STATUS ROUTINE
FDSTAT EQU FDORIG+27H  ;DISK JOCKEY 2D STATUS ROUTINE
FDERR  EQU FDORIG+2AH  ;DISK JOCKEY 2D ERROR, FLASH LED
FDDEN  EQU FDORIG+2DH  ;DISK JOCKEY 2D SET DENSITY ROUTINE
FDSIDE EQU FDORIG+30H  ;DISK JOCKEY 2D SET SIDE ROUTINE
FDRAM  EQU FDORIG+400H ;DISK JOCKEY 2D RAM ADDRESS
DBLSID EQU 20H        ;SIDE BIT FROM CONTROLLER
IO      EQU FDORIG+3F8H ;START OF I/O REGISTERS
DREG   EQU IO+1
CMDREG EQU IO+4
CLRCMD EQU 0D0H
```

```
*****
*
* DEVICE SPECIFICATION TABLE FOR THE DISK JOCKEY 2D/B
*
*****
```

```

FDDST: DB MAXFD      ;NUMBER OF LOGICAL DRIVES
        DW FDWARM    ;WARM BOOT
        DW FDTRAN    ;SECTOR TRANSLATION
        DW FDLDRV    ;SELECT DRIVE 1
        DW FDSEL2    ;SELECT DRIVE 2
        DW FDLHOME   ;HOME DRIVE
        DW FDSEEK    ;SEEK TO SPECIFIED TRACK
        DW FDSSEC    ;SET SECTOR
        DW FDDMA     ;SET DMA ADDRESS
        DW FDREAD    ;READ A SECTOR
        DW FDWRITE   ;WRITE A SECTOR
        DW NOBAD     ;NO BAD SECTOR MAP
```

```
*****
*
* FLOPPY DISK WARM BOOT LOADER
*
*****
```

```

*****
FDWARM: MOV      C,A
        CALL     FDSEL      ;SELECT DRIVE A
        MVI     C,0        ;SELECT SIDE 0
        CALL     FDSIDE
WRMFAIL:CALL     FDHOME     ;TRACK 0, SINGLE DENSITY
        JC      WRMFAIL    ;LOOP IF ERROR

                                ;THE NEXT BLOCK OF CODE RE-INITIALIZES
                                ; THE WARM BOOT LOADER FOR TRACK 0
                                ;INITIALIZE THE SECTOR TO READ - 2
        MVI     A,5-2
        STA     NEWSEC
        LXI     H,CCP-100H  ;FIRST REVOLUTION DMA - 100H
        SHLD    NEWDMA
                                ;LOAD ALL OF TRACK 0

T0BOOT: MVI     A,5-2      ;FIRST SECTOR - 2
NEWSEC  EQU     $-1
        INR     A          ;UPDATE SECTOR #
        INR     A
        CPI     27         ;SIZE OF TRACK IN SECTORS + 1
        JC     NOWRAP     ;SKIP IF NOT AT END OF TRACK
        JNZ     T1BOOT    ;DONE WITH THIS TRACK
        SUI     27-6      ;BACK UP TO SECTOR 6
        LXI     H,CCP-80H  ;MEMORY ADDRESS OF SECTOR - 100H
        SHLD    NEWDMA
NOWRAP: STA     NEWSEC     ;SAVE THE UPDATED SECTOR #
        MOV     C,A
        CALL     FDSEC     ;SET UP THE SECTOR
        LXI     H,CCP-100H ;MEMORY ADDRESS OF SECTOR - 100H
NEWDMA  EQU     $-2
        LXI     D,100H     ;UPDATE DMA ADDRESS
        DAD     D
NOWRP:  SHLD    NEWDMA     ;SAVE THE UPDATED DMA ADDRESS
        MOV     B,H
        MOV     C,L
        CALL     FDDMA     ;SET UP THE NEW DMA ADDRESS
        LXI     B,RETRIES*100H+0;MAXIMUM # OF ERRORS, TRACK #
WRMFRED:PUSH    B
        CALL     FDSEEK    ;SET UP THE PROPER TRACK
        CALL     FDREAD    ;READ THE SECTOR
        POP     B
        JNC     T0BOOT    ;CONTINUE IF NO ERROR
        DCR     B
        JNZ     WRMFRED    ;KEEP TRYING IF ERROR
        JMP     FDERR      ;TOO MANY ERRORS, FLASH THE LIGHT

;LOAD TRACK 1, SECTOR 1, SECTOR 3 (PARTIAL), SECTOR 2 (1024 BYTE SECTORS)

T1BOOT: MVI     C,1        ;TRACK 1
        CALL     FDSEEK
        LXI     B,CCP+0B00H ;ADDRESS FOR SECTOR 1
        LXI     D,10*100H+1 ;RETRY COUNT + SECTOR 1
        CALL     WRMREAD
        LXI     B,CCP+0F00H ;ADDRESS FOR SECTOR 2

```

```

LXI    D,10*100H+3    ;RETRY COUNT + SECTOR 3
CALL   WRMREAD

LXI    B,0300H        ;SIZE OF PARTIAL SECTOR
LXI    D,CCP+1300H    ;ADDRESS FOR SECTOR 3
LXI    H,CCP+0F00H    ;ADDRESS OF SECTOR 3

WRMCPY: MOV    A,M      ;GET A BYTE AND
        STAX   D        ;  SAVE IT
        INX   D        ;BUMP POINTERS
        INX   H
        DCX   B        ;BUMP COUNTER
        MOV   A,B      ;CHECK IF DONE
        ORA   C
        JNZ   WRMCPY   ;  IF NOT, LOOP

LXI    B,CCP+0F00H    ;ADDRESS FOR SECTOR 2
LXI    D,10*100H+2    ;RETRY COUNT + SECTOR 2
CALL   WRMREAD

XRA    A              ;CLEAR ERROR INDICATOR
RET

WRMREAD: PUSH   D
        CALL   FDDMA   ;SET DMA ADDRESS
        POP    B
        CALL   FDSEC   ;SET SECTOR
WRMFRD: PUSH   B      ;SAVE ERROR COUNT
        CALL   FDREAD  ;READ A SECTOR
        JC     WRMERR  ;DO RETRY STUFF ON ERROR
        CALL   FDSTAT  ;SECTOR SIZE MUST BE 1024 BYTES
        ANI   0CH     ;MASK LENGTH BITS
        SUI   0CH     ;CARRY (ERROR) WILL BE SET IF < 0C0H
WRMERR: POP    B      ;FETCH RETRY COUNT
        RNC     ;RETURN IF NO ERROR
        DCR   B      ;BUMP ERROR COUNT
        JNZ   WRMFRD
        JMP   FDERR   ;ERROR, FLASH THE LIGHT

FDTRAN: INX    B
        PUSH   D      ;SAVE TABLE ADDRESS
        PUSH   B      ;SAVE SECTOR #
        CALL   FDGET  ;GET DPH FOR CURRENT DRIVE
        LXI   D,10    ;LOAD DPH POINTER
        DAD   D
        MOV   A,M
        INX   H
        MOV   H,M
        MOV   L,A
        MOV   A,M    ;GET # OF CP/M SECTORS/TRACK
        ORA   A      ;CLEAR CARRY
        RAR     ;DIVIDE BY TWO
        SUB   C      ;SUBTRACT SECTOR NUMBER
        PUSH  PSW    ;SAVE ADJUSTED SECTOR
        JM   SIDETWO
SIDEA:  POP    PSW   ;DISCARD ADJUSTED SECTOR

```

```

POP      B      ;RESTORE SECTOR REQUESTED
POP      D      ;RESTORE ADDRESS OF XLT TABLE
SIDEONE: XCHG   ;HL <- &(TRANSLATION TABLE)
DAD      B      ;BC = OFFSET INTO TABLE
MOV      L,M    ;HL <- PHYSICAL SECTOR
MVI      H,0
RET

SIDETWO: CALL   FDGSID   ;CHECK OUT NUMBER OF SIDES
JZ       SIDEA   ;SINGLE SIDED
POP      PSW    ;RETRIEVE ADJUSTED SECTOR
POP      B
CMA      ;MAKE SECTOR REQUEST POSITIVE
INR      A
MOV      C,A    ;MAKE NEW SECTOR THE REQUESTED SECTOR
POP      D
CALL    SIDEONE
MVI      A,80H  ;SIDE TWO BIT
ORA      H      ;      AND SECTOR
MOV      H,A
RET

FDLDRV: STA    FDLOG   ;SAVE LOGICAL DRIVE
MOV      C,A    ;SAVE DRIVE #
MVI      A,0    ;HAVE THE FLOPPIES BEEN ACCESSED YET ?
FLOPFLG EQU  $-1
ANA      A
JNZ      FLOPOK

MVI      B,17   ;FLOPPIES HAVN'T BEEN ACCESSED
LXI      H,FDBOOT ;CHECK IF 2D CONTROLLER IS INSTALLED
MVI      A,(JMP)
CLOPP:  CMP      M      ;MUST HAVE 17 JUMPS
JNZ      ZRET
INX      H
INX      H
INX      H
DCR      B
JNZ      CLOPP
LXI      D,FDINIT ;INITIALIZATION SEQUENCE
LXI      H,FDORIG+7E2H ;LOAD ADDRESS
LXI      B,30    ;BYTE COUNT
CALL    MOVBYT   ;LOAD CONTROLLER RAM
MVI      A,0FFH  ;START 1791
STA      DREG
MVI      A,CLRCMD ;1791 RESET
STA      CMDREG
MVI      A,1     ;SET 2D INITIALIZED FLAG
STA      FLOPFLG

FLOPOK: CALL    FLUSH   ;FLUSH BUFFER SINCE WE ARE USING IT
LDA      FDLOG   ;SELECT NEW DRIVE
MOV      C,A
CALL    FDSEL
CALL    FDLHOME  ;RECALIBRATE THE DRIVE
LXI      H,1     ;SELECT SECTOR 1 OF TRACK 2

```

```

SHLD TRUESEC
INX H
SHLD CPMTRK
XRA A ;MAKE SURE WE ARE DOING A READ
STA RDWR
CALL FILL ;FILL IN BUFFER WITH SECTOR
JC ZRET ;TEST FOR ERROR RETURN
CALL FDSTAT ;GET STATUS ON CURRENT DRIVE
STA FDLNST ;SAVE DRIVE STATUS
ANI 0CH ;MASK IN SECTOR SIZE BITS
PUSH PSW ;USED TO SELECT A DPB
RAR
LXI H,XLTS ;TABLE OF XLT ADDRESSES
MOV E,A
MVI D,0
DAD D
PUSH H ;SAVE POINTER TO PROPER XLT
CALL FDGET ;GET POINTER TO PROPER DPH
POP D
LXI B,2 ;COPY XLT POINTER INTO DPH
CALL MOVBYT
LXI D,8 ;OFFSET TO DPB POINTER IN DPH
DAD D ;HL <- &DPH.DPB
PUSH H
CALL FDGSID ;GET POINTER TO SIDE FLAG TABLE ENTRY
LDA FDLNST ;GET DRIVE STATUS
ANI DBLSID ;CHECK DOUBLE SIDED BIT
MOV M,A ;SAVE SIDES FLAG
LXI D,DPB128S ;BASE FOR SINGLE SIDED DPB'S
JZ SIDOK
LXI D,DPB128D ;BASE OF DOUBLE SIDED DPB'S
SIDEOK: XCHG
POP D ;(HL) -> DPB BASE, (DE) -> &DPH.DPB
POP PSW ;OFFSET TO CORRECT DPB
RAL
RAL ;MAKE 0, 10, 20, 30
MOV C,A
MVI B,0 ;MAKE OFFSET
DAD B ;(HL) IS NOW A DPB POINTER
XCHG ;PUT PROPER DPB ADDRESS IN DPH.DPB
MOV M,E
INX H
MOV M,D
LXI H,15 ;OFFSET TO DPB.SIZ
DAD D
MOV C,M ;FETCH SECTOR SIZE CODE
SIDEOK: LDA FDLOG ;RETURN PROPER DPH
LXI D,DPHFD0
JMP RETDPH

FDSEL2: STA FDLOG
MOV C,A
JMP FDSEL

FDLHOME: MVI C,0 ;SELECT SIDE 0
CALL FDSIDE

```

```

        JMP      FDHOME          ;DO ACTUAL HOME

FDSSEC: PUSH    B                ;SAVE SECTOR NUMBER
        MOV     A,B             ;CHECK SIDE SELECT BIT
        RLC                    ;MOVE HIGH BIT TO BIT ZERO
        ANI    1
        MOV     C,A
        CALL   FDSIDE          ;CALL SELECT SIDE 0 = SIDE A, 1 = SIDE B
        POP    B
        JMP    FDSEC

FDGSID: LXI    H,FDLSID        ;SIDE FLAG TABLE
        LDA    FDLOG          ;DRIVE NUMBER
        PUSH   D
        MOV    E,A            ;MAKE OFFSET
        MVI   D,0
        DAD   D               ;OFFSET TO PROPER ENTRY
        POP   D
        MOV   A,M            ;SET UP FLAGS
        ORA   A
        RET

FDINIT: DW     0                ;INITIALIZATION BYTES LOADED ONTO 2D/B
        DW    1800H           ;HEAD LOADED TIMEOUT
        DW    0                ;DMA ADDRESS
        DB    0                ;DOUBLE SIDED FLAG
        DB    0                ;READ HEADER FLAG
        DB    07EH           ;DRIVE SELECT CONSTANT
        DB    0                ;DRIVE NUMBER
        DB    8                ;CURRENT DISK
        DB    0                ;HEAD LOADED FLAG
        DB    9                ;DRIVE 0 PARAMETERS
        DB    0FFH           ;DRIVE 0 TRACK ADDRESS
        DB    9                ;DRIVE 1 PARAMETERS
        DB    0FFH           ;DRIVE 1 TRACK ADDRESS
        DB    9                ;DRIVE 2 PARAMETERS
        DB    0FFH           ;DRIVE 2 TRACK ADDRESS
        DB    9                ;DRIVE 3 PARAMETERS
        DB    0FFH           ;DRIVE 3 TRACK ADDRESS
        DB    9                ;CURRENT PARAMETERS
        DB    0                ;SIDE DESIRED
        DB    1                ;SECTOR DESIRED
        DB    0                ;TRACK DESIRED

        DB    0                ;HEADER IMAGE, TRACK
        DB    0                ;SECTOR
        DB    0                ;SIDE
        DB    0                ;SECTOR
        DW    0                ;CRC

FDLOG:  DB    0
FDLDST: DB    0                ;FLOPPY DRIVE STATUS BYTE

FDLSID: REPT   MAXFD
        DB    0FFH           ;DOUBLE SIDED FLAG 0 = SINGLE, 1 = DOUBLE
        ENDM

```

ENDIF

IF (MAXFD NE 0) OR (MAXDM NE 0)

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

```
F0F7 FFF0 XLTS: DW XLT128 ;XLT FOR 128 BYTE SECTORS
F0F9 1AF1 DW XLT256 ;XLT FOR 256 BYTE SECTORS
F0FB 4FF1 DW XLT512 ;XLT FOR 512 BYTE SECTORS
F0FD 8CF1 DW XLT124 ;XLT FOR 1024 BYTE SECTORS
```

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

```
F0FF 00 XLT128: DB 0
F100 01070D1319 DB 1,7,13,19,25
F105 050B1117 DB 5,11,17,23
F109 03090F15 DB 3,9,15,21
F10D 02080E141A DB 2,8,14,20,26
F112 060C1218 DB 6,12,18,24
F116 040A1016 DB 4,10,16,22
```

```
F11A 00 XLT256: DB 0
F11B 0102131425 DB 1,2,19,20,37,38
F121 0304151627 DB 3,4,21,22,39,40
F127 0506171829 DB 5,6,23,24,41,42
F12D 0708191A2B DB 7,8,25,26,43,44
F133 090A1B1C2D DB 9,10,27,28,45,46
F139 0B0C1D1E2F DB 11,12,29,30,47,48
F13F 0D0E1F2031 DB 13,14,31,32,49,50
F145 0F10212233 DB 15,16,33,34,51,52
F14B 11122324 DB 17,18,35,36
```

```
F14F 00 XLT512: DB 0
F150 0102030411 DB 1,2,3,4,17,18,19,20
F158 2122232431 DB 33,34,35,36,49,50,51,52
F160 0506070815 DB 5,6,7,8,21,22,23,24
F168 2526272835 DB 37,38,39,40,53,54,55,56
F170 090A0B0C19 DB 9,10,11,12,25,26,27,28
F178 292A2B2C39 DB 41,42,43,44,57,58,59,60
F180 0D0E0F101D DB 13,14,15,16,29,30,31,32
F188 2D2E2F30 DB 45,46,47,48
```



```

F18C 00      XLT124: DB      0
F18D 0102030405 DB      1,2,3,4,5,6,7,8
F195 191A1B1C1D DB      25,26,27,28,29,30,31,32
F19D 3132333435 DB      49,50,51,52,53,54,55,56
F1A5 090A0B0C0D DB      9,10,11,12,13,14,15,16
F1AD 2122232425 DB      33,34,35,36,37,38,39,40
F1B5 393A3B3C3D DB      57,58,59,60,61,62,63,64
F1BD 1112131415 DB      17,18,19,20,21,22,23,24
F1C5 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.
*
*****
    
```

```

F1CD 1A00      DPB128S:DW      26      ;CP/M SECTORS/TRACK
F1CF 03        DB          3      ;BSH
F1D0 07        DB          7      ;BLM
F1D1 00        DB          0      ;EXM
F1D2 F200      DW      242      ;DSM
F1D4 3F00      DW          63      ;DRM
F1D6 C0        DB      0C0H     ;AL0
F1D7 00        DB          0      ;AL1
F1D8 1000      DW          16      ;CKS
F1DA 0200      DW          2      ;OFF
F1DC 01        DB          1      ;128 BYTE SECTORS
    
```

```

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

```

F1DD 3400      DPB256S:DW      52      ;CP/M SECTORS/TRACK
F1DF 04        DB          4      ;BSH
F1E0 0F        DB          15     ;BLM
F1E1 01        DB          1      ;EXM
F1E2 F200      DW      242      ;DSM
F1E4 7F00      DW      127      ;DRM
F1E6 C0        DB      0C0H     ;AL0
F1E7 00        DB          0      ;AL1
F1E8 2000      DW          32      ;CKS
F1EA 0200      DW          2      ;OFF
F1EC 02        DB          2      ;256 BYTE SECTORS
    
```

```

*****
    
```

```

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

```

F1ED 3C00      DPB512S:DW      60      ;CP/M SECTORS/TRACK
F1EF 04        DB          4      ;BSH
F1F0 0F        DB          15     ;BLM
F1F1 00        DB          0      ;EXM
F1F2 1801     DW          280     ;DSM
F1F4 7F00     DW          127     ;DRM
F1F6 C0        DB        0C0H    ;AL0
F1F7 00        DB          0      ;AL1
F1F8 2000     DW          32     ;CKS
F1FA 0200     DW          2      ;OFF
F1FC 03        DB          3      ;512 BYTE SECTORS
    
```

```

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

```

F1FD 4000     DP1024S:DW     64      ;CP/M SECTORS/TRACK
F1FF 04        DB          4      ;BSH
F200 0F        DB          15     ;BLM
F201 00        DB          0      ;EXM
F202 2B01     DW          299     ;DSM
F204 7F00     DW          127     ;DRM
F206 C0        DB        0C0H    ;AL0
F207 00        DB          0      ;AL1
F208 2000     DW          32     ;CKS
F20A 0200     DW          2      ;OFF
F20C 04        DB          4      ;1024 BYTE SECTORS
    
```

```

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

```

F20D 3400     DPB128D:DW     52      ;CP/M SECTORS/TRACK
F20F 04        DB          4      ;BSH
F210 0F        DB          15     ;BLM
F211 01        DB          1      ;EXM
F212 F200     DW          242     ;DSM
F214 7F00     DW          127     ;DRM
F216 C0        DB        0C0H    ;AL0
F217 00        DB          0      ;AL1
F218 2000     DW          32     ;CKS
F21A 0200     DW          2      ;OFF
F21C 01        DB          1      ;128 BYTE SECTORS
    
```

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

F21D 6800 DPB256D:DW 104 ;CP/M SECTORS/TRACK  
F21F 04 DB 4 ;BSH  
F220 0F DB 15 ;BLM  
F221 00 DB 0 ;EXM  
F222 E601 DW 486 ;DSM  
F224 FF00 DW 255 ;DRM  
F226 F0 DB 0F0H ;AL0  
F227 00 DB 0 ;AL1  
F228 4000 DW 64 ;CKS  
F22A 0200 DW 2 ;OFF  
F22C 02 DB 2 ;256 BYTE SECTORS

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

F22D 7800 DPB512D:DW 120 ;CP/M SECTORS/TRACK  
F22F 04 DB 4 ;BSH  
F230 0F DB 15 ;BLM  
F231 00 DB 0 ;EXM  
F232 3102 DW 561 ;DSM  
F234 FF00 DW 255 ;DRM  
F236 F0 DB 0F0H ;AL0  
F237 00 DB 0 ;AL1  
F238 4000 DW 64 ;CKS  
F23A 0200 DW 2 ;OFF  
F23C 03 DB 3 ;512 BYTE SECTORS

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

F23D 8000 DP1024D:DW 128 ;CP/M SECTORS/TRACK  
F23F 04 DB 4 ;BSH  
F240 0F DB 15 ;BLM  
F241 00 DB 0 ;EXM  
F242 5702 DW 599 ;DSM  
F244 FF00 DW 255 ;DRM  
F246 F0 DB 0F0H ;AL0  
F247 00 DB 0 ;AL1  
F248 4000 DW 64 ;CKS  
F24A 0200 DW 2 ;OFF  
F24C 04 DB 4 ;1024 BYTE SECTORS

ENDIF

```
*****
*
* THE FOLLOWING EQUATES RELATE THE MORROW DESIGNS DJDMA
* CONTROLLER.
*
*****
```

```

0050 =      IF      (MAXDM NE 0) OR (MAXMF NE 0)
DMCHAN EQU  50H      ;DEFAULT CHANNEL ADDRESS
00EF =      DMKICK EQU 0EFH ;KICK I/O PORT ADDRESS

0020 =      RDSECT EQU  20H      ;READ SECTOR COMMAND
0021 =      WRSECT EQU  21H      ;WRITE A SECTOR COMMAND
0022 =      GSTAT  EQU  22H      ;GET DRIVE STATUS
0023 =      DMSDMA EQU  23H      ;SET DMA ADDRESS
0024 =      INTRQC EQU  24H      ;SET INTERRUPT REQUEST
0025 =      DMHALTC EQU  25H      ;HALT COMMAND
0026 =      BRACHA EQU  26H      ;CHANNEL BRANCH
0027 =      SETCHA EQU  27H      ;SET CHANNEL ADDRESS
0028 =      SETCRC EQU  28H      ;SET CRC RETRY COUNT
0029 =      RDTRCK EQU  29H      ;READ TRACK COMMAND
002A =      WRTRCK EQU  2AH      ;WRITE TRACK COMMAND
002B =      SEROUT EQU  2BH      ;SERIAL OUPUT THROUGH BIT BANGER SERIAL PORT
002C =      SENABL EQU  2CH      ;ENABLE SERIAL INPUT
002D =      TRKSIZ EQU  2DH      ;SET NUMBER OF TRACKS
002E =      SETLOG EQU  2EH      ;SET LOGICAL DRIVES
00A0 =      READM  EQU  0A0H     ;READ FROM CONTROLLER MEMORY
00A1 =      WRITEM EQU  0A1H     ;WRITE TO CONTROLLER MEMORY

0066 =      DMFSTP EQU  3*341/10 ;FAST STEPPING RATE CONSTANT IS 3 MS * 34.1
01FF =      DMFSET EQU  15*341/10;FAST SETTLING RATE CONSTANT IS 15 MS * 34.1

0080 =      N$DUBL EQU  80H      ;DOUBLE DENSITY
0040 =      N$2SIDE EQU 40H      ;2 SIDED DRIVE

003E =      SERIN  EQU  03EH     ;ADDRESS OF SERIAL INPUT DATA, (STATUS - 1)

```

```
*****
*
* DEVICE SPECIFICATION TABLE FOR THE DISK JOCKEY DMA FLOPPY
*
*****
```

```

F24D 01      DMDST:  IF      MAXDM NE 0
F24E 64F2    DW      MAXDM      ;NUMBER OF LOGICAL DRIVES
F250 DEF2    DW      DMWARM     ;WARM BOOT
F252 13F3    DW      DMTRAN     ;SECTOR TRANSLATION
F254 D6F2    DW      DMLDRV     ;SELECT DRIVE 1
F256 3CF4    DW      DMSELR     ;SELECT DRIVE 2
F258 3EF4    DW      DMHOME     ;HOME DRIVE
F25A AEF3    DW      DMSEEK     ;SEEK TO SPECIFIED TRACK
F25C BBF3    DW      DMSSEC     ;SET SECTOR
          DW      DMDMA      ;SET DMA ADDRESS

```

```

F25E 9FF4      DW      DMREAD      ;READ A SECTOR
F260 9CF4      DW      DMWRITE     ;WRITE A SECTOR
F262 0BED      DW      NOBAD        ;NO BAD SECTOR MAP

0B00 =         DMTRCK EQU      22*128    ;AMOUNT OF CODE ON TRACK 0

F264 CDD6F2    DMWARM: CALL    DMSELR      ;SELECT DRIVE 0
F267 215000    LXI      H,DMCHAN    ;SET UP BRANCH
F26A 3626      MVI      M,BRACHA
F26C 23        INX      H
F26D 3691      MVI      M,(LOW DMWCHN) ;LOW ADDRESS BYTE
F26F 23        INX      H
F270 36F2      MVI      M,(HIGH DMWCHN) ;HIGH ADDRESS BYTE
F272 23        INX      H
F273 3600      MVI      M,0
F275 21B0F2    DMWBAD: LXI      H,DMWEND-1 ;POINTER TO END OF COMMAND STRUCTURE
F278 CDC4F3    CALL     DOCMD        ;READ IN TRACKS
F27B 3A9CF2    LDA      DMWST        ;GET TRACK READ STATUS
F27E E640      ANI      40H
F280 CA75F2    JZ       DMWBAD        ;LOOP ON 'TERRIBLE' ERRORS LIKE NO DISK
F283 010003    LXI      B,300H        ;3/4 K BYTES OF SECTOR 3 NEEDS TO BE MOVED
F286 116DF5    LXI      D,BUFFER      ;SECTOR 3 IS IN OUR BUFFER
F289 2100E7    LXI      H,CCP+1300H   ; AND THIS IS WHERE WE WANT IT TO GO...
F28C CDE6EE    CALL     MOVBYT
F28F AF        XRA      A
F290 C9        RET

F291 23        DMWCHN: DB      DMSDMA      ;SET TRACK 0 DMA ADDRESS
F292 00D2      DW      CCP-512        ;FIRST TRACK DMA ADDRESS - BOOT LOADER
F294 00        DB      0
F295 29        DB      RDTRCK        ;READ TRACK COMMAND
F296 00        DB      0             ;TRACK 0
F297 00        DB      0             ;SIDE 0
F298 00        DB      0             ;DRIVE 0
F299 B4F2      DW      DMWSEC        ;SECTOR LOAD/STATUS MAP
F29B 00        DB      0
F29C 00        DMWST:  DB      0             ;TRACK READ STATUS
F29D 23        DB      DMSDMA
F29E 00DF      DW      CCP+DMTRCK    ;DMA ADDRESS FOR TRACK 1
F2A0 00        DB      0
F2A1 29        DB      RDTRCK
F2A2 01        DB      1             ;TRACK 1
F2A3 00        DB      0             ;SIDE 0
F2A4 00        DB      0             ;DRIVE 0
F2A5 CEF2      DW      DMWSEC+26     ;MAP IS LOADED RIGHT AFTER TRACK 0 STATUS MAP
F2A7 00        DB      0
F2A8 00        DB      0             ;TRACK READ STATUS
F2A9 23        DB      DMSDMA
F2AA 6DF5      DW      BUFFER        ;SECTOR 3 GETS LOADED IN SYSTEM BUFFER
F2AC 00        DB      0
F2AD 20        DB      RDSECT
F2AE 01        DB      1             ;TRACK 1
F2AF 03        DB      3             ;SIDE 0, SECTOR 3
F2B0 00        DB      0             ;DRIVE 0
F2B1 00        DMWEND: DB      0             ;READ STATUS
F2B2 0000      DW      0             ;ROOM FOR THE HALT

```

```

F2B4 FFFFFFFF DMWSEC: DW      0FFFFH, 0FFFFH      ;DO NOT LOAD BOOT LOADER
F2B8 00000000 DW      0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ;22 SECTORS TO BE LOADED
F2CE 0000FFFFFF DW      0, 0FFFFH, 0FFFFH, 0FFFFH   ;FIRST 2 SECTORS ON TRACK 2

F2D6 32E6F4   DMSELR: STA      DMLOG
F2D9 0600     MVI      B,0      ;8 INCH LOGICAL DRIVES START AT ZERO
F2DB C39EF3   JMP      DMSEL2

F2DE 03       DMTRAN: INX      B
F2DF D5       PUSH     D        ;SAVE TABLE ADDRESS
F2E0 C5       PUSH     B        ;SAVE SECTOR #
F2E1 CD71F3   CALL     DMGET
F2E4 110A00   LXI      D,10
F2E7 19       DAD      D
F2E8 7E       MOV      A,M
F2E9 23       INX      H
F2EA 66       MOV      H,M
F2EB 6F       MOV      L,A
F2EC 7E       MOV      A,M      ;GET # OF CP/M SECTORS/TRACK
F2ED B7       ORA      A        ;CLEAR CARY
F2EE 1F       RAR
F2EF 91       SUB      C        ;DIVIDE BY TWO
F2F0 F5       PUSH     PSW      ;SAVE ADJUSTED SECTOR
F2F1 FAFDF2   JM       DMSIDE2
F2F4 F1       DMSIDEA:POP    PSW      ;DISCARD ADJUSTED SECTOR
F2F5 C1       POP      B        ;RESTORE SECTOR REQUESTED
F2F6 D1       POP      D        ;RESTOR ADDRESS OF XLT TABLE
F2F7 EB       DMSIDE1:XCHG   ;HL <- &(TRANSLATION TABLE)
F2F8 09       DAD      B        ;BC = OFFSET INTO TABLE
F2F9 6E       MOV      L,M      ;HL <- PHYSICAL SECTOR
F2FA 2600     MVI      H,0
F2FC C9       RET

F2FD CD5FF4   DMSIDE2:CALL    DMSTAT
F300 E620     ANI      20H
F302 CAF4F2   JZ       DMSIDEA
F305 F1       POP      PSW      ;RETRIEVE ADJUSTED SECTOR
F306 C1       POP      B
F307 2F       CMA
F308 3C       INR      A        ;MAKE SECTOR REQUEST POSITIVE
F309 4F       MOV      C,A      ;MAKE NEW SECTOR THE REQUESTED SECTOR
F30A D1       POP      D
F30B CDF7F2   CALL     DMSIDE1
F30E 3E80     MVI      A,80H    ;SIDE TWO BIT
F310 B4       ORA      H        ;
F311 67       MOV      H,A      ;
F312 C9       RET

F313 32E6F4   DMLDRV: STA      DMLOG
F316 CDD3F3   CALL     DMINIT   ;TEST FOR A DRIVE
F319 DA81EC   JC       ZRET
F31C 210100   LXI      H,1      ;SELECT SECTOR 1 OF TRACK 2
F31F 22F0F4   SHLD    TRUESEC
F322 23       INX      H
F323 22EEF4   SHLD    CPMTRK

```

```

F326 AF          XRA      A          ;MAKE SURE WE ARE DOING A READ
F327 3274ED     STA      RDWR
F32A CDF0ED     CALL     FILL        ;FLUSH BUFFER AND REFILL
F32D DA81EC     JC       ZRET        ;TEST FOR ERROR RETURN

F330 CD5FF4     CALL     DMSTAT        ;GET STATUS ON CURRENT DRIVE
F333 E60C       ANI      0CH        ;MASK IN SECTOR SIZE BITS
F335 F5         PUSH     PSW          ;USED TO SELECT A DPB
F336 1F         RAR
F337 21F7F0     LXI      H,XLTS        ;TABLE OF XLT ADDRESSES
F33A 5F         MOV      E,A
F33B 1600       MVI      D,0
F33D 19         DAD      D
F33E E5         PUSH     H          ;SAVE POINTER TO PROPER XLT
F33F CD71F3     CALL     DMGET
F342 D1         POP      D
F343 010200     LXI      B,2          ;NUMBER OF BYTES TO MOVE
F346 CDE6EE     CALL     MOVBYT       ;MOVE THE ADDRESS OF XLT
F349 110800     LXI      D,8          ;OFFSET TO DPB POINTER
F34C 19         DAD      D          ;HL <- &DPH.DPB
F34D E5         PUSH     H
F34E CD5FF4     CALL     DMSTAT
F351 E620       ANI      20H        ;CHECK DOUBLE SIDED BIT
F353 11CDF1     LXI      D,DPB128S   ;BASE FOR SINGLE SIDED DPB'S
F356 CA5FF3     JZ       DMSOK
F359 CD7AF3     CALL     SETHIGH     ;SET CONTROLLER TO KNOW ABOUT FAST STEPPING
F35C 110DF2     LXI      D,DPB128D   ;BASE OF DOUBLE SIDED DPB'S
F35F EB         DMSOK: XCHG        ;HL <- DBP BASE, DE <- &DPH.DPB
F360 D1         POP      D          ;RESTORE DE (POINTER INTO DPH)
F361 F1         POP      PSW        ;OFFSET TO CORRECT DPB
F362 17         RAL
F363 17         RAL
F364 4F         MOV      C,A
F365 0600       MVI      B,0
F367 09         DAD      B
F368 EB         XCHG        ;PUT DPB ADDRESS IN DPH
F369 73         MOV      M,E
F36A 23         INX      H
F36B 72         MOV      M,D
F36C 210F00     LXI      H,15
F36F 19         DAD      D
F370 4E         MOV      C,M
F371 3AE6F4     DMGET: LDA      DMLOG
F374 115DF5     LXI      D,DPHDM0
F377 C3FCEE     JMP      RETDPH

;
; THE CURRENT DRIVE IS DOUBLE SIDED.  THUS IS IT SAFE TO SET THE
; STEPPING RATE TO 3 MS WITH 15 MS SETTLING.
;

F37A 2AE6F4     SETHIGH:LHLD    DMLOG        ;GET THE CURRENT DRIVE NUMBER
F37D 2600       MVI      H,0        ;DRIVE NUMBER IS A BYTE
F37F 29         DAD      H          ;TEN BYTES PER PARAMETER TABLE ENTRY
F380 54         MOV      D,H
F381 5D         MOV      E,L

```

```

F382 29      DAD      H
F383 29      DAD      H
F384 19      DAD      D
F385 114FEA  LXI      D,DPARAM+1      ;PARAMETER TABLE ADDRESS
F388 19      DAD      D              ;SKIP THE TRACK SIZE BYTE
F389 3600    MVI      M,0              ;FORCE REPARAMITIZATION OF THIS DRIVE
F38B 23      INX      H              ;OFFSET TO THE STEPPING RATE CONSTANT
F38C 3666    MVI      M,(LOW DMFSTP)      ;FAST STEPPING RATE CONSTANT
F38E 23      INX      H
F38F 3600    MVI      M,(HIGH DMFSTP)
F391 110500  LXI      D,5              ;SKIP OVER THE RESERVED FIELDS
F394 19      DAD      D
F395 36FF    MVI      M,(LOW DMFSET)      ;FAST SETTLING RATE CONSTANT
F397 23      INX      H
F398 3601    MVI      M,(HIGH DMFSET)
F39A CDFCF3  CALL     DMPARM           ;SET DRIVE PARAMETERS FOR THE SA850
F39D C9      RET

```

ENDIF

```

*****
*
* DRIVE SPECIFICATION TABLE FOR DJDMA 5 1/4 INCH DRIVES
*
*****

```

```

MFDST:  IF      MAXMF NE 0
        DB      MAXMF          ;NUMBER OF LOGICAL DRIVES
        DW      MFWARM         ;WARM BOOT
        DW      MFTRAN        ;SECTOR TRANSLATION
        DW      MFLDRV        ;SELECT DRIVE 1
        DW      MFSEL2        ;SELECT DRIVE 2
        DW      DMHOME        ;HOME DRIVE
        DW      MFSEEK        ;SEEK TO SPECIFIED TRACK
        DW      MFSSEC        ;SET SECTOR
        DW      DMDMA         ;SET DMA ADDRESS
        DW      DMREAD        ;READ A SECTOR
        DW      DMWRITE       ;WRITE A SECTOR
        DW      NOBAD         ;NO BAD SECTOR MAP

MFTRCK  EQU      9*512        ;AMOUNT OF CODE ON TRACK 0

MFWARM: CALL     MFSEL2        ;SELECT DRIVE 0
        LXI     H,DMCHAN      ;SET UP BRANCH
        MVI     M,BRACHA
        INX     H
        MVI     M,(LOW MFWCHN) ;LOW ADDRESS BYTE
        INX     H
        MVI     M,(HIGH MFWCHN) ;HIGH ADDRESS BYTE
        INX     H
        MVI     M,0
MFWFAL: LXI     H,MFWEND-1    ;POINTER TO END OF COMMAND STRUCTURE
        CALL    DOCMD         ;READ IN TRACKS
        LDA     MFWST         ;CHECK OUT DRIVE STATUS
        ANI     40H           ;TEST FOR OK
        JZ      MFWFAL        ;FAILED, LOOP

```



```

XRA      A          ;RETURN NO ERROR
RET

MFWCHN:  DB      DMSDMA      ;SET TRACK 0 DMA ADDRESS
          DW      CCP-512     ;FIRST TRACK DMA ADDRESS - BOOT LOADER
          DB      0
          DB      RDTRCK     ;READ TRACK COMMAND
          DB      0          ;TRACK 0
          DB      0          ;SIDE 0
          DB      0          ;DRIVE 0
          DW      MFWSEC     ;SECTOR LOAD/STATUS MAP
          DB      0
MFWST    DB      0          ;TRACK READ STATUS
          DB      DMSDMA
          DW      CCP+MFTRCK ;DMA ADDRESS FOR TRACK 1
          DB      0
          DB      RDTRCK
          DB      1          ;TRACK 1
          DB      0          ;SIDE 0
          DB      0          ;DRIVE 0
          DW      MFWSEC+10   ;MAP IS LOADED RIGHT AFTER TRACK 0 STATUS MAP
          DB      0
MFWEND:  DB      0          ;TRACK READ STATUS
          DW      0          ;ROOM FOR THE HALT

MFWSEC:  DW      0FFH, 0, 0, 0, 0 ;DO NOT LOAD BOOT LOADER
          DW      0, 0FFFFH, 0FFFFH, 0FFFFH, 0FFFFH ;FIRST TWO SECTORS LOADED

MFSSEC:  DCR      C          ;MINNIE FLOPPY SECTORS START AT ZERO
          LDA      DBLFLG    ;GET DOUBLE SIDED FLAGS
          ORA      A
          JZ      DMSSEC     ;NOPE, SINGLE SIDED
          MVI      B, 80H    ;SET HIGH BIT FOR DOUBLE SIDED SELECT
          JMP     DMSSEC

DBLFLG:  DB      0

MFSEEK:  XRA      A          ;CLEAR DOUBLE SIDED SELECT
          STA      DBLFLG
          LDA      MFPCON
          ANI      N$2SIDE
          JZ      DMSEEK     ;ONLY SINGLE SIDED

          MOV      A, C      ;MOVE SELECTED TRACK IN (A)
          SBI      35        ;SUBTRACT BY TRACK BY NUMBER OF TRACKS
          JC      DMSEEK     ;LESS THAN TRACK 35

          MOV      D, A      ;SAVE ADJUSTED TRACK NUMBER
          MVI      A, 34
          SUB      D          ;ADJUST TO COUNT TRACKS BACK OUT
          MOV      C, A      ;RESAVE NEW TRACK NUMBER

          MVI      A, 0FFH   ;SET DOUBLE SIDED FLAG
          STA      DBLFLG
          JMP     DMSEEK

```

```
MFSEL2: STA MFLOG
        MOV C,A ;GET PROPER PHYSICAL CONFIGURATION BYTE
        MVI B,0
        LXI H,MFSCON
        DAD B
        MOV A,M
        STA MFPCON
        MOV A,C ;SHHH, PRETEND THAT NOTHING HAPPENED

        MVI B,4 ;5 1/4 INCH DRIVES START AT DRIVE 4
        JMP DMSEL2

MFTRAN: LDA MFPCON
        ANI N$DUBL
        LXI H,MFXLTD ;POINT TO DOUBLE SIDED SECTOR TRANSLATION TABLE
        JNZ MFTDUBL ;SINGLE DENSITY SECTOR TRANSLATION
        LXI H,MFXLTS
MFTDUBL: DAD B ;ADD OFFSET SECTOR NUMBER TO TABLE
        MOV L,M ;PICK UP SECTOR NUMBER FROM TABLE
        MVI H,0 ;MSB OF SECTOR NUMBER EQUAL 0
        RET

MFLDRV: STA MFLOG
        CALL DMINIT ;TEST FOR A CONTROLLER
        JC ZRET

        LDA MFLOG ;GET PROPER PHYSICAL CONFIGURATION BYTE
        MOV C,A
        MVI B,0
        LXI H,MFSCON
        DAD B
        MVI A,N$DUBL
        MOV M,A
        STA MFPCON

        LXI H,1 ;SELECT SECTOR 1 OF TRACK 0
        SHLD TRUESEC
        DCX H
        SHLD CPMTRK
        XRA A ;MAKE SURE WE ARE DOING A READ
        STA RDWR
        CALL FILL ;FLUSH BUFFER AND REFILL
        JC ZRET ;TEST FOR ERROR RETURN
        LDA BUFFER+5CH ;GET DISKETTE CONFIGURATION BYTE

        PUSH PSW ;SAVE CONFIGURATION BYTE
        LXI H,1
        SHLD CPMTRK ;LOAD TRACK 1 SECTOR 1
        CALL FILL ;THIS IS TO FIX BUG WITH DJDMA FIRMWARE ON
        JC ZRET ; RETURNING SINGLE DENSITY STATUS ON TRACK 0
        POP PSW

        ORA A
        JNZ MFL9 ;NON ZERO
```

```

MVI    A,90H          ;DOUBLE DENSITY DEFAULT CONFIGURATION
CALL   DMSTAT        ;IF ZERO THEN DETERMINE SECTOR SIZE
ANI    80H           ;CHECK DENSITY BIT
JNZ    MFL9          ;ITS DOUBLE DENSITY
MVI    A,10H         ;SINGLE DENSITY DEFAULT CONFIGURATION BYTE

MFL9:  MOV    C,A     ;MOVE CONFIGURATION BYTE INTO (C)

MFL2:  LXI    H,MFS   ;ADDRESS OF CONFIGURATION TABLE -> (HL)
MOV    A,M         ;GET AN ENTRY
ORA    A           ;CHECK FOR END OF THE TABLE
JZ     ZRET        ;YES, SELECT ERROR
CMP    C           ;CHECK IF ENTRY MATCHES SELECTED DRIVE
JZ     MFL3        ;SKIP ONFIGURATION BYTE
INX   H            ;SKIP DRIVE TYPE
INX   H            ;SKIP DPB ADDRESS
INX   H
INX   H
JMP   MFL2

MFL3:  INX   H
MOV    A,M         ;PICK UP DRIVE TYPE
STA   MFPCON
MOV    E,A

PUSH   H
LDA   MFLOG        ;GET PROPER PHYSICAL CONFIGURATION BYTE
MOV    C,A
MVI   B,0
LXI   H,MFSCON
DAD   B
MOV   M,E
POP   H

INX   H
MOV   A,M
INX   H
MOV   H,M
MOV   L,A         ;DPB ADDRESS -> (HL)
PUSH  H           ;SAVE DPB ADDRESS
CALL  MFGDPH      ;GET DPH
LXI   D,10        ;OFFSET TO DPB ADDRESS IN DPH
DAD   D
POP   D
MOV   M,E         ;STORE DPB ADDRESS IN DPH
INX   H
MOV   M,D
CALL  MFGDPH
PUSH  H
CALL  DMSTAT      ;GET STATUS
POP   H
ANI   80H         ;CHECK DENSITY BIT
MVI   C,3         ;512 BYTE SECTORS
RNZ
MVI   C,2         ;256 BYTE SECTORS
RET

```

```

MFGDPH LDA MFLOG
        LXI D,DPHMF0
        JMP RETDPH

MFPCON: DB 0 ;PHYSICAL CONFIGURATION BYTE
MFLOG:  DB 0

MFSCON: DB 0, 0, 0, 0 ;SAVED PHYSICAL CONFIGURATION BYTES

MFS:   DB 10H ;NORTH STAR CP/M 1.4
        DB 0 ;SINGLE DENSITY, 35 TRACKS, SINGLE SIDED
        DW DPBMF0 ;1K GROUPS

        DB 90H ;NORTH STAR CP/M 1.4
        DB N$DUBL ;DOUBLE DENSITY, 35 TRACKS, SINGLE SIDED
        DW DPBMF1 ;1K GROUPS

        DB 0B0H ;NORTH STAR CP/M 2.X
        DB N$DUBL ;DOUBLE DENSITY, 35 TRACKS, SINGLE SIDED
        DW DPBMF2 ;2K GROUPS

        DB 0F0H ;NORTH STAR CP/M 2.X
        DB N$DUBL+N$2SIDE ;DOUBLE DENSITY, 35 TRACKS, DOUBLE SIDED
        DW DPBMF3 ;2K GROUPS

        DB 0E5H ;NORTH STAR CP/M 1.4
        DB N$DUBL ;DOUBLE DENSITY, 35 TRACKS, SINGLE SIDED
        DW DPBMF1 ;1K GROUPS

        DB 0A0H ;NORTH STAR CP/M 2.X (FAKE 40 TRACK)
        DB N$DUBL ;DOUBLE DENSITY, 35 TRACKS, SINGLE SIDED
        DW DPBMF2 ;2K GROUPS

        DB 0D0H ;NORTH STAR CP/M 2.X (FAKE 40 TRACK)
        DB N$DUBL+N$2SIDE ;DOUBLE DENSITY, 35 TRACKS, DOUBLE SIDED
        DW DPBMF3 ;2K GROUPS

        DB 0 ;END OF CONFIGURATION TABLE

MFXLTD DB 1, 2, 3, 4
        DB 21,22,23,24
        DB 5, 6, 7, 8
        DB 25,26,27,28
        DB 9,10,11,12
        DB 29,30,31,32
        DB 13,14,15,16
        DB 33,34,35,36
        DB 17,18,19,20
        DB 37,38,39,40

MFXLTS DB 1, 2
        DB 3, 4
        DB 5, 6
        DB 7, 8
        DB 9,10

```

```

DB      11,12
DB      13,14
DB      15,16
DB      17,18
DB      19,20
ENDIF

```

```

*****
*
* COMMON ROUTINES FOR THE DJDMA WITH 8 AND 5 1/4 INCH DRIVES
*
*****

```

```

F39E 4F      DMSEL2: MOV      C,A          ;MOVE DRIVE INTO (C)
F39F 215000  LXI      H,DMCHAN
F3A2 362E    MVI      M,SETLOG      ;SET LOGICAL DRIVES
F3A4 23      INX      H
F3A5 70      MOV      M,B          ;DRIVE IN (B)
F3A6 C5      PUSH     B
F3A7 CDC4F3  CALL     DOCMD
F3AA C1      POP      B
F3AB C35AF4  JMP      DMSEL

F3AE C5      DMSSEC: PUSH    B          ;SAVE SECTOR NUMBER
F3AF 78      MOV      A,B
F3B0 07      RLC
F3B1 E601    ANI      1
F3B3 4F      MOV      C,A
F3B4 CD4DF4  CALL     DMSIDE
F3B7 C1      POP      B
F3B8 C343F4  JMP      DMSEC

F3BB 215000  DMDMA  LXI      H,DMCHAN      ;DEFAULT CHANNEL ADDRESS
F3BE 3623    MVI      M,DMSDMA          ;SET DMA ADDRESS
F3C0 23      INX      H
F3C1 71      MOV      M,C          ;LOW BYTE FIRST
F3C2 23      INX      H
F3C3 70      MOV      M,B          ;HIGH BYTE NEXT

F3C4 AF      DOCMD  XRA      A
F3C5 23      INX      H
F3C6 77      MOV      M,A
F3C7 23      DOCMD2 INX      H
F3C8 3625    MVI      M,DMHALTC
F3CA 23      INX      H
F3CB 77      MOV      M,A
F3CC D3EF    OUT     DMKICK
F3CE B6      TESTS  ORA      M
F3CF CACEF3  JZ      TESTS
F3D2 C9      RET

F3D3 215000  DMINIT: LXI     H,DMCHAN      ;SEE IF CONTROLLER WILL HALT
F3D6 3625    MVI     M,DMHALTC
F3D8 23      INX     H
F3D9 3600    MVI     M,0
F3DB D3EF    OUT     DMKICK          ;START CONTROLLER

```

```

F3DD 110000      LXI      D,0           ;SET UP TIMEOUT COUNTER
F3E0 7E          DMINWT  MOV      A,M
F3E1 B7          ORA      A
F3E2 C2EDF3      JNZ      DMIOK          ;CONTROLLER HAS RESPONDED
F3E5 1B          DCX      D           ;BUMP TIMEOUT COUNTER
F3E6 7A          MOV      A,D
F3E7 B3          ORA      E
F3E8 C2E0F3      JNZ      DMINWT
F3EB 37          STC                ;SET ERROR FLAG
F3EC C9          RET

F3ED E5          DMIOK  PUSH     H           ;SET DRIVE PARAMETERS
F3EE CDFCF3      CALL    DMPARM
F3F1 E1          POP      H
F3F2 2B          DCX      H           ;BACK TO START OF COMMAND
F3F3 3628        MVI      M,SETCRC      ;SET CRC ERROR RETRY COUNT TO ONE
F3F5 23          INX      H
F3F6 3601        MVI      M,1
F3F8 AF          XRA      A
F3F9 C3C7F3      JMP      DOCMD2        ;DO COMMAND

;
; SET FLOPPY DRIVE PARAMETERS
;
; THIS ROUTINE READS THE DPARAM TABLE AND IF THE A DRIVE HAS NOT
; PREVIOUSLY BEEN CALIBRATED THEN THAT DRIVES TRACK COUNT,
; STEPPING RATE, AND HEAD SETTling TIME ARE LOADED.
;

F3FC 3E08        DMPARM: MVI      A,8           ;EIGHT DRIVES
F3FE 114013      LXI      D,1340H        ;START WITH DRIVE 0'S TABLE
F401 214FEA      LXI      H,DPARAM+1      ;DRIVE PARAMETER TABLE

F404 F5          DMSTR0: PUSH     PSW           ;SAVE THE DRIVE COUNT
F405 7E          MOV      A,M           ;LOAD FLAGS
F406 B7          ORA      A           ;DOES THE DRIVE NEED TO BE CALIBRATED?
F407 C22CF4      JNZ      DMSTR1        ;NO, DO NOT FIDDLE AROUND
F40A E5          PUSH     H           ;SAVE THE PARAMETER TABLE POINTER
F40B D5          PUSH     D           ;SAVE THE CONTROLLERS TABLE POINTER
F40C 35          DCR      M           ;SET TO CALIBRATED MODE (0FFH)
F40D 2B          DCX      H           ;BACK UP TO THE TRACK SIZE BYTE
F40E 22D2F4      SHLD    DMNTRK        ;SET THE NUMBER OF TRACKS POINTER
F411 23          INX      H
F412 23          INX      H
F413 22DAF4      SHLD    DMSPAR        ;SET THE STEPPING CONSTANTS POINTER
F416 EB          XCHG                ;SET THE LOCAL PARAMETER TABLE POINTER
F417 22D7F4      SHLD    DMLOC0        ;OFFSET TO THE STEPPING PARAMETERS
F41A 23          INX      H
F41B 23          INX      H
F41C 23          INX      H
F41D 23          INX      H
F41E 22DFF4      SHLD    DMLOC1
F421 21D1F4      LXI      H,DMWCON      ;WRITE THE DRIVE CONSTANTS OUT
F424 111100      LXI      D,17          ;HALT STATUS OFFSET
F427 CDBC F4      CALL    DMDOIT
F42A D1          POP      D           ;RETRIEVE THE TABLE POINTERS

```

```

F42B E1          POP      H

F42C 010A00     DMSTR1: LXI      B,10          ;BUMP PARAMETER TABLE POINTER
F42F 09         DAD      B
F430 EB        XCHG
F431 011000     LXI      B,16          ;BUMP CONTROLLER TABLES POINTER
F434 09         DAD      B
F435 EB        XCHG

F436 F1        POP      PSW          ;RETRIEVE DRIVE COUNT
F437 3D        DCR      A          ;BUMP COUNT
F438 C204F4     JNZ      DMSTR0          ;SET UP NEXT DRIVE

F43B C9        RET

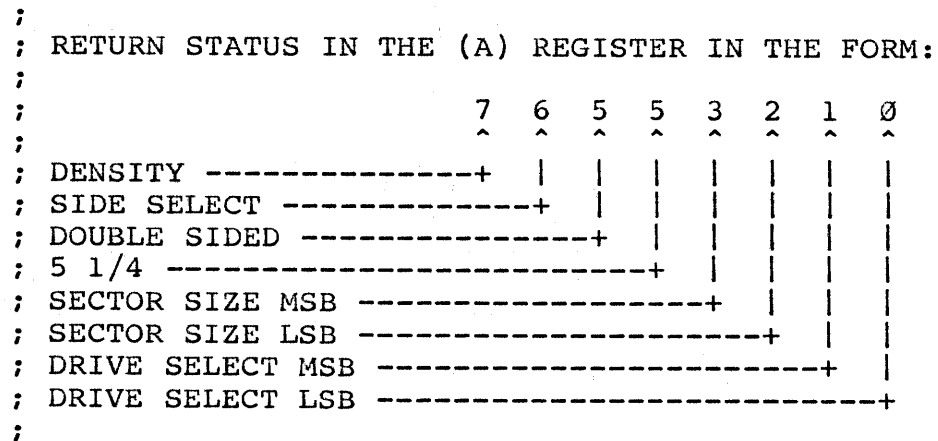
F43C AF        DMHOME XRA      A
F43D 4F        MOV      C,A          ;PUT A ZERO INTO (C) FOR TRACK ZERO

F43E 79        DMSEEK MOV      A,C          ;ENTER WITH TRACK IN (C)
F43F 32E3F4     STA      LLTRK          ;SAVE FOR USE LATER
F442 C9        RET

F443 3AE4F4     DMSEC  LDA      LLSS          ;LOAD SECTOR
F446 E680     ANI      80H          ;SAVE SIDE SELECT BIT
F448 B1        STORES ORA      C
F449 32E4F4     STA      LLSS
F44C C9        RET

F44D 79        DMSIDE: MOV     A,C          ;MOVE SIDE BIT INTO (A)
F44E E601     ANI      1
F450 0F        RRC          ;MOVE AROUND TO BIT 7
F451 4F        MOV      C,A          ;RESAVE IN (C)
F452 3AE4F4     LDA      LLSS
F455 E67F     ANI      7FH          ;MASK OUT OLD SIDE SELECT BIT
F457 C348F4     JMP      STORES

F45A 79        DMSEL:  MOV     A,C          ;MOVE DRIVE INTO (A)
F45B 32E5F4     STA      LLDRV
F45E C9        DMDEN:  RET          ;DOUBLE DENSITY ONLY
    
```



```

F45F 215000     DMSTAT LXI      H,DMCHAN
    
```

```

F462 3622      MVI     M,GSTAT      ;SET UP READ STATUS
F464 23        INX     H
F465 3AE5F4    LDA     LLDRV      ;GET LAST SELECTED DRIVE
F468 77        MOV     M,A        ;STORE DRIVE IN COMMAND
F469 23        INX     H        ;SKIP OVER RETURNED STATUS
F46A 23        INX     H
F46B 23        INX     H
F46C CDC4F3    CALL    DOCMD      ;ISSUE COMMAND
F46F 3AE4F4    LDA     LLSS      ;GET SIDE BIT OF LAST OPERATION
F472 E680      ANI     80H
F474 0F        RRC
F475 4F        MOV     C,A
F476 215100    LXI     H,DMCHAN+1 ;POINT TO DRIVE
F479 7E        MOV     A,M      ;LOAD DRIVE
F47A B1        ORA     C
F47B E604      ANI     4        ;MASK UPPER DRIVE SELECT BIT FOR 5 1/4
F47D 07        RLC
F47E 07        RLC      ;MOVE TO BIT 4
F47F B6        ORA     M      ;PUT TOGETHER WITH LOWER DRIVE BITS
F480 B1        ORA     C
F481 4F        MOV     C,A
F482 23        INX     H
F483 3E10      MVI     A,10H      ;DOUBLE DENSITY BIT
F485 A6        ANA     M
F486 07        RLC      ;20H
F487 07        RLC      ;40H
F488 07        RLC      ;80H FOR DENSITY BIT
F489 B1        ORA     C
F48A 4F        MOV     C,A
F48B 23        INX     H
F48C 3E03      MVI     A,3        ;SECTOR LENGTH MASK
F48E A6        ANA     M      ;AND IN
F48F 07        RLC      ;MOVE TO BITS 2 & 3
F490 07        RLC
F491 B1        ORA     C
F492 4F        MOV     C,A
F493 23        INX     H
F494 3E04      MVI     A,4        ;MASK FOR DOUBLE SIDED BIT
F496 A6        ANA     M
F497 07        RLC      ;8
F498 07        RLC      ;10
F499 07        RLC      ;20
F49A B1        ORA     C
F49B C9        RET

F49C 3E21      DMWRITE MVI     A,WRSECT
F49E 01        DB     01      ;UGH...
F49F 3E20      DMREAD  MVI     A,RDSECT
F4A1 215000    LXI     H,DMCHAN
F4A4 11E2F4    LXI     D,LLTRK-1
F4A7 0604      MVI     B,4
F4A9 77        CLOAD  MOV     M,A
F4AA 23        INX     H
F4AB 13        INX     D
F4AC 1A        LDAX   D
F4AD 05        DCR     B

```



F4AE	C2A9F4	JNZ	CLOAD
F4B1	2B	DCX	H
F4B2	CDC4F3	CALL	DOCMD
F4B5	3A5400	LDA	DMCHAN+4
F4B8	FE80	CPI	80H
F4BA	3F	CMC	
F4BB	C9	RET	

```

;
; EXECUTE A DJDMA COMMAND, NO COMMAND STATUS IS RETURNED
;
; ENTRY:
; DE = OFFSET TO THE HALT STATUS
; HL = POINTER TO THE START OF THE COMMAND
;
; RETURNS:
; NOTHING
;

```

F4BC	3E26	DMDOIT:	MVI	A, BRACHA	; BRANCH CHANNEL COMMAND
F4BE	325000		STA	DMCHAN	
F4C1	225100		SHLD	DMCHAN+1	; LOAD COMMAND VECTOR
F4C4	AF		XRA	A	; CLEAR EXTENDED ADDRESS
F4C5	325300		STA	DMCHAN+3	

F4C8	19		DAD	D	; OFFSET TO THE HALT STATUS
F4C9	77		MOV	M, A	; CLEAR THE HALT STATUS INDICATOR

F4CA	D3EF		OUT	DMKICK	; START THE CONTROLLER
------	------	--	-----	--------	------------------------

F4CC	B6	DMWAIT:	ORA	M	; WAIT FOR THE OPERATION COMPLETE STATUS
F4CD	CACCF4		JZ	DMWAIT	

F4D0	C9		RET		
------	----	--	-----	--	--

F4D1	A1	DMWCON:	DB	WRITEM	; WRITE TRACK SIZE
F4D2	0000	DMNTRK:	DW	0	; NUMBER OF TRACKS + DESYNC
F4D4	00		DB	0	; X-ADDRESS
F4D5	0200		DW	2	; TWO BYTES
F4D7	0000	DMLOC0:	DW	0	; LOCAL CONTROLLER ADDRESS

F4D9	A1		DB	WRITEM	; WRITE STEPPING RATE DATA
F4DA	0000	DMSPAR:	DW	0	; POINTER TO THE STEPPING PARAMETERS
F4DC	00		DB	0	
F4DD	0800		DW	8	
F4DF	0000	DMLOC1:	DW	0	

F4E1	25		DB	DMHALTC	; CONTROLLER HALT
F4E2	00		DB	0	; STATUS

```

;
; DRIVER VARIABLES
;

```

F4E3	00	LLTRK	DB	0	
F4E4	01	LLSS	DB	1	

F4E5 00 LLDRV DB 0  
 F4E6 00 DMLOG DB 0

ENDIF

\*\*\*\*\*  
 \*  
 \* THE FOLLWING EQUATES ARE FOR THE HDDMA HARD DISK CONTROLLER \*  
 \*  
 \*\*\*\*\*

IF MAXMW NE 0 ;HDDMA CONTROLLER PRESENT ?  
 IF ST506 ;SPECIFICATIONS FOR A SEAGATE TECHNOLOGY 506  
 CYL EQU 153 ;NUMBER OF CYLINDERS  
 HEADS EQU 4 ;NUMBER OF HEADS PER CYLINDER  
 PRECOMP EQU 64 ;CYLINDER TO START WRITE PRECOMENSATION  
 LOWCURR EQU 128 ;CYLINDER TO START LOW CURRENT  
 STEPDLY EQU 30 ;STEP DELAY (0-12.7 MILLISECONDS)  
 STEPRCL EQU 30 ;RECALIBRATE STEP DELAY  
 HEADDLY EQU 0 ;SETTLE DELAY (0-25.5 MILLISECONDS)  
 ENDIF

IF ST412 ;SPECIFICATIONS FOR A SEAGATE ST412  
 CYL EQU 306  
 HEADS EQU 4  
 PRECOMP EQU 128  
 LOWCURR EQU 128  
 STEPDLY EQU 0  
 STEPRCL EQU 30  
 HEADDLY EQU 0  
 ENDIF

IF CM5619 ;SPECIFICATIONS FOR AN CMI 5619  
 CYL EQU 306  
 HEADS EQU 6  
 PRECOMP EQU 128  
 LOWCURR EQU 128  
 STEPDLY EQU 2  
 STEPRCL EQU 30  
 HEADDLY EQU 0  
 ENDIF

SECTSIZ EQU 7 ;SECTOR SIZE CODE (MUST BE 7 FOR THIS CBIOS)  
 ; 0 = 128 BYTE SECTORS  
 ; 1 = 256 BYTE SECTORS  
 ; 3 = 512 BYTE SECTORS  
 ; 7 = 1024 BYTE SECTORS (DEFAULT)  
 ; F = 2048 BYTE SECTORS

;DEFINE CONTROLLER COMMANDS  
 DMAREAD EQU 0 ;READ SECTOR  
 DMAWRIT EQU 1 ;WRITE SECTOR  
 DMARHED EQU 2 ;FIND A SECTOR  
 DMAWHED EQU 3 ;WRITE HEADERS (FORMAT A TRACK)  
 DMALCON EQU 4 ;LOAD DISK PARAMETERS  
 DMASSTA EQU 5 ;SENSE DISK DRIVE STATUS

```

DMANOOP EQU 6 ;NULL CONTROLLER OPERATION

RESET EQU 54H ;RESET CONTROLLER
ATTN EQU 55H ;SEND A CONTROLLER ATTENTION

CHAN EQU 50H ;DEFAULT CHANNEL ADDRESS
STEPOUT EQU 10H ;STEP DIRECTION OUT
STEPIN EQU 0 ;STEP DIRECTION IN
BAND1 EQU 40H ;NO PRECOMP, HIGH CURRENT
BAND2 EQU 0C0H ;PRECOMP, HIGH CURRENT
BAND3 EQU 80H ;PRECOMP, LOW CURRENT
TRACK0 EQU 1 ;TRACK ZERO STATUS
WFLT EQU 2 ;WRITE FAULT FROM DRIVE
DREADY EQU 4 ;DRIVE READY
SEKCOMP EQU 8 ;SEEK COMPLETE
    
```

```

*****
*
* DRIVE SPECIFICATION TABLE FOR THE HD DMA HARD DISK CONTROLLER *
*
*****
    
```

```

MWDST: DB MAXMW*MWLOG ;NUMBER OF LOGICAL DRIVES
        DW MWWARM ;WARM BOOT
        DW MWTRAN ;SECTOR TRANSLATION
        DW MWLDRV ;SELECT LOGICAL DRIVE 1 (FIRST TIME SELECT)
        DW MWDRV ;SELECT LOGICAL DRIVE 2 (GENERAL SELECT)
        DW MWHOME ;HOME CURRENT SELECTED DRIVE
        DW MWSEEK ;SEEK TO SELECTED TRACK
        DW MWSEC ;SELECT SECTOR
        DW MWDMA ;SET DMA ADDRESS
        DW MWREAD ;READ A SECTOR
        DW MWWRITE ;WRITE A SECTOR
        IF HEADS > 2 ;TEST IF DRIVE IS BIG ENOUGH FOR A BAD SPOT MAP
        DW MWBAD ;RETURN BAD SECTOR MAP INFO
        ELSE
        DW NOBAD
        ENDIF
    
```

```

*****
*
* THE FOLLOWING ARE THE LOWEST LEVEL DRIVERS FOR THE MORROW *
* DESIGNS HARD DISK DMA CONTROLLER. *
*
*****
    
```

```

MWWARM XRA A
        CALL MWDRV ;SELECT DRIVE A
        CALL MWHOME ;HOME AND RESET THE DRIVE
        LXI B,0 ;MAKE SURE WE ARE ON TRACK 0
        CALL MWSEEK
        XRA A
        STA MWHEAD ;SELECT HEAD ZERO
        STA MWSECTR ;SELECT SECTOR 1
        LXI H,BUFFER ;LOAD SECTOR 1 INTO BUFFER
        SHLD DMADMA
    
```

```

CALL    MWWREAD      ;READ CCP INTO BUFFER
RC      ;RETURN IF ERROR
LXI     D,BUFFER+200H
LXI     H,CCP
LXI     B,200H      ;MOVE 200H BYTES
CALL    MOVBYT
LXI     H,CCP-200H  ;INITIAL DMA ADDRESS
PUSH    H
XRA     A
PUSH    A      ;SAVE FIRST SECTOR -1
MWWL0D POP    PSW   ;RESTORE SECTOR
POP     H      ;RESTORE DMA ADDRESS
INR     A
STA     MWSECTR
CPI     6      ;PAST BDOS ?
RZ     ;YES, ALL DONE
INR     H      ;UPDATE DMA ADDRESS BY 1024 BYTES
INR     H
INR     H
INR     H
SHLD   DMADMA
PUSH    H
PUSH    PSW
CALL    MWWREAD      ;READ IN A SECTOR
JNC    MWWL0D
RET     ;RETURN WITH ERROR

MWWREAD MVI     C,RETRIES ;RETRY COUNTER
MWWERR  PUSH    B      ;SAVE THE RETRY COUNT
CALL    MWREAD      ;READ THE SECTOR
POP     B
RNC
DCR     C      ;UPDATE THE ERROR COUNT
JNZ    MWWERR      ;KEEP TRYING IF NOT TOO MANY ERRORS
STC
RET     ;SET ERROR FLAG

MWLDRV  STA     MWCURL   ;SAVE CURRENT LOGICAL DRIVE
CALL    MWRESET     ;RESET CONTROLLER CARD
JC     ZRET        ;CONTROLLER FAILURE

LDA     MWCURL
CALL    MWDRV      ;SELECT DRIVE
JC     ZRET        ;SELECT ERROR

CALL    MWSTAT     ;GET DRIVE STATUS
ANI     DREADY     ;CHECK IF DRIVE READY
JNZ    ZRET

CALL    MWHOME     ;HOME DRIVE

LXI     D,DPHMW0   ;START OF HARD DISK DPH'S
LDA     MWCURL
MOV     L,A
MVI     H,0
DAD     H

```

```

DAD H
DAD H
DAD H
DAD D ;(HL) = POINTER TO DPH
MVI C,4 ;RETURN SECTOR SIZE OF 1024
RET

MWDRV STA MWCURL
CALL MWDLOG
MOV A,C
STA MWDRIIVE ;SAVE NEW SELECTED DRIVE
MWSEL MVI A,DMANOOP
JMP MWPREP ;EXECUTE DISK COMMAND

MWDLOG: MVI C,0
MWLLX: SUI MWLOG
RC
INR C
JMP MWLLX

MWSTAT MVI A,DMASTA ;SENSE STATUS OPERATION CODE
JMP MWPREP ;EXECUTE DISK COMMAND

MWHOME CALL MWRESET ;RESET CONTROLLER, DO A LOAD CONSTANTS
LXI H,DMARG1 ;LOAD ARGUMENTS
MVI M,STEPCL ;LOAD STEP DELAY (SLOW RATE)
INX H
MVI M,HEADDLY ;HEAD SETTLE DELAY
CALL MWISSUE ;DO LOAD CONSTANTS AGAIN
CALL MWPTR ;GET POINTER TO CURRENT CYLINDER NUMBER
MVI M,0FFH ;FAKE AT CYLINDER 65535 FOR MAX HEAD TRAVEL
INX H
MVI M,0FFH
LXI B,0 ;SEEK TO CYLINDER 0
CALL MWSEEK ;RECAL SLOWLY
JMP MWRESET ;BACK TO FAST STEPPING MODE

MWBAD: LXI H,MWBTAB ;RETURN POINTER TO BAD SECTOR LOCATION
RET

MWBTAB: DW 0 ;TRACK 0
DW 19 ;HEAD 2, SECTOR 0 = (2 * SPT + 0) + 1

MWSEEK CALL MWPTR ;GET TRACK POINTER
MOV E,M ;GET OLD TRACK NUMBER
INX H
MOV D,M
DCX H
MOV M,C ;STORE NEW TRACK NUMBER
INX H
MOV M,B
MOV L,C ;BUILD CYLINDER WORD
MOV H,B
SHLD DMARG0 ;SET COMMAND CHANNEL CYLINDER NUMBER
MOV A,D
INR A

```

```

LXI      H,0FFFFH
JNZ      MWSKIP0
MVI      C,STEPOUT
JMP      MWSKIP

MWSKIP0:MOV  H,B          ;(HL) = NEW TRACK, (DE) = OLD TRACK
MOV      L,C
CALL     MWHLMDDE
MVI      C,STEPOUT
MOV      A,H
ANI      80H            ;CHECK HIT BIT FOR NEGITIVE DIRECTION
JNZ      MWSOUT        ;STEP IN
MVI      C,0
JMP      MWSKIP

MWSOUT:  CALL  MWNEGHL
MWSKIP:  SHLD  DMASTEP
LDA      MWDRIVE
ORA      C
STA      DMASEL0

MVI      A,DMANOOP    ;NO-OPERATION COMMAND FOR THE CHANNEL
CALL     MWPREP       ;STEP TO PROPER TRACK
LXI      H,0          ;CLEAR STEP COUNTER
SHLD    DMASTEP
RET

MWDMA   MOV  H,B          ;SET DMA ADDRESS
MOV     L,C
SHLD   DMADMA
RET

MWSEC   MOV  A,C          ;LOAD SECTOR NUMBER
DCR    A                ;RANGE IS ACTAULLY 0-16
CALL   MWDSPT          ;FIGURE OUT HEAD NUMBER -> (C)
ADI    MWSPT           ;MAKE SECTOR NUMBER
STA    MWSECTR
MOV    A,C
STA    MWHEAD          ;SAVE HEAD NUMBER
RET

MWDSPT  MVI  C,0          ;CLEAR HEAD COUNTER
MWDSPTX SUI  MWSPT       ;SUBTRACT A TRACKS WORTH OF SECTORS
RC      ;RETURN IF ALL DONE
INR    C                ;BUMP TO NEXT HEAD
JMP    MWDSPTX

MWRESET LHLD  CHAN        ;SAVE THE COMMAND CHANNEL FOR A WHILE
SHLD   TEMPB
LDA    CHAN+2
STA    TEMPB+2
OUT    RESET           ;SEND RESET PULSE TO CONTROLLER
LXI    H,DMACHAN      ;ADDRESS OF COMMAND CHANNEL
SHLD   CHAN           ;DEFAULT CHANNEL ADDRESS
XRA    A
STA    CHAN+2         ;CLEAR EXTENDED ADDRESS BYTE
SHLD   40H            ;SET UP A POINTER TO THE COMMAND CHANNEL

```

```

STA      42H
LHLD    DMARG0      ;SAVE THE TRACK NUMBER
PUSH    H
LXI     H,DMASEL1   ;LOAD ARGUMENTS
LDA     MWDRIVE     ;GET THE CURRENTLY SELECTED DRIVE
ORI     03CH        ;RAISE *STEP AND *DIR
MOV     M,A         ;SAVE IN DRIVE SELECT REGISTER
LXI     D,5         ;OFFSET TO DMARG1
DAD     D
MVI     M,STEPDLY   ;LOAD STEP DELAY
INX     H
MVI     M,HEADDLY   ;HEAD SETTLE DELAY
INX     H
MVI     M,SECTSIZ   ;SECTOR SIZE CODE
INX     H
MVI     M,DMALCON   ;LOAD CONSTANTS COMMAND
CALL    MWISSUE     ;DO LOAD CONSTANTS
POP     H           ;RESTORE THE TRACK NUMBER
SHLD    DMARG0
PUSH    PSW         ;SAVE STATUS
LHLD    TEMPB       ;RESTORE MEMORY USED FOR THE CHANNEL POINTER
SHLD    CHAN
LDA     TEMPB+2
STA     CHAN+2
POP     PSW
RET

MWRITE  MVI     A,DMAREAD      ;LOAD DISK READ COMMAND
        JMP     MWPREP

MWRITE  MVI     A,DMAWRIT     ;LOAD DISK WRITE COMMAND

MWPREP: STA     DMAOP         ;SAVE COMMAND CHANNEL OP CODE

        MVI     C,BAND1
        LHLD    DMARG0
        LXI     D,PRECOMP
        CALL    MWHLCD
        JC

        MVI     C,BAND2
        LXI     D,LOWCURR
        CALL    MWHLCD
        JC

MWPREPS MVI     C,BAND3       ;CYLINDER > LOW CURRENT
        LDA     MWHEAD       ;LOAD HEAD ADDRESS
        STA     DMARG2
        CMA
        ANI     7           ;NEGATIVE LOGIC FOR THE CONTROLLER
        RLC          ;3 BITS OF HEAD SELECT
        RLC          ;SHOVE OVER TO BITS 2 - 4
        ORA     C           ;ADD ON LOW CURRENT AND PRECOMP BITS
        MOV     C,A
        LDA     MWDRIVE     ;LOAD DRIVE ADDRESS
        ORA     C           ;SLAP IN DRIVE BITS

```

```

STA      DMASEL1      ;SAVE IN COMMAND CHANNEL HEAD SELECT
LDA      MWSECTR      ;LOAD SECTOR ADDRESS
STA      DMARG3

IF      0      ;SET TO 1 FOR MW ERROR REPORTER
MWISSUE CALL MWDOIT   ;DO DESIRED OPERATION
RNC      ;DO NOTHING IF NO ERROR
PUSH    PSW        ;SAVE ERROR INFO
CALL    HEXOUT     ;PRINT STATUS
CALL    DSPOUT     ; AND A SPACE
LXI     H,DMACHAN
MVI     C,16       ;16 BYTES OF STATUS
MWERR:  PUSH B
        PUSH H
        MOV  A,M
        CALL HEXOUT ;PRINT A BYTE OF THE STATUS LINE
        CALL SPOUT
        POP  H
        POP  B
        INX H       ;BUMP COMMAND CHANNEL POINTER
        DCR  C
        JNZ  MWERR
        MVI  C,0AH   ;TERMINATE WITH A CR LF
        CALL POUT
        MVI  C,0DH
        CALL POUT
        POP  PSW    ;RESTORE ERROR STATUS
        RET

DSPOUT:  CALL SPOUT ;PRINT TWO SPACES
SPOUT:  MVI  C,' '  ;PRINT A SPACE
        JMP  POUT

HEXOUT:  PUSH PSW   ;POOR PERSONS NUMBER PRINTER
        RRC
        RRC
        RRC
        RRC
        CALL NIBOUT
        POP  PSW
NIBOUT:  ANI  0FH
        ADI  '0'
        CPI  '9'+1
        JC   NIBOK
        ADI  27H
NIBOK:  MOV  C,A
        JMP  POUT

MWDOIT  EQU  $
        ELSE
MWISSUE EQU  $      ;DO A DISK COMMAND, HANDLE TIMEOUTS + ERRORS
        ENDIF

```



```

LXI    H,DMASTAT    ;CLEAR STATUS BYTE
MVI    M,0
OUT    ATTN         ;START THE CONTROLLER
LXI    D,0          ;TIME OUT COUNTER (65536 RETRIES)
MWILOOP MOV  A,M     ;GET STATUS
ORA    A           ;SET UP CPU FLAGS
RM     ;RETURN NO ERROR (CARRY RESET)
STC
RNZ    ;RETURN ERROR STATUS
XTHL  ;WASTE SOME TIME
XTHL
XTHL
XTHL
DCX    D           ;BUMP TIMEOUT COUNTER
MOV    A,D
ORA    E
JNZ    MWILOOP     ;LOOP IF STILL BUSY
STC
RET

MWPTR  LDA    MWDRIIVE ;GET CURRENTLY SELECT DRIVES TRACK ADDRESS
RLC
MOV    E,A
MVI    D,0
LXI    H,MWTAB
DAD    D           ;OFFSET INTO TRACK TABLE
RET

MWTRAN: MOV  H,B
MOV    L,C
INX    H
RET

MWNEGHL:MOV  A,H
CMA
MOV    H,A
MOV    A,L
CMA
MOV    L,A
INX    H
RET

MWHLMDE:XCHG
CALL  MWNEGHL
XCHG
DAD    D
RET

MWHLCDE:MOV  A,H
CMP    D
RNZ
MOV    A,L
CMP    E
RET

MWTAB  EQU    $    ;COLLECTION OF TRACK ADDRESSES

```

```

REPT MAXMW
DB 0FFH ;INITIALIZE TO (WAY OUT ON THE END OF THE DISK)
DB 0FFH
ENDM
DB 0FFH

MWCURL DB 0 ;CURRENT LOGICAL DRIVE
MWDRIVE DB 0FFH ;CURRENTLY SELECTED DRIVE
MWHEAD DB 0 ;CURRENTLY SELECTED HEAD
MWSECTR DB 0 ;CURRENTLY SELECTED SECTOR

DMACHAN EQU $ ;COMMAND CHANNEL AREA
DMASEL0 DB 0 ;DRIVE SELECT
DMASTEP DW 0 ;RELATIVE STEP COUNTER
DMASEL1 DB 0 ;HEAD SELECT
DMADMA DW 0 ;DMA ADDRESS
DB 0 ;EXTENDED ADDRESS
DMARG0 DB 0 ;FIRST ARGUMENT
DMARG1 DB 0 ;SECOND ARGUMENT
DMARG2 DB 0 ;THIRD ARGUMENT
DMARG3 DB 0 ;FOURTH ARGUMENT
DMAOP DB 0 ;OPERATION CODE
DMASTAT DB 0 ;CONTROLLER STATUS BYTE
DMALNK DW DMACHAN ;LINK ADDRESS TO NEXT COMMAND CHANNEL
DB 0 ;EXTENDED ADDRESS

```

ENDIF

```

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

```

```

IF NOSTAND NE 0 ;UNALLOCATED WRITTING VARIABLES
F4E7 00 UNALOC: DB 0 ;UNALLOCATED WRITE IN PROGRESS FLAG
F4E8 0000 OBLOCK: DW 0 ;LAST UNALLOCATED BLOCK NUMBER WRITTEN
F4EA 00 UNADRV: DB 0 ;DRIVE THAT THE BLOCK BELONGS TO
ENDIF

F4EB 0000 CPMSEC: DW 0 ;CP/M SECTOR #

F4ED 00 CPMDRV: DB 0 ;CP/M DRIVE #
F4EE 0000 CPMTRK: DW 0 ;CP/M TRACK #
F4F0 0000 TRUESEC:DW 0 ;PHYSICAL SECTOR THAT CONTAINS CP/M SECTOR

F4F2 00 ERROR: DB 0 ;BUFFER'S ERROR STATUS FLAG
F4F3 00 BUFDRV: DB 0 ;DRIVE THAT BUFFER BELONGS TO
F4F4 0000 BUFTRK: DW 0 ;TRACK THAT BUFFER BELONGS TO
F4F6 0000 BUFSEC: DW 0 ;SECTOR THAT BUFFER BELONGS TO

F4F8 0000 ALTTRK: DW 0 ;ALTERNATE TRACK
F4FA 0000 ALTSEC: DW 0 ;ALTERANTE SECTOR
F4FC 00 LASTDRV:DB 0 ;LAST SELECTED DRIVE

```

\*\*\*\*\*

```

*
* DPB AND DPH AREA.
*
*****
    
```

```

                IF      MAXHD NE 0

0000 #          DPHDSK SET      0                ;GENERATE DPH'S FOR THE HDCA HARD DISKS
                REPT    MAXHD
                LDSK   SET      0
                REPT    HDLOG
                DPHGEN  HD,%DPHDSK,DPBHD,%LDSK
                LDSK   SET      LDSK+1
                DPHDSK SET      DPHDSK+1
                ENDM
                ENDM

F4FD+=          DPHHD0 EQU      $
F4FD+0000       DW      0
F4FF+0000000000 DW      0,0,0
F505+77F9       DW      DIRBUF
F507+2DF5       DW      DPBHD0
F509+07FA       DW      CSVHD0
F50B+07FA       DW      ALVHD0
F50D+=          DPHHD1 EQU      $
F50D+0000       DW      0
F50F+0000000000 DW      0,0,0
F515+77F9       DW      DIRBUF
F517+3DF5       DW      DPBHD1
F519+06FB       DW      CSVHD1
F51B+06FB       DW      ALVHD1
F51D+=          DPHHD2 EQU      $
F51D+0000       DW      0
F51F+0000000000 DW      0,0,0
F525+77F9       DW      DIRBUF
F527+4DF5       DW      DPBHD2
F529+05FC       DW      CSVHD2
F52B+05FC       DW      ALVHD2
    
```

```

                IF      HDPART NE 0                ;USE NON-STANDARD PARTITIONING
    
```

```

*****
*
* HDSECTP IS THE NUMBER OF 128 BYTE SECTORS PER CYLINDER.
*
* HDTRKS IS THE TOTAL NUMBER OF DATA CYLINDERS. EG. IT IS
* THE NUMBER OF CYLIDERS ON THE DRIVE MINUS THE NUMBER OF
* CYLINDERS THAT ARE USED FOR THE SYSTEM. IF THE NUMBER OF
* 'SYSTEM TRACKS' IS NOT ONE THEN THE INITIAL VALUE OF
* 'OFF' SHOULD BE ADJUSTED ACCORDINGLY.
*
* HDTRKS = TRACKS - 1
*
*****
    
```

```

                IF      M10 NE 0
                HDSECTP EQU      336                ;SECTORS PER TRACK
    
```

```

HDTRKS EQU 243 ;TOTAL DATA TRACKS
ENDIF

IF M20 NE 0
HDSECTP EQU 672
HDTRKS EQU 243
ENDIF

IF M26 NE 0
HDSECTP EQU 1024
HDTRKS EQU 201
ENDIF

LDSK SET 0 ;USE NON-STANDARD PARTITIONING
TRACKS SET HDTRKS/HDLOG ;NUMBER OF TRACKS PER PARTITION
DSM SET HDSECTP/8*TRACKS/4-1 ;NUMBER OF GROUPS PER PARTITION
OFF SET 1

REPT HDLOG
DPBGEN HD,%LDSK,%HDSECTP,5,31,1,%DSM,511,0FFH,0FFH,0,%OFF,3
OFF SET OFF+TRACKS
LDSK SET LDSK+1
ENDM

ELSE ;ELSE USE STANDARD DPB'S

DPBHD0 IF M26 NE 0
DW 1024 ;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 3 ;SECSIZ

DPBHD1 DW 1024 ;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 64 ;OFF
DB 3 ;SECSIZ

DPBHD2 DW 1024 ;CP/M SECTORS/TRACK
DB 5 ;BSH
DB 31 ;BLM
DB 1 ;EXM
DW 2047 ;DSM

```

```

DW      511      ;DRM
DB      0FFH     ;AL0
DB      0FFH     ;AL1
DW      0        ;CKS
DW      127     ;OFF
DB      3        ;SECSIZ
ENDIF

```

```

IF      M10 NE 0
DPBHD0 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      3         ;SECSIZ

```

```

DPBHD1 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      3         ;SECSIZ
ENDIF

```

```

IF      M20 NE 0
F52D A002   DPBHD0 DW      672      ;CP/M SECTORS/TRACK
F52F 05     DB      5            ;BSH
F530 1F     DB      31          ;BLM
F531 01     DB      1           ;EXM
F532 F407   DW      2036       ;DSM
F534 FF01   DW      511       ;DRM
F536 FF     DB      0FFH      ;AL0
F537 FF     DB      0FFH      ;AL1
F538 0000   DW      0         ;CKS
F53A 0100   DW      1         ;OFF
F53C 03     DB      3         ;SECSIZ

```

```

F53D A002   DPBHD1 DW      672      ;CP/M SECTORS/TRACK
F53F 05     DB      5            ;BSH
F540 1F     DB      31          ;BLM
F541 01     DB      1           ;EXM
F542 F407   DW      2036       ;DSM
F544 FF01   DW      511       ;DRM
F546 FF     DB      0FFH      ;AL0
F547 FF     DB      0FFH      ;AL1
F548 0000   DW      0         ;CKS

```

```

F54A 6200      DW      98      ;OFF
F54C 03        DB      3       ;SECSIZ

F54D A002      DPBHD2 DW      672     ;CP/M SECTORS/TRACK
F54F 05        DB      5       ;BSH
F550 1F        DB      31      ;BLM
F551 01        DB      1       ;EXM
F552 0404      DW     1028     ;DSM
F554 FF01      DW     511      ;DRM
F556 FF        DB     0FFH     ;AL0
F557 FF        DB     0FFH     ;AL1
F558 0000      DW      0       ;CKS
F55A C300      DW     195      ;OFF
F55C 03        DB      3       ;SECSIZ

```

```

ENDIF
ENDIF
ENDIF ;END OF HD DPH'S AND DPB'S

```

```

IF      MAXMF NE 0

```

```

DPBGEN MF, 0, 20, 3, 7, 0, 04FH, 63, 0C0H, 0, 16, 3, 2
DPBGEN MF, 1, 40, 3, 7, 0, 0A4H, 63, 0C0H, 0, 16, 2, 3
DPBGEN MF, 2, 40, 4, 15, 1, 051H, 63, 80H, 0, 16, 2, 3
DPBGEN MF, 3, 40, 4, 15, 1, 0A9H, 63, 80H, 0, 16, 2, 3

```

```

DN      SET      0
        REPT     MAXMF
        DPHGEN   MF, %DN, DPBMF, %DN
DN      SET      DN+1
        ENDM
        ENDIF

```

```

IF      MAXFD NE 0
DN      SET      0
        REPT     MAXFD
        DPHGEN   FD, %DN, 0, 0
DN      SET      DN+1
        ENDM
        ENDIF

```

```

0000 #      DN      IF      MAXDM NE 0
            SET      0
            REPT     MAXDM
            DPHGEN   DM, %DN, 0, 0
            DN      SET      DN+1
            ENDM

```

```

F55D+=      DPHDM0 EQU     $
F55D+0000   DW      0
F55F+000000000000 DW    0,0,0
F565+77F9   DW     DIRBUF
F567+0000   DW      00
F569+86FC   DW     CSVDM0
F56B+C6FC   DW     ALVDM0
            ENDM

```

```

IF      MAXMW NE 0

```

```

*****
*
* MWSECTP IS THE NUMBER OF 128 BYTE SECTORS PER CYLINDER.
* MWSECTP = 72 * HEADS
*
* MWTRKS IS THE TOTAL NUMBER OF DATA CYLINDERS.
* MWTRKS = TRACKS - 1
*
*****

```

```

IF ST506 NE 0
MWSECTP EQU 288 ;SECTORS PER TRACK
MWTRKS EQU 152 ;TOTAL DATA TRACKS
ENDIF

```

```

IF ST412 NE 0
MWSECTP SET 288
MWTRKS SET 305
ENDIF

```

```

IF CM5619 NE 0
MWSECTP SET 432
MWTRKS SET 305
ENDIF

```

```

DPHDSK SET 0 ;GENERATE DPH'S FOR THE HDDMA HARD DISKS
REPT MAXMW
LDSK SET 0
REPT MWLOG
DPHGEN MW,%DPHDSK,DPBMW,%LDSK
DPHDSK SET DPHDSK+1
LDSK SET LDSK+1
ENDM
ENDM

```

```

IF MWPART NE 0 ;GENERATE DPB'S FOR A HDDMA HARD DISK

```

```

LDSK SET 0 ;USE NON-STANDARD PARTITIONING
TRACKS SET MWTRKS/MWLOG ;NUMBER OF TRACKS PER PARTITION
DSM SET MWSECTP/8*TRACKS/4-1 ;NUMBER OF GROUPS PER PARTITION
OFF SET 1

```

```

REPT MWLOG
DPBGEN MW,%LDSK,%MWSECTP,5,31,1,%DSM,1023,0FFH,0FFH,0,%OFF,4
OFF SET OFF+TRACKS
LDSK SET LDSK+1
ENDM

```

```

ELSE ;USE STANDARD PARTITIONING

```

```

OFF SET 1 ;INITIAL SYSTEM TRACK OFFSET
TRKOFF SET 8192/(MWSECTP/8)+1 ;THE NUMBER OF TRACKS IN A PARTITION
BLOCKS SET MWSECTP/8*MWTRKS ;THE NUMBER OF BLOCKS ON THE DRIVE
PSIZE SET TRKOFF*(MWSECTP/8) ;THE NUMBER OF BLOCKS IN A PARTITION
LDSK SET 0

```

```

REPT   BLOCKS/8192      ;GENERATE SOME 8 MEGABYTE DPB'S
DPBGEN MW,%LDSK,%MWSECPT,5,31,1,2047,1023,0FFH,0FFH,0,%OFF,4
OFF    SET             OFF+TRKOFF
BLOCKS SET             BLOCKS-PSIZE
LDSK   SET             LDSK+1
      ENDM
BLOCKS SET             BLOCKS/4
      IF             BLOCKS GT 256      ;IF THERE IS ANY STUFF LEFT, THEN USE IT
BLOCKS SET             BLOCKS-1
      DPBGEN        MW,%LDSK,%MWSECPT,5,31,1,%BLOCKS,1023,0FFH,0FFH,0,%OFF,4
      ENDIF
      ENDIF
      ENDIF

```

```
F56D = BUFFER EQU $
```

```

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

```

```

F56D 1B7E PROMPT: DB 1BH,7EH ;SWITCH TV950 TO WS MODE
F56F 801A DB 80H,CLEAR ;CLEAN BUFFER AND SCREEN
F571 0D0A0A DB ACR,ALF,ALF
F574 4D6F72726F DB 'Morrow Designs '
F583 36 DB '0'+MSIZE/10 ;CP/M MEMORY SIZE
F584 34 DB '0'+(MSIZE MOD 10)
F585 4B2043502F DB 'K CP/M ' ;CP/M VERSION NUMBER
F58C 32 DB CPMREV/10+'0'
F58D 2E DB '.'
F58E 32 DB (CPMREV MOD 10)+'0'
F58F 20 DB ' '
F590 45 DB (REVNUM/10)+'A'-1
F591 33 DB (REVNUM MOD 10)+'0'
F592 0D0A DB ACR,ALF

```

```

;
; PRINT A MESSAGE LIKE:
;
; AB: DJDMA 8", CD: DJDMA 5 1/4", E: HDDMA M5
;

```

```

0000 # MSDRV SET 0 ;START WITH DRIVE A:
      MSBUMP MACRO NDRIVES ;PRINT A DRIVE NAME
      IF DN GT 1
      DB ', '
      ENDIF
      REPT NDRIVES
      DB MSDRV+'A'
MSDRV SET MSDRV+1
      ENDM
      DB ': '
      ENDM

```



```

PRHEX MACRO DIGIT ;WRITE A BYTE IN HEX
PRNIB DIGIT/10H
PRNIB DIGIT
ENDM

PRNIB MACRO DIGIT ;WRITE A DIGIT IN HEX
TEMP SET DIGIT AND 0FH
IF TEMP < 10
DB TEMP + '0'
ELSE
DB TEMP - 10 + 'A'
ENDIF
ENDM

0001 # DN SET 1 ;GENERATE THE DRIVE MESSAGES

REPT 16 ;RUN OFF AT LEAST 16 DRIVES

IF DN EQ HDORDER ;GENERATE THE HDCA'S MESSAGE
MSBUMP MAXHD*HDLOG
DB 'HDCA '
IF MAXHD GT 1
DB '(' , MAXHD+'0' , ')'
ENDIF
IF M10 NE 0
IF M10M NE 0
DB 'Memorex'
ELSE
DB 'Fujitsu'
ENDIF
DB ' M10'
ENDIF
IF M20 NE 0
DB 'Fujitsu M20'
ENDIF
IF M26 NE 0
DB 'Shugart M26'
ENDIF
ENDIF

IF DN EQ MWORDER ;GENERATE THE HDDMA'S MESSAGE
MSBUMP MAXMW*MWLOG
DB 'HDDMA'
IF MWQUIET EQ 0
DB ' '
IF MAXMW GT 1
DB '(' , MAXMW+'0' , ')'
ENDIF
IF ST506 NE 0
DB 'M5'
ENDIF
IF ST412 NE 0
DB 'M10'
ENDIF
IF CM5619 NE 0

```

```

DB      'M16'
ENDIF
ENDIF
ENDIF

```

```

IF      DN EQ FDORDER      ;GENERATE THE 2D/B MESSAGE
MSBUMP MAXFD
DB      'DJ2D/B @'
PRHEX  FDORIG/100H
PRHEX  FDORIG
ENDIF

```

```

IF      DN EQ DMORDER      ;GENERATE THE DJDMA 8 MESSAGE
MSBUMP MAXDM
DB      'DJDMA 8"'
ENDIF

```

```

IF      DN EQ MFORDER      ;GENERATE THE DJDMA 5 1/4 MESSAGE
MSBUMP MAXMF
DB      'DJDMA 5 1/4"'
ENDIF

```

```

DN      SET      DN+1
        ENDM

```

```

F594+41 DB      MSDRV+'A'
F595+42 DB      MSDRV+'A'
F596+43 DB      MSDRV+'A'
F597+3A20 DB      ':'
F599+4844434120 DB      'HDCA '
F59E+46756A6974 DB      'Fujitsu M20'
F5A9+2C20 DB      ','
F5AB+44 DB      MSDRV+'A'
F5AC+3A20 DB      ':'
F5AE+444A444D41 DB      'DJDMA 8"'

```

```

F5B6 0D0A DB      ACR,ALF
F5B8 00 DB      0 ;END OF MESSAGE

```

```

*****
*
* CBOOT IS THE COLD BOOT LOADER. ALL OF CP/M HAS BEEN LOADED IN *
* WHEN CONTROL IS PASSED HERE.
*
*****

```

```

F5B9 310001 CBOOT: LXI      SP,TPA      ;SET UP STACK

F5BC AF      XRA      A      ;CLEAR COLD BOOT FLAG
F5BD 32DBEB STA      CWFLG
F5C0 3244EA STA      GROUP      ;CLEAR GROUP SELECT BYTE
F5C3 32EDF4 STA      CPMDRV      ;SELECT DISK A:
F5C6 320400 STA      CDISK

F5C9 2103EA LXI      H,BIOS+3      ;PATCH COLD BOOT TO WARM CODE
F5CC 2201EA SHLD     BIOS+1

```

```

F5CF 3A43EA      LDA      IOBYT      ;INITIALIZE THE IOBYTE
F5D2 320300      STA      IOBYTE

F5D5 116DF9      LXI      D,BADMAP    ;CLEAR OUT BAD MAP
F5D3 12          STAX     D
F5D9 216EF9      LXI      H,BADMAP+1
F5DC 010900      LXI      B,9*BADSIZ ;32 MAP ENTRIES
F5DF CDE6EE      CALL     MOVBYT
F5E2 36FF        MVI      M,0FFH     ;END MARKER

                IF      CONTYP NE 6      ;NON IOBYTE INITS
                IF      CONTYP NE 0      ;DO NOT CALL TTYSET FOR PROM'S
F5E4 CDF3F5      CALL     TTYSET     ;INITIALIZE THE TERMINAL
                ENDIF

                IF      LSTTYP NE 0      ;DO NOT CALL LSTSET FOR PROM'S
F5E7 CD55F6      CALL     LSTSET     ;INITIALIZE THE LIST DEVICE
                ELSE      ;DO IOBYTE INITS
                LXI      H,DEVSET        ;DEVICE SETUP ROUTINE POINTER TABLE
CBOOT0: MOV      E,M          ;LOAD A ROUTINE ADDRESS
                INX      H
                MOV      D,M
                INX      H
                MOV      A,D          ;TEST FOR THE END OF THE TABLE
                ORA      E
                JZ       CBOOT2
                PUSH     H          ;SAVE THE TABLE POINTER
                LXI      H,CBOOT1     ;RETURN ADDRESS
                PUSH     H
                XCHG
                PCHL
                ;'CALL' A DEVICE SETUP ROUTINE
CBOOT1: POP      H          ;RESTORE THE TABLE POINTER
                JMP      CBOOT0

DEVSET: DW      TTYSET, CRTSET, UC1SET ;DEVICE SETUP ROUTINE POINTERS
                DW      PTRSET, UR1SET, UR2SET
                DW      PTPSET, UP1SET, UP2SET
                DW      LPTSET, UL1SET, 0

CBOOT2 EQU      $
                ENDIF

F5EA 216DF5      LXI      H,PROMPT    ;PREP FOR SENDING SIGNON MESSAGE
F5ED CD05EF      CALL     MESSAGE    ;SEND THE PROMPT
F5F0 C386EB      JMP      GOCPM
    
```

```

*****
*
*  CONSOLE AND LIST DEVICE INITIALIZATION ROUTINES FOLLOW.
*
*****
    
```

```

                IF      CONTYP EQ 2      ;MULTI I/O, DECISION I
    
```

```

*****
    
```

```

*
* TERMINAL INITIALIZATION ROUTINE. THIS ROUTINE READS THE SENSE *
* SWITCH ON THE WB-14 AND SETS THE SPEED ACCORDINGLY. *
*
*****

```

```

F5F3 CDCDEA      TTYSET: CALL    SELG0      ;SELECT GROUP 0
F5F6 DB49        IN      SENSESW     ;GET SENSE SWITCH (FF ON A MULTIO)
F5F8 F5          PUSH    PSW
F5F9 CDD3EA      CALL    SELCON     ;SELECT CONSOLE
F5FC F1          POP     PSW
F5FD F5          PUSH    PSW
F5FE CD0BF6      CALL    TINI0      ;INITIALIZE THE CONSOLE
F601 F1          POP     PSW
F602 F5          PUSH    PSW
F603 CDDBEA      CALL    SELRDR     ;SELECT THE READER/PUNCH
F606 F1          POP     PSW
F607 CD0BF6      CALL    TINI0      ;INITIALIZE THE READER/PUNCH
F60A C9          RET

F60B E6E0        TINI0:  ANI      0E0H      ;MASK IN UPPER THREE BITS
F60D 07          RLC
F60E 07          RLC
F60F 07          RLC
F610 FE07        CPI      7          ;CHECK FOR SENSE = 7 (DEFAULT SETTING)
F612 CA23F6      JZ      DFBAUD     ;USE DEFAULT BAUD RATE

F615 2147F6      LXI      H,BTAB     ;POINTER TO BAUD RATE TABLE
F618 87          ADD     A
F619 5F          MOV     E,A
F61A 1600        MVI     D,0
F61C 19          DAD     D          ;GET A POINTER INTO BAUD RATE TABLE
F61D 5E          MOV     E,M
F61E 23          INX     H          ;BUMP TO HIGH BYTE OF WORD
F61F 56          MOV     D,M
F620 C327F6      JMP     SETIT      ;SET BAUD RATE

F623 2A45EA      DFBAUD: LHL D,DEFCON ;USE DEFAULT BAUD RATE
F626 EB          XCHG

F627 3E87        SETIT:  MVI     A,DLAB+WLS1+WLS0+STB ;ENABLE DIVISOR ACCESS LATCH
F629 D34B        OUT     LCR      ;SET THE BAUD RATE IN (DE)
F62B 7A          MOV     A,D
F62C D349        OUT     DLM      ;SET UPPER DIVISOR
F62E 7B          MOV     A,E
F62F D348        OUT     DLL      ;SET LOWER DIVISOR

F631 3E07        MVI     A,WLS1+WLS0+STB ;CLEAR DIVISOR LATCH
F633 D34B        OUT     LCR
F635 AF          XRA     A
F636 D349        OUT     IER      ;SET NO INTERRUPTS
F638 D34D        OUT     LSR      ;CLEAR STATUS
F63A 3E03        MVI     A,DTRENB+RTSENB ;ENABLE DTR AND RTS OUTPUTS TO TERMINAL
F63C D34C        OUT     MCR
F63E DB4E        IN      MSR      ;CLEAR MODEM STATUS REGISTER
F640 DB4D        IN      LSR      ;CLEAR LINE STATUS REGISTER

```

```
F642 DB48      IN      RBR      ;CLEAR RECIEVER BUFFERS
F644 DB48      IN      RBR
F646 C9        RET
```

```
F647 1704     BTAB:   DW      1047   ;110 BAUD      000
F649 8001     DW      384     ;300          001
F64B 6000     DW      96      ;1200         010
F64D 3000     DW      48      ;2400         011
F64F 1800     DW      24      ;4800         100
F651 0C00     DW      12      ;9600         101
F653 0600     DW      6       ;19200        110
                        ;DEFCON      111
```

```
ENDIF ;MULTI I/O, DECISION I
```

```
IF      CONTYP EQ 3 ;2D/B CONSOLE INITIALIZATION
```

```
TTYSET: CALL  FDTSTAT ;CLEAN INPUT BUFFER
        RNZ   ;ALL EMPTY
        CALL  FDCIN
        JMP   TTYSET
```

```
ENDIF ;2D/B CONSOLE
```

```
IF      CONTYP EQ 4
TTYSET: CALL  DMINIT ;SEE IF CONTROLLER PRESENT
        RC    ;NO CONTROLLER, RETURN
        LXI  D,DMACI ;CONSOLE INITIALIZATION SEQUENCE
        LXI  H,DMCHAN
        LXI  B,10 ;COMMAND LENGTH
        CALL MOVBYT
        DCX  H
        XRA  A ;CLEAR SERIAL INPUT STATUS
        STA  SERIN+1
        JMP  DOCMD2 ;DO STUFF AND RETURN
```

```
DMACI:  DB    WRITEM ;ZOT MONITOR DISABLE FLAG
        DW    TTYSET ;ANY NON-ZERO BYTE WILL DO
        DB    0
        DW    1 ;ONE BYTE
        DW    13F5H ;MAGICAL PLACE IN MONITOR
        DB    SENABL ;ENABLE SERIAL INPUT
        DB    1
```

```
ENDIF
```

```
*****
*
* INITIALIZE THE NORTH STAR MOTHER BOARD, LEFT SERIAL PORT, RIGHT
* SERIAL PORT, AND NORTH STAR RAM PARITY.
*
*****
```

```
IF      CONTYP EQ 6 ;NORTH STAR DRIVERS
```

```
TTYSET: ;SET UP THE PARALLEL PORT + MOTHERBOARD
```

```

XRA      A                ;INITIALIZE MOTHER BOARD
OUT      6
OUT      6
OUT      6
OUT      6

MVI      A,30H            ;RESET THE PARALLEL PORT INPUT FLAG
OUT      NSPSTA
MVI      A,60H            ;SET THE PARALLEL PORT OUTPUT FLAG
OUT      NSPSTA
MVI      A,ACR            ;FORCE A CR OUT THE PARALLEL PORT
CALL     NSPOUT

                                ;INITIALIZE THE LEFT SERIAL PORT
                                ;SEE THE EQUATES FOR BIT DEFINATIONS

MVI      A,NSLIN1
OUT      NSLSTA
MVI      A,NSLIN2
OUT      NSLSTA
XRA      A                ;CLEAR THE INPUT/OUTPUT BUFFERS
OUT      NSLDAT
IN       NSLDAT
IN       NSLDAT

                                ;INITIALIZE THE RIGHT SERIAL PORT
                                ;SEE THE EQUATES FOR BIT DEFINATIONS

MVI      A,NSRIN1
OUT      NSRSTA
MVI      A,NSRIN2
OUT      NSRSTA
XRA      A                ;CLEAR THE INPUT/OUTPUT BUFFERS
OUT      NSRDAT
IN       NSRDAT
IN       NSRDAT

                                ;RESET PARITY ON NORTH STAR RAMS
                                ;DISABLE PARITY LOGIC

IF       NSRAM NE 0
MVI      A,40H
OUT      NSRAM
LXI      H,0                ;STARTING ADDRESS

NSET0:   MOV      A,M        ;GET A BYTE
        MOV      M,A        ;REWRITE, SET PROPER PARITY
        INR     L           ;BUMP THE ADDRESS POINTER
        JNZ     NSET0

NSET1:   INR     H           ;SKIP TO THE NEXT MEMORY PAGE
        JZ      NSET2       ;SKIP IF ALL DONE
        MVI     A,(HIGH $) + 1 ;IS THE POINTER ABOVE US?
        CMP     H           ;SET CARRY IF POINTER IS <= OUR PAGE+1
        JC      NSET0       ;RESET THE NEXT PAGES PARITY
        MOV     A,M         ;TEST FOR A PROM OR NO MEMORY
        MOV     B,A         ;SAVE THE ORIGINAL BYTE
        CMA     A           ;SEE IF THIS LOCATION WILL CHANGE
        MOV     M,A
        CMP     M           ;TEST FOR A CHANGE
        MOV     M,B         ;RESTORE THE ORIGINAL VALUE
        JZ      NSET0       ;VALUE COMPLEMENTED, MUST BE RAM
        ORA     A           ;TEST FOR NO MEMORY PRESENT

```

```

JZ      NSET1      ;SKIP TO THE NEXT PAGE IF NO MEMORY
LXI     D,700H    ;SKIP 2K BYTES OF 'PROM'
DAD     D
JNC     NSET1     ;DO A PAGE CHECK IF NO OVERFLOW

NSET2:  MVI     A,41H    ;RE-ENABLE PARITY ON THE MEMORY BOARDS
        OUT     NSRAM
        ENDIF

CRTSET: ;NULL ROUTINES
PTRSET:
PTPSET:
UC1SET:
UR1SET:
UR2SET:
UP1SET:
UP2SET:
LPTSET:
UL1SET:

        RET
        ENDIF      ;NORTH STAR DRIVERS

        IF      (LSTTYP GE 2) AND (LSTTYP LE 5) ;SERIAL MULTI I/O LIST DRIVERS

F655 CDE3EA  LSTSET: CALL  SELLST      ;SELECT PRINTER GROUP
F658 3E80    MVI     A,DLAB      ;ACCESS DIVISOR LATCH
F65A D34B    OUT     LCR
F65C 2A47EA  LHLD   DEFLST      ;GET LST: BAUD RATE DIVISOR
F65F 7C      MOV     A,H
F660 D349    OUT     DLM      ;SET UPPER BAUD RATE
F662 7D      MOV     A,L
F663 D348    OUT     DLL
F665 3E07    MVI     A,STB+WLS0+WLS1 ;2 STOP BITS + 8 BIT WORD
F667 D34B    OUT     LCR
F669 3E03    MVI     A,DTRENB+RTSENB ;DTR + RTS ENABLED
F66B D34C    OUT     MCR
F66D DB48    IN      RBR      ;CLEAR INPUT BUFFER
F66F AF      XRA     A
F670 D349    OUT     IER      ;NO INTERRUPTS
F672 C9      RET

        ENDIF

F673 00FF00  DB      0,0FFH,0

0C76 =      CODELEN EQU  ($-BIOS)      ;LENGTH OF CBIOS CODE

        IF      CODELEN GT 1000H      ;TEST FOR SYSGEN PROBLEMS
        'FATAL ERROR, system is too big for SYSGEN rev. 4.X'
DBGTMP SET  CODELEN      ;CBIOS CODE LENGTH ! <DEBUG>
        ENDIF

        IF      DEBUG
DBGTMP SET  CODELEN      ;CBIOS CODE LENGTH ! <DEBUG>
        ENDIF

```

```
F676 DS 512-($-BUFFER) ;BUFFER FOR 512 BYTE SECTORS
      IF (MAXFD NE 0) OR (MAXDM NE 0) OR (MAXMW NE 0)
F76D DS 512 ;ADDITIONAL SPACE FOR 1K SECTOR DEVICES
      ENDIF
```

```
*****
*
* EACH BAD MAP ENTRY CONSISTS OF 9 BYTES:
* LOGICAL DRIVE NUMBER (1 BYTE)
* TRACK NUMBER OF BAD SECTOR (2 BYTES)
* SECTOR NUMBER OF BAD SECTOR (2 BYTES)
* TRACK NUMBER OF ALTERNATE SECTOR (2 BYTES)
* SECTOR NUMBER OF ALTERNATE SECTOR (2 BYTES)
*
```

```
F96D BADMAP: DS BADSIZ*9+1 ;32 ENTRIES + END MARKER
F977 DIRBUF: DS 128 ;DIRECTORY BUFFER
F9F7 TEMPB: DS 16 ;A LITTLE TEMPORARY BUFFER
```

```
*****
*
* ALLOCATION AND CHECKED DIRECTORY TABLE AREA
*
```

```
      IF MAXHD NE 0
      IF HDPART NE 0 ;USE NON-STANDARD PARTITIONING

TRACKS SET HDTRKS/HDLOG ;NUMBER OF TRACKS PER PARTITION
DSM SET HDSECTP/8*TRACKS/4-1 ;NUMBER OF GROUPS PER PARTITION
ALV SET (DSM/8)+1

DN SET 0
REPT MAXHD*HDLOG ;GENERATE CKS AND ALV TABLES
ALLOC HD,%DN,%ALV,0
DN SET DN+1
ENDM

      ELSE ;STANDARD PARTITIONING

0000 # DN SET 0
      REPT MAXHD
      IF M26 NE 0
      ALLOC HD,%DN,252,0
DN SET DN+1
      ALLOC HD,%DN,252,0
DN SET DN+1
      ALLOC HD,%DN,256,0
DN SET DN+1
      ENDIF

      IF M10 NE 0
```



```

        DN      ALLOC  HD,%DN,159,0
              SET    DN+1
              ALLOC  HD,%DN,161,0
        DN      SET    DN+1
              ENDIF

              IF     M20 NE 0
        DN      ALLOC  HD,%DN,255,0
              SET    DN+1
              ALLOC  HD,%DN,255,0
        DN      SET    DN+1
              ALLOC  HD,%DN,129,0
        DN      SET    DN+1
              ENDIF
              ENDM

FA07+      CSVHD0: DS    0
FA07+      ALVHD0: DS   255
FB06+      CSVHD1: DS    0
FB06+      ALVHD1: DS   255
FC05+      CSVHD2: DS    0
FC05+      ALVHD2: DS   129
              ENDIF

              ENDIF

        DN      IF     MAXFD NE 0
              SET    0

              REPT  MAXFD
        DN      ALLOC  FD,%DN,75,64
              SET    DN+1
              ENDM
              ENDIF

0000 #     DN      IF     MAXDM NE 0
              SET    0

              REPT  MAXDM
        DN      ALLOC  DM,%DN,75,64
              SET    DN+1
              ENDM

FC86+      CSVDM0: DS    64
FC86+      ALVDM0: DS    75
              ENDIF

        DN      IF     MAXMF NE 0
              SET    0
              REPT  MAXMF
        DN      ALLOC  MF,%DN,22,16
              SET    DN+1
              ENDM
              ENDIF

              IF     MAXMW NE 0
              IF     MWPART NE 0

```

;USE NON-STANDARD PARTITIONING

```

TRACKS SET MWTRKS/MWLOG ;NUMBER OF TRACKS PER PARTITION
DSM SET MWSECTP/8*TRACKS/4-1 ;NUMBER OF GROUPS PER PARTITION
ALV SET (DSM/8)+1
DN SET 0

```

```

REPT MAXMW*MWLOG ;GENERATE CKS AND ALV TABLES
ALLOC MW,%DN,%ALV,0
DN SET DN+1
ENDM

```

```

ELSE ;USE STANDARD PARTITIONING

```

```

DN SET 0
TRKOFF SET 8192/(MWSECTP/8)+1
PSIZE SET TRKOFF*(MWSECTP/8)

```

```

REPT MAXMW

```

```

BLOCKS SET MWSECTP/8*MWTRKS

```

```

REPT BLOCKS/8192 ;GENERATE SOME 8 MEGABYTE ALV'S
ALLOC MW,%DN,256,0
BLOCKS SET BLOCKS-PSIZE
DN SET DN+1
ENDM

```

```

BLOCKS SET BLOCKS/4

```

```

IF BLOCKS GT 256 ;USE THE REMAINDER
BLOCKS SET BLOCKS-1
ALV SET (BLOCKS/8)+1
ALLOC MW,%DN,%ALV,0
DN SET DN+1
ENDIF
ENDM

```

```

ENDIF
ENDIF

```

```

0014 = BIOSLEN EQU (HIGH ($-BIOS))+1 ;BIOS LENGTH IN PAGES

```

```

IF BIOSLEN GT BIOSLN ;TEST FOR OVERFLOW
'FATAL ERROR, system overflow. BIOSLN must be at least'
DBGTMP SET BIOSLEN ;BIOSLN! <DEBUG>
ENDIF

```

```

IF DEBUG
DBGTMP SET BIOSLN ;CURRENT BIOSLN! <DEBUG>
IF BIOSLN GT BIOSLEN
DBGTMP SET BIOSLEN ;OPTIMAL BIOSLN! <DEBUG>
ENDIF
ENDIF

```

```

FD11 END

```

0006 AACK	0007 ABEL	0008 ABS	EFDD ACCOK	000D ACR
007F ADEL	001B AESC	0003 AETX	000C AFF	0009 AHT
000A ALF	EE8A ALL	EE83 ALT	EAA6 ALT2	EEDB ALTMIS
F4FA ALTSEC	F4F8 ALTTRK	FCC6 ALVDM0	FA07 ALVHD0	FB06 ALVHD1
FC05 ALVHD2	0000 ANUL	001E ARS	0020 ASP	001F AUS
EAED AUTOFLG	0000 AUTOLF	EAEB AUTOST	000B AVT	EE47 AWRITIN
ECE6 BADE	ECCE BADL	F96D BADMAP	ED0F BADPTR	ECE9 BADRET
0001 BADSIZ	0004 BANK	0E00 BDOSLN	DC00 BDOS	0014 BIOSLEN
0016 BIOSLN	EA00 BIOS	0026 BRACHA	F647 BTAB	F4F3 BUFDRV
0080 BUFF	F56D BUFFER	F4F6 BUFSEC	F4F4 BUFTRK	ED90 BUFWRN
F0DD BUILD	0006 CBAUD	E9E7 CBLOCK	F5B9 CBOOT	D400 CCP
0800 CCPLN	0004 CDISK	0040 CHECK	EA09 CIN	EBD0 CLDBOT
EBB7 CLDCMND	001A CLEAR	004A CLK	F4A9 CLOAD	0000 CM5619
0C76 CODELEN	EAF2 COLDCM	EAF3 COLDEND	EAAE COLDMES	0004 COMPLT
0001 CONGRP	EAA5 CONIN	EAA8 CONIN1	EAC2 CONIST	EAB4 CONOUT
EAB7 CONOUT1	EA06 CONST	0002 CONTYP	EA9E COSTRP	EB85 COUNT
EA0C COUT	0004 COVER	000A CPERI	ED71 CPMDMA	F4ED CPMDRV
0016 CPMREV	F4EB CPMSEC	F4EE CPMTRK	0010 CRRDY	0020 CRSTB
1020 CRSTRD	FC86 CSVDM0	FA07 CSVHD0	FB06 CSVHD1	FC05 CSVHD2
0010 CTS	EBDB CWFLG	0002 D10	0004 D11	0008 D12
0001 D9	0048 DAISI0	0049 DAISI1	0048 DAISY0	0049 DAISY1
000A DBAD	EFA0 DCRC	0000 DEBUG	EA45 DEFCON	EA47 DEFLST
0020 DENABLE	F623 DFBAUD	006E DFRMLN	0004 DHOME	EB3A DIAOST
EB5F DIOUT	EB54 DIOUTA	EB74 DIOUTB	F977 DIRBUF	ED3D DIVDONE
EFA2 DIVLOG	ED2F DIVLOOP	EFA4 DIVLX	F015 DIVSPT	F017 DIVSX
0080 DLAB	0048 DLL	0049 DLM	EA4C DMARAP	0050 DMCHAN
F45E DMDEN	F3BB DMDMA	F4BC DMDOIT	F24D DMDST	01FF DMFSET
0066 DMFSTP	F371 DMGET	0025 DMHALTC	F43C DMHOME	F3D3 DMINIT
F3E0 DMINWT	F3ED DMIOK	00EF DMKICK	F313 DMLDRV	F4D7 DMLOC0
F4DF DMLOC1	F4E6 DMLOG	F4D2 DMNTRK	0002 DMORDER	F3FC DMPARM
F49F DMREAD	0023 DMSDMA	F443 DMSEC	F43E DMSEEK	F39E DMSSEL2
F45A DMSSEL	F2D6 DMSCLR	F2F7 DMSIDE1	F2FD DMSIDE2	F2F4 DMSIDEA
F44D DMSIDE	F35F DMSOK	F4DA DMSPAR	F3AE DMSSEC	F45F DMSTAT
F404 DMSTR0	F42C DMSTR1	F2DE DMTRAN	0B00 DMTRCK	F4CC DMWAIT
F264 DMWARM	F275 DMWBAD	F291 DMWCHN	F4D1 DMWCON	F2B1 DMWEND
F49C DMWRITE	F2B4 DMWSEC	F29C DMWST	F3C4 DOCMD	F3C7 DOCMD2
EC0A DOHOME	EBF8 DONOP	F23D DP1024D	F1FD DP1024S	EA4E DPARAM
F20D DPB128D	F1CD DPB128S	F21D DPB256D	F1DD DPB256S	F22D DPB512D
F1ED DPB512S	F52D DPBHD0	F53D DPBHD1	F54D DPBHD2	F55D DPHDM0
F4FD DPHHD0	F50D DPHHD1	F51D DPHHD2	EC92 DPHTAB	EA39 DRCONF
0008 DREAD	0001 DR	0020 DRVRDY	0007 DSDMA	0002 DSEL1
0003 DSEL2	0020 DSR	0006 DSSEC	0001 DSTRAN	0005 DSTRK
EA3B DSTTAB	0001 DTRENB	ED4C DTSLOP	0000 DWBOOT	0009 DWRITE
0008 ENINT	0005 ENTRY	F4F2 ERROR	0000 FDORDER	EDF0 FILL
ED8F FLUSH	EE31 FNALOC	EE50 FREAD	0001 FUJITSU	EE4B FWRTIN
EC74 GDPH	ECA2 GETBAD	F011 GETSPT	EB86 GOCPM	EA44 GROUP
004F GRPSEL	0022 GSTAT	0000 GZERO	F0EC HDADD	0001 HDCA
0004 HDCLOK	0051 HDCMND	0050 HDCNTL	F0EB HDCUR	EFB1 HDD2
0053 HDDATA	F0EE HDDISK	EFF0 HDDMA	EFAB HDDRV	EF12 HDDST
0001 HDFREN	0052 HDFUNC	EFBE HDHOME	EF95 HDL2	EF63 HDLDRV
0003 HDLOG	0001 HDORDER	0050 HDORG	0000 HDPART	F09C HDPREP
F0D2 HDPTR	F01E HDREAD	0051 HDRESLT	0004 HDRLEN	0002 HDRUN
F000 HDSEC	F0F0 HDSECT	EFCC HDSEEK	0052 HDSKOMP	0015 HDSPT
0050 HDSTAT	EF82 HDTDEL	F0F1 HDTRAK	EF5F HDTRAN	EFDA HDTRK2
EF29 HDWARM	0008 HDWPRT	F053 HDWRITE	EF3E HDWRLD	EF4E HDWRRD
EF51 HDWR	F0EF HEAD	0078 HINC	EBFF HOME	0000 IDBUFF

0049 IER	0040 INDEX	ED80 INTO	0024 INTRQC	0003 IOBYTE
EA43 IOBYT	EA3F IOCONF	0008 ISBUFF	EE5C JUMPBUF	EE60 JUMPER
EE67 JUMPL	EA4B LASTCH	F4FC LASTDRV	0060 LBAUD	004B LCR
F4E5 LLDRV	F4E4 LLSS	F4E3 LLTRK	0006 LPERI	004D LSR
EA49 LSTAND	0003 LSTGRP	EB19 LSTOST	EB0C LSTOUT	F655 LSTSET
0003 LSTTYP	EA4A LSTXOR	0000 M10F	0000 M10	0000 M10M
0001 M20	0000 M26	EEEC M8080	0400 MAXCHRS	0001 MAXDM
0000 MAXFD	0001 MAXHD	0004 MAXLOG	0000 MAXMF	0000 MAXMW
0630 MAXRGT	0048 MBASE	004C MCR	EF05 MESSAGE	0000 MFORDER
EEE3 MOV128	EEE6 MOVBYT	ED5B MOVE	0040 MSIZE	004E MSR
FFFF MULTIO	0000 MULTR3	0000 MWORDER	0000 MWPART	0000 MWQUIET
0009 MWSPT	0040 N2SIDE	0080 NDUBL	ED0B NOBAD	0001 NOSTAND
00FB NSTEP	00FC NULL	00A0 NUMTABS	F4E8 OBLOCK	3700 OFFSETC
EB3F OKIOST	EB42 OKOUT	EB49 OKOUT1	ECF5 OMES	0002 OPDONE
ED79 OUTOF	ECEC OVERFLO	0002 PAPER	0008 PFRDY	0010 PFSTB
0810 PFSTRD	EA0F POUT	ED9F PREP	F082 PROCESS	F56D PROMPT
0080 PSELECT	0004 PSTEP	0020 PWRDY	0040 PWSTB	2040 PWSTRD
0048 RBR	0020 RDSECT	0029 RDTRCK	ED74 RDWR	ED1A READ
00A0 READM	0080 READY	EB26 REDIR	EB29 REDIR0	EB2E REDIR1
ED21 REDWRT	0080 REST	0010 RESTOR	EEFC RETDPH	000A RETRIES
EDA9 RETRYLP	0002 RETRY	0035 REVNUM	0001 RIBBON	0040 RLIFT
0001 RSECT	F03E RTLOOP	0002 RTSENB	ED1E RWENT	EDD2 RWOP
0001 S0	0002 S1	0200 SECLEN	ED60 SECPSEC	ED22 SECSIZ
EC13 SECTRAN	EAD3 SELCON	EACD SELG0	EAE3 SELLST	EADB SELRDR
002C SENABL	0049 SENSESW	003E SERIN	002B SEROUT	0027 SETCHA
0028 SETCRC	EC47 SETD0	EC4C SETD1	EC56 SETD2	EC60 SETD3
EBF9 SETDMA	EC1C SETDRV	F37A SETHIGH	F627 SETIT	002E SETLOG
EBF3 SETSEC	F0F5 SETTLE	EC0D SETTRK	0000 SIZE	EFE1 SLOOP
0003 SMASK	0000 ST412	0000 ST506	0004 STB	F448 STORES
F9F7 TEMPB	F3CE TESTS	0048 THR	0020 THRE	F60B TINIO
0001 TKZERO	0008 TMOUT	0100 TPA	002D TRKSIZ	F4F0 TRUESEC
F5F3 TTYSET	F4EA UNADRV	F4E7 UNALOC	0030 VINC	EAF3 WARMCM
EAF4 WARMEND	EAF0 WARMES	EA03 WBOOT	EBDC WBOOT	0000 WBOT
0010 WFAULT	0001 WLS0	0002 WLS1	0000 WMDRIVE	ED11 WRITE
00A1 WRITEM	ED8A WRITYP	0021 WRSECT	002A WRTRCK	EFF6 WSDONE
0005 WSECT	F05F WTLOOP	F18C XLT124	F0FF XLT128	F11A XLT256
F14F XLT512	F0F7 XLTS	0013 XOFF	0011 XON	EEF7 Z80MOV
F0C0 ZKEY	EC81 ZRET			

\*\*\*\*\*

\*  
\* MORROW DESIGNS CP/M VERS 2.2 COLD BOOT LOADER.  
\* CBIOS REVISION E.2, 3/4/82.  
\*

\* THE FOLLOWING ROUTINES WILL BOOT CP/M FROM THE  
\* DISK JOCKEY 2D REV. B 8 INCH DISK CONTROLLER (DJ2D/B),  
\* DISK JOCKEY DMA 8 + 5 1/4 INCH CONTROLLER (DJDMA),  
\* HARD DISK CONTROLLER REVISION 3 (HDC3),  
\* OR THE HARD DISK DMA (HDDMA) DISK CONTROLLERS.  
\*

\* PROVISIONS HAVE BEEN MADE FOR A MICRONIX BOOT LOADER.  
\* THIS LOADER ALWAYS GETS LOADED TO 0100H.  
\*

\* 8 INCH FLOPPY DISK BOOT LOADER FOR THE  
\* MORROW DESIGNS DISK JOCKEY 2D/B (DJ2DB)  
\*

\* THE 'ORDER' COLUMN IS THE INTERLEAVE SEQUENCE USED BY THE  
\* LOADER DURING THE LOAD.  
\*

TRACK	SECTOR	SYSGEN	LOAD	ORDER	NAME
0	1	900	FF00	0	BOOT LOADER
0	2	980			UNUSED
0	3	A00			
0	4	A80			
0	5	B00	9100	1	CCP
0	6	B80	9180	12	
0	7	C00	9200	2	
0	8	C80	9280	13	
0	9	D00	9300	3	
0	10	D80	9380	14	
0	11	E00	9400	4	
0	12	E80	9480	15	
0	13	F00	9500	5	
0	14	F80	9580	16	
0	15	1000	9600	6	
0	16	1080	9680	17	
0	17	1100	9700	7	
0	18	1180	9780	18	
0	19	1200	9800	8	
0	20	1280	9880	19	
0	21	1300	9900	9	BDOS
0	22	1380	9980	20	
0	23	1400	9A00	10	
0	24	1480	9A80	21	
0	25	1500	9B00	11	
0	26	1580	9B80	22	

\* TRACK 1 IS RECORDED IN DOUBLE DENSITY FORMAT. THERE ARE  
\* 1024 BYTES PER SECTOR.  
\*

1	1	1600	9C00	4	
1	2	1A00	A000	1	
1	3	1E00	A400	5	CBIOS (@ A700H)
1	4	2200	A800	2	

\*  
\*

```

*      1      5      2600  AC00      6
*      1      6      2A00  B000      3
*      1      7      2E00  B400      7
*      1      8      3200  B800      UNUSED
*

```

```

* THREE SPARE SECTORS (TRACK 0, SECTORS 2 TO 4) HAVE BEEN
* PROVIDED FOR A MORE ADVANCED BOOT LOADER AT A LATER DATE.
*

```

```

* THE WARM BOOT LOADER STARTS ON TRACK 0, SECTOR 5 AND
* CONTINUES THROUGH TO TRACK 1 SECTOR 3. ONLY THE FIRST
* 3/4 K BYTES OF TRACK 1, SECTOR 3 IS LOADED SINCE CP/M
* REQUIRES THAT THE WARM BOOT LOADER LOAD UP TO THE START
* OF (BUT NOT PAST) THE CBIOS JUMP TABLE.
*

```

```

*
*           8 INCH FLOPPY DISK BOOT LOADER FOR THE
*           MORROW DESIGNS DISK JOCKEY DMA (DJDMA)
*

```

```

* THE LOADING IS IDENTICAL TO THAT OF OF THE DJ2DB EXCEPT
* THAT THE LOADER ITSELF IS LOADED AT 80H AND THE 'ORDER'
* COLUMN DOES NOT APPLY. THE DJDMA IS CAPABLE OF LOADING A
* COMPLETE TRACK AT A TIME AND THUS IT MERELY ZAPS THE
* TRACKS IN ALL AT ONCE.
*

```

```

*
*           5 1/4 INCH FLOPPY DISK BOOT LOADER FOR THE
*           MORROW DESIGNS DISK JOCKEY DMA (DJDMA)
*

```

```

* THE COLD BOOT LOADER (TRACK 0, SECTOR 0) IS LOADED INTO
* RAM AT 80H. THIS LOADER WILL START LOADING FROM TRACK 0,
* SECTOR 1 AND STOPS AT TRACK 1, SECTOR 9. THE LOAD
* SEQUENCE IS AS FOLLOWS:
*

```

TRACK	SECTOR	SYSGEN	LOAD ORDER	NAME
0	0	900	80 0	COLD BOOT
0	1	B00	9500 1	CCP
0	2	D00	9700 2	
0	3	F00	9900 3	
0	4	1100	9B00 4	
0	5	1300	9D00 5	BDOS
0	6	1500	9F00 6	
0	7	1700	A100 7	
0	8	1900	A300 8	
0	9	1B00	A500 9	
1	0	1D00	A700 10	
1	1	1F00	A900 11	
1	2	2100	AB00 12	CBIOS
1	3	2300	AD00 13	
1	4	2500	AF00 14	
1	5	2700	B100 15	
1	6	2900	B300 16	
1	7	2B00	B500 17	
1	8	2D00	B700 18	
1	9	2F00	B900 19	

\* THE WARM BOOT STARTS FROM TRACK 0 SECTOR 1 AND CONTINUES  
 \* THROUGH TO TRACK 1 SECTOR 1.

\* SHUGART SA4000 DISK INTERFACE BOOT LOADER FOR THE  
 \* MORROW DESIGNS HARD DISK CONTROLLER REV. 3 (HDC3)

\* THE COLD BOOT LOADER (TRACK 0, SECTOR 1) IS LOADED INTO  
 \* RAM IN THE VERY LAST PART OF THE CBIOS. THIS AREA IS  
 \* USED FOR UNINITIALIZED TABLES AND THUS IS A SAFE PLACE  
 \* FOR THE LOADER. THIS COLD BOOT LOADER WILL START LOADING  
 \* THE CCP FROM TRACK 0, SECTOR 2 AND WILL FINISH UP WITH  
 \* THE LAST PART OF THE CBIOS ON TRACK 0, SECTOR 20.

TRACK	SECTOR	SYSGEN	LOAD ORDER	NAME
0	1	900	FC00 1	COLD BOOT
0	2	B00	9500 2	CCP
0	3	D00	9700 3	
0	4	F00	9900 4	
0	5	1100	9B00 5	
0	6	1300	9D00 6	BDOS
0	7	1500	9F00 7	
0	8	1700	A100 8	
0	9	1900	A300 9	
0	10	1B00	A500 10	
0	11	1D00	A700 11	
0	12	1F00	A900 12	
0	13	2100	AB00 13	CBIOS
0	14	2300	AD00 14	
0	15	2500	AF00 15	
0	16	2700	B100 16	
0	17	2900	B300 17	
0	18	2B00	B500 18	
0	19	2D00	B700 19	
0	20	2F00	B900 20	
0	21	3100		UNUSED

\* THE WARM BOOT LOAD SEQUENCE STARTS AT TRACK 0, SECTOR 2  
 \* AND GOES STRAIGHT THROUGH TO SECTOR 12. THERE IS STILL  
 \* PLENTY OF ROOM LEFT IN THIS LOADER FOR MORE ADVANCED  
 \* THINGS LIKE SECTOR INTERLEAVING ALTHOUGH THIS IS HARDLY  
 \* NECESSARY ON A HARD DISK.

\* SHUGART SA1000 DISK INTERFACE BOOT LOADER FOR THE  
 \* MORROW DESIGNS HARD DISK DMA CONTROLLER (HDDMA)

TRACK	SECTOR	SYSGEN	LOAD ORDER	NAME
0	1	900	100 0	COLD BOOT + CCP
0	2	D00	9300 1	
0	3	1100	9700 2	
0	4	1500	9B00 3	BDOS (@ 9D00)
0	5	1900	9F00 4	
0	6	1D00	A300 5	
0	7	2100	A700 6	CBIOS
0	8	2500	AB00 7	

```

*      0      9      2900      AF00      8      *
*
*      1      10     2D00     B300     9      *
*
* SINCE 1K BYTE SECTORS WERE IMPLEMENTED ON THIS DISK;
* TRACK 0, SECTOR 1 CONTAINS BOTH THE COLD BOOT LOADER AND
* PART OF THE CCP. THE COLD BOOT LOADER RELOCATES THIS
* PEICE OF THE CCP TO IT PROPER RESTING PLACE AS PART OF
* THE BOOT PROCESS.
*

```

\*\*\*\*\*

\*\*\*\*\*

```

* THE FOLLOWING TABLE GIVES A GENERAL IDEA AS TO WHERE THE
* VARIOUS PARTS OF OF THE OPERATING SYSTEM ARE IN MEMORY.
*

```

```

* THE ONLY CHANGES TO THE MAP THAT I SEE IN THE FUTURE IS
* THE INCREASING THE SPACE FOR THE UNINITIALIZED TABLES
* FOLLOWING THE CBIOS. THE AMOUNT OF CODE AND TABLE SPACE
* THAT CAN ACTUALLY BE LOADED FROM THE DISK IS FIXED BY THE
* AMOUNT OF SPACE AVAILABLE ON THE SYSTEM TRACKS.
*

```

```

* OUR MOST RESTRICTIVE (SMALLEST) DRIVE IS THE 5 1/4 INCH
* 'MINNIE FLOPPY'. THIS DRIVE HAS 20 512 BYTE SECTORS FOR
* A TOTAL OF 10K BYTES ON THE SYSTEM TRACKS. THE 8 INCH
* FLOPPY DISK DRIVE IS ALSO VERY CLOSE TO BEING FILLED UP.
*

```

```

* SINCE 512 BYTES ARE RESERVED FOR THE COLD BOOT LOADER WE
* HAVE A TOTAL OF 9.5K BYTES FOR THE OPERATING SYSTEM. OUT
* OF THIS 5.5K BYTES ARE USED BY THE (CCP + BDOS) LEAVING
* US WITH A TOTAL OF 4K BYTES OF LOADED CODE AND DATA SPACE
* TO PLAY WITH. RIGHT NOW WE ARE USING ALL OF THIS SPACE
* SO ANY MAJOR ADDITIONS WILL HAVE TO RESULT IN A LITTLE
* (LOT?) OF CODE SHUFFELING OR IN THE CREATION OF A CBIOS
* THAT SIMPLY WILL NOT FIT ON A SMALL DISK DRIVE.
*

```

	SYSGEN	48K	56K	60K	62K	64K	
	IMAGE	CP/M	CP/M	CP/M	CP/M	CP/M	
	900	~~~~	~~~~	~~~~	~~~~	~~~~	LOADER
	B00	9500	B500	C500	CD00	D500	CCP
	1300	9D00	BD00	CD00	D500	DD00	BDOS
	2100	AB00	CB00	DB00	E300	ED00	CBIOS
	3100	BB00	DB00	EB00	F300	FB00	TABLES
	35FF	BFFF	DFFF	EFFE	F7FF	FFFF	THE END
	~~~~	8D00	AD00	BD00	C500	CD00	DDT

\*\*\*\*\*

```

0000 = MICRON EQU 0 ;SET TO 1 FOR MICRONIX BOOT LOADER

```

```

IF MICRON EQ 0

```

```

0040 = MSIZE EQU 64 ;MEMORY SIZE OF TARGET CP/M

```



```

0016 = BIOSLN EQU 16H ;BIOS LENGTH
0011 = CODLEN EQU 11H ;CODE LENGTH

0800 = CCPLN EQU 800H
0E00 = BDOSLN EQU 0E00H

0000 = SIZE EQU (MSIZE*1024)
D400 = CCP EQU SIZE-(BIOSLN*100H+CCPLN+BDOSLN)
DC00 = BDOS EQU CCP+CCPLN
EA00 = BIOS EQU CCP+CCPLN+BDOSLN

EA00 = CBOOT EQU BIOS ;COLD BOOT ADDRESS FOR CP/M
D400 = LOADDR EQU CCP ;LOAD ADDRESS FOR FLOPPY

ELSE ;MICRONIX BOOT LOADER

CBOOT EQU 0100H ;COLD BOOT ADDRESS FOR THE LOADER
LOADDR EQU 0100H

;IF THE LOAD ADDRESS IS MOVED FORWARD FROM
;0100 THEN THE STARTING EXTENDED ADDRESS FOR
;THE DJDMA BOOT LOADER SHOULD BE ADJUSTED.

ENDIF

000A = RETRIES EQU 10 ;MAXIMUM # OF DISK RETRIES

*****
*
* ONLY ONE OF THE FOLLOWING EQUATES SHOULD BE SET. THE
* OTHERS SOULD BE 0. THESE EQUATES DEFINE THE BOOT LOADER
* THAT IS TO BE USED.
*
*****

0001 = MAXHD EQU 1 ;SET TO BOOT AN HDC3 CONTROLLER
0000 = MAXMW EQU 0 ;SET TO BOOT A HDDMA CONTROLLER
0000 = MAXFD EQU 0 ;SET TO BOOT A DJ2D/B CONTROLLER
0000 = MAXDM EQU 0 ;SET TO BOOT A DJDMA CONTROLLER WITH 8 INCH
0000 = MAXMF EQU 0 ;SET TO BOOT A DJDMA CONTROLLER WITH 5 1/4 INCH

*****
*
* THE FOLLOWING EQUATES ARE FOR THE HARD DISK CONTROLLER 3.
*
*****

IF MAXHD NE 0
0050 = HDORG EQU 50H ;HARD DISK CONTROLLER
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 = TKZ 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 = INDX 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 = SECLN 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

```

```

*****
*
* THE FOLLOWING EQUATES ARE FOR THE HARD DISK DMA.
*
*****

```

```

IF MAXMW NE 0
CYL EQU 153 ;SPECIFICATIONS FOR A SEAGATE TECHNOLOGY 506
HEADS EQU 4 ;NUMBER OF HEADS PER CYLINDER
SPT EQU 9 ;SECTORS PER TRACK
PRECOMP EQU 64 ;CYLINDER TO START WRITE PRECOMENSATION
LOWCURR EQU 128 ;CYLINDER TO START LOW CURRENT
STEPDLY EQU 30 ;STEP DELAY (0-12.7 MILLISECONDS)
HEADDLY EQU 20 ;SETTLE DELAY (0-255 MILLISECONDS)
SECTSIZ EQU 7 ;SECTOR SIZE CODE (MUST BE 3 FOR THIS CBIOS)
; 0 = 128 BYTE SECTORS
; 1 = 256 BYTE SECTORS
; 3 = 512 BYTE SECTORS
; 7 = 1024 BYTE SECTORS (DEFAULT FOR CP/M)
; F = 2048 BYTE SECTORS

;DEFINE CONTROLLER COMMANDS
DMAREAD EQU 0 ;READ SECTOR
DMAWRIT EQU 1 ;WRITE SECTOR
DMARHED EQU 2 ;FIND A SECTOR
DMAWHED EQU 3 ;WRITE HEADERS (FORMAT A TRACK)
DMALCON EQU 4 ;LOAD DISK PARAMETERS
DMASSTA EQU 5 ;SENSE DISK DRIVE STATUS
DMANOOP EQU 6 ;NULL CONTROLLER OPERATION

RESET EQU 54H ;RESET CONTROLLER
ATTN EQU 55H ;SEND A CONTROLLER ATTENTION

```

```

CHAN EQU 50H ;DEFAULT CHANNEL ADDRESS
STEPOUT EQU 10H ;STEP DIRECTION OUT
STEPIN EQU 0 ;STEP DIRECTION IN
BAND1 EQU 40H ;NO PRECOMP, HIGH CURRENT
BAND2 EQU 0C0H ;PRECOMP, HIGH CURRENT
BAND3 EQU 80H ;PRECOMP, LOW CURRENT
TRACK0 EQU 1 ;TRACK ZERO STATUS
WFAULT EQU 2 ;WRITE FAULT FROM DRIVE
DREADY EQU 4 ;DRIVE READY
ENDIF

```

```

*****
*
* THE FOLLOWING EQUATES ARE FOR THE DISK JOCKEY 2D/B.
*
*****

```

```

IF MAXFD NE 0
ORIGIN EQU 0F800H ;ORGIN OF DJ 2D MOD B PROM
DJRAM EQU ORIGIN+400H
;DISK JOCKEY 2D MOD B ROUTINES
TKZERO EQU ORIGIN+9H ;TRACK 0 SEEK
TRKSET EQU ORIGIN+0CH ;SET TRACK
SETSEC EQU ORIGIN+0FH ;SET SECTOR
SETDMA EQU ORIGIN+12H ;SET DMA ADDRESS
DREAD EQU ORIGIN+15H ;READ SECTOR
DMAST EQU ORIGIN+24H ;GET DMA ADDRESS
STATUS EQU ORIGIN+27H ;DISK STATUS
DSKERR EQU ORIGIN+2AH ;FLASH ERROR LIGHT
SETDEN EQU ORIGIN+2DH ;SET DENSITY
ENDIF

```

```

*****
*
* THE FOLLOWING EQUATES ARE FOR THE DISK JOCKEY DMA IF WANTED.
*
*****

```

```

IF (MAXDM NE 0) OR (MAXMF NE 0)
DJKICK EQU 0EFH ;KICK PORT FOR DJDMA CONTROLLER
CHANNL EQU 50H
;8 INCH BOOT LOADER
TRKOFF EQU 22*128 ;NUMBER OF BYTES LOADED FROM TRACK 0
ELSE ;5 1/4 INCH BOOT LOADER
TRKOFF EQU 9*512 ;NUMBER OF BYTES LOADED FROM TRACK 0
ENDIF
SETDMA EQU 23H ;SET DMA ADDRESS
DJHALT EQU 25H ;HALT CONTROLLER
DJBRAN EQU 26H ;BRANCH CONTROLLER COMMAND
REDTRK EQU 29H ;READ TRACK COMMAND
ENDIF

```

```

*****
*
* DEFINE THE ORIGIN ADDRESS FOR THE VARIOUS BOOT LOADERS.
*
*****

```

```

FE00 = BOOT IF MAXHD NE 0 ;HDC3
EQU BIOS+(BIOSLN*100H)-512 ;VERY LAST PART OF CP/M SYSTEM
ENDIF

```

```

BOOT IF MAXMW NE 0 ;HDDMA
EQU 100H
ENDIF

```

```

BOOT IF MAXFD NE 0 ;DJ2D/B
EQU DJRAM+300H ;UPPER 3/4 OF ON BOARD FLOPPY RAM
ENDIF

```

```

BOOT IF (MAXDM NE 0) OR (MAXMF NE 0) ;DJDMA
EQU 80H
ENDIF

```

```

0B00 = OFFSET EQU 900H-BOOT ;DDT OFFSET

```

```

FE00 ORG BOOT

```

```

*****
*
* COLD BOOT LOADER FOR A HARD DISK.
*
*****

```

```

FE00 31FEFF IF (MAXHD NE 0) OR (MAXMW NE 0)
LXI SP,CSTKHD ;SET UP STACK AT END OF THIS SECTOR

```

```

FE03 010213 IF MAXHD NE 0
FE06 CD0CFE LXI B,19*100H+2 ;B = SECTOR COUNT, C = SECTOR #
FE09 C300EA CALL CLODHD
JMP CBOOT ;GO TO CP/M

```

```

ELSE
LXI H,BOOT+200H ;COPY PART OF CCP UP
LXI D,LOADDR
LXI B,200H
MOVLOP: MOV A,M ;GET A BYTE
STAX D ;SAVE IT
INX H ;BUMP POINTERS
INX D
DCX B ;BUMP COUNTER
MOV A,B ;TEST FOR END
ORA C
JNZ MOVLOP
LXI B,10*100H+2 ;B = SECTOR COUNT, C = SECTOR #
CALL CLODHD
JMP CBOOT ;GO TO CP/M

```

```

                                ENDIF
FE0C C5      CLODHD  PUSH  B      ;SAVE SECTOR AND COUNT
FE0D 79      MOV    A,C      ;LOAD SECTOR
FE0E 3290FE  STA    HDSEC
FE11 2100D2  LXI    H,LOADDR-200H ;GET DMA ADDRESS (SELF MODIFYING)
FE12 =      CDMAHD  EQU    $-2   ;STORAGE FOR PREVIOUS DMA ADDRESS
                                IF    MAXHD NE 0
FE14 110002  LXI    D,200H      ;OFFSET TO NEW DMA ADDRESS
                                ELSE
                                LXI    D,400H
                                ENDIF
FE17 19      DAD    D      ;ADD IN OFFSET, HL = NEW DMA ADDRESS
FE18 2212FE  SHLD   CDMAHD     ;SAVE NEW DMA ADDRESS
FE1B CD25FE  CALL   CRDHD     ;ATTEMPT A READ
FE1E C1      POP    B      ;RECOVER SECTOR NUMBER AND COUNT
                                ;    B = COUNT, C = NUMBER
FE1F 05      DCR    B      ;UPDATE SECTOR COUNT
FE20 C8      RZ
FE21 0C      INR    C      ;BUMP SECTOR NUMBER
FE22 C30CFE  JMP    CLODHD    ;CONTINUE READING

```

```

*****
*
* CRDHD DOES THE ACTUAL READ FROM THE CONTROLLER, THE DMA
* ADDRESS AND SECTOR # HAVE ALREADY BEEN SET UP.
*
*****

```

```

FE25 01010A CRDHD  LXI    B,RETRIES*100H+1 ;LOW LEVEL HDC3 DRIVERS
FE28 C5      CRHD   PUSH   B      ;MAXIMUM # OF ATTEMPTS
FE29 CD35FE  CALL   HDREAD   ;SAVE ERROR COUNT
FE2C C1      POP    B      ;ATTEMPT THE READ
FE2D D0      RNC
FE2E 05      DCR    B      ;RESTORE THE ERROR COUNT
FE2F C228FE  JNZ    CRHD     ;RETURN IF NO ERROR
FE32 C332FE  JMP    $        ;UPDATE ERROR COUNT
                                ;TRY AGAIN IF NOT TOO MANY ERRORS
                                ;DYNAMIC ERROR HALT
FE35 CD7CFE  HDREAD  CALL   HDPREP   ;PREPARE THE SECTOR HEADER IMAGE
FE38 D8      RC
FE39 3E01    MVI    A,RSECT   ;ERROR EXIT
FE3B D351    OUT   HDCMND   ;READ SECTOR COMMAND
FE3D CD62FE  CALL   PROCESS   ;PROCESS THE READ
FE40 D8      RC
FE41 AF      XRA    A      ;ERROR EXIT
FE42 D351    OUT   HDCMND   ;POINTER TO DATA BUFFER
FE44 0680    MVI    B,SECLN/4 ;NUMBER OF BYTES TO READ
FE46 2A12FE  LHLD   CDMAHD   ;GET DESTINATION OF DATA
FE49 DB53    IN    HDDATA  ;TWO DUMMY DATA BYTES
FE4B DB53    IN    HDDATA
FE4D DB53    RTLOOP  IN    HDDATA  ;MOVE FOUR BYTES
FE4F 77      MOV    M,A      ;BYTE ONE
FE50 23      INX   H
FE51 DB53    IN    HDDATA  ;BYTE TWO
FE53 77      MOV    M,A

```

```

FE54 23          INX      H
FE55 DB53        IN       HDDATA      ;BYTE THREE
FE57 77          MOV      M,A
FE58 23          INX      H
FE59 DB53        IN       HDDATA      ;BYTE FOUR
FE5B 77          MOV      M,A
FE5C 23          INX      H
FE5D 05          DCR      B             ;UPDATE BYTE COUNT
FE5E C24DFE      JNZ      RTLOOP
FE61 C9          RET

FE62 DB50        PROCESS IN       HDSTAT      ;WAIT FOR COMMAND TO FINISH
FE64 47          MOV      B,A
FE65 E602        ANI      OPDONE
FE67 CA62FE      JZ       PROCESS
FE6A 3E07        MVI      A,DSKCLK      ;TURN ON DISK CLOCK
FE6C D350        OUT      HDCNTL
FE6E DB50        IN       HDSTAT
FE70 E608        ANI      TMOUT          ;TIMED OUT ?
FE72 37          STC
FE73 C0          RNZ
FE74 DB51        IN       HDRESLT
FE76 E602        ANI      RETRY          ;ANY RETRIES ?
FE78 37          STC
FE79 C0          RNZ
FE7A AF          XRA      A             ;NO ERROR EXIT
FE7B C9          RET

FE7C DB50        HDPREP  IN       HDSTAT      ;IS DRIVE READY ?
FE7E E620        ANI      DRVRDY
FE80 37          STC
FE81 C0          RNZ
FE82 3E08        MVI      A,ISBUFF      ;INITIALIZE POINTER TO HEADER BUFFER
FE84 D351        OUT      HDCMND
FE86 3EFC        MVI      A,NULL
FE88 D352        OUT      HDFUNC      ;SELECT DRIVE A
FE8A AF          XRA      A
FE8B D353        OUT      HDDATA      ;FORM HEAD BYTE
FE8D D353        OUT      HDDATA      ;FORM TRACK BYTE
FE8F 3E00        MVI      A,0             ;FORM SECTOR BYTE
FE90 =          HDSEC  EQU      $-1
FE91 D353        OUT      HDDATA
FE93 3E80        MVI      A,80H          ;FORM KEY
FE95 D353        OUT      HDDATA
FE97 3E07        MVI      A,DSKCLK      ;TURN ON DISK CLOCK
FE99 D350        OUT      HDCNTL
FE9B 3E0F        MVI      A,WENABL      ;WRITE ENABLE ON
FE9D D350        OUT      HDCNTL
FE9F C9          RET

FFFE            ORG      BOOT+200H-2      ;LAST WORD ON SECTOR IS LOAD ADDRESS

FFFE =          CSTKHD EQU      $
FFFE 00FE       DW      BOOT
                ENDIF

```

```

CRDHD  IF      MAXMW NE 0      ;LOW LEVEL HDDMA ROUTINES
CALL   HDSETUP ;SET UP PARAMETERS
LXI    B,RETRIES*100H+1 ;MAXIMUM # OF ATTEMPTS
CRHD   PUSH   B              ;SAVE ERROR COUNT
CALL   HDISSUE ;ATTEMPT THE READ
POP    B              ;RESTORE THE ERROR COUNT
RNC    ;RETURN IF NO ERROR
DCR    B              ;UPDATE ERROR COUNT
JNZ    CRHD          ;TRY AGAIN IF NOT TOO MANY ERRORS
ERROR  JMP     $           ;DYNAMIC ERROR HALT

HDSETUP SHLD   DMADMA      ;SET UP DMA ADDRESS
CALL    HDRESET ;RESET CONTROLLER
LDA     HDSEC   ;SET LOGICAL SECTOR NUMBER
DCR    A        ;RANGE IS ACTAULLY 0-16
CALL    DIVSPT  ;FIGURE OUT HEAD NUMBER -> (C)
ADI     SPT     ;MAKE REAL SECTOR NUMBER
STA     DMARG3
MOV     A,C
STA     DMARG2      ;SAVE HEAD NUMBER
CMA    ;NEGATIVE LOGIC FOR THE CONTROLLER
ANI     7          ;3 BITS OF HEAD SELECT
RLC    ;SHOVE OVER TO BITS 2 - 4
RLC
STA     DMASEL1    ;SAVE IN COMMAND CHANNEL HEAD SELECT
RET

DIVSPT  MVI     C,0      ;CLEAR HEAD COUNTER
DIVSPTX SUI     SPT     ;SUBTRACT A TRACKS WORTH OF SECTORS
RC      ;RETURN IF ALL DONE
INR    C          ;BUMP TO NEXT HEAD
JMP    DIVSPTX

HDRESET MVI     A,(RET)  ;ONE TIME CODE
STA     HDRESET
OUT    RESET      ;SEND RESET PULSE TO CONTROLLER
LXI    H,DMACHAN  ;ADDRESS OF COMMAND CHANNEL
SHLD   CHAN       ;DEFAULT CHANNEL ADDRESS
XRA    A
STA     CHAN+2    ;CLEAR EXTENDED ADDRESS BYTE
XTHL   ;WAIT FOR RESET (AROUND 10 USEC'S)
XTHL
CALL    HDISSUE   ;DO LOAD CONSTANTS
JC      ERROR     ;CONTROLLER NOT PRESENT
MVI    A,DMASSTA ;SENSE STATUS COMMAND
STA     DMAOP

RDYCHEK CALL    HDISSUE
ANI     DREADY    ;CHECK FOR DRIVE READY
JNZ    RDYCHEK    ;LOOP IF NOT READY
LXI    H,0FFFFH
SHLD   DMASTEP    ;DO RECALIBRATE
CALL    HDISSUE
LXI    H,0
SHLD   DMASTEP    ;CLEAR STEP COUNTER
SHLD   DMARG0     ;CLEAR CYLINDER #
SHLD   DMARG3     ;CLEAR SECTOR # + READ DISK COMMAND

```

```

RET
HDISSUE LXI    H,DMASTAT    ;STATUS BYTE
        MVI    M,0
        OUT   ATTN        ;START CONTROLLER
        LXI    D,0        ;TIME OUT COUNTER
        MOV    B,E        ;CONTROLLER BUSY STATUS
HDILOOP MOV    A,M        ;GET STATUS
        ORA    A          ;SET UP CPU FLAGS
        RM
        STC
        RNZ
XTHL   XTHL        ;RETURN ERROR STATUS
XTHL   XTHL        ;WASTE SOME TIME
XTHL   XTHL
XTHL   XTHL
DCX    D          ;BUMP TIMEOUT COUNTER
MOV    A,D
ORA    E
JNZ   HDILOOP    ;LOOP IF STILL BUSY
STC
RET

HDSEC  DB    0        ;CURRENTLY SELECTED SECTOR

DMACHAN EQU    $        ;COMMAND CHANNEL AREA
DMASEL0 DB    10H      ;DRIVE SELECT (STEP OUT, DRIVE 0)
DMASTEP DW    0        ;RELATIVE STEP COUNTER
DMASEL1 DB    0        ;HEAD SELECT
DMADMA  DW    0        ;DMA ADDRESS
        DB    0        ;EXTENDED ADDRESS
DMARG0  DB    0        ;FIRST ARGUMENT
DMARG1  DB    STEPDLY  ;SECOND ARGUMENT (STEPPING TIME)
DMARG2  DB    HEADDLY  ;THIRD ARGUMENT (SETTLE TIME)
DMARG3  DB    SECTSIZ  ;FOURTH ARGUMENT (SECTOR SIZE)
DMAOP   DB    DMALCON  ;OPERATION CODE
DMASTAT DB    0        ;CONTROLLER STATUS BYTE
DMALNK  DW    DMACHAN  ;LINK ADDRESS TO NEXT COMMAND CHANNEL
        DB    0        ;EXTENDED ADDRESS

CSTKHD  ORG    BOOT+200H
        EQU    $        ;STACK AREA AT END OF SECTOR
        ENDIF
        ENDIF

*****
*
* COLD BOOT LOADER FOR THE DISK JOCKEY 2D REVISION B CONTROLLER *
*
*****

T0BOOT IF    MAXFD NE 0
NEWSEC MVI    A,5-2    ;FIRST SECTOR - 2
        EQU    $-1
        INR   A        ;UPDATE SECTOR #
        INR   A

```



```

TRKSIZ  CPI      27          ;SIZE OF TRACK IN SECTORS + 1
        EQU      $-1
        JC       NOWRAP     ;SKIP IF NOT AT END OF TRACK
        JNZ      T1BOOT     ;DONE WITH THIS TRACK
EXIT    EQU      $-2
        SUI      27-6       ;BACK UP TO SECTOR 6
BACKUP  EQU      $-1
        LXI      H,LOADDR-80H ;MEMORY ADDRESS OF SECTOR - 100H
NXTDMA  EQU      $-2
        SHLD     NEWDMA
NOWRAP  STA      NEWSEC     ;SAVE THE UPDATED SECTOR #
        MOV      C,A
        CALL     SETSEC     ;SET UP THE SECTOR
        LXI      H,LOADDR-100H ;MEMORY ADDRESS OF SECTOR - 100H
NEWDMA  EQU      $-2
        LXI      D,100H     ;UPDATE DMA ADDRESS
SECSIZ  EQU      $-2
        DAD      D
NOWRP   SHLD     NEWDMA     ;SAVE THE UPDATED DMA ADDRESS
        MOV      B,H
        MOV      C,L
        CALL     SETDMA     ;SET UP THE NEW DMA ADDRESS
        LXI      B,RETRIES*100H+0;MAXIMUM # OF ERRORS, TRACK #
NXTRTY  EQU      $-2
FREAD  PUSH     B
        CALL     TRKSET     ;SET UP THE PROPER TRACK
        CALL     DREAD     ;READ THE SECTOR
        POP      B
        JNC      T0BOOT     ;CONTINUE IF NO ERROR
        DCR      B
        JNZ      FREAD     ;KEEP TRYING IF ERROR
        JMP      DSKERR     ;TOO MANY ERRORS, FLASH THE LIGHT

T1BOOT  LXI      H,CBOOT     ;WE JUMP TO CBOOT NEXT TIME
        SHLD     EXIT
        MVI      C,1        ;SELECT DOUBLE DENSITY
        CALL     SETDEN
        XRA      A          ;FIRST SECTOR - 2
        STA      NEWSEC
        MVI      A,8        ;SIZE OF (LOGICAL) TRACK + 1
        STA      TRKSIZ
        DCR      A          ;NUMBER OF SECTORS TO BACK UP
        STA      BACKUP
        LXI      H,LOADDR+0700H ;DMA START ADDRESS FOR FIRST REVOLUTION - 2048
        SHLD     NEWDMA
        LXI      H,LOADDR+0300H ;DMA START ADDRESS FOR SECOND REVOLUTION - 2048
        SHLD     NXTDMA
        LXI      H,2048     ;DIFFERENCE BETWEEN DMA ADDRESSES
        SHLD     SECSIZ
        LXI      H,RETRIES*100H+1;MAXIMUM # OF ERRORS, TRACK #
        SHLD     NXTRTY
        JMP      T0BOOT     ;GO LOAD IN TRACK 1
ENDIF

```

```

*****
*

```

```

* COLD BOOT LOADER FOR THE DISK JOCKEY DMA CONTROLLER *
*
*****

```

```

IF      (MAXDM NE 0) OR (MAXMF NE 0)      ;SET UP DJDMA LOADER

MVI     A,DJBRAN      ;LOAD BRANCH CHANNEL COMMAND
STA     CHANNL
LXI     H,COMMND     ;LOAD NEW COMMAND CHANNEL ADDRESS
SHLD   CHANNL+1
XRA     A
STA     CHANNL+3

DJSTRT: OUT     DJKICK      ;START CONTROLLER
DJWAIT: LDA     DJDONE     ;GET FINAL STATUS
ORA     A            ;0 = STILL BUSY
JZ      DJWAIT      ;LOOP IF BUSY
LXI     H,SECTB0     ;CHECK FOR BAD LOAD
LXI     B,40FFH     ;B = OK, C = LOADED
LXI     D,ENDTBL-SECTB0 ;ERROR COUNT + # OF SECTORS
DJLOOP: MOV    A,M      ;LOAD SECTOR CODE
CMP     C            ;CHECK FOR 0FFH (ALREADY LOADED)
JZ      DJCONT      ;SKIP IF LOAD WAS 'OK'
MOV     M,C          ;LOAD 'LOADED' FLAG
CMP     B            ;CHECK FOR 'OK' STATUS
JZ      DJCONT      ;SKIP IF LOAD OK
INR     M            ;MAKE FLAG = 0
INR     D            ;BUMP ERROR COUNTER
DJCONT: DCR     E       ;BUMP SECTOR COUNT
INX     H            ;BUMP TABLE POINTER
JNZ     DJLOOP
MOV     A,D          ;CHECK OUT ERROR COUNTER
ORA     A
JZ      CBOOT       ;START CP/M IF NO ERRORS
DCR     M            ;DROP RETRY COUNTER
JNZ     DJSTRT      ;RETRY LOAD OPERATION
JMP     $            ;DYNAMIC ERROR HALT

COMMND: DB     SETDMA     ;SET DMA ADDRESS
        DW     LOADDR-512 ;START AT CCP
        IF     MICRON EQ 0
        DB     0          ;EXTENDED ADDRESS FOR CP/M
        ELSE
        DB     0FFH      ;WRAP AROUND FROM FFFF00 TO 000100
        ENDIF

        DB     REDTRK    ;READ TRACK
        DB     0         ;TRACK 0
        DB     0         ;SIDE 0
        DB     0         ;DRIVE 0
        DW     SECTB0    ;SECTOR TABLE 0
        DB     0         ;EXTENDED ADDRESS
        DB     0         ;RETURNED STATUS

        DB     SETDMA     ;SET DMA ADDRESS
        DW     LOADDR+TRKOFF ;LOAD ADDRESS FOR TRACK 1

```

```

DB      0      ;EXTENDED ADDRESS

DB      REDTRK  ;READ TRACK
DB      1      ;TRACK 1
DB      0      ;SIDE 0
DB      0      ;DRIVE 0
DW      SECTB1 ;SECTOR TABLE 1
DB      0      ;EXTENDED ADDRESS
DB      0      ;RETURNED STATUS

DJDONE: DB      DJHALT ;HALT CONTROLLER
        DB      0      ;RETURNED STATUS

        ORG     BOOT+5DH ;BOOT + 5CH CONTAINS 'CONFIGURATION BYTE'

        IF      MAXDM NE 0 ;BOOTING FROM 8 INCH DRIVES
SECTB0: DW      0FFFFH, 0FFFFH ;DO NOT LOAD BOOT LOADER
        DW      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ;22 SECTORS TO BE LOADED

SECTB1: DW      0, 0, 0, 0FF00H ;FIRST SEVEN SECTORS

        DB      RETRIES ;RETRY COUNTER
        ELSE    ;BOOTING FROM 5 1/5 INCH DRIVES
SECTB0: DW      0FFH, 0, 0, 0, 0 ;DO NOT LOAD BOOT LOADER

SECTB1: DW      0, 0, 0, 0, 0 ;LOAD TEN SECTORS

        DB      RETRIES ;RETRY COUNTER
        ENDIF

ENDTBL EQU      $-1 ;END OF TABLE MARKER

        ENDIF

0000    END

```

0E00	BDOSLN	DC00	BDOS	0016	BIOSLN	EA00	BIOS	FE00	BOOT
EA00	CBOOT	D400	CCP	0800	CCPLN	FE12	CDMAHD	FE0C	CLODHD
0011	CODLEN	0004	COMPLT	FE25	CRDHD	FE28	CRHD	FFFE	CSTKHD
0020	DRVRDY	0007	DSKCLK	0051	HDCMND	0050	HDCNTL	0053	HDDATA
0052	HDFUNC	0050	HDORG	FE7C	HDPREP	FE35	HDREAD	0051	HDRESLT
0004	HDRLEN	FE90	HDSEC	0050	HDSTAT	0000	IDBUFF	0040	INDX
0008	ISBUFF	D400	LOADDR	0000	MAXDM	0000	MAXFD	0001	MAXHD
0000	MAXMF	0000	MAXMW	00F7	MDIR	0000	MICRON	0040	MSIZE
00FB	NSTEP	00FC	NULL	0B00	OFFSET	0002	OPDONE	FE62	PROCESS
0004	PSTEP	000A	RETRIES	0002	RETRY	0001	RSECT	FE4D	RTLOOP
0005	SCENBL	0200	SECLN	0000	SIZE	0001	TKZ	0008	TMOUT
000F	WENABL	0010	WFAULT	000B	WRESET	0005	WSECT		