

CBIOSH (FIRST) PRN

8/22/82

CBIOSH, ASM

AG - (LIE) 62K

```

*****
*
* CBIOS FOR CP/M VER 2.2 FOR DISK JOCKEY 2D MOD. B
* CONTROLLER. HANDLES DISKETTES WITH SECTOR SIZES OF 128
* BYTES SINGLE DENSITY, 256, 512, 1024 BYTES DOUBLE
* DENSITY. THERE ARE CONDITIONAL ASSEMBLIES FOR THE
* DISKUS HARD DISK CONTROLLER.
*
* NOTE: THE SYSTEM 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 CAN BE 128, 256, OR 1024 BYTES IN
* EITHER SINGLE OR DOUBLE DENSITY.
*
* SOFTWARE ENGINEERING, MORROW DESIGNS 11/81
*****

```

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

```

```

*****
* 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.
*
*****

```

0002 = CONTYP EQU 2

```

*****
*
* 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 MULTI I/O SERIAL, NO PROTOCOL.
*            3 MULTI I/O SERIAL, CTS PROTOCOL.
*            4 MULTI I/O SERIAL, DSR PROTOCOL.
*            5 MULTI I/O SERIAL, XON / XOFF PROTOCOL.
*            6 MULTI I/O PARALLEL, CENTRONICS.
*            7 MULTI I/O PARALLEL, DIABLO HYTYPE II.
*

```

VR-1412

VISI-READ

```

*
* NOTE: THE DECISION BOARD IS FUNCTIONALLY IDENTICAL TO THE
* MULTI I/O BOARD FOR PRINTER I/O. SELECTIONS 2 - 6
* WILL WORK ON THE DECISION I.
*
*****

```

```
0003 = LSTTYP EQU 3
```

```

*****
*
* 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 THEN YOU SHOULD SET MULTR3 TO 0.
*
*****

```

```
0000 = MULTR3 EQU 0 ;0 = DECISION, 1 = MULTI I/O
```

```
0001 = CONGRP IF CONTYP EQ 2
        EQU 1 ;COSOLE PORT (1 = P1, 2 = P2, 3 = P3)
        ENDIF
```

```
0003 = LSTGRP IF LSTTYP GE 2
        EQU 3 ;PRINTER PORT (1 = P1, 2 = P2, 3 = P3)
        ENDIF
```

```

*****
*
* 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
```

```
0001 = MAXHD EQU 1 ;FLOPPIES ARE E,F,G,H
0004 = MAXFLOP EQU 4 ;SET TO NUMBER OF HARD DISKS
;SET TO NUMBER OF FLOPPIES
```

```
0000 = M26 EQU 0 ;SET ONLY ONE OF THESE VARIABLES
```

```
0001 = M20 EQU 1
```

```
0000 = M10F EQU 0
```

```
0000 = M10M EQU 0
```

```
0000 = M10 EQU M10F OR M10M
```

```

*****
*
* 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
*
*****

```

VR-1412

VISI READ

* 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

IF STDLOG NE 0
 LOGDSK EQU STDLOG ;SET TO NUMBER OF USER SELECTED
 ELSE ; LOGICAL DISKS

0003 = LOGDSK EQU 3*M26+3*M20+2*M10 ;DEFAULT LOGICAL DISKS PER DRIVE

ENDIF

0001 = FUJITSU EQU M20 OR M10F

0014 = MREV EQU 26*M26+20*M20+10*M10 ;HARD DISK TYPE

0015 = HDSPT EQU 32*M26+21*M20+21*M10 ;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 IF MAXFLOP NE 0 ;INCLUDE DISCUS 2D ?

FC00 = DJRAM EQU ORIGIN+400H ;DISK JOCKEY 2D RAM ADDRESS

F800 = DJBOOT EQU ORIGIN ;DISK JOCKEY 2D INITIALIZATION

F803 = DJCIN EQU ORIGIN+3H ;DISK JOCKEY 2D CHARACTER INPUT ROUTINE

F806 = DJCOUT EQU ORIGIN+6H ;DISK JOCKEY 2D CHARACTER OUTPUT ROUTINE

F809 = DJHOME EQU ORIGIN+9H ;DISK JOCKEY 2D TRACK ZERO SEEK

F80C = DJTRK EQU ORIGIN+0CH ;DISK JOCKEY 2D TRACK SEEK ROUTINE

F80F = DJSEC EQU ORIGIN+0FH ;DISK JOCKEY 2D SET SECTOR ROUTINE

F812 = DJDMA EQU ORIGIN+012H ;DISK JOCKEY 2D SET DMA ADDRESS

F815 = DJREAD EQU ORIGIN+15H ;DISK JOCKEY 2D READ ROUTINE

F818 = DJWRITE EQU ORIGIN+18H ;DISK JOCKEY 2D WRITE ROUTINE

F81B = DJSEL EQU ORIGIN+1BH ;DISK JOCKEY 2D SELECT DRIVE ROUTINE

F821 = DJTSTAT EQU ORIGIN+21H ;DISK JOCKEY 2D TERMINAL STATUS ROUTINE

F827 = DJSTAT EQU ORIGIN+27H ;DISK JOCKEY 2D STATUS ROUTINE

F82A = DJERR EQU ORIGIN+2AH ;DISK JOCKEY 2D ERROR, FLASH LED

F82D = DJDEN EQU ORIGIN+2DH ;DISK JOCKEY 2D SET DENSITY ROUTINE

F830 = DJSIDE EQU ORIGIN+30H ;DISK JOCKEY 2D SET SIDE ROUTINE

0008 = DBLSID EQU 8 ;SIDE BIT FROM CONTROLLER

FBF8 = IO EQU ORIGIN+3F8H ;START OF I/O REGISTERS

FBF9 = DREG EQU IO+1

FBFC = CMDREG EQU IO+4

00D0 = CLRCMD EQU 0D0H

ENDIF

* THE FOLLOWING BLOCK WILL DEFINE CERAIN 2DB ENTRY POINTS IN *
 * CASE THE USER IS NOT ACTUALLY USING THE 2DB'S DISKS BUT IS *

VR-1412

MSI-READ

* USING THE 2DB'S CONSOLE DRIVER PROM. *

* * *

IF (MAXFLOP EQ 0) AND (CONTYP EQ 3)

ORIGIN EQU 0F800H
 DJCIN EQU ORIGIN+3H ;DISK JOCKEY 2D CHARACTER INPUT ROUTINE
 DJCOUT EQU ORIGIN+6H ;DISK JOCKEY 2D CHARACTER OUTPUT ROUTINE
 DJTSTAT EQU ORIGIN+21H ;DISK JOCKEY 2D TERMINAL STATUS ROUTINE

ENDIF

* * *

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

* * *

		IF	MAXHD NE 0	;WANT HARD DISK INCLUDED ?
0050 =	HDORG	EQU	50H	;HARD DISK CONTROLLER ORIGIN
0050 =	HDSTAT	EQU	HDORG	;HARD DISK STATUS
0050 =	HDCNTL	EQU	HDORG	;HARD DISK CONTROL
0053 =	HDDATA	EQU	HDORG+3	;HARD DISK DATA
0052 =	HDFUNC	EQU	HDORG+2	;HARD DISK FUNCTION
0051 =	HDCMND	EQU	HDORG+1	;HARD DISK COMMAND
0051 =	HDRESLT	EQU	HDORG+1	;HARD DISK RESULT
0002 =	RETRY	EQU	2	;RETRY BIT OF RESULT
0001 =	TKZERO	EQU	1	;TRACK ZERO BIT OF STATUS
0002 =	OPDONE	EQU	2	;OPERATION DONE BIT OF STATUS
0004 =	COMPLT	EQU	4	;COMPLETE BIT OF STATUS
0008 =	TMOUT	EQU	8	;TIME OUT BIT OF STATUS
0010 =	WFAULT	EQU	10H	;WRITE FAULT BIT OF STATUS
0020 =	DRVRDY	EQU	20H	;DRIVE READY BIT OF STATUS
0040 =	INDEX	EQU	40H	;INDEX BIT OF STATUS
0004 =	PSTEP	EQU	4	;STEP BIT OF FUNCTION
00FB =	NSTEP	EQU	0FBH	;STEP BIT MASK OF FUNCTION
0004 =	HDRLN	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 FOLOWING EQUATES WILL DEFINE THE DECISION I MOTHER *

* BOARD I/O OR THE MULTI I/O ENVIRONMENTS IF NEEDED. *

* * *

VR-1412

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

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 IF MULTR3 EQ 0 ;DAISY OUTPUT PORTS ARE DIFFERENT

0049 = DAISI1 EQU MBASE+1 ; FOR DECISION I AND MULTI I/O.

;THESE TWO ARE THE DECISION I PORTS

ELSE ; AND THESE ARE THE MULTI I/O'S.

DAISI0 EQU MBASE+1

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 = RESTORE 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 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 NO AUTO LINE FEED.

