

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

001D =	REVNUM	EQU	29	;CBIOS REVISION NUMBER 2.9
0016 =	CPMREV	EQU	22	;CP/M REVISION NUMBER 2.2
0001 =	CONGRP	EQU	1	;CONSOLE PORT (1 = P1, 2 = P2, 3 = P3)
0003 =	LSTGRP	EQU	3	;PRINTER PORT (1 = P1, 2 = P2, 3 = P3)

CBIOSH (LAST) PRN
8/22/82

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

0001 =	FIRST	EQU	1	;0 = FLOPPIES ARE A,B,C,D DRIVES AND ; HARD DISK ARE E,F,G,H ;1 = HARD DISKS ARE A,B,C,D DRIVES AND ; FLOPPIES ARE E,F,G,H
0001 =	MAXHD	EQU	1	;SET TO NUMBER OF HARD DISKS
0001 =	MAXFLOP	EQU	1	;SET TO NUMBER OF FLOPPIES

0001 =	M20	EQU	1	
--------	-----	-----	---	--

*
* THE NEXT EQUATE WILL SET THE NUMBER OF LOGICAL DISKS ON A
* PHYSICAL HARD DISK DRIVE. THE USER MUST SET STDLOG TO A
* VAULE GREATER THAN OR EQUALE TO 2 FOR AN M10 OR 3 FOR AN
* M20 OR M26. THE REASON FOR THIS IS THAT CP/M CAN NOT
* ADDRESS MORE THAN 8 MEGABYTES PER LOGICAL DISK AND
* SPLITTING A DISK TO LESS THEN 2 OR 3 PARTS WOULD MAKE
* PARTITIONS THAT ARE GREATER THAN 8 MEGABYTES IN LENGTH.
*

? 0000 =	STDLOG	EQU	0	;SET TO 0 TO USE STANDARD LOGICAL DISKS
0003 =	LOGDSK	EQU	3	;DEFAULT LOGICAL DISKS PER DRIVE
? 0001 =	FUJITSU	EQU	1	
? 0014 =	MREV	EQU	20	;HARD DISK TYPE
0015 =	HDSPT	EQU	21	;SECTORS PER TRACK

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

F800 =	ORIGIN	EQU	0F800H	
FC00 =	DJRAM	EQU	ORIGIN+400H	;DISK JOCKEY 2D RAM ADDRESS
F800 =	DJBOOT	EQU	ORIGIN	;DISK JOCKEY 2D INITIALIZATION
F803 =	DJCIN	EQU	ORIGIN+3H	;DISK JOCKEY 2D CHARACTER INPUT ROUTINE
F806 =	DJCOU	EQU	ORIGIN+6H	;DISK JOCKEY 2D CHARACTER OUTPUT ROUTINE
F809 =	DJHOME	EQU	ORIGIN+9H	;DISK JOCKEY 2D TRACK ZERO SEEK
F80C =	DJTRK	EQU	ORIGIN+0CH	;DISK JOCKEY 2D TRACK SEEK ROUTINE

VR-1412

```

F80F = DJSEC EQU ORIGIN+0FH ;DISK JOCKEY 2D SET SECTOR ROUTINE
F812 = DJDMA EQU ORIGIN+012H ;DISK JOCKEY 2D SET DMA ADDRESS
F815 = DJREAD EQU ORIGIN+15H ;DISK JOCKEY 2D READ ROUTINE
F818 = DJWRITE EQU ORIGIN+18H ;DISK JOCKEY 2D WRITE ROUTINE
F81B = DJSEL EQU ORIGIN+1BH ;DISK JOCKEY 2D SELECT DRIVE ROUTINE
F821 = DJTSTAT EQU ORIGIN+21H ;DISK JOCKEY 2D TERMINAL STATUS ROUTINE
F827 = DJSTAT EQU ORIGIN+27H ;DISK JOCKEY 2D STATUS ROUTINE
F82A = DJERR EQU ORIGIN+2AH ;DISK JOCKEY 2D ERROR, FLASH LED
F82D = DJDEN EQU ORIGIN+2DH ;DISK JOCKEY 2D SET DENSITY ROUTINE
F830 = DJSIDE EQU ORIGIN+30H ;DISK JOCKEY 2D SET SIDE ROUTINE
0008 = DBLSID EQU 8 ;SIDE BIT FROM CONTROLLER
FBF8 = IO EQU ORIGIN+3F8H ;START OF I/O REGISTERS
FBF9 = DREG EQU IO+1
FBFC = CMDREG EQU IO+4
00D0 = CLRCMD EQU 0D0H
    
```

```

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

```

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

0048 = MBASE EQU 48H ;BASE ADDRESS OF MULTI I/O OR DECISION I
004F = GRPSEL EQU MBASE+7 ;GROUP SELECT PORT
    
```

VR-1412

MSI-READ

```

0048 = DLL EQU MBASE ;DIVISOR (LSB)
0049 = DLM EQU MBASE+1 ;DIVISOR (MSB)
0049 = IER EQU MBASE+1 ;INTERUPT ENABLE REGISTER
004A = CLK EQU MBASE+2 ;WB14 PRINTER SELECT PORT
004B = LCR EQU MBASE+3 ;LINE CONTROL REGISTER
004D = LSR EQU MBASE+5 ;LINE STATUS REGISTER
004E = MSR EQU MBASE+6
0048 = RBR EQU MBASE ;READ DATA BUFFER
0048 = THR EQU MBASE ;TRANSMITTER DATA BUFFER
0080 = DLAB EQU 80H ;DIVISOR LATCH ACCESS BIT
0020 = THRE EQU 20H ;STATUS LINE THRE BIT
0010 = CTS EQU 10H ;CLEAR TO SEND
0020 = DSR EQU 20H ;DATA SET READY
0001 = DR EQU 1 ;LINE STATUS DR BIT
0001 = WLS0 EQU 1 ;WORD LENGTH SELECT BIT 0
0002 = WLS1 EQU 2 ;WORD LENGTH SELECT BIT 1 FOR 8 BIT WORD
0004 = STB EQU 4 ;STOP BIT COUNT - 2 STOP BITS

```

; DEFINE MULTI I/O PORTS ADDRESSES FOR GROUP ZERO

```

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

```

```

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

```

; DEFINE GROUP SELECT BITS

```

0001 = S0 EQU 01H ;GROUP NUMBER (0-3)
0002 = S1 EQU 02H
0003 = SMASK EQU 03H
0004 = BANK EQU 04H
0008 = ENINT EQU 08H
0010 = RESTOR EQU 10H ;PRINTER RESTORE ON MULTI I/O
0020 = DENABLE EQU 20H ;DRIVER ENABLE ON MULTI I/O

```

```

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

```

```

003E = MSIZE EQU 62 ;MEMORY SIZE OF TARGET CP/M
A800 = BIAS EQU (MSIZE-20)*1024 ;MEMORY OFFSET FROM 20K SYSTEM
CD00 = CCP EQU 2500H+BIAS ;CONSOLE COMMAND PROCESSOR
D500 = BDOS EQU CCP+800H ;BDOS ADDRESS
E300 = BIOS EQU CCP+1600H ;CBIOS ADDRESS
3E00 = OFFSETC EQU 2100H-BIOS ;OFFSET FOR SYSGEN
0004 = CDISK EQU 4 ;ADDRESS OF LAST LOGGED DISK
0080 = BUFF EQU 80H ;DEFAULT BUFFER ADDRESS
0100 = TPA EQU 100H ;TRANSIENT MEMORY
0000 = INTIOBY EQU 0 ;INITIAL IOBYTE

```

VR-1412

VS1-READ

```

0003 = IOBYTE EQU 3 ;IOBYTE LOCATION
0000 = WBOT EQU 0 ;WARM BOOT JUMP ADDRESS
0005 = ENTRY EQU 5 ;BDOS ENTRY JUMP ADDRESS
    
```

```

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

```

000A = ACR EQU 'J'-64 ;CARRIAGE RETURN
000D = ALF EQU 'M'-64 ;LINE FEED
000A = RETRIES EQU 10 ;MAX RETRIES ON DISK I/O BEFORE ERROR
001A = CLEAR EQU 'Z'-64 ;CLEAR SCREEN ON AN ADM 3
0000 = ANUL EQU 0 ;NULL
    
```

```

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

```

E300          ORG      BIOS          ;CBIOS STARTING ADDRESS

E300 C3A1EA    JMP      CBOOT         ;COLD BOOT ENTRY POINT
E303 C3FBE3    WBOOTE  JMP      WBOOT         ;WARM BOOT ENTRY POINT
E306 C363E3    CONST   JMP      CSTTY        ;CONSOLE STATUS ROUTINE
E309 C33BE3    CIN     JMP      CIFLSH       ;CONSOLE INPUT
E30C C351E3    COUT    JMP      COTTY        ;CONSOLE OUTPUT
E30F C372E3    JMP      LIST          ;LIST DEVICE OUTPUT
E312 C39FE3    JMP      PUNCH         ;PUNCH DEVICE OUTPUT
E315 C3A2E3    JMP      READER        ;READER DEVICE INPUT
E318 C346E4    JMP      HOME          ;HOME DRIVE
E31B C388E4    JMP      SETDRV        ;SELECT DISK
E31E C348E4    JMP      SETTRK       ;SET TRACK
E321 C33AE4    JMP      SETSEC       ;SET SECTOR
E324 C340E4    JMP      SETDMA       ;SET DMA ADDRESS
E327 C3C7E5    JMP      READ          ;READ THE DISK
E32A C3C0E5    JMP      WRITE        ;WRITE THE DISK
E32D C38BE3    JMP      LISTST       ;LIST DEVICE STATUS
E330 C34DE4    JMP      SECTTRAN     ;SECTOR TRANSLATION
E333 C31BF8    DJDRV   JMP      DJSEL          ;HOOKUP FOR SINGLE.COM PROGRAM

E336 0600     DEFCON  DW      6          ;CONSOLE BAUD RATE
E338 0C00     DEFLST  DW     12         ;PRINTER BAUD RATE

E33A 00       GROUP   DB      0          ;GROUP BYTE

E33B CD3AE6    CIFLSH  CALL     FLUSH        ;FLUSH DISK BUFFERS ON INPUT
E33E 3A3AE3    CITYY   LDA      GROUP        ;GET GROUP BYTE
E341 F601     ORI     CONGRP       ;SELECT CONSOLE
E343 D34F     OUT    GRPSEL
    
```

VR-1412

VISI-READ

```

E345 DB4D      CONIN1  IN      LSR      ;READ STATUS REGISTER
E347 E601      ANI      DR        ;WAIT TILL CHARACTER READY
E349 CA45E3    JZ        CONIN1
E34C DB48      IN      RBR      ;READ CHARACTER
E34E E67F      ANI      7FH     ;STRIP PARITY
E350 C9        RET

E351 3A3AE3    COTTY   LDA      GROUP    ;GET GROUP BYTE
E354 F601      ORI      CONGRP   ;SELECT CONSOLE
E356 D34F      OUT      GRPSEL

E358 DB4D      CONOUT1 IN      LSR      ;READ STATUS
E35A E620      ANI      THRE    ;WAIT TILL TRANSMITTER BUFFER EMPTY
E35C CA58E3    JZ        CONOUT1
E35F 79        MOV      A,C      ;CHARACTER IS IN (C)
E360 D348      OUT      THR     ;OUTPUT TO TRANSMITTER BUFFER
E362 C9        RET

E363 3A3AE3    CSTTY   LDA      GROUP    ;GET GROUP BYTE
E366 F601      ORI      CONGRP   ;SELECT CONSOLE
E368 D34F      OUT      GRPSEL

E36A DB4D      IN      LSR      ;READ STATUS REGISTER
E36C E601      ANI      DR        ;NO CHARACTER READY
E36E C8        RZ        ;CHARACTER READY
E36F 3EFF      MVI      A,0FFH
E371 C9        RET

E372 3A3AE3    LIST    LDA      GROUP    ;GET GROUP BYTE
E375 F603      ORI      LSTGRP   ;SELECT LIST DEVICE
E377 D34F      OUT      GRPSEL

E379 DB4D      LL       IN      LSR      ;WAIT TILL TRANSMITTER BUFFER EMPTY
E37B E620      ANI      THRE
E37D CA79E3    JZ        LL

E380 DB4E      IN      MSR      ;WAIT TILL CLEAR TO SEND
E382 E610      ANI      CTS
E384 CA79E3    JZ        LL

E387 79        MOV      A,C
E388 D348      OUT      THR
E38A C9        RET

E38B 3A3AE3    LISTST LDA      GROUP    ;GET GROUP BYTE
E38E F603      ORI      LSTGRP   ;SELECT LIST DEVICE
E390 D34F      OUT      GRPSEL

E392 DB4D      IN      LSR      ;CHECK IF TRANSMITTER BUFFER EMPTY
E394 E648      ANI      THR
E396 C8        RZ        ;RETURN NOT READY

E397 DB4E      IN      MSR
E399 E610      ANI      CTS
E39B C8        RZ        ;RETURN NOT READY IF CTS IS FALSE

```

VR-1412

VSI-READ

E39C 3EFF MVI A,0FFH
 E39E C9 RET ;PRINTER READY

E39F C30CE3 PUNCH JMP COUT

E3A2 C309E3 READER JMP CIN

 *
 * 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).
 *

E3A5 218000 GOCPM LXI H,BUFF ;SET UP INITIAL DMA ADDRESS

E3A8 CD40E4 CALL SETDMA
 E3AB 3EC3 MVI A,(JMP) ;INITIALIZE JUMP TO WARM BOOT
 E3AD 320000 STA WBOT

E3B0 320500 STA ENTRY ;INITIALIZE JUMP TO BDOS
 E3B3 2103E3 LXI H,WBOOTE ;ADDRESS IN WARM BOOT JUMP
 E3B6 220100 SHLD WBOT+1

E3B9 2106D5 LXI H,BDOS+6 ;ADDRESS IN BDOS JUMP
 E3BC 220600 SHLD ENTRY+1

E3BF AF XRA A ;A ← 0

E3C0 3217EE STA BUFSEC ;DISK JOCKEY BUFFER EMPTY

E3C3 323BE6 STA BUFWRN ;SET BUFFER NOT DIRTY FLAG

E3C6 3A0400 LDA CDISK ;JUMP TO CP/M WITH CURRENTLY SELECTED DISK IN C

E3C9 4F MOV C,A

E3CA 3AF7E3 LDA CWFLG

E3CD B7 ORA A

E3CE 11F9E3 LXI D,COLDBEG ;BEGINNING OF INITIAL COMMAND

E3D1 3E01 MVI A,COLDEND-COLDBEG+1 ;LENGTH OF COMMAND

E3D3 CADBE3 JZ CLDCMND

E3D6 11FAE3 LXI D,WARBEG

E3D9 3E01 MVI A,WARMEND-WARBEG+1

E3DB 2108CD CLDCMND LXI H,CCP+8 ;COMMAND BUFFER

E3DE 3207CD STA CCP+7

E3E1 47 MOV B,A

E3E2 CD03E7 CALL MOVLOP

E3E5 3AF7E3 LDA CWFLG

E3E8 B7 ORA A

E3E9 3AF8E3 LDA AUTOFLG

E3EC CAF0E3 JZ CLDBOT

E3EF 1F RAR

E3F0 1F CLDBOT RAR

E3F1 DA00CD JC CCP

E3F4 C303CD JMP CCP+3 ;ENTER CP/M

E3F7 00 CWFLG DB 0 ;COLD/WARM BOOT FLAG

 *
 * THE FOLLOWING BYTE DETERMINES IF AN INITIAL COMMAND IS TO BE *
 *

VR-1412

* 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. *

E3F8 00 AUTOFLG DB 0 ;AUTO COMMAND FEATURE

* IF THERE IS A COMMAND INSERTED HERE, IT WILL BE GIVEN IF THE *
 * AUTO FEATURE IS ENABLED. *

* FOR EXAMPLE: *

* COLDBEG DB 'MBASIC MYPROG' *
 * COLDEND DB 0 *

* WILL EXECUTE MICROSOFT BASIC, AND MBASIC WILL EXECUTE THE *
 * "MYPROG" BASIC PROGRAM. *

COLDBEG DB '' ;COLD BOOT COMMAND GOES HERE

E3F9 00 COLDEND DB 0

WARMBEG DB '' ;WARM BOOT COMMAND GOES HERE

E3FA 00 WARMEND DB 0

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

E3FB 310001 WBOOT LXI SP,TPA ;SET UP STACK POINTER

E3FE 3E01 MVI A,1

E400 32F7E3 STA CWFLG ;SET COLD/WARM BOOT FLAG

IF (MAXHD NE 0) AND FIRST ;SUPPLY WARM BOOT FROM HARD DISK ?

E403 AF XRA A

E404 4F MOV C,A

E405 2100CB LXI H,CCP-200H ;INITIAL DMA ADDRESS

E408 E5 PUSH H

E409 3246E8 STA HEAD

E40C 3E01 MVI A,1

E40E F5 PUSH PSW ;SAVE FIRST SECTOR - 1

E40F CD18E7 CALL HDDRV ;SELECT DRIVE A

E412 0E00 MVI C,0

E414 CD37E7 CALL HDTRK ;HOME THE DRIVE

E417 F1 WARMLOD POP PSW ;RESTORE SECTOR

E418 E1 POP H ;RESTORE DMA ADDRESS

VR-1412

VSI-READ

```

E419 3C      INR      A
E41A 322AE8  STA      HDSECTR
E41D FE0C    CPI      12          ;PAST BDOS ?
E41F CAA5E3  JZ       GOCPM          ;YES, ALL DONE
E422 24      INR      H          ;UPDATE DMA ADDRESS
E423 24      INR      H
E424 22A1E7  SHLD     HDADD
E427 E5      PUSH     H
E428 F5      PUSH     PSW
E429 01000A  WARMRD  LXI      B,RETRIES*100H+0 ;RETRY COUNTER
E42C C5      WRMREAD PUSH     B          ;SAVE THE RETRY COUNT
E42D CD87E7  CALL     HDREAD        ;READ THE SECTOR
E430 C1      POP      B
E431 D217E4  JNC     WARMLOD        ;TEST FOR ERROR
E434 05      DCR     B          ;UPDATE THE ERROR COUNT
E435 C22CE4  JNZ     WRMREAD        ;KEEP TRYING IF NOT TO MANY ERRORS
E438 76      HLT     ;ERROR HALT
E439 00      DB      0          ;TRY NOT TO SCREW UP DECISION CPU'S
                ENDIF
    
```

```

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

```

E43A 60      SETSEC  MOV      H,B
E43B 69      MOV      L,C
E43C 220FEE  SHLD     CPMSEC
E43F C9      DONOP  RET          ;NULL SINGLE.COM HOOKUP FOR NO FLOPPIES
    
```

```

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

```

E440 60      SETDMA  MOV      H,B          ;HL <- BC
E441 69      MOV      L,C
E442 221BE6  SHLD     CPMDMA        ;CP/M DMA ADDRESS
E445 C9      RET
    
```

```

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

```

E446 0E00    HOME   MVI      C,0          ;TRACK TO SEEK TO
    
```

```

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

```
E448 79      SETTRK MOV      A,C          ;A <- TRACK #
E449 3212EE STA      CPMTRK        ;CP/M TRACK #
E44C C9      RET
```

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

```
E44D 3A11EE SECTRAN LDA      (MAXHD NE 0) AND (MAXFLOP NE 0) ;BOTH TYPES ?
                        CPMDRV          ;GET THE DRIVE NUMBER
```

```
E450 FE03      CPI      FIRST
                        MAXHD*LOGDSK   ;OVER THE # OF HARD DISKS ?
```

```
E452 DA84E4      JC      TRANHD
                  ELSE
                  CPI      MAXFLOP     ;OVER THE # OF FLOPPIES ?
```

```
JNC      TRANHD
ENDIF
ENDIF
```

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

```
SECTRAN EQU      $
ENDIF
```

```
E455 03      TRANFP IF      MAXFLOP NE 0   ;FLOPPY TRANSLATION
```

```
E456 D5      INX      B
E457 C5      PUSH     D          ;SAVE TABLE ADDRESS
```

```
E458 CD9FE5  CALL     GETDPB          ;GET DPB ADDRESS INTO HL
E45B 7E      MOV      A,M        ;GET # OF CP/M SECTORS/TRACK
```

```
E45C B7      ORA      A          ;CLEAR CARY
E45D 1F      RAR      ;DIVIDE BY TWO
```

```
E45E 91      SUB      C
E45F F5      PUSH     PSW        ;SAVE ADJUSTED SECTOR
```

```
E460 FA6CE4  JM      SIDETWO
E463 F1      SIDEA  POP      PSW   ;DISCARD ADJUSTED SECTOR
```

```
E464 C1      POP      B          ;RESTORE SECTOR REQUESTED
E465 D1      POP      D          ;RESTOR ADDRESS OF XLT TABLE
```

```
E466 EB      SIDEONE XCHG        ;HL <- &(TRANSLATION TABLE)
```

```
E467 09      DAD      B          ;BC = OFFSET INTO TABLE
```

```
E468 6E      MOV      L,M        ;HL <- PHYSICAL SECTOR
E469 2600    MVI      H,0
E46B C9      RET
```

```
E46C 010F00  SIDETWO LXI      B,15        ;OFFSET TO SIDE BIT
```

```
E46F 09      DAD      B
E470 7E      MOV      A,M
```

```
E471 E608    ANI      8          ;TEST FOR DOUBLE SIDED
```

```
E473 CA63E4  JZ      SIDEA          ;MEDIA IS ONLY SINGLE SIDED
E476 F1      POP      PSW        ;RETRIEVE ADJUSTED SECTOR
```

VR-1412

VSI-READ

E477 C1 POP B
 E478 2F CMA ;MAKE SECTOR REQUEST POSITIVE
 E479 3C INR A

E47A 4F MOV C,A ;MAKE NEW SECTOR THE REQUESTED SECTOR
 E47B D1 POP D
 E47C CD66E4 CALL SIDEONE
 E47F 3E80 MVI A,80H ;SIDE TWO BIT
 E481 B4 ORA H ; AND SECTOR
 E482 67 MOV H,A

E483 C9 RET
 ENDIF

IF MAXHD NE 0 ;HARD DISK TRANSLATION ROUTINE

E484 60 TRANHD MOV H,B
 E485 69 MOV L,C

E486 23 INX H
 E487 C9 RET
 ENDIF

 *
 * SETDRV SELECTS THE NEXT DRIVE TO BE USED IN READ/WRITE
 * OPERATIONS. IF THE DRIVE HAS NEVER BEEN SELECTED BEFORE, A
 * PARAMETER TABLE IS CREATED WHICH CORRECTLY DESCRIBES THE
 * DISKETTE CURRENTLY IN THE DRIVE. DISKETTES CAN BE OF FOUR
 * DIFFERENT SECTOR SIZES:
 * 1) 128 BYTES SINGLE DENSITY.
 * 2) 256 BYTES DOUBLE DENSITY.
 * 3) 512 BYTES DOUBLE DENSITY.
 * 4) 1024 BYTES DOUBLE DENSITY.
 *

E488 79 SETDRV MOV A,C ;SAVE THE DRIVE #
 E489 3211EE STA CPMDRV
 E48C FE04 CPI MAXFLOP+(MAXHD*LOGDSK) ;CHECK FOR A VALID DRIVE #
 E48E D290E5 JNC ZRET ;ILLEGAL DRIVE #
 E491 7B MOV A,E ;TEST IF DRIVE EVER LOGGED IN BEFORE
 E492 E601 ANI 1
 E494 C277E5 JNZ SETDRV1 ;BIT 0 OF E = 0 -> NEVER SELECTED BEFORE
 IF (MAXHD NE 0) AND (MAXFLOP NE 0) ;BOTH TYPES ?
 E497 3A11EE LDA CPMDRV ;GET THE DRIVE NUMBER

IF FIRST
 E49A FE03 CPI MAXHD*LOGDSK ;OVER THE # OF HARD DISKS ?
 E49C DA4AE5 JC DRVHD
 E49F D603 SUI MAXHD*LOGDSK
 ELSE
 CPI MAXFLOP ;OVER THE # OF FLOPPIES ?
 JNC SUBFP

ENDIF
 ENDIF

IF (MAXFLOP NE 0) AND FIRST
 E4A1 4F MOV C,A ;SAVE DRIVE #

```

E4A2 3E00      MVI      A,0           ;HAVE THE FLOPPIES BEEN ACCESSED YET ?
E4A3 =        FLOPFLG EQU    $-1
E4A4 A7        ANA      A
E4A5 C2F5E4    JNZ      FLOPOK
E4A8 0611      MVI      B,17          ;FLOPPIES HAVN'T BEEN ACCESSED
E4AA 2100F8    LXI      H,DJBOOT    ;CHECK IF 2D CONTROLLER IS INSTALLED
E4AD 3EC3      MVI      A,(JMP)
E4AF BE        CLOPP    CMP      M
E4B0 C290E5    JNZ      ZRET
E4B3 23        INX      H
E4B4 23        INX      H
E4B5 23        INX      H
E4B6 05        DCR      B
E4B7 C2AFE4    JNZ      CLOPP
E4BA 11D2E4    LXI      D,DJINIT    ;INITIALIZATION SEQUENCE
E4BD 21E2FF    LXI      H,ORIGIN+7E2H ;LOAD ADDRESS
E4C0 061E      MVI      B,30         ;BYTE COUNT
E4C2 CD03E7    CALL     MOVLOP
E4C5 3EFF      MVI      A,0FFH      ;START 1791
E4C7 32F9FB    STA     DREG
E4CA 3ED0      MVI      A,CLRCMD    ;1791 RESET
E4CC 32FCFB    STA     CMDREG
E4CF C3F0E4    JMP     DJNEXT
E4D2 0000001800DJINIT DB 0,0,0,18H,0,0,8,0,7EH,0,8,0,9,0FFH,9,0FFH
E4E2 09FF09FF09 DB 9,0FFH,9,0FFH,9,0,1,0,0,0,0,0,0,0
E4F0 3E01      DJNEXT  MVI      A,1           ;SAVE 2D INITIALIZED FLAG
E4F2 32A3E4    STA     FLOPFLG
                ENDIF
                IF      MAXFLOP NE 0
E4F5 210100    FLOPOK  LXI      H,1           ;SELECT SECTOR 1 OF TRACK 1
E4F8 2213EE    SHLD   TRUESEC
E4FB 3E01      MVI      A,1
E4FD 3212EE    STA     CPMTRK
E500 CDCDE6    CALL     FILL          ;FLUSH BUFFER AND REFILL
E503 DA90E5    JC      ZRET          ;TEST FOR ERROR RETURN
E506 CD27F8    CALL     DJSTAT        ;GET STATUS ON CURRENT DRIVE
E509 E60C      ANI     0CH           ;STRIP OFF UNWANTED BITS
E50B F5        PUSH   PSW           ;USED TO SELECT A DPB
E50C 1F        RAR
E50D 21B8E5    LXI     H,XLTS        ;TABLE OF XLT ADDRESSES
E510 5F        MOV     E,A
E511 1600      MVI     D,0
E513 19        DAD     D
E514 E5        PUSH   H           ;SAVE POINTER TO PROPER XLT
E515 CD9FE5    CALL   GETDPB        ;GET DPH POINTER INTO DE
E518 EB        XCHG
E519 D1        POP     D
E51A 0602      MVI     B,2           ;NUMBER OF BYTES TO MOVE
E51C CD03E7    CALL   MOVLOP        ;MOVE THE ADDRESS OF XLT
E51F 110800    LXI     D,8           ;OFFSET TO DPB POINTER
E522 19        DAD     D           ;HL <- &DPH.DPB
E523 E5        PUSH   H
E524 2A07F8    LHLD   ORIGIN+7      ;GET ADDRESS OF DJ TERMINAL OUT ROUTINE
E527 23        INX     H           ;BUMP TO LOOK AT ADDRESS OF

```

VR-1412

VISI-READ

; UART STATUS LOCATION

```

E528 7E      MOV      A,M
E529 EE03    XRI      3          ;ADJUST FOR PROPER REV DJ
E52B 6F      MOV      L,A
E52C 26FB    MVI      H,(ORIGIN+300H)/100H
E52E 7E      MOV      A,M
E52F E608    ANI      DBLSID      ;CHECK DOUBLE SIDED BIT
E531 111FE9  LXI      D,DPB128S   ;BASE FOR SINGLE SIDED DPB'S
E534 C23AE5  JNZ      SIDEOK
E537 115FE9  LXI      D,DPB128D   ;BASE OF DOUBLE SIDED DPB'S
E53A EB      SIDEOK  XCHG      ;HL <- DBP BASE, DE <- &DPH.DPB
E53B D1      POP      D          ;RESTORE DE (POINTER INTO DPH)
E53C F1      POP      PSW      ;OFFSET TO CORRECT DPB
E53D 17      RAL
E53E 17      RAL
E53F 4F      MOV      C,A
E540 0600    MVI      B,0
E542 09      DAD      B
E543 EB      XCHG      ;PUT DPB ADDRESS IN DPH
E544 73      MOV      M,E
E545 23      INX      H
E546 72      MOV      M,D
                ENDIF
E547 C377E5  IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
                JMP      SETDRV1      ;SKIP OVER THE HARD DISK SELECT
                IF      NOT FIRST
                SUBFP  SUI      MAXFLOP      ;ADJUST THE DRIVE #
                ENDIF
                ENDIF
E54A CD96E5  DRVHD  IF      MAXHD NE 0
                CALL     DIVLOG      ;DIVIDE BY LOGICAL DISKS PER DRIVE
E54D 79      MOV      A,C
E54E 324CE8  STA      HDDISK
E551 CD3AE8  CALL     DRVPTR
E554 7E      MOV      A,M
E555 3C      INR      A
E556 C277E5  JNZ      SETDRV1
E559 F6FC    ORI      NULL          ;SELECT DRIVE
E55B D352    OUT     HDFUNC
E55D 3E05    MVI      A,SCENBL      ;ENABLE THE CONTROLLER
E55F D350    OUT     HDCNTL
E561 0EEF    MVI      C,239         ;WAIT APPROX 2 MINUTES FOR DISK TO READY
E563 210000  LXI      H,0
E566 2B      TDELAY DCX      H
E567 7C      MOV      A,H
E568 B5      ORA      L
E569 CC94E5  CZ      DCRC
E56C C8      RZ
E56D DB50    IN      HDSTAT      ;TEST IF READY YET
E56F E620    ANI      DRVRDY
E571 C266E5  JNZ      TDELAY
                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
    
```

```

INDX1  IN      HDSTAT
        ANA     C
        CMP     B          ;LOOP UTIL INDEX LEVEL CHANGES
    
```

```

INDX2  JZ      INDX1
        INX     H
        IN      HDSTAT    ;START COUNTING UNTIL INDEX RETURNS TO
        ANA     C          ;      PREVIOUS STATE
        CMP     B
        JNZ     INDX2
    
```

```

SHLD   SETTLE      ;SAVE THE COUNT FOR TIMEOUT DELAY
ENDIF
    
```

```

E574 CD29E7      CALL HDHOME
                ENDIF
    
```

```

E577 CD9FE5      SETDRVI CALL GETDPB      ;GET ADDRESS OF DPB IN HL
E57A 010F00      LXI   B,15          ;OFFSET TO SECTOR SIZE
E57D 09          DAD    B
    
```

```

E57E 7E          MOV   A,M          ;GET SECTOR SIZE
    
```

```

E57F E607        ANI   7H
E581 32CCE5      STA   SECSIZ
E584 7E          MOV   A,M
    
```

```

E585 1F          RAR
E586 1F          RAR
    
```

```

E587 1F          RAR
E588 1F          RAR
    
```

```

E589 E60F        ANI   0FH
E58B 320AE6      STA   SECPSEC
    
```

```

E58E EB          XCHG      ;HL <- DPH
E58F C9          RET
    
```

```

E590 210000      ZRET  LXI   H,0          ;SELDRV ERROR EXIT
E593 C9          RET
    
```

```

E594 0D          DCRC   IF      MAXHD NE 0
E595 C9          RET      C          ;CONDITIONAL DECREMENT C ROUTINE
    
```

```

E596 0E00        DIVLOG MVI   C,0
E598 D603        DIVLOGX SUI   LOGDSK
E59A D8          RC
E59B 0C          INR   C
E59C C398E5      JMP   DIVLOGX
                ENDIF
    
```

```

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

```

E59F 3A11EE      GETDPB LDA   CPMDRV
    
```

VR-1412
VISI-READ

```

E5A2 6F      MOV     L,A           ;FORM OFFSET
E5A3 2600    MVI     H,0
E5A5 29      DAD     H
E5A6 29      DAD     H
E5A7 29      DAD     H
E5A8 29      DAD     H
E5A9 11CFE9  LXI     D,DPBASE   ;BASE OF DPH'S
E5AC 19      DAD     D
E5AD E5      PUSH    H           ;SAVE ADDRESS OF DPH
E5AE 110A00  LXI     D,10         ;OFFSET TO DPB
E5B1 19      DAD     D
E5B2 7E      MOV     A,M           ;GET LOW BYTE OF DPB ADDRESS
E5B3 23      INX     H
E5B4 66      MOV     H,M           ;GET LOW BYTE OF DPB
E5B5 6F      MOV     L,A
E5B6 D1      POP     D
E5B7 C9      RET
    
```

```

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

```

E5B8 51E8    XLTS    IF     MAXFLOP NE 0
E5BA 6CE8    DW     XLT128   ;XLT FOR 128 BYTE SECTORS
E5BC A1E8    DW     XLT256   ;XLT FOR 256 BYTE SECTORS
E5BE DEE8    DW     XLT512   ;XLT FOR 512 BYTE SECTORS
                DW     XLT124   ;XLT FOR 1024 BYTE SECTORS
                ENDF
    
```

```

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

```

E5C0 79      WRITE  MOV     A,C           ;SAVE WRITE COMMAND TYPE
E5C1 3232E6  STA     WRITTP
E5C4 3E01    MVI     A,1           ;SET WRITE COMMAND
E5C6 06      DB     (MVI) OR (B*8) ;THIS "MVI B" INSTRUCTION CAUSES
                                ; THE FOLLOWING "XRA A" TO
                                ; BE SKIPPED OVER.
    
```

```

*****
*
* READ ROUTINE TO BUFFER DATA FROM THE DISK. IF THE SECTOR
* REQUESTED FROM CP/M IS IN THE BUFFER, THEN THE DATA IS SIMPLY
* TRANSFERRED FROM THE BUFFER TO THE DESIRED DMA ADDRESS. IF
*
*****
    
```

VR-1412

VISI-READ

* 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. *
 * * * * *

```
E5C7 AF READ XRA A ;SET THE COMMAND TYPE TO READ
E5C8 321EE6 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. *
 * * * * *

```
E5CB 0600 REDWRT MVI B,0 ;THE 0 IS MODIFIED TO CONTAIN THE LOG2
E5CC = SECSIZ EQU $-1 ; OF THE PHYSICAL SECTOR SIZE/128
; ON THE CURRENTLY SELECTED DISK.
```

```
E5CD 2A0FEE LHL D CPMSEC ;GET THE DESIRED CP/M SECTOR #
```

```
E5D0 7C MOV A,H
E5D1 E680 ANI 80H ;SAVE ONLY THE SIDE BIT
E5D3 4F MOV C,A ;REMEMBER THE SIDE
```

```
E5D4 7C MOV A,H
E5D5 E67F ANI 7FH ;FORGET THE SIDE BIT
E5D7 67 MOV H,A
```

```
E5D8 2B DCX H ;TEMPORARY ADJUSTMENT
E5D9 05 DIVLOOP DCR B ;UPDATE REPEAT COUNT
E5DA CAE7E5 JZ DIVDONE
```

```
E5DD B7 ORA A
```

```
E5DE 7C MOV A,H
E5DF 1F RAR
```

```
E5E0 67 MOV H,A
E5E1 7D MOV A,L
```

```
E5E2 1F RAR ;DIVIDE THE CP/M SECTOR # BY THE SIZE
; OF THE PHYSICAL SECTORS
```

```
E5E3 6F MOV L,A
E5E4 C3D9E5 JMP DIVLOOP ;
```

```
E5E7 23 DIVDONE INX H
E5E8 7C MOV A,H
E5E9 B1 ORA C ;RESTORE THE SIDE BIT
```

```
E5EA 67 MOV H,A
E5EB 2213EE SHLD TRUESEC ;SAVE THE PHYSICAL SECTOR NUMBER
E5EE 2111EE LXI H,CPMDRV ;POINTER TO DESIRED DRIVE,TRACK, AND SECTOR
```

```
E5F1 1115EE LXI D,BUFDRV ;POINTER TO BUFFER DRIVE,TRACK, AND SECTOR
E5F4 0605 MVI B,5 ;COUNT LOOP
E5F6 05 DTSLOP DCR B ;TEST IF DONE WITH COMPARE
```

```
E5F7 CA05E6 JZ MOVE ;YES, MATCH. GO MOVE THE DATA
E5FA 1A LDAX D ;GET A BYTE TO COMPARE
E5FB BE CMP M ;TEST FOR MATCH
```

```
E5FC 23 INX H ;BUMP POINTERS TO NEXT DATA ITEM
E5FD 13 INX D
```

VR-1412

VISI HEAD

E5FE CAF6E5 JZ DTSLOP ;MATCH, CONTINUE TESTING

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

E601 CDCDE6 CALL FILL ;FILL THE BUFFER WITH CORRECT PHYSICAL SECTOR
 E604 D8 RC ;NO GOOD, RETURN WITH ERROR INDICATION

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

E605 3A0FEE MOVE LDA CPMSEC ;GET THE CP/M SECTOR TO TRANSFER
 E608 3D DCR A ;ADJUST TO PROPER SECTOR IN BUFFER
 E609 E600 ANI 0 ;STRIP OFF HIGH ORDERED BITS
 E60A = SECPSEC EQU \$-1 ;THE 0 IS MODIFIED TO REPRESENT THE # OF
 ; CP/M SECTORS PER PHYSICAL SECTORS

E60B 6F MOV L,A ;PUT INTO HL
 E60C 2600 MVI H,0
 E60E 29 DAD H ;FORM OFFSET INTO BUFFER

E60F 29 DAD H
 E610 29 DAD H
 E611 29 DAD H
 E612 29 DAD H
 E613 29 DAD H
 E614 29 DAD H

E615 110FEA LXI D,BUFFER ;BEGINNING ADDRESS OF BUFFER
 E618 19 DAD D ;FORM BEGINNING ADDRESS OF SECTGR TO TRANSFER
 E619 EB XCHG ;DE = ADDRESS IN BUFFER
 E61A 210000 LXI H,0 ;GET DMA ADDRESS, THE 0 IS MODIFIED T/
 ; CONTAIN THE DMA ADDRESS

E61B = CPMDMA EQU \$-2
 E61D 3E00 MVI A,0 ;THE ZERO GETS MODIFIED TO CONTAIN
 ; A ZERO IF A READ, OR A 1 IF WRITE

E61E = RDWR EQU \$-1
 E61F A7 ANA A ;TEST WHICH KIND OF OPERATION
 E620 C228E6 JNZ INTO ;TRANSFER DATA INTO THE BUFFER
 E623 CD01E7 OUTOF CALL MOVER

E626 AF XRA A
 E627 C9 RET

E628 EB INTO XCHG ;
 E629 CD01E7 CALL MOVER ;MOVE THE DATA, HL = DESTINATION
 ; DE = SOURCE

E62C 3E01 MVI A,1
 E62E 323BE6 STA BUFWRTN ;SET BUFFER WRITTEN INTO FLAG
 E631 3E00 MVI A,0 ;CHECK FOR DIRECTORY WRITE

E632 = WRITYP EQU \$-1
 E633 3D DCR A

VR-1412

VISH-READ


```

E634 3E00      MVI      A,0
E636 3232E6    STA      WRITTYP      ;SET NO DIRECTORY WRITE
E639 C0        RNZ              ;NO ERROR EXIT
    
```

```

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

```

E63A 3E00      FLUSH    MVI      A,0      ;THE 0 IS MODIFIED TO REFLECT IF
                                ; THE BUFFER HAS BEEN WRITTEN INTO
E63B =         BUFWRN EQU    $-1
E63C A7        ANA      A          ;TEST IF WRITTEN INTO
E63D C8        RZ              ;NOT WRITTEN, ALL DONE
    
```

```

                                IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E63E 2118F8    LXI      H,DJWRITE  ;WRITE OPERATION FOR DISK JOCKEY
E641 11BCE7    LXI      D,HDWRITE  ;WRITE OPERATION FOR HARD DISK
E644 CD10E7    CALL     DECIDE
    
```

```

                                ELSE
                                IF      MAXHD NE 0
                                LXI      H,HDWRITE
                                ENDIF
                                IF      MAXFLOP NE 0
                                LXI      H,DJWRITE
                                ENDIF
                                ENDIF
    
```

```

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

```

E647 F3        PREP     DI          ;RESET INTERRUPTS
E648 AF        XRA      A          ;RESET BUFFER WRITTEN FLAG
E649 323BE6    STA      BUFWRN
E64C 22AEE6    SHLD    RETRYOP      ;SET UP THE READ/WRITE OPERATION
E64F 060A      MVI      B,RETRIES ;MAXIMUM NUMBER OF RETRIES TO ATTEMPT
E651 C5        RETRYLP PUSH    B      ;SAVE THE RETRY COUNT
E652 3A15EE    LDA      BUFDRV     ;GET DRIVE NUMBER INVOLVED IN THE OPERATION
    
```

```

                                IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
                                IF      FIRST
E655 FE03      CPI      MAXHD*LOGDSK
E657 DA5CE6    JC       NOADJST
E65A D603      SUI      MAXHD*LOGDSK
                                ELSE
                                CPI      MAXFLOP
                                JC       NOADJST
                                SUI      MAXFLOP
                                —ENDIF
    
```

VR-1412

VISH-READ

```

E65C 4F      NOADJST MOV    C,A
E65D 2133E3  LXI    H,DJDRV      ;SELECT DRIVE
E660 1118E7  LXI    D,HDDRV
E663 CD0CE7  CALL   DECIDGO
ELSE
MOV    C,A
IF     MAXHD NE 0
CALL   HDDRV
ENDIF
IF     MAXFLOP NE 0
CALL   DJDRV      ;SELECT THE DRIVE
ENDIF
ENDIF

E666 3A16EE  LDA    BUFTRK
E669 A7      ANA    A      ;TEST FOR TRACK ZERO
E66A 4F      MOV    C,A
E66B C5     PUSH   B

IF     (MAXHD NE 0) AND (MAXFLOP NE 0)
E66C 2109F8  LXI    H,DJHOME
E66F 1129E7  LXI    D,HDHOME
E672 CC0CE7  CZ     DECIDGO
ELSE
IF     MAXHD NE 0
CZ     HDHOME
ENDIF
IF     MAXFLOP NE 0
CZ     DJHOME      ;HOME THE DRIVE IF TRACK 0
ENDIF
ENDIF

E675 C1     POP    B      ;RESTORE TRACK #

IF     (MAXHD NE 0) AND (MAXFLOP NE 0)
E676 210CF8  LXI    H,DJTRK
E679 1137E7  LXI    D,HDTRK
E67C CD0CE7  CALL   DECIDGO
ELSE
IF     MAXHD NE 0
CALL   HDTRK
ENDIF
IF     MAXFLOP NE 0
CALL   DJTRK      ;SEEK TO PROPER TRACK
ENDIF
ENDIF

E67F 2A17EE  LHLD   BUFSEC
E682 7C     MOV    A,H      ;GET SECTOR INVOLVED IN OPERATION
E683 07     RLC      ;BIT 0 OF A EQUALS SIDE #
E684 E601   ANI    1      ;STRIP OFF UNNECESSARY BITS
E686 4F     MOV    C,A      ;C <- SIDE #

IF     (MAXHD NE 0) AND (MAXFLOP NE 0)
E687 2130F8  LXI    H,DJSIDE

```

VR-1412

VISI-READ

E68A 1160E7 LXI D,HDSIDE
 E68D CD0CE7 CALL DECIDGO

ELSE
 IF MAXHD NE 0
 CALL HDSIDE
 ENDIF

IF MAXFLOP NE 0
 CALL DJSIDE ;SELECT THE SIDE
 ENDIF
 ENDIF

E690 2A17EE LHL D, BUFSEC
 E693 7C MOV A, H
 E694 E67F ANI 7FH ;STRIP OFF SIDE BIT
 E696 47 MOV B, A ;C ← SECTOR #
 E697 4D MOV C, L

IF (MAXHD NE 0) AND (MAXFLOP NE 0)
 E698 210FF8 LXI H, DJSEC
 E69B 1169E7 LXI D, HDSEC
 E69E CD0CE7 CALL DECIDGO

ELSE
 IF MAXHD NE 0
 CALL HDSEC
 ENDIF

IF MAXFLOP NE 0
 CALL DJSEC ;SELECT THE SIDE
 ENDIF
 ENDIF

E6A1 010FEA LXI B, BUFFER ;SET THE DMA ADDRESS

IF (MAXHD NE 0) AND (MAXFLOP NE 0)
 E6A4 2112F8 LXI H, DJDMA
 E6A7 115BE7 LXI D, HDDMA
 E6AA CD0CE7 CALL DECIDGO

ELSE
 IF MAXHD NE 0
 CALL HDDMA
 ENDIF

IF MAXFLOP NE 0
 CALL DJDMA ;SELECT THE SIDE
 ENDIF
 ENDIF

E6AD CD0000 CALL 0 ;GET OPERATION ADDRESS
 E6AE = RETRYOP EQU \$-2
 E6B0 C1 POP B ;RESTORE THE RETRY COUNTER
 E6B1 3E00 MVI A, 0 ;NO ERROR EXIT STATUS
 E6B3 D0 RNC ;RETURN NO ERROR
 E6B4 05 DCR B ;UPDATE THE RETRY COUNTER
 E6B5 37 STC ;ASSUME RETRY COUNT EXPIRED
 E6B6 3EFF MVI A, 0FFH ;ERROR RETURN
 E6B8 C8 RZ ;RETURN SAD NEWS
 E6B9 78 MOV A, B
 E6BA FE05 CPI RETRIES/2 ;RESEEK AFTER HALF RETRIES DONE

VR-1412

E6BC C251E6
E6BF C5

JNZ RETRYLP ;TRY AGAIN
PUSH B

E6C0 2109F8 IF (MAXHD NE 0) AND (MAXFLOP NE 0)
E6C3 1129E7 LXI H,DJHOME
E6C6 CC0CE7 LXI D,HDHOME
CZ DECIDGO

ELSE
IF MAXHD NE 0
CZ HDHOME
ENDIF
IF MAXFLOP NE 0
CZ DJHOME ;HOME THE DRIVE IF TRACK 0
ENDIF
ENDIF

E6C9 C1
E6CA C351E6

POP B
JMP RETRYLP ;TRY AGAIN

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

E6CD CD3AE6 FILL CALL FLUSH ;FLUSH BUFFER FIRST
E6D0 D8 RC ;CHECK FOR ERROR
E6D1 1111EE LXI D,CPMDRV ;UPDATE THE DRIVE, TRACK, AND SECTOR
E6D4 2115EE LXI H,BUFDRV
E6D7 0604 MVI B,4 ;NUMBER OF BYTES TO MOVE
E6D9 CD03E7 CALL MOVLOP ;COPY THE DATA

E6DC 3A1EE6 LDA RDWR
E6DF A7 ANA A
E6E0 CAF5E6 JZ FREAD
E6E3 3A32E6 LDA WRITYP
E6E6 3D DCR A
E6E7 3D DCR A
E6E8 C8 RZ
E6E9 CD9FE5 CALL GETDPB
E6EC 110F00 LXI D,15
E6EF 19 DAD D
E6F0 7E MOV A,M
E6F1 E603 ANI 3
E6F3 3D DCR A
E6F4 C8 RZ

E6F5 = FREAD EQU \$
IF (MAXHD NE 0) AND (MAXFLOP NE 0)
E6F5 2115F8 LXI H,DJREAD
E6F8 1187E7 LXI D,HDREAD
E6FB CD10E7 CALL DECIDE
ELSE
IF MAXHD NE 0
LXI H,HDREAD
ENDIF

VR1412

VISI-READ

```

IF      MAXFLOP NE 0
LXI     H,DJREAD      ;SELECT THE SIDE
ENDIF
ENDIF

```

```

E6FE C347E6      JMP     PREP      ;SELECT DRIVE, TRACK, AND SECTOR.
                  ;      THEN READ THE BUFFER

```

```

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

```

```

E701 0680      MOVER  MVI     B,128      ;LENGTH OF TRANSFER
E703 1A        MOVLOP LDAX   D          ;GET A BTE OF SOURCE
E704 77        MOV     M,A          ;MOVE IT
E705 13        INX    D          ;BUMP POINTERS
E706 23        INX    H
E707 05        DCR    B          ;UPDATE COUNTER
E708 C203E7    JNZ    MOVLOP      ;CONTINUE MOVING UNTIL DONE
E70B C9        RET

```

```

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

```

```

IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E70C CD10E7    DECIDGO CALL  DECIDE  ;WHICH CONTROLLER ?
E70F E9        PCHL
ENDIF

```

```

IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E710 3A15EE    DECIDE  LDA     BUFDIV  ;GET PROPER ROUTINE INTO H&L, BASED
                  ;      ON CURRENTLY SELECTED DRIVE
E713 FE03      CPI     MAXHD*LOGDSK
E715 D0        RNC
ELSE
E716 EB        CPI     MAXFLOP
E717 C9        RC
ENDIF
XCHG
RET
ENDIF

```

```

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

```

```

IF      MAXHD NE 0
E718 79        HDDRV  MOV     A,C      ;SELECT HARD DISK DRIVE

```

VR-1412

VISI-READ

```

E719 CD96E5 CALL DIVLOG ;GET THE PHYSICAL DRIVE #
E71C 79 MOV A,C
E71D 324CE8 STA HDDISK ;SELECT THE DRIVE
E720 F6FC ORI NULL
E722 D352 OUT HDFUNC
E724 3E0F MVI A,WENABL
E726 D350 OUT HDCNTL
E728 C9 RET
    
```

```

E729 CD3AE8 HDHOME CALL DRVPTR
E72C 3600 MVI M,0 ;SET TRACK TO ZERO
E72E DB50 IN HDSTAT ;TEST STATUS
E730 E601 ANI TKZERO ;AT TRACK ZERO ?
E732 C8 RZ ;YES
    
```

```

STEP0 IF NOT FUJITSU
IN HDSTAT ;TEST STATUS
ANI TKZERO ;AT TRACK ZERO ?
    
```

```

JZ DELAY
MVI A,1
STC
    
```

```

CALL ACCOK ;TAKE ONE STEP OUT
JMP STEP0
    
```

ELSE

```

E733 AF XRA A
E734 C348E7 JMP ACCOK
ENDIF
    
```

```

DELAY IF NOT FUJITSU
SETTLE LXI H,0 ;GET DELAY
DELOOP EQU $-2
    
```

```

DCX H ;WAIT 20MS
    
```

```

MOV A,H
ORA L
INX H
DCX H
JNZ DELOOP
    
```

```

RET
ENDIF
    
```

```

E737 CD3AE8 HDTRK CALL DRVPTR ;GET POINTER TO CURRENT TRACK
E73A 5E MOV E,M ;GET CURRENT TRACK
E73B 71 MOV M,C ;UPDATE THE TRACK
E73C 7B MOV A,E ;NEED TO SEEK AT ALL ?
E73D 91 SUB C
E73E C8 RZ
E73F 3F CMC ;GET CARRY INTO DIRECTION
    
```

```

E740 DA45E7 JC HDTRK2
E743 2F CMA
E744 3C INR A
    
```

```

E745 C348E7 HDTRK2 IF FUJITSU
JMP ACCOK
    
```

```

ELSE
HDTRK2 CALL ACCOK
    
```

```

JMP     DELAY
ENDIF

```

```

E748 47      ACCOK  MOV     B,A           ;PREP FOR BUILD
E749 CD45E8  CALL    BUILD
E74C E6FB    SLOOP  ANI     NSTEP        ;GET STEP PULSE LOW
E74E D352    OUT     HDFUNC       ;OUTPUT LOW STEP LINE
E750 F604    ORI     PSTEP        ;SET STEP LINE HIGH
E752 D352    OUT     HDFUNC       ;OUTPUT HIGH STEP LINE
E754 05      DCR     B           ;UPDATE REPEAT COUNT
E755 C24CE7  JNZ     SLOOP        ;KEEP GOING THE REQUIRED # OF TRACKS
E758 C361E7  JMP     WSDONE

```

```

E75B 60      HDDMA  MOV     H,B           ;SAVE THE DMA ADDRESS
E75C 69      MOV     L,C

```

```

E75D 22A1E7  SHLD   HDADD
E760 =       HD SIDE EQU     $
E760 C9      RET

```

```

E761 DB50    WSDONE  IN      HDSTAT        ;WAIT FOR SEEK COMPLETE TO FINISH
E763 E604    ANI     COMPLT
E765 CA61E7  JZ      WSDONE
E768 C9      RET

```

```

E769 79      HDSEC  MOV     A,C
E76A CD7EE7  CALL    DIVSPT
E76D C615    ADI     HDSPT
E76F A7      ANA     A
E770 CC7AE7  CZ      GETSPT
E773 322AE8  STA     HDSECTR
E776 79      MOV     A,C
E777 3246E8  STA     HEAD
E77A 3E15    GETSPT MVI     A,HDSPT
E77C 0D      DCR     C
E77D C9      RET

```

```

E77E 0E00    DIVSPT MVI     C,0
E780 D615    DIVSPTX SUI     HDSPT
E782 D8      RC
E783 0C      INR     C
E784 C380E7  JMP     DIVSPTX

```

```

E787 CD05E8  HDREAD  CALL    HDPREP
E78A D8      RC
E78B AF      XRA     A
E78C D351    OUT     HDCMND
E78E 2F      CMA
E78F D353    OUT     HDDATA
E791 D353    OUT     HDDATA
E793 3E01    MVI     A,RSECT      ;READ SECTOR COMMAND
E795 D351    OUT     HDCMND
E797 CDEBE7  CALL    PROCESS
E79A D8      RC
E79B AF      XRA     A
E79C D351    OUT     HDCMND
E79E 0680    MVI     B, SECLN/4

```

```

E7A0 210000 LXI H,0
E7A1 = HDADD EQU $-2
E7A3 DB53 IN HDDATA
E7A5 DB53 IN HDDATA
E7A7 DB53 RTLOOP IN HDDATA ;MOVE FOUR BYTES
E7A9 77 MOV M,A
E7AA 23 INX H
E7AB DB53 IN HDDATA
E7AD 77 MOV M,A
E7AE 23 INX H
E7AF DB53 IN HDDATA
E7B1 77 MOV M,A
E7B2 23 INX H
E7B3 DB53 IN HDDATA
E7B5 77 MOV M,A
E7B6 23 INX H
E7B7 05 DCR B
E7B8 C2A7E7 JNZ RTLOOP
E7BB C9 RET

E7BC CD05E8 HDWRITE CALL HDPREP ;PREPARE HEADER
E7BF D8 RC
E7C0 AF XRA A
E7C1 D351 OUT HDCMND
E7C3 2AA1E7 LHLD HDADD
E7C6 0680 MVI B,SECLN/4
E7C8 7E WTLOOP MOV A,M ;MOVE 4 BYTES
E7C9 D353 OUT HDDATA
E7CB 23 INX H
E7CC 7E MOV A,M
E7CD D353 OUT HDDATA
E7CF 23 INX H
E7D0 7E MOV A,M
E7D1 D353 OUT HDDATA
E7D3 23 INX H
E7D4 7E MOV A,M
E7D5 D353 OUT HDDATA
E7D7 23 INX H
E7D8 05 DCR B
E7D9 C2C8E7 JNZ WTLOOP
E7DC 3E05 MVI A,WSECT ;ISSUE WRITE SECTOR COMMAND
E7DE D351 OUT HDCMND
E7E0 CDEBE7 CALL PROCESS
E7E3 D8 RC
E7E4 3E10 MVI A,WFAULT
E7E6 A0 ANA B
E7E7 37 STC
E7E8 C8 RZ
E7E9 AF XRA A
E7EA C9 RET

E7EB DB50 PROCESS IN HDSTAT ;WAIT FOR COMMAND TO FINISH
E7ED 47 MOV B,A
E7EE B602 ANI OPDONE
E7F0 CAEBE7 JZ PROCESS
E7F3 3E07 MVI A,DSKCLK
    
```

VR-1412

VISI-READ


```

E7F5 D350      OUT      HDCNTL
E7F7 DB50      IN       HDSTAT
E7F9 E608      ANI      TMOUT          ;TIMED OUT ?
E7FB 37        STC
E7FC C0        RNZ
E7FD DB51      IN       HDRESLT
E7FF E602      ANI      RETRY          ;ANY RETRIES ?
E801 37        STC
E802 C0        RNZ
E803 AF        XRA      A
E804 C9        RET
E805 DB50      HDPREP  IN       HDSTAT
E807 E620      ANI      DRVRDY
E809 37        STC
E80A C0        RNZ
E80B 3E08      MVI      A,ISBUFF          ;INITIALIZE POINTER
E80D D351      OUT      HDCMND
E80F CD45E8    CALL     BUILD
E812 F60C      ORI      0CH
E814 D352      OUT      HDFUNC
E816 3A46E8    LDA      HEAD
E819 D353      OUT      HDDATA          ;FORM HEAD BYTE
E81B CD3AE8    CALL     DRVPTR
E81E 7E        MOV      A,M              ;FORM TRACK BYTE
E81F D353      OUT      HDDATA
E821 A7        ANA      A
E822 0680      MVI      B,80H
E824 CA29E8    JZ       ZKEY
E827 0600      MVI      B,0
E829 3E00      ZKEY   MVI      A,0          ;FORM SECTOR BYTE
E82A =         HDSECTR EQU     $-1
E82B D353      OUT      HDDATA
E82D 78        MOV      A,B
E82E D353      OUT      HDDATA
E830 3E07      MVI      A,DSKCLK
E832 D350      OUT      HDCNTL
E834 3E0F      MVI      A,WENABL
E836 D350      OUT      HDCNTL
E838 AF        XRA      A
E839 C9        RET
E83A 2A4CE8    DRVPTR  LHLD     HDDISK
E83D EB        XCHG
E83E 1600      MVI      D,0
E840 2150E8    LXI     H,DRIVES
E843 19        DAD     D
E844 C9        RET
E845 3E00      BUILD  MVI      A,0
E846 =         HEAD   EQU     $-1
E847 17        RAL
E848 17        RAL
E849 17        RAL
E84A 17        RAL
E84B F600      ORI      0

```

VR-1412

VISI-READ

```

E84C =      HDDISK EQU      $-1
E84D EEF0   XRI      0F0H
E84F C9     RET
    
```

```

E850 =      DRIVES EQU      $
          REPT    MAXHD
          DB      0FFH
          ENDM
E850+FF    DB      0FFH
    
```

ENDIF

```

*****
*
* XLT TABLES (SECTOR SKEW TABLES) FOR CP/M 2.0. THESE TABLES
* DEFINE THE SECTOR TRANSLATION THAT OCCURS WHEN MAPPING CP/M
* SECTORS TO PHYSICAL SECTORS ON THE DISK. THERE IS ONE SKEW
* TABLE FOR EACH OF THE POSSIBLE SECTOR SIZES. CURRENTLY THE
* TABLES ARE LOCATED ON TRACK 0 SECTORS 6 AND 8. THEY ARE
* LOADED INTO MEMORY IN THE CBIOS RAM BY THE COLD BOOT ROUTINE.
*
*****
    
```

```

E851 00      XLT128  IF      MAXFLOP NE 0
          DB      0
E852 01070D1319 DB      1,7,13,19,25
E857 050B1117   DB      5,11,17,23
E85B 03090F15   DB      3,9,15,21
E85F 02080E141A DB      2,8,14,20,26
E864 060C1218   DB      6,12,18,24
E868 040A1016   DB      4,10,16,22
    
```

```

E86C 00      XLT256  DB      0
E86D 0102131425 DB      1,2,19,20,37,38
E873 0304151627 DB      3,4,21,22,39,40
E879 0506171829 DB      5,6,23,24,41,42
E87F 0708191A2B DB      7,8,25,26,43,44
E885 090A1B1C2D DB      9,10,27,28,45,46
E88B 0B0C1D1E2F DB      11,12,29,30,47,48
E891 0D0E1F2031 DB      13,14,31,32,49,50
E897 0F10212233 DB      15,16,33,34,51,52
E89D 11122324   DB      17,18,35,36
    
```

```

E8A1 00      XLT512  DB      0
E8A2 0102030411 DB      1,2,3,4,17,18,19,20
E8AA 2122232431 DB      33,34,35,36,49,50,51,52
E8B2 0506070815 DB      5,6,7,8,21,22,23,24
E8BA 2526272835 DB      37,38,39,40,53,54,55,56
E8C2 090A0B0C19 DB      9,10,11,12,25,26,27,28
E8CA 292A2B2C39 DB      41,42,43,44,57,58,59,60
E8D2 0D0E0F101D DB      13,14,15,16,29,30,31,32
E8DA 2D2E2F30   DB      45,46,47,48
    
```

```

E8DE 00      XLT124  DB      0
E8DF 0102030405 DB      1,2,3,4,5,6,7,8
E8E7 191A1B1C1D DB      25,26,27,28,29,30,31,32
E8EF 3132333435 DB      49,50,51,52,53,54,55,56
    
```

VR-1412

VISI HEAD

```

E8F7 090A0B0C0D DB 9,10,11,12,13,14,15,16
E8FF 2122232425 DB 33,34,35,36,37,38,39,40
E907 393A3B3C3D DB 57,58,59,60,61,62,63,64
E90F 1112131415 DB 17,18,19,20,21,22,23,24
E917 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.
*
*****
    
```

```

E91F 1A00 DPB128S DW 26 ;CP/M SECTORS/TRACK
E921 03 DB 3 ;BSH
E922 07 DB 7 ;BLM
E923 00 DB 0 ;EXM
E924 F200 DW 242 ;DSM
E926 3F00 DW 63 ;DRM
E928 C0 DB 0C0H ;AL0
E929 00 DB 0 ;AL1
E92A 1000 DW 16 ;CKS
E92C 0200 DW 2 ;OFF
E92E 01 DB 1H ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
;LOG2(#BYTES PER SECTOR/128) + 1 +
;8 IF DOUBLE SIDED.
    
```

```

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

```

E92F 3400 DPB256S DW 52 ;CP/M SECTORS/TRACK
E931 04 DB 4 ;BSH
E932 0F DB 15 ;BLM
E933 00 DB 0 ;EXM
E934 F200 DW 242 ;DSM
E936 7F00 DW 127 ;DRM
E938 C0 DB 0C0H ;AL0
E939 00 DB 0 ;AL1
E93A 2000 DW 32 ;CKS
E93C 0200 DW 2 ;OFF
E93E 12 DB 12H ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
;LOG2(#BYTES PER SECTOR/128) + 1 +
;8 IF DOUBLE SIDED.
    
```

```

*****
    
```

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

E93F	3C00	DPB512S	DW	60	;CP/M SECTORS/TRACK
E941	04		DB	4	;BSH
E942	0F		DB	15	;BLM
E943	00		DB	0	;EXM
E944	1801		DW	280	;DSM
E946	7F00		DW	127	;DRM
E948	C0		DB	0C0H	;AL0
E949	00		DB	0	;AL1
E94A	2000		DW	32	;CKS
E94C	0200		DW	2	;OFF
E94E	33		DB	33H	;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) + ;LOG2(#BYTES PER SECTOR/128) + 1 + ;8 IF DOUBLE SIDED.

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

E94F	4000	DP1024S	DW	64	;CP/M SECTORS/TRACK
E951	04		DB	4	;BSH
E952	0F		DB	15	;BLM
E953	00		DB	0	;EXM
E954	2B01		DW	299	;DSM
E956	7F00		DW	127	;DRM
E958	C0		DB	0C0H	;AL0
E959	00		DB	0	;AL1
E95A	2000		DW	32	;CKS
E95C	0200		DW	2	;OFF
E95E	74		DB	74H	;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) + ;LOG2(#BYTES PER SECTOR/128) + 1 + ;8 IF DOUBLE SIDED.

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

E95F	3400	DPB128D	DW	52	;CP/M SECTORS/TRACK
E961	04		DB	4	;BSH
E962	0F		DB	15	;BLM
E963	01		DB	1	;EXM
E964	F200		DW	242	;DSM
E966	7F00		DW	127	;DRM
E968	C0		DB	0C0H	;AL0
E969	00		DB	0	;AL1

VR-1412

VISI-READ

E96A 2000 DW 32 ;CKS
E96C 0200 DW 2 ;OFF
E96E 09 DB 9H

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

E96F 6800 DPB256D DW 104 ;CP/M SECTORS/TRACK
E971 04 DB 4 ;BSH
E972 0F DB 15 ;BLM
E973 00 DB 0 ;EXM
E974 E601 DW 486 ;DSM
E976 FF00 DW 255 ;DRM
E978 F0 DB 0F0H ;AL0
E979 00 DB 0 ;AL1
E97A 4000 DW 64 ;CKS
E97C 0200 DW 2 ;OFF
E97E 1A DB 1AH

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

E97F 7800 DPB512D DW 120 ;CP/M SECTORS/TRACK
E981 04 DB 4 ;BSH
E982 0F DB 15 ;BLM
E983 00 DB 0 ;EXM
E984 3102 DW 561 ;DSM
E986 FF00 DW 255 ;DRM
E988 F0 DB 0F0H ;AL0
E989 00 DB 0 ;AL1
E98A 4000 DW 64 ;CKS
E98C 0200 DW 2 ;OFF
E98E 3B DB 3BH

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

E98F 8000 DP1024D DW 128 ;CP/M SECTORS/TRACK
E991 04 DB 4 ;BSH
E992 0F DB 15 ;BLM
E993 00 DB 0 ;EXM
E994 5702 DW 599 ;DSM
E996 FF00 DW 255 ;DRM
E998 F0 DB 0F0H ;AL0

VR-1412

VISI-READ

```

E999 00      DB      0      ;AL1
E99A 4000    DW      64     ;CKS
E99C 0200    DW      2      ;OFF
E99E 7C      DB      7CH

```

ENDIF

```

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

```

```

IF      STDLOG EQ 0      ;USE STANDARD FORMAT
IF      MAXHD NE 0

```

```

E99F A002    DPBHD1  DW      672     ;CP/M SECTORS/TRACK
E9A1 05      DB      5      ;BSH
E9A2 1F      DB      31     ;BLM
E9A3 01      DB      1      ;EXM
E9A4 DF07    DW      2015    ;DSM
E9A6 FF01    DW      511     ;DRM
E9A8 FF      DB      0FFH    ;AL0
E9A9 FF      DB      0FFH    ;AL1
E9AA 0000    DW      0      ;CKS
E9AC 0100    DW      1      ;OFF
E9AE 33      DB      33H     ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +

```

```

;LOG2(#BYTES PER SECTOR/128) + 1 +
;8 IF DOUBLE SIDED.
E9AF A002    DPBHD2  DW      672     ;CP/M SECTORS/TRACK

```

```

E9B1 05      DB      5      ;BSH
E9B2 1F      DB      31     ;BLM
E9B3 01      DB      1      ;EXM
E9B4 DF07    DW      2015    ;DSM
E9B6 FF01    DW      511     ;DRM
E9B8 FF      DB      0FFH    ;AL0
E9B9 FF      DB      0FFH    ;AL1
E9BA 0000    DW      0      ;CKS
E9BC 6200    DW      98     ;OFF
E9BE 33      DB      33H     ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
;LOG2(#BYTES PER SECTOR/128) + 1 +
;8 IF DOUBLE SIDED.

```

```

E9BF A002    DPBHD3  DW      672     ;CP/M SECTORS/TRACK
E9C1 05      DB      5      ;BSH
E9C2 1F      DB      31     ;BLM
E9C3 01      DB      1      ;EXM
E9C4 0404    DW      1028    ;DSM
E9C6 FF01    DW      511     ;DRM
E9C8 FF      DB      0FFH    ;AL0
E9C9 FF      DB      0FFH    ;AL1
E9CA 0000    DW      0      ;CKS
E9CC C300    DW      195    ;OFF
E9CE 33      DB      33H     ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
;LOG2(#BYTES PER SECTOR/128) + 1 +
;8 IF DOUBLE SIDED.

```

VR-1412

VSI-READ

ENDIF
ENDIF
ENDIF

```
*****
*
* THE FOLLOWING DPB'S ARE USED WHEN THE USER SELECTES THE
* NUMBER OF LOGICAL DRIVES. THESE MACROS DIVIDE EVENLY THE
* SPACE PER LOGICAL DRIVE WHERE THE STANDARD FORMAT TRIES
* TO CREATE THE LEAST AMOUNT OF LOGICAL DRIVES WITH THE
* MOST SPACE PER LOGICAL DRIVE.
*
*****
```

```
IF STDLOG NE 0
    MDPBHD MACRO L,D
    DPBHD&L DW SECPT
    DB BSH
    DB BLM
    DB EXM
    IF LDSK NE 0
    DW (TOTBLS/LOGDSK)
    ELSE
    DW (TOTBLS/LOGDSK)-1 ;RESERVED CPM TRACK
    ENDIF
    DW DRM
    DB AL0
    DB AL1
    DW CKS
    DW (TRACKS/LOGDSK)*&D+1
    DB SLOG
    ENDM
```

```
IF MAXHD NE 0
    IF M20 NE 0
    SECPT EQU 672
    TOTBLS EQU 5124
    TRACKS EQU 244
    ENDIF
    BSH EQU 5
    BLM EQU 31
    EXM EQU 1
    DRM EQU 511
    AL0 EQU 0FFH
    AL1 EQU 0FFH
    CKS EQU 0
    SLOG EQU 33H
```

```
LDSK SET 0
    REPT MAXHD
DPBDRV SET 0
    REPT STDLOG
    MDPBHD %LDSK,%DPBDRV
```

VR-1412

VISI-READ

```
LDSK SET LDSK+1
DPBDRV SET DPBDRV+1
ENDM
ENDM
ENDIF
ENDIF
```

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

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

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

```
E9CF = DPBASE EQU $
```

```
0000 # DN IF STDLOG EQ 0 ✓
    SET 0
    IF FIRST ✓
    REPT MAXHD ;GENERATE HARD DISK DPH'S FOLLOWED
    HEADER %DN,DPBHD1 ; BY FLOPPY DPH'S
    DN SET DN+1
    HEADER %DN,DPBHD2
    DN SET DN+1
    IF M20 NE 0 ✓
    DN HEADER %DN,DPBHD3
    SET DN+1
    ENDIF
```

```
ENDM
E9CF+0000 DW 0 ;TRANSLATION TABLE FILLED IN LATER
E9D1+0000000000 DW 0,0,0 ;SCRATCH
E9D7+19EE DW DIRBUF ;DIRECTORY BUFFER
E9D9+9FE9 DW DPBHD1 ;DPB FILLED IN LATER
E9DB+95EF DW CSV0 ;DIRECTORY CHECK VECTOR
E9DD+99EE DW ALV0 ;ALLOCATION VECTOR
```

VR-1412 VSI-READ


```

E9DF+0000 DW 0 ;TRANSLATION TABLE FILLED IN LATER
E9E1+0000000000 DW 0,0,0 ;SCRATCH
E9E7+19EE DW DIRBUF ;DIRECTORY BUFFER
E9E9+AFE9 DW DPBHD2 ;DPB FILLED IN LATER
E9EB+91F0 DW CSV1 ;DIRECTORY CHECK VECTOR
E9ED+95EF DW ALV1 ;ALLOCATION VECTOR
E9EF+0000 DW 0 ;TRANSLATION TABLE FILLED IN LATER
E9F1+0000000000 DW 0,0,0 ;SCRATCH
E9F7+19EE DW DIRBUF ;DIRECTORY BUFFER
E9F9+BFE9 DW DPBHD3 ;DPB FILLED IN LATER
E9FB+12F1 DW CSV2 ;DIRECTORY CHECK VECTOR
E9FD+91F0 DW ALV2 ;ALLOCATION VECTOR

```

```

DN REPT MAXFLOP
HEADER %DN,0
SET DN+1

```

PE 7015 HDTK2 JMP ACCOR

```

E9FF+0000 DW 0 ;TRANSLATION TABLE FILLED IN LATER
EA01+0000000000 DW 0,0,0 ;SCRATCH
EA07+19EE DW DIRBUF ;DIRECTORY BUFFER
EA09+0000 DW 0 ;DPB FILLED IN LATER
EA0B+5DF1 DW CSV3 ;DIRECTORY CHECK VECTOR
EA0D+12F1 DW ALV3 ;ALLOCATION VECTOR

```

UE 74B @30000 JMP DELAY

```

DN ELSE
REPT MAXFLOP ;GENERATE FLOPPY DPH'S FOLLOWED BY
HEADER %DN,0 ; HARD DISK DPH'S
SET DN+1
ENDM

```

```

DN REPT MAXHD
HEADER %DN,DPBHD1
SET DN+1

```

```

DN REPT MAXHD
HEADER %DN,DPBHD2
SET DN+1

```

```

DN REPT MAXHD
HEADER %DN,DPBHD3
SET DN+1

```

```

ENDIF
ENDIF
ENDIF

```

```

DN IF STDLOG NE 0
IF FIRST
SET MAXFLOP
REPT MAXHD
REPT STDLOG ;GENERATE HARD DISK DPH'S FOLLOWED

```

```

DN HEADER %DN,DPBHD,%(DN-MAXFLOP) ;BY FLOPPY DPH'S
SET DN+1
ENDM

```

```

DN SET 0 ;FLOPPIES ALWAYS START AT ZERO
REPT MAXFLOP

```

```

DN HEADER %DN,0,0
SET DN+1
ENDM

```

```

ELSE ;GENERATE FLOPPIES BEFORE HARD DISK

```

VR-1412

```

DN      SET      0
        REPT     MAXFLOP
        HEADER   %DN,0,0
DN      SET      DN+1
        ENDM
DN      SET      MAXFLOP
        REPT     MAXHD
        REPT     STDLOG
        HEADER   %DN,DPBHD,%(DN-MAXFLOP)
DN      SET      DN+1
        ENDM
        ENDM
        ENDIF
        ENDIF
    
```

```
EA0F = BUFFER EQU $
```

```

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

```

EA0F 801A      PROMPT DB      80H, CLEAR ;CLEAN BUFFER AND SCREEN
EA11 0A0D0A0D0A DB      ACR,ALF,ACR,ALF,ACR,ALF
EA17 4D6F72726F DB      'Morrow Designs '
EA26 36        DB      '0'+MSIZE/10 ;CP/M MEMORY SIZE
EA27 32        DB      '0'+(MSIZE MOD 10)
EA28 4B2043502F DB      'K CP/M ' ;CP/M VERSION NUMBER
EA2F 32        DB      CPMREV/10+'0'
EA30 2E        DB      '.'
EA31 32        DB      (CPMREV MOD 10)+'0'
EA32 2C20436269 DB      ', Cbios rev '
EA3E 322E      DB      REVNUM/10+'0', '.' ;CBIOS REVISION NUMBER
EA40 39        DB      REVNUM MOD 10+'0'
EA41 2E        DB      '.'
EA42 32        DB      MREV/10+'0'
EA43 30        DB      MREV MOD 10+'0'
EA44 0A0D      DB      ACR,ALF
EA46 466F7220  DB      'For '
EA4A 6120446973 DB      'a Disk Jockey 2D/B'
EA5C 20616E6420 DB      ' and '
EA61 6120      DB      'a '
EA63 46756A6974 DB      'Fujitsu M20 '
EA6F 6861726420 DB      'hard disk'
EA78 2E        DB      '.'
EA79 0A0D      DB      ACR,ALF
EA7B 4465636973 DB      'Decision I'
EA85 2061732063 DB      ' as console'
EA90 2E        DB      '.'
EA91 0A0D      DB      ACR, ALF
EA93 00        DB      0 ;END OF MESSAGE
    
```

```

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

VR-1412 VIS-READ

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

EA94	7E	MESSAGE	MOV	A,M	;GET A CHARACTER OF THE MESSAGE
EA95	23		INX	H	;BUMP TEXT POINTER
EA96	B7		ORA	A	;TEST FOR END
EA97	C8		RZ		;RETURN IF DONE
EA98	E5		PUSH	H	;SAVE POINTER TO TEXT
EA99	4F		MOV	C,A	;OUTPUT CHARACTER IN C
EA9A	CD0CE3		CALL	COU	;OUTPUT THE CHARACTER
EA9D	E1		POP	H	;RESTORE THE POINTER
EA9E	C394EA		JMP	MESSAGE	;CONTINUE UNTIL NULL REACHED

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

EAA1	310001	CBOOT	LXI	SP,TPA	;SET UP STACK
EAA4	AF		XRA	A	;CLEAR COLD BOOT FLAG
EAA5	32F7E3		STA	CWFLG	
EAA8	323AE3		STA	GROUP	;CLEAR GROUP SELECT BYTE
EAAB	2100FC		IF	MAXFLOP NE 0	;IF 2D/B IS THERE THEN MAKE RAM COPY
			LXI	H,DJRAM	; OF THE JUMP TABLE.
EAAE	1100F8		LXI	D,ORIGIN	
EAB1	0633		MVI	B,33H	;SIZE OF JUMP TABLE
EAB3	CD03E7		CALL	MOVLOP	;COPY TABLE
			ENDIF		
EAB6	3E00		MVI	A,INTIOBY	
EAB8	320300		STA	IOBYTE	
EABB	CDE0EA		CALL	TINIT	;INITIALIZE THE TERMINAL
EABE	CD0CEB		CALL	LINIT	;INITIALIZE THE LIST DEVICE
EAC1	210FEA		LXI	H,PROMPT	;PREP FOR SENDING SIGNON MESSAGE
EAC4	CD94EA		CALL	MESSAGE	;SEND THE PROMPT
EAC7	AF		XRA	A	;SELECT DISK A
EAC8	3211EE		STA	CPMDRV	
EACB	320400		STA	CDISK	
EACE	32A3E4		IF	(MAXFLOP NE 0) AND FIRST	
			STA	FLOPFLG	
			ENDIF		
EAD1	2103E3		LXI	H,BIOS+3	;PATCH COLD BOOT TO WARM CODE
EAD4	2201E3		SHLD	BIOS+1	
EAD7	C3A5E3		JMP	GOCPM	

VR-1412

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

EADA 1B7B3F3137TV950 DB LBH,7BH,3FH,'171';SEND 'ESC { ? 1 7 1' TO TELEVIDEO 950

EAE0 21DAEA TINIT LXI H,TV950 ;SET TELEVIDEO 950 TO 19.2KB
 EAE3 CD94EA CALL MESSAGE

EAE6 3A3AE3 LDA GROUP ;GET GROUP BYTE
 EAE9 F601 ORI CONGRP ;SELECT CONSOLE DEVICE
 EAEB D34F OUT GRPSEL

EAED DB48 IN RBR ;CLEAR RECIEVER BUFFERS

EAEF DB48 IN RBR
 EAF1 AF XRA A
 EAF2 D34D OUT LSR ;CLEAR STATUS
 EAF4 D349 OUT IER ;SET NO INTERRUPTS

EAF6 2A36E3 LHLD DEFCON ;GET DEFAULT BAUD RATE
 EAF9 EB XCHG

EAF A 3E87 MVI A,DLAB+WLS1+WLS0+STB ;ENABLE DIVISOR ACCESS LATCH
 EAF C D34B OUT LCR ;SET THE BAUD RATE IN (DE)

EAF E 7A MOV A,D
 EAF F D349 OUT DLM ;SET UPPER DIVISOR

EB0 1 7B MOV A,E
 EB0 2 D348 OUT DLL ;SET LOWER DIVISOR
 EB0 4 3E07 MVI A,WLS1+WLS0+STB
 EB0 6 D34B OUT LCR

EB0 8 AF DONE XRA A ;CLEAR STATUS REGISTER
 EB0 9 D34D OUT LSR
 EB0 B C9 RET

EB0 C 3A3AE3 LINIT LDA GROUP ;GET GROUP BYTE
 EB0 F F603 ORI LSTGRP ;SELECT LIST DEVICE
 EB1 1 D34F OUT GRPSEL

EB1 3 3E80 MVI A,DLAB ;ACCESS DIVISOR LATCH
 EB1 5 D34B OUT LCR
 EB1 7 2A38E3 LHLD DEFLST ;GET LST: BAUD RATE DIVISOR

EB1 A 7C MOV A,H
 EB1 B D349 OUT DLM ;SET UPPER BAUD RATE
 EB1 D 7D MOV A,L

EB1 E D348 OUT DLL
 EB2 0 3E07 MVI A,STB+WLS0+WLS1
 EB2 2 D34B OUT LCR

EB2 4 DB48 IN RBR ;CLEAR INPUT BUFFER
 EB2 6 AF XRA A
 EB2 7 D349 OUT IER ;NO INTERUPTS

EB2 9 C9 RET

EB2 A 00FF00 DB 0,0FFH,0

EB2 D DS 512-(\$-BUFFER) ;MAXIMUM SIZE BUFFER FOR 512 BYTE SECTORS

*Redo To use
 several lines.*

Add 'd' at end.

*WAIT XRA A
 DEC A
 JNZ WAIT C*

VR-1412

VISI-READ

EC0F IF MAXFLOP NE 0 ;ADDITIONAL SPACE FOR FLOPPIES 1K SECTORS
DS 512
ENDIF

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

EE0F 0000 CPMSEC DW 0 ;CP/M SECTOR #
EE11 00 CPMDRV DB 0 ;CP/M DRIVE #
EE12 00 CPMTRK DB 0 ;CP/M TRACK #
EE13 0000 TRUESEC DW 0 ;DISK JOCKEY SECTOR THAT CONTAINS CP/M SECTOR
EE15 00 BUFDRV DB 0 ;DRIVE THAT BUFFER BELONGS TO
EE16 00 BUFTRK DB 0 ;TRACK THAT BUFFER BELONGS TO
EE17 0000 BUFSEC DW 0 ;SECTOR THAT BUFFER BELONGS TO

EE19 DIRBUF DS 128 ;DIRECTORY BUFFER

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

0000 # DN SET 0

IF STDLOG EQ 0 ✓
IF NOT FIRST NO
REPT MAXFLOP
ALLOC %DN,75,64
SET DN+1
ENDM
REPT MAXHD
IF M20 NE 0 ✓
ALLOC %DN,252,0
SET DN+1
ALLOC %DN,252,0
SET DN+1
ALLOC %DN,129,0
SET DN+1
ENDIF
ENDM

ELSE
REPT MAXHD
IF M20 NE 0 ✓
ALLOC %DN,252,0
SET DN+1
ALLOC %DN,252,0
SET DN+1
ALLOC %DN,129,0
SET DN+1
ENDIF

VR-1412

VISI-READ

```

EE99+      ALV0      DS      252
EF95+      CSV0      DS      0
EF95+      ALV1      DS      252
F091+      CSV1      DS      0
F091+      ALV2      DS      129
F112+      CSV2      DS      0
              REPT    MAXFLOP
              ALLOC   %DN,75,64
              DN
F112+      ALV3      DS      75
F15D+      CSV3      DS      64

```

ENDIF
~~ENDIF~~

```

              IF      STDLOG NE 0
              IF      MAXHD NE 0      ;MAKE UP HARD DISK ALLOCATION VECTORS
              DN      SET      MAXFLOP      ;HARD DISKS ALWAYS START AFTER FLOPPIES
              REPT    MAXHD
              REPT    STDLOG
              DN      ALLOC   %DN,((TOTBLS/LOGDSK)/8)+1,0
              SET      DN+1
              ENDM
              ENDM
              ENDIF

```

```

              DN      IF      MAXFLOP NE 0      ;MAKE UP FLOPPY ALLOCATION VECTORS
              SET      0
              REPT    MAXFLOP      ;FLOPPIES FIRST
              DN      ALLOC   %DN,75,64
              SET      DN+1
              ENDM
              ENDM
              ENDM

```

F19D END

E748 ACCOK	000A ACR	000D ALF	EE99 ALV0	EF95 ALV1
F091 ALV2	F112 ALV3	0000 ANUL	E3F8 AUTOFLG	0004 BANK
D500 BDOS	A800 BIAS	E300 BIOS	EE15 BUFDRV	0080 BUFF
EA0F BUFFER	EE17 BUFSEC	EE16 BUFTRK	E63B BUFWRN	E845 BUILD
EAA1 CBOOT	CD00 CCP	0004 CDISK	E33B CIFLSH	E309 CIN
E33E CITYY	E3F0 CLDBOT	E3DB CLDCMND	001A CLEAR	004A CLK
E4AF CLOPP	00D0 CLRCMD	FBFC CMDREG	E3F9 COLDBEG	E3F9 COLDEND
0004 COMPLT	0001 CONGRP	E345 CONIN1	E358 CONOUT1	E306 CONST
E351 COTTY	E30C COUT	E61B CPMDMA	EE11 CPMDRV	0016 CPMREV
EE0F CPMSEC	EE12 CPMTRK	E363 CSTTY	EF95 CSV0	F091 CSV1
F112 CSV2	F15D CSV3	0010 CTS	E3F7 CWFLG	0048 DAISIO
0049 DAIS11	0048 DAISY0	0049 DAISY1	0008 DBLSID	E594 DCRC
E710 DECIDE	E70C DECIDGO	E336 DEFCON	E338 DEFLST	0020 DENABLE
EE19 DIRBUF	E5E7 DIVDONE	E596 DIVLOG	E598 DIVLOGX	E5D9 DIVLOOP
E77E DIVSPT	E780 DIVSPTX	F800 DJBOOT	F803 DJCIN	F806 DJCOUT
F82D DJDEN	F812 DJDMA	E333 DJDRV	F82A DJERR	F809 DJHOME
E4D2 DJINIT	E4F0 DJNEXT	FC00 DJRAM	F815 DJREAD	F80F DJSEC
F81B DJSEL	F830 DJSIDE	F827 DJSTAT	F80C DJTRK	F821 DJTSTAT
F818 DJWRITE	0080 DLAB	0048 DLL	0049 DLM	EB08 DONE
E43F DONOP	E98F DP1024D	E94F DP1024S	E95F DPB128D	E91F DPB128S
E96F DPB256D	E92F DPB256S	E97F DPB512D	E93F DPB512S	E9CF DPBASE
E99F DPBHD1	E9AF DPBHD2	E9BF DPBHD3	FBF9 DREG	0001 DR
E850 DRIVES	E54A DRVHD	E83A DRVPTR	0020 DRVRDY	0007 DSKCLK
0020 DSR	E5F6 DTSLOP	0008 ENINT	0005 ENTRY	E6CD FILL
0001 FIRST	E4A3 FLOPFLG	E4F5 FLOPOK	E63A FLUSH	E6F5 FREAD
0001 FUJITSU	E59F GETDPB	E77A GETSPT	E3A5 GOCPM	E33A GROUP
004F GRPSEL	0000 GZERO	E7A1 HDADD	0051 HDCMND	0050 HDCNTL
0053 HDDATA	E84C HDDISK	E75B HDDMA	E718 HDDRV	0052 HDFUNC
E729 HDHOME	0050 HDORG	E805 HDPREP	E787 HDREAD	0051 HDRESLT
0004 HDRLEN	E769 HDSEC	E82A HDSECTR	E760 HDSIDE	0015 HDSPT
0050 HDSTAT	E737 HDTRK	E745 HDTRK2	E7BC HDWRITE	E846 HEAD
E446 HOME	0000 IDBUFF	0049 IER	0040 INDEX	0000 INTIOBY
E628 INTO	FBF8 IO	0003 IOBYTE	0008 ISBUFF	004B LCR
EB0C LINIT	E372 LIST	E38B LISTST	E379 LL	0003 LOGDSK
004D LSR	0003 LSTGRP	0001 M20	0001 MAXFLOP	0001 MAXHD
0048 MBASE	00F7 MDIR	EA94 MESSAGE	E605 MOVE	E701 MOVER
E703 MOVLOP	0014 MREV	003E MSIZE	004E MSR	E65C NOADJST
00FB NSTEP	00FC NULL	3E00 OFFSETC	0002 OPDONE	F800 ORIGIN
E623 OUTOF	E647 PREP	E7EB PROCESS	EA0F PROMPT	0004 PSTEP
E39F PUNCH	0048 RBR	E61E RDWR	E3A2 READER	E5C7 READ
E5CB REDWRT	0010 RESTOR	000A RETRIES	0002 RETRY	E651 RETRYLP
E6AE RETRYOP	001D REVNUM	0001 RSECT	E7A7 RTLOOP	0001 S0
0002 SI	0005 SCENBL	0200 SECLN	E60A SECPSEC	E5CC SECSIZ
E44D SECTRAN	0049 SENSESW	E440 SETDMA	E488 SETDRV	E577 SETDRV1
E43A SETSEC	E448 SETTRK	E463 SIDEA	E53A SIDEOK	E466 SIDEONE
E46C SIDETWO	E74C SLOOP	0003 SMASK	0004 STB	0000 STDLOG
E566 TDELAY	0048 THR	0020 THRE	EAE0 TINIT	0001 TKZERO
0008 TMOUT	0100 TPA	E455 TRANFP	E484 TRANHD	EE13 TRUESEC
EADA TV950	E3FA WARMBEG	E3FA WARMEND	E417 WARMLOD	E429 WARMRD
E303 WBOOTE	E3FB WBOOT	0000 WBOT	000F WENABL	0010 WFAULT
0001 WLS0	0002 WLS1	000B WRESET	E5C0 WRITE	E632 WRITTYP
E42C WRMREAD	E761 WSDONE	0005 WSECT	E7C8 WTLOOP	E8DE XLT124
E851 XLT128	E86C XLT256	E8A1 XLT512	E5B8 XLTS	E829 ZKEY
E590 ZRET				