```

ENDIF

```

*****
*
* 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

```

VR-1412

```

0100 = TPA EQU 100H ;TRANSIENT MEMORY
0000 = INTIOBY EQU 0 ;INITIAL IOBYTE
0003 = IOBYTE EQU 3 ;IOBYTE LOCATION
0000 = WBOT EQU 0 ;WARM BOOT JUMP ADDRESS
0005 = ENTRY EQU 5 ;BDOS ENTRY JUMP ADDRESS
    
```

```

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

```

000A = RETRIES EQU 10 ;MAX RETRIES ON DISK I/O BEFORE ERROR
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 = ACR EQU 'J'-64 ;CARRIAGE RETURN
000B = AVT EQU 'K'-64 ;VERTICAL TAB
000C = APF EQU 'L'-64 ;FORM FEED
000D = ALF EQU 'M'-64 ;LINE FEED
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 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 C3F8EA JMP CBOOT ;COLD BOOT ENTRY POINT
E303 C3FEE3 WBOOTE JMP WBOOT ;WARM BOOT ENTRY POINT

IF CONTPR NE 0
E306 C366E3 CONST JMP CSTTY ;CONSOLE STATUS ROUTINE
E309 C33BE3 CIN JMP CIFLSH ;CONSOLE INPUT
E30C C354E3 COUT JMP COTTY ;CONSOLE OUTPUT

ELSE
CIN JMP $ ;CONSOLE STATUS ROUTINE PROM POINTER
CIN JMP $ ;CONSOLE INPUT PROM POINTER
COUT JMP $ ;CONSOLE OUTPUT PROM POINTER

ENDIF
    
```

VR-1412

VISI-READ


```

E30F C375E3      IF LSTTYP NE 0
                JMP     LIST          ;LIST DEVICE OUTPUT
                ELSE
                JMP CONT      ;LIST DEVICE OUTPUT
                ENDIF
    
```

```

E312 C3A2E3      JMP     PUNCH          ;PUNCH DEVICE OUTPUT
E315 C3A5E3      JMP     READER          ;READER DEVICE INPUT
E318 C349E4      JMP     HOME            ;HOME DRIVE
E31B C38BE4      JMP     SETDRV         ;SELECT DISK
E31E C34BE4      JMP     SETTRK        ;SET TRACK
E321 C33DE4      JMP     SETSEC        ;SET SECTOR
E324 C343E4      JMP     SETDMA       ;SET DMA ADDRESS
E327 C3CAE5      JMP     READ          ;READ THE DISK
E32A C3C3E5      JMP     WRITE         ;WRITE THE DISK
    
```

```

E32D C38EE3      IF LSTTYP NE 0
                JMP     LISTST        ;LIST DEVICE STATUS
                ELSE
                JMP S          ;LIST DEVICE STATUS
                ENDIF
    
```

```

E330 C350E4      JMP     SECTRAN         ;SECTOR TRANSLATION
    
```

```

E333 C31BF8      DJDRV IF MAXFLOP NE 0
                JMP     DJSEL         ;HOOKUP FOR SINGLE.COM PROGRAM
                ELSE
                JMP DONOP
                ENDIF
    
```

* THE FOLLOWING TWO WORDS DEFINE THE DEFAULT BAUD RATE FOR *
 * THE CONSOLE AND THE LST: DEVICES. THESE WORDS MUST *
 * IMMEDIATLY FOLLOW THE CBIOS JUMP TABLE SO THAT THE USER *
 * CAN EASLY 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 *
 * VALUE NEEDED FOR THE DEFCON OR DEFLST WORDS. *
 *

BAUD RATE	DEFCON	BAUD RATE	DEFCON
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

```

E336 0C00      DEFCON DW 12 6          ;CONSOLE BAUD RATE
    
```

VR-1412

VISI READ

E338 0C00 DEFLST DW 12 ;PRINTER BAUD RATE

 * THE NEXT BYTE IS TO MAKE SURE THAT THE GROUP SELECT BYTE
 * ON THE MULTI 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 A DRIVER-ENABLE)
 * THEN YOU SHOULD MODIFY THE SAME BIT IN THE GROUP BYTE
 * PROVIDED. 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.

E33A 00 GROUP DB 0 ;GROUP BYTE

 * 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.

E33B CD3DE6 C1FLSH CALL FLUSH ;FLUSH DISK BUFFERS ON INPUT
 E33E C341E3 JMP CITYTY

 * CONTYP: 1 BLANK SPACE FOR CONSOLE DRIVER
 * NOTE: IF THE USER PLANS TO UTILIZE THIS SPACE THEN THE
 * ONE TIME CODE SUCH AS TINIT SOULD BE PLACED JUST BELOW
 * THE CBOOT ROUTINE. THIS SPACE (BELOWE CBOOT) IS RECYLED

VR-1412
 VSI-READ

* FOR USE AS A DISK BUFFER AFTER CBOOT IS DONE. *

* * *

IF CONTYP EQ 1

TINIT EQU \$;MAKE IT EASY TO FIND THIS PLACE

COTTY EQU \$

CITY EQU \$

CSTTY EQU \$

RET

DS 127

ENDIF ;BLANK SPACE

* * *

* 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 CURRENT DIVISOR LATCH VALUE WILL BE CHECKED. *

* IF THE DIVISOR SEEMS TO BE OK THEN NO ACTION WILL BE TAKEN *

* AS FAR AS THE BAUD RATE SETTING GOES. IF THE DIVISOR IS NOT *

* OK THEN 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 COMPILING WITH MULTR3 (A MULTI I/O) THEN *

* THE SWITCHES WILL NOT BE AVAILABLE SO THE BAUD RATE *

* WILL BE TAKEN FROM DEFCON. *

* * *

* * *

* DUE TO ITS LENGTH, THE TINIT ROUTINE DRIVER IS BELOWE THE *

* CBOOT ROUTINE. *

* * *

VR-1412

VIS-READ

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

```
E341 3A3AE3  CTTY  LDA    GROUP    ;GET GROUP BYTE
E344 F601    ORI    CONGRP    ;SELECT CONSOLE
E346 D34F    OUT    GRPSEL

E348 DB4D    CONIN1 IN     LSR      ;READ STATUS REGISTER
E34A E601    ANI    DR        ;WAIT TILL CHARACTER READY
E34C CA48E3  JZ     CONIN1

E34F DB48    IN     RBR      ;READ CHARACTER
E351 E67F    ANI    7FH     ;STRIP PARITY
E353 C9      RET
```

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

```
E354 3A3AE3  CTTY  LDA    GROUP    ;GET GROUP BYTE
E357 F601    ORI    CONGRP    ;SELECT CONSOLE
E359 D34F    OUT    GRPSEL

E35B DB4D    CONOUT1 IN    LSR      ;READ STATUS
E35D E620    ANI    THRE     ;WAIT TILL TRANSMITTER BUFFER EMPTY
E35F CA5BE3  JZ     CONOUT1
E362 79      MOV    A,C      ;CHARACTER IS IN (C)
E363 D348    OUT    THR      ;OUTPUT TO TRANSMITTER BUFFER
E365 C9      RET
```

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

```
E366 3A3AE3  CSTTY LDA    GROUP    ;GET GROUP BYTE
E369 F601    ORI    CONGRP    ;SELECT CONSOLE
E36B D34F    OUT    GRPSEL

E36D DB4D    IN     LSR      ;READ STATUS REGISTER
E36F E601    ANI    DR        ;NO CHARACTER READY
E371 C8      RZ
E372 3EFF    MVI    A,0FFH    ;CHARACTER READY
E374 C9      RET

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

```
*****
```

VR-1412

*
* CONTYP: 3 2DB CONSOLE DRIVER
*

IF CONTYP EQ 3

COTTY JMP DJCOUT ;CONSOLE OUTPUT

CITY JMP DJCIN ;CONSOLE INPUT

CSTTY CALL DJTSTAT ;CONSOLE STATUS

MVI A,0FFH

RZ

INR A

RET

ENDIF ;2DB

UE A41
ACK, ALF

*

* 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 MULTI I/O SERIAL, NO PROTOCOL.

* 3 MULTI I/O SERIAL, CTS PROTOCOL.

* 4 MULTI I/O SERIAL, DSR PROTOCOL.

* 5 MULTI I/O SERIAL, XON / XOFF PROTOCOL.

* 6 MULTI I/O PARALLEL, CENTRONICS.

* 7 MULTI I/O PARALLEL, DIABLO HYTYPE II.

*

*

* LSTTYP: 1 BLANK SPACE FOR PRINTER DRIVER

* NOTE: IF THE USER PLANS TO UTILIZE THIS SPACE THEN THE

* ONE TIME CODE SUCH AS LINIT SOULD BE PLACED JUST BELOW

* THE CBOOT ROUTINE. THIS SPACE (BELOWE CBOOT) IS RECYLED

* FOR USE AS A DISK BUFFER AFTER CBOOT IS DONE.

*

IF LSTTYP EQ 1

LINIT EQU \$;MAKE IT EASY TO FIND THIS PLACE

LIST EQU \$

LISTST EQU \$

RET

DS 127

VR-1412

VISI-READ

ENDIF ;BLANK SPACE

```

*****
*
* LSTTYP: 2 SERIAL PRINTER, NO PROTOCOL
*
* LSTTYP: 3 SERIAL PRINTER, CTS PROTOCOL
*
* LSTTYP: 4 SERIAL PRINTER, DSR PROTOCOL
*
* LSTTYP: 5 SERIAL PRINTER, XON / XOFF PROTOCOL
*
*****

```

IF (LSTTYP GE 2) AND (LSTTYP LE 5)

```

E375 3A3AE3 LIST LDA GROUP ;GET GROUP BYTE
E378 F603 ORI LSTGRP ;SELECT LIST DEVICE
E37A D34F OUT GRPSEL

```

```

E37C DB4D LL IN LSR
E37E E620 ANI THRE ;WAIT TILL TRANSMITTER BUFFER EMPTY
E380 CA7CE3 JZ LL

```

```

*****
*
* THE CTS DRIVER IS USED FOR A PRINTER WITH HARDWARE
* HANDSHAKING (TI 810). IT SHOULD BE CONNECTED TO THE CTS
* INPUT ON THE LIST DEVICE SERIAL PORT.
*
*****

```

```

E383 DB4E IF LSTTYP EQ 3 ;CTS PROTOCOL
E385 E610 IN MSR
E387 CA7CE3 ANI CTS ;WAIT TILL CLEAR TO SEND
JZ LL
ENDIF

```

```

*****
*
* THE DSR DRIVER IS USED FOR A PRINTER WITH HARDWARE
* HANDSHAKING (TI 810). IT SHOULD BE CONNECTED TO THE DSR
* INPUT ON THE LIST DEVICE SERIAL PORT.
*
*****

```

```

IF LSTTYP EQ 4 ;DSR PROTOCOL
IN MSR
ANI DSR
JZ LL ;WAIL TILL DSR COMES UP
ENDIF

```

```

*****
*
* THE XON/XOFF DRIVER IS USED FOR A PRINTER WITH SOFTWARE
* HANDSHAKING (DIABLO 630).
*
*****

```

VR-1412

VS-READ

*

```

XLOOP  IF      LSTTYP EQ 5      ;XON / XOFF PROTOCOL
        CALL   LISTST         ;CHECK PRINTER STATUS
        ORA   A
        JZ    XLOOP          ;LOOP IF NOT READY
        ENDF
    
```

```

E38A 79      MOV    A,C
E38B D348    OUT   THR
E38D C9      RET
    
```

```

E38E 3A3AE3 LISTST LDA    GROUP      ;GET GROUP BYTE
E391 F603    ORI    LSTGRP      ;SELECT LIST DEVICE
E393 D34F    OUT   GRPSEL
    
```

```

E395 DB4D    IN    LSR          ;CHECK IF TRANSMITTER BUFFER EMPTY
E397 E648    ANI   THR
E399 C8      RZ              ;RETURN NOT READY
    
```

```

E39A DB4E    IF      LSTTYP EQ 3    ;CTS PROTOCOL
E39C E610    IN    MSR
E39E C8      ANI   CTS
            RZ              ;RETURN NOT READY IF CTS IS FALSE
            ENDF
    
```

```

            IF      LSTTYP EQ 4    ;DSR PROTOCOL
            IN    MSR
            ANI   DSR
            RZ              ;RETURN NOT READY IF DSR IS TRUE
            ENDF
    
```

```

LSTFLG  IF      LSTTYP EQ 5      ;XON / XOFF PROTOCOL
        MVI   B,XON          ;LAST CHARACTER RECIEVED FROM PRINTER
        EQU  $-1
    
```

```

        IN    LSR
        ANI   DR              ;CHECK FOR A CHARACTER
        JZ    XSKIP          ;NO CHARACTER PRESENT
    
```

```

        IN    RBR            ;GET CHARACTER
        ANI   7FH
        MOV   B,A            ;SAVE
        STA  LSTFLG         ;KLUDGE FLAG (LAST CHARACTER RECIEVED)
    
```

```

XSKIP   MOV   A,B
        SUI  XOFF           ;CHECK FOR XOFF CHAR (CONTROL S)
        JNZ  XSDONE        ;PRINTER READY
        RET                ;PRINTER NOT READY
    
```

```

XSDONE EQU  $              ;PRINTER READY FOR DATA
        ENDF
    
```

```

E39F 3EFF    MVI   A,0FFH
E3A1 C9      RET              ;PRINTER READY
            ENDF            ;MULTI I/O SERIAL
    
```

VR-1412
VISI READ

```
*****
*
* LSTTYP: 6      CENTRONICS PARALLEL PRINTER DRIVER.
*
*****
```

IF LSTTYP EQ 6

```
*****
*
* DECISION I DIABLO PARALLEL TO CENTRONICS PARALLEL INTERFACE.
*
* THE FOLLOWING CABLE MUST BE MADE FROM THE 50 PIN DIABLO
* CONECTOR TO THE 35 PIN CENTRONICS.
*
```

CENTRONICS			MULTI I/O	
PIN	SIGNAL		PIN	SIGNAL
* 1	/STROBE	<-	46	/D9
* 2	DATA1	<-	37	/D1
* 3	DATA2	<-	36	/D2
* 4	DATA3	<-	39	/D3
* 5	DATA4	<-	33	/D4
* 6	DATA5	<-	40	/D5
* 7	DATA6	<-	42	/D6
* 8	DATA7	<-	43	/D7
* 9	DATA8	<-	45	/D8
* 10	/ACKNLG	->	12	/CHECK
* 11	BUSY	->	28	/PRINTER READY
* 12	PE	->	3	/PAPER
* 13	SLCT	->	4	/RIBBON
* 14	/AUTO FEED XT	<-	1	/D10
* 15	NC			
* 16	0V	<->	2	GND
* 17	CHASSIS GND			
* 18	NC			
* 19	/STROBE RTN	<->	8	GND
* 20	DATA1 RTN	<->	11	GND
* 21	DATA2 RTN	<->	14	GND
* 22	DATA3 RTN	<->	16	GND
* 23	DATA4 RTN	<->	18	GND
* 24	DATA5 RTN	<->	20	GND
* 25	DATA6 RTN	<->	22	GND
* 26	DATA7 RTN	<->	25	GND
* 27	DATA8 RTN	<->	38	GND
* 28	/ACKNLG RTN	<->	41	GND
* 29	BUSY RTN	<->	44	GND
* 30	PE RTN	<->	47	GND
* 31	/INIT	<-	9	/D11
* 32	/ERROR	->	5	/COVER
* 33	GND	<->	35	GND
* 34	NC			
* 35	/SLCT IN	<-	10	/D12
			35 <->	24 /SELECT

```
* IMPORTANT: FOR THIS INTERFACE TO WORK /SELECT (24) ON THE
* PARALLEL CONECTOR MUST BE TIED TO GROUND (35).
```

VR-1412

VIS-READ

*

```
LIST LDA GROUP ;GET GROUP BYTE
OUT GRPSEL
```

```
RL IN DAISY0 ;WAIT TILL PRINTER READY AND SELECTED
ANI READY+PAPER
JZ RL
```

```
PL IN DAISY0 ;TEST IF OUT OF PAPER
ANI RIBBON
JNZ PL
```

```
EL IN DAISY0
ANI COVER
JNZ EL
```

```
MOV A,C ;MOVE CHARACTER INTO (A)
OUT DAISI1 ;LATCH DATA
MVI A,D11+D10+D9 ;MAKE SURE STROBE IS HIGH
```

```
OUT DAISI0
DCR A ;PULSE STROBE LOW
OUT DAISI0
```

```
INR A
OUT DAISI0
```

```
ACK IN DAISY0 ;WAIT TILL READY AGAIN
ANI READY
JZ ACK
```

```
RET
```

```
LISTST LDA GROUP ;GET GROUP BYTE
OUT GRPSEL ;SELECT GROUP ZERO
```

```
IN DAISY0 ;WAIT TILL PRINTER READY AND SELECTED
ANI READY+PAPER
```

```
RZ
IN DAISY0 ;TEST IF OUT OF PAPER
ANI RIBBON
```

```
RZ
IN DAISY0
```

```
ANI COVER
XRI COVER
```

```
RZ
DCR A
RET
```

```
ENDIF ;CENTRONICS PARALLEL
```

```
IF LSTTYP EQ 7 ;DIABLO HYTYP II
```

* DIABLO 1610 SIMULATOR FOR THE MORROW DESIGNS / THINKER TOYS *
* MULT I/O BOARD. THE SIMULATOR MAKES THE PARALLEL HYTYP II *
* LOOK LIKE A SERIAL 1610. *

VR-1412

VISI-READ

```

*
* THIS ROUTINE DOES ALL OF THE CHARACTER DECODING, ESCAPE
* SEQUENCES FORWARD, BACKWARD, ETC. THE LIST OF ESCAPE
* SEQUENCES, AND SPECIAL CHARACTERS RECOGNIZED IS:
*
*      ADEL          IGNORED
*      ANUL          IGNORED
*      AACK          IGNORED (WHEN RECEIVED)
*      ABEL          IGNORED
*      AFF           FORM FEED
*      AETX          ETX/ACK HANDSHAKE
*      AHT           HORIZONTAL TAB
*      ALF           LINE FEED
*      ASP           SPACE
*      ABS           BACKSPACE
*      ACR           CARRIAGE RETURN
*      AESC 0        IGNORED
*      AESC 1        SET TAB STOP AT CURRENT PRINT POSITION
*      AESC 2        CLEAR ALL TAB STOPS
*      AESC 3        GRAPHICS MODE ON
*      AESC 4        GRAPHICS MODE OFF
*      AESC 5        FORWARD PRINT
*      AESC 6        BACKWARD PRINT
*      AESC 8        CLEAR TAB STOP
*      AESC 9        SET LEFT MARGIN
*      AESC A        IGNORED
*      AESC B        IGNORED
*      AESC D        NEGATIVE HALF LINE FEED
*      AESC U        HALF LINE FEED
*      AESC ALF      NEGATIVE LINE FEED
*      AESC AHT C    ABSOLUTE HORIZONTAL TAB
*      AESC AVT C    ABSOLUTE VERTICAL TAB
*      AESC ARS C    SET VMI
*      AESC AUS C    SET HMI
*
*****

```

```

LIST  LDA      GROUP      ;SET PRINTER INITIALIZED FLAG
      ORI      DENABLE
      STA      GROUP
      MOV      A,C        ;GET THE CHARACTER TO PRINT
      ANI      7FH       ;STRIP OFF PARITY
      RZ
      CPI      ADEL      ;IGNORE DELETE
      RZ
      MOV      C,A        ;SAVE CHARACTER
      LDA      ESCFLG
      LXI      H,LEVEL0   ;LEVEL ZERO CHARACTERS
      ANA      A
      MOV      A,C        ;SCAN FOR CHAR IN A
      JZ       LOOKUP     ;LOOK UP ACTIVITY FOR THIS CHARACTER
      LDA      ESCFLG
      LXI      H,LEVEL1   ;SINGLE CHARACTER ESCAPE SEQUENCES
      CPI      AESC
      MOV      A,C        ;SCAN FOR CHAR IN A
      JZ       LOOKUP     ;EXECUTE SINGLE LEVEL ESCAPE SEQUENCE

```

VR-1412

VSI-READ

```
LXI H,LEVEL2 ;TWO CHARACTER ESCAPE SEQUENCE
LDA ESCFLG
```

```
*****
* LOOKUP SCANS THE TABLE POINTED AT BY HL LOOKING FOR A MATCH *
* OF THE CHARACTER IN REGISTER A. *
*****
```

```
LOOKUP DCR M ;TEST IF END OF TABLE
      INR M
      JZ GOTHER ;EXECUTE THE DEFAULT FUNCTION
      CMP M ;OTHERWISE TEST FOR A MATCH
      JZ GOTHER
      INX H ;BUMP OVER CHARACTER
      INX H ;BUMP OVER FUNCTION ADDRESS
      INX H
      JMP LOOKUP
GOTHER INX H ;BUMP OVER CHARACTER
      MOV A,M ;GET LOW BYTE OF FUNCTION ADDRESS
      INX H
      MOV H,M ;GET HIGH BYTE OF FUNCTION ADDRESS
      MOV L,A ;FORM ADDRESS OF FUNCTION
      PCHL ;EXECUTE IT
```

```
*****
* EACH OF THE FOLLOWING TABLES CONTAINS ENTRIES OF THE FORM: *
* 1 BYTE CHARACTER TO MATCH *
* 2 BYTES OF ADDRESS TO EXECUTE *
* TERMINATED BY A FIRST BYTE OF 0. *
*****
```

```
LEVEL0 DB AESC
      DW DOAESC ;BEGINNING OF AN ESCAPE SEQUENCE
      DB AFF
      DW DOAFF ;FORM FEED
      DB AETX
      DW DOAETX
      DB AHT
      DW DOAHT ;HORIZONTAL TAB
      DB ALF
      DW DOALF ;LINE FEED
      DB ASP
      DW DOASP ;SPACE
      DB ABS
      DW DOABS ;BACK SPACE
      DB ACR
      DW DOACR ;CARRIAGE RETURN
      DB 0
      DW DOCHAR ;ANY OTHER CHARACTER
```

```
LEVEL1 DB '1'
      DW SETHTAB ;SET HORIZONTAL TAB
      DB '2'
      DW CLRALL ;CLEAR ALL HORIZONTAL TABS
      DB '3'
      DW SETGRP ;GRAPHICS MODE
```

VR-1412

VISI-READ

```

DB      '4'
DW      CLRGRP          ;CLEAR GRAPHICS MODE
DB      '5'
DW      CLRDIR          ;FORWARD PRINTING
DB      '6'
DW      SETDIR          ;BACKWARD PRINTING
DB      '8'
DW      CLRHTAB         ;CLEAR HORIZONTAL TAB
DB      '9'
DW      SETLMAR         ;SET LEFT MARGIN
DB      '0'
DW      FUNC1           ;NO OPERATION LEVEL 1
DB      'A'
DW      FUNC1
DB      'B'
DW      FUNC1
DB      'a'
DW      FUNC1
DB      'b'
DW      FUNC1
DB      'D'
DW      NEGHLF          ;NEGATIVE HALF LINE FEED
DB      'U'
DW      POSHLF          ;HALF LINE FEED
DB      ALF
DW      NEGLF           ;NEGATIVE LINE FEED
DB      AHT
DW      SETTWO          ;TWO CHARACTER ESCAPE SEQUENCE
DB      AVT
DW      SETTWO
DB      ARS
DW      SETTWO
DB      AUS
DW      SETTWO
DB      0
DW      FUNC1

```

```

LEVEL2 DB      AHT
        DW      ABSHTAB      ;ABSOLUTE HORIZONTAL TAB
        DB      AVT
        DW      ABSVTAB      ;ABSOLUTE VERTICAL TAB
        DB      ARS
        DW      SETVMI
        DB      AUS
        DW      SETHMI
        DB      0
        DW      FUNC2

```

* THE FOLLOWING ROUTINES EXECUTE ESCAPE SEQUENCES, ETC. *

```

SETTWO DOAESC  MOV     A,C          ;GET THE ESCAPE CHARACTER
        STA     ESCFLG
FUNC0   RET

```

VR-1412

VIS-READ

DOAETX RET

DOALF	CALL	LFVMI	;GET LINE FEED VMI
ADJVP	XCHG		
	LHLD	DLVPOS	;GET VERTICAL MOTION DISPLACEMENT
	DAD	D	
	SHLD	DLVPOS	
	RET		

LFVMI	LDA	GRHFLG	
	ANA	A	
	LXI	H,1	;ONLY 1/48 IF IN GRAPHICS MODE
	RNZ		
	LHLD	VMI	;GET VERTICAL MOTION INDEX
	RET		

NEGLF	CALL	LFVMI	;GET LINE FEED VMI
	CALL	NEGHL	
	CALL	ADJVP	
	JMP	FUNCI	

DOASP	CALL	SPHMI	;GET SPACE HORIZONTAL MOTION
SPDIR	LDA	DIRFLG	;FORWARD OR BACKWARDS ?
	ANA	A	

ADJHP	CNZ	NEGHL	;NEGATE HL
	XCHG		;ADJUST HORIZONTAL POSITION
	LHLD	DLHPOS	;GET CURRENT ADJUSTMENT
	DAD	D	;UPDATE IT
	SHLD	DLHPOS	;AND SAVE
	RET		

SPHMI	LDA	GRHFLG	;IN GRAPHICS MODE ?
	ANA	A	
	LXI	H,2	;ONLY 1/60 IF IN GRAPHICS MODE
	RNZ		
	LHLD	HMI	
	RET		

DOABS	CALL	SPHMI	;SPACE INCREMENT
	CALL	NEGHL	;NEGATIVE TO START WITH
	JMP	SPDIR	;ADJUST BACKWARDS

DOACR	XRA	A	
	STA	DIRFLG	;FORWARD PRINTING
	STA	GRHFLG	;NO GRAPHICS MODE
	LHLD	HPOS	;GET CURRENT OFFSET
	XCHG		
	LHLD	LMAR	;GET LEFT MARGIN
	CALL	HLMDE	
	SHLD	DLHPOS	;DON'T MOVE YET THOUGH
	MVI	A,AUTOLF	;IN AUTO LINE FEED MODE ?
	ANA	A	
	JNZ	DOALF	;DO LINE FEED ALSO
	RET		

```

DOCHAR  MOV    L,C
        MVI    H,0
        CALL   WHEEL          ;PRINT THE CHARACTER IN REGISTER C
        LDA    GRHFLG
        ANA    A
        LXI    H,0          ;DON'T MOVE IF IN GRAPHICS MODE
        JNZ    SPDIR
        LHL   HMI
        JMP    SPDIR
    
```

```

CLRALL  EQU    $          ;CLEAR ALL HORIZONTAL TABS
        LXI    H,TABSTP   ;BEGINNING OF TAB STOP ARRAY
        MVI    D,TABLEN   ;SIZE OF TAB ARRAY (BYTES)
NOTBLP  MVI    M,80H      ;RESET TABS (RESET TO 0 LATER)
KLUDGE  EQU    $-1       ;USED ON FIRST RESET (WARMBOOT)
        INX    H          ;NEXT TAB STOP
        DCR    D          ;UPDATE REPEAT COUNT
        JNZ    NOTBLP    ;CONTINUE ZEROING
FUNC2   EQU    $
FUNC1   XRA    A          ;CLEAR ESCAPE SEQUENCE FLAG
        STA    ESCFLG
        RET
    
```

```

SETGRP  MVI    A,1        ;SET GRAPHICS MODE ON
        STA    GRHFLG
        JMP    FUNC1
    
```

```

CLRGRP  XRA    A          ;TURN GRAPHICS MODE OFF
        STA    GRHFLG
        JMP    FUNC1
    
```

```

CLRDIR  XRA    A          ;FORWARD PRINT MODE
        STA    DIRFLG
        JMP    FUNC1
    
```

```

SETDIR  MVI    A,A        ;SET BACKWARD PRINTING MODE
        STA    DIRFLG
        JMP    FUNC1
    
```

```

SETLMAR LHL   HPOS        ;GET CURRENT POSITION
        XCHG
        LHL   DLHPOS     ;GET OFFSET
        DAD    D
        SHLD  LMAR
        JMP    FUNC1
    
```

```

SETVMI  MOV    L,C        ;SET THE MOTION INDEX
        MVI    H,0
        DCX    H
        SHLD  VMI
        JMP    FUNC2
    
```

```

SETHMI  MOV    L,C
        MVI    H,0
        DCX    H
        SHLD  HMI
    
```

JMP FUNC2

POSHLF CALL HLFVMI ;HALF LINE FEED VMI

CALL ADJVP
JMP FUNC1

NEGHLF CALL HLFVMI ;NEGATIVE HALF LINE FEED

CALL NEGHL
CALL ADJVP
JMP FUNC1

HLFVMI LHL D VMI ;GET VMI FOR FULL LINE FEED

DIVID2 MOV A,H ;HIGH BYTE
ORA A ;CLEAR THE CARRY
RAR

MOV H,A
MOV A,L
RAR
MOV L,A
RET

ABSHTAB MOV E,C ;ABSOLUTE HORIZONTAL TAB

MVI D,0
DCX D ;FORM 16 BIT TAB COLUMN
CALL NEWDLH
JMP FUNC2

NEWDLH LHL D HMI ;MULTIPLY BY HMI

CALL HLTDE
XCHG
LHL D HPOS ;AND SUBTRACT CURRENT HORIZONTAL POSITION
XCHG
CALL HLMDE

SHLD DLHPOS
RET

ABSVTAB MOV E,C ;ABSOLUTE VERTICAL TAB

MVI D,0
DCX D
LHL D VMI ;MULTIPLY BY VMI
CALL HLTDE
XCHG
LHL D VPOS ;AND SUBTRACT THE CURRENT VERTICAL POSITION
XCHG
CALL HLMDE

SHLD DLVPOS
JMP FUNC2

SEHTAB CALL TABCOL ;SET HORIZONTAL TAB

ORA M ;OR IN TAB STOP
MOV M,A ; AND SAVE
JMP FUNC1

TABCOL LHL D HPOS ;COMPUTE ADDRESS OF CURRENT CHARACTER COL

XCHG
LHL D DLHPOS

VR-1412

VISI-READ

```

DAD      D          ;GET LOGICAL POSITION
XCHG
LHLD    HMI        ;AND DIVIDE BY HMI TO GET CHARACTER COLUMN

```

```

XCHG
CALL    HLDDE

```

```

MTABP   ;MAKE A TAB POINTER
;HL -> TAB COLUMN DESIRED (1-160)
;HL <- ADDRESS OF TAB STOP

```

```

; A <- BIT MASK FOR TAB STOP
LXI     D,8        ;NUMBER OF STOPS PER BYTE
CALL    HLDDE      ;HL/DE -> HL, HL MOD DE -> DE

```

```

MOV     C,E        ;SAVE
INR    C          ;MAKE RANGE (1-8)
LXI     D,TABSTP   ;TAB ARRAY

```

```

DAD     D          ;MAKE ARRAY POINTER
XRA    A
STC

```

```

MTAB0   RAR
DCR    C          ;BUMP BIT COUNTER
JNZ    MTAB0
RET

```

```

CLRHTAB CALL TABCOL ;CLEAR HORIZONTAL TAB

```

```

CMA
ANA    M          ;MASK OUT TAB STOP
MOV    M,A

```

```

JMP    FUNC1

```

```

DOAHT   LHLD      HPOS ;COMPUTE ADDRESS OF CURRENT CHARACTER COL

```

```

XCHG
LHLD   DLHPOS
DAD    D          ;GET LOGICAL POSITION

```

```

XCHG
LHLD   HMI        ;AND DIVIDE BY HMI TO GET CHARACTER COLUMN

```

```

XCHG
CALL   HLDDE

```

```

TABLOP LXI     D,NUMTABS
INX    H          ;START WITH NEXT POSITION

```

```

CALL   HLCDE      ;COMPARE POSITION WITH NUMBER OF TABS
JNC    TOFAR      ;PAST LAST TAB
PUSH   H          ;SAVE COL POINTER

```

```

CALL   MTABP      ;GENERATE TAB POINTER
ANA    M          ;CHECK OUT TAB STOP
POP    H          ;RESTORE COL POINTER

```

```

JZ     TABLOP     ;LOOP IF STOP NOT SET

```

```

XCHG
JMP    NEWDLH     ;SET NEW COL POSITION AND RETURN

```

```

TOFAR   LHLD     HPOS ;GO ALL THE WAY TO THE RIGHT

```

```

XCHG
LXI    H,MAXRGT

```

```

CALL   HLMDE
SHLD  DLHPOS

```

```

RET

```

```

DOAFF  LXI     H,DFRMLN ;MULTIPLY FORMS LENGTH BY 48

```



```

LXI    D,48
CALL   HLTDE
LXI    D,10
CALL   HLDDE      ;AND DIVIDE IT BY 10
PUSH   H          ;SAVE THIS RESULT
LHLD   VPOS      ;GET LOGICAL VERTICAL POSITION
XCHG

```

```

LHLD   DLVPOS
DAD    D
POP    D
PUSH   D          ;GET COPY OF FORMS LENGTH
CALL   HLDDE      ;HL MOD DE
XCHG

```

```

POP    D
XCHG
CALL   HLMDE
XCHG
LHLD   DLVPOS
DAD    D
SHLD   DLVPOS
JMP    PAPER

```

* NEGHL FORMS THE TWOS COMPLEMENT OF HL. *

```

NEGHL  MOV    A,H
        CMA
        MOV    H,A
        MOV    A,L
        CMA
        MOV    L,A
        INX   H
        RET

```

* HLMDE SUBTRACTS DE FROM HL AND RETURNS. *

```

HLMDE  XCHG
        CALL   NEGHL
        XCHG
        DAD    D
        RET

```

* HLCDE COMPARES HL WITH DE. ON RETURN THE Z FLAG IS SET IF *
* THEY ARE EQUAL, THE CARRY FLAG IS SET IF HL IS LESS THAN DE. *

```

HLCDE  MOV    A,H
        CMP    D
        RNZ
        MOV    A,L
        CMP    E
        RET

```

VR-1412

VISI-READ

 * DIVIDE THE NUMBER IN HL BY THE NUMBER IN DE. RETURN THE *
 * QUOTIENT IN HL AND THE REMAINDER IN DE. *

HLDDE MOV A,D ;START BY NEGATING DE AND
 CMA ; MOVING THE LEFT OPERAND TO BC
 MOV B,A

MOV A,E
 CMA
 MOV C,A

INX B
 MVI A,16 ;REPEAT COUNT IN REG A
 LXI D,0 ;INITIAL REMAINDER IS ZERO

DIV3 DCR A ;TEST IF DONE
 RM ;ALL DONE ?
 DAD H ;SHIFT RIGHT OPERAND TO THE LEFT

XCHG
 PUSH PSW ;SAVE CARRY
 DAD H ;SHIFT LEFT OPERAND TO THE LEFT

POP PSW
 JNC DIV1 ;DOES IT FIT ?
 INX H

DIV1 PUSH H
 DAD B
 JNC DIV2

XCHG
 INX H
 XTHL

DIV2 POP H
 JMP DIV3
 POP H

XCHG
 JMP DIV3

 * MULTIPLY THE CONTENTS OF HL BY THE CONTENTS OF DE. *

HLTDE MOV C,L
 MOV B,H

MULT LXI H,0
 MOV A,B
 ORA C

RZ
 MOV A,B
 ORA A

RAR
 MOV B,A
 MOV A,C

RAR
 MOV C,A
 CC DADDE

XCHG
 DAD H

VR-1412

VISI:RFAD

XCHG
 JMP MULT
 DADDE DAD D
 RET

 * THE ROUTINES BELOW ACTUALLY INTERFACE TO THE PRINTER,
 * CAUSING PAPER FEED, CARRIAGE, AND PRINT WHEEL MOTION.

CARRG LHL D DLHPOS ;CHECK FOR ANY ACCUMULATED MOTION
 MOV A,H
 ORA L
 RZ
 LHL D HPOS ;CHECK FOR TOO MUCH MOTION

XCHG
 LHL D DLHPOS
 DAD D
 MOV A,H
 ANA A
 JP LFTOK

LHL D HPOS
 CALL NEGHL
 SHLD DLHPOS
 LFTOK LHL D HPOS

XCHG
 LHL D DLHPOS
 DAD D
 LXI D,MAXRGT
 CALL HLCDE
 JC RGTOK
 LHL D HPOS ;OTHERWISE MOVE ONLY TO MAXRIGHT
 XCHG

LXI H,MAXRGT
 CALL HLMDE
 SHLD DLHPOS
 RGTOK LHL D HPOS ;UPDATE THE HORIZONTAL POSITION

XCHG
 LHL D DLHPOS
 DAD D
 SHLD HPOS
 LHL D DLHPOS ;CHECK IF REQUIRED MOTION IS TO THE LEFT
 MOV A,H
 ANA A
 MVI C,0

JP POSH
 CALL NEGHL
 MVI C,D11
 POSH

XCHG
 LXI H,0
 SHLD DLHPOS ;RESET THE HORIZONTAL INCREMENT
 XCHG
 MOV A,L
 ANI 1
 JZ NOHHLF ;NO HALF SPACES
 MOV A,C

VR-1412

VISI-READ

	ORI	D12	
	MOV	C,A	
NOHHLF	CALL	DIVID2	
	MOV	A,H	
	ANI	D9+D10	
	ORA	C	
	MOV	H,A	
	LXI	D,CRSTRD	
	JMP	CMND	
PAPR	LHLD	DLVPOS	;CHECK FOR ANY PAPER MOTION
	MOV	A,H	
	ORA	L	
	RZ		;NO MOTION
	MOV	A,H	
	ANA	A	
	MVI	C,0	
	JP	POSV	
POSV	CALL	NEGHL	
	MVI	C,D11	
	MOV	A,H	
	ANI	D9+D10	
	ORA	C	
	MOV	H,A	
	PUSH	H	;SAVE PAPER MOTION
	LHLD	VPOS	
	XCHG		
	LHLD	DLVPOS	;GET LOGICAL POSITION
	DAD	D	
	PUSH	H	;SAVE FOR NOW
	LXI	H,DFRMLN	;GET DEFAULT FORM LENGTH
	LXI	D,48	
	CALL	HLTDE	;MULTIPLY BY 48
	LXI	D,10	
	CALL	HLDDE	;DIVIDE BY 10
	POP	D	
	XCHG		
	CALL	HLDDE	;COMPUTE HL MOD DE
	XCHG		
	SHLD	VPOS	;SAVE NEW VERTICAL POSITION
	LXI	H,0	
	SHLD	DLVPOS	;RESET VERTICAL MOTION
	POP	H	
	LXI	D,PFSTRD	;PAPER FEED STROBE
	JMP	CMND	
WHEEL	PUSH	H	
	CALL	CARRG	;POSITION THE CARRIAGE FIRST
	CALL	PAPR	
	POP	H	
	LXI	D,PWSTRD	
CMND	LDA	GROUP	;GET GROUP BYTE
	OUT	GRPSEL	;SELECT GROUP ZERO
CMND0	IN	DAISY0	

```

ANA      D
JZ       CMND0
MOV      A,L           ;NEGATE LOW DATA BITS
CMA
MOV      L,A
MOV      A,H
ANI      D9+D10+D11+D12 ;MASK IN DATA BITS ONLY
CMA
IF       MULTR3       ;MASK OUT RIBBON LIFT BIT ON MULTI I/O
ANI      0FFH-RESTORE
ENDIF
MOV      H,A
MOV      A,L
OUT      DAIS11       ;OUTPUT LOW BITS
MOV      A,H
OUT      DAIS10       ;OUTPUT HIGH BITS
XRA      E           ;SLAP STROBE BITS IN
OUT      DAIS10
MOV      A,H         ;AND DROP STROBES BACK DOWN
OUT      DAIS10
RET
    
```

```

*****
* NEW LIST DEVICE STATUS ROUTINE. RETURNS 0FFH IF THE PRINTER *
* CAN EXCEPT ANOTHER CHARACTER, OTHERWISE IT RETURNS 0. *
*****
    
```

```

LISTST  LDA  GROUP      ;CHECK PRINTER INITIALIZED FLAG
        ANI  DENABLE
        RZ           ;0 = PRINTER NOT INITIALIZED
        LDA  GROUP      ;GET GROUP BYTE
        OUT  GRPSEL     ;SELECT GROUP ZERO
        LXI  D,PWSTRD
        IN   DAISY0
        ANA  D
        XRA  A
        RZ
        CMA
        RET
    
```

```

*****
* DYNAMIC DATA LOCATIONS USED BY THE SIMULATOR. *
*****
    
```

```

HMI     DW     0       ;HORIZONTAL MOTION INDEX. SET BY LINIT
                          ; AND ESCAPE SEQUENCES.
VMI     DW     0       ;VERTICAL MOTION INDEX. SET BY LINIT
                          ; AND ESCAPE SEQUENCES.
VPOS    DW     0       ;VERTICAL POSITION. SET BY PLATEN MOTION
DLVPOS  DW     0       ;DELTA VPOS. SET BY PLATEN MOTION
HPOS    DW     0       ;HORIZONTAL POSITION. SET BY CARRIAGE MOTION
DLHPOS  DW     0       ;DELTA HPOS. SET BY CARRIAGE MOTION
LMAR    DW     0       ;LEFT MARGIN
DIRFLG  DB     0       ;DIRECTION FLAG
GRHFLG  DB     0       ;GRAPHICS MODE FLAG
ESCFLG  DB     0       ;ESCAPE SEQUENCE IN PROGRESS FLAG
    
```

VR-1412

VISI-READ

TABSTP DS NUMTABS/8+1 ;TAB STOPS BIT ARRAY
 TABLEN EQU NUMTABS/8+1 ;LENGTH OF TABS ARRAY

ENDIF

 *
 * THE FOLLOWING ROUTINES ARE USED TO MAKE THE READER AND PUNCH *
 * DEVICES PEFORM I/O THROUGH THE CONSOLE. THE USER MAY PATCH *
 * HERE FOR THEIR PARTICULAR DEVICES. *
 *

E3A2 C30CE3 PUNCH JMP COUT

E3A5 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). *
 *

E3A8	218000	GOCPM	LXI	H,BUFF	;SET UP INITIAL DMA ADDRESS
E3AB	CD43E4		CALL	SETDMA	
E3AE	3EC3		MVI	A,(JMP)	;INITIALIZE JUMP TO WARM BOOT
E3B0	320000		STA	WBOT	
E3B3	320500		STA	ENTRY	;INITIALIZE JUMP TO BDOS
E3B6	2103E3		LXI	H,WBOOTE	;ADDRESS IN WARM BOOT JUMP
E3B9	220100		SHLD	WBOT+1	
E3BC	2106D5		LXI	H,BDOS+6	;ADDRESS IN BDOS JUMP
E3BF	220600		SHLD	ENTRY+1	
E3C2	AF		XRA	A	;A ← 0
E3C3	324AEE		STA	BUFSEC	;DISK JOCKEY BUFFER EMPTY
E3C6	323EE6		STA	BUFWRN	;SET BUFFER NOT DIRTY FLAG
E3C9	3A0400		LDA	CDISK	;JUMP TO CP/M WITH CURRENTLY SELECTED DISK IN C
E3CC	4F		MOV	C,A	
E3CD	3AFAE3		LDA	CWFLG	
E3D0	B7		ORA	A	
E3D1	11FCE3		LXI	D,COLDBEG	;BEGINNING OF INITIAL COMMAND
E3D4	3E01		MVI	A,COLDEND-COLDBEG+1	;LENGTH OF COMMAND
E3D6	CADEE3		JZ	CLDCMND	
E3D9	11FDE3		LXI	D,WARMBEG	
E3DC	3E01		MVI	A,WARMEND-WARMBEG+1	
E3DE	2108CD	CLDCMND	LXI	H,CCP+8	;COMMAND BUFFER
E3E1	3207CD		STA	CCP+7	
E3E4	47		MOV	B,A	
E3E5	CD06E7		CALL	MOVLOP	
E3E8	3AFAE3		LDA	CWFLG	
E3EB	B7		ORA	A	
E3EC	3AFBE3		LDA	AUTOFLG	
E3EF	CAF3E3		JZ	CLDBOT	
E3F2	1F		RAR		

VR-1412

VISI READ

```
E3F3 1F CLDBOT RAR
E3F4 DA00CD JC CCP
E3F7 C303CD JMP CCP+3 ;ENTER CP/M
```

```
E3FA 00 CWFLG DB 0 ;COLD/WARM BOOT FLAG
```

```
*****
*
* THE FOLLOWING BYTE DETERMINES IF AN INITIAL COMMAND IS TO BE
* GIVEN TO CP/M ON WARM OR COLD BOOTS. THE VALUE OF THE BYTE IS
* USED TO GIVE THE COMMAND TO CP/M:
*
* 0 = NEVER GIVE COMMAND.
* 1 = GIVE COMMAND ON COLD BOOTS ONLY.
* 2 = GIVE THE COMMAND ON WARM BOOTS ONLY.
* 3 = GIVE THE COMMAND ON WARM AND COLD BOOTS.
*****
```

```
E3FB 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.
*****
```

```
E3FC 00 COLDBEG DB '' ;COLD BOOT COMMAND GOES HERE
E3FD 00 COLDEND DB 0
E3FE 00 WARBEG DB '' ;WARM BOOT COMMAND GOES HERE
E3FF 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.
*****
```

```
E3FE 310001 WBOOT LXI SP,TPA ;SET UP STACK POINTER
E401 3E01 MVI A,1
E403 32FAE3 STA CWFLG ;SET COLD/WARM BOOT FLAG

E406 AF IF (MAXHD NE 0) AND FIRST ;SUPPLY WARM BOOT FROM HARD DISK ?
E407 4F XRA A
E408 2100CB MOV C,A
E409 2100CB LXI H,CCP-200H ;INITIAL DMA ADDRESS
```

VR-1412

VISI-READ

```

E40B E5      PUSH  H
E40C 3249E8  STA   HEAD
E40F 3E01    MVI   A,1
E411 F5      PUSH  PSW      ;SAVE FIRST SECTOR - 1
E412 CD1BE7  CALL  HDDRV     ;SELECT DRIVE A
E415 0E00    MVI   C,0
E417 CD3AE7  CALL  HDTRK     ;HOME THE DRIVE
E41A F1      WARMLOD POP  PSW      ;RESTORE SECTOR
E41B E1      POP   H      ;RESTORE DMA ADDRESS
E41C 3C      INR   A
E41D 322DE8  STA   HDSECTR
E420 FE0C    CPI   12      ;PAST BDOS ?
E422 CAABE3  JZ    GOCPM     ;YES, ALL DONE
E425 24      INR   H      ;UPDATE DMA ADDRESS
E426 24      INR   H
E427 22A4E7  SHLD  HDADD
E42A E5      PUSH  H
E42B F5      PUSH  PSW
E42C 01000A  WARMRD LXI   B,RETRIES*100H+0 ;RETRY COUNTER
E42F C5      WRMREAD PUSH B      ;SAVE THE RETRY COUNT
E430 CD8AE7  CALL  HDREAD    ;READ THE SECTOR
E433 C1      POP   B
E434 D21AE4  JNC   WARMLOD   ;TEST FOR ERROR
E437 05      DCR   B      ;UPDATE THE ERROR COUNT
E438 C22FE4  JNZ   WRMREAD   ;KEEP TRYING IF NOT TO MANY ERRORS
E43B 76      HLT
E43C 00      DB    0      ;TRY NOT TO SCREW UP DECISION CPU'S
    
```

ENDIF

```

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

```

IF (MAXFLOP NE 0) AND NOT FIRST ;SUPPLY WARM BOOT FROM 2D ?
MVI C,0
CALL DJSEL      ;SELECT DRIVE A
WRMFAIL CALL DJHOME ;TRACK 0, SINGLE DENSITY
JC WRMFAIL     ;LOOP IF ERROR
MVI C,0        ;SELECT SIDE 0
CALL DJSIDE
    
```

;THE NEXT BLOCK OF CODE RE-INITIALIZES
; THE WARM BOOT LOADER FOR TRACK 0.

```

MVI A,5-2      ;INITIALIZE THE SECTOR TO READ - 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
    
```

VR-1412

VSI-READ


```

JC      NOWRAP      ;SKIP IF NOT AT END OF TRACK
JNZ     T1BOOT      ;DONE WITH THIS TRACK
SUI     27-6        ;BACK UP TO SECTOR 6
NOWRAP  LXI         H,CCP-80H      ;MEMORY ADDRESS OF SECTOR - 100H
        SHLD        NEWDMA
        STA         NEWSEC      ;SAVE THE UPDATED SECTOR #
        MOV         C,A
        CALL        DJSEC       ;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        DJDMA       ;SET UP THE NEW DMA ADDRESS
WRMFRED LXI         B,RETRIES*100H+0;MAXIMUM # OF ERRORS, TRACK #
        PUSH        B
        CALL        DJTRK      ;SET UP THE PROPER TRACK
        CALL        DJREAD     ;READ THE SECTOR
        POP         B
        JNC         T0BOOT     ;CONTINUE IF NO ERROR
        DCR         B
        JNZ         WRMFRED    ;KEEP TRYING IF ERROR
        JMP         DJERR      ;TO 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        DJTRK
        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
        JMP         GOCPM      ;ALL DONE, DO LAST INITS...

```

```

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

```

ENDIF

```

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

```

```

E43D 60 SETSEC MOV H,B
E43E 69 MOV L,C
E43F 2242EE SHLD CPMSEC
E442 C9 DONOP RET ;NULL SINGLE.COM HOOKUP FOR NO FLOPPIES

```

```

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

```

```

E443 60 SETDMA MOV H,B ;HL <- BC
E444 69 MOV L,C
E445 221EE6 SHLD CPMDMA ;CP/M DMA ADDRESS
E448 C9 RET

```

```

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

```

```

E449 0E00 HOME MVI C,0 ;TRACK TO SEEK TO

```

```

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

```

VR-1412

VIS-READ

```

E44B 79      SETTRK  MOV      A,C          ;A <- TRACK #
E44C 3245EE  STA      CPMTRK      ;CP/M TRACK #
E44F C9      RET
    
```

```

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

```

E450 3A44EE  SECTRAN LDA      (MAXHD NE 0) AND (MAXFLOP NE 0) ;BOTH TYPES ?
                        CPMDRV      ;GET THE DRIVE NUMBER
    
```

```

E453 FE03    IF      FIRST
E455 DA87E4  CPI      MAXHD*LOGDSK ;OVER THE # OF HARD DISKS ?
                        JC      TRANHD
    
```

```

                        ELSE
                        CPI      MAXFLOP ;OVER THE # OF FLOPPIES ?
                        JNC     TRANHD
                        ENDIF
                        ENDIF
    
```

```

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

```

                        SECTRAN EQU      $
                        ENDIF
    
```

```

E458 03      IF      MAXFLOP NE 0 ;FLOPPY TRANSLATION
E459 D5      TRANFP  INX      B
E45A C5      PUSH     D ;SAVE TABLE ADDRESS
E45B CDA2E5  PUSH     B ;SAVE SECTOR #
                        CALL    GETDPB ;GET DPB ADDRESS INTO HL
E45E 7E      MOV      A,M ;GET # OF CP/M SECTORS/TRACK
E45F B7      ORA      A ;CLEAR CARY
E460 1F      RAR      ;DIVIDE BY TWO
    
```

```

E461 91      SUB      C
E462 F5      PUSH     PSW ;SAVE ADJUSTED SECTOR
E463 FA6FE4  JM      SIDETWO
    
```

```

E466 F1      SIDEA  POP     PSW ;DISCARD ADJUSTED SECTOR
E467 C1      POP     B ;RESTORE SECTOR REQUESTED
E468 D1      POP     D ;RESTOR ADDRESS OF XLT TABLE
E469 EB      SIDEONE XCHG ;HL <- &(TRANSLATION TABLE)
E46A 09      DAD     B ;BC = OFFSET INTO TABLE
E46B 6E      MOV     L,M ;HL <- PHYSICAL SECTOR
    
```

```

E46C 2600    MVI     H,0
E46E C9      RET
    
```

```

E46F 010F00  SIDETWO LXI     B,15 ;OFFSET TO SIDE BIT
E472 09      DAD     B
E473 7E      MOV     A,M
    
```

```

E474 E608    ANI     8 ;TEST FOR DOUBLE SIDED
E476 CA66E4  JZ      SIDEA ;MEDIA IS ONLY SINGLE SIDED
E479 F1      POP     PSW ;RETRIEVE ADJUSTED SECTOR
    
```

```

E47A C1      POP     B
E47B 2F      CMA     ;MAKE SECTOR REQUEST POSITIVE
    
```

VR-1412

```

E47C 3C      INR      A
E47D 4F      MOV      C,A          ;MAKE NEW SECTOR THE REQUESTED SECTOR
E47E D1      POP      D
E47F CD69E4  CALL     SIDEONE
E482 3E80    MVI      A,80H          ;SIDE TWO BIT
E484 B4      ORA      H              ;      AND SECTOR
E485 67      MOV      H,A
E486 C9      RET
                ENDIF
    
```

```

E487 60      IF      MAXHD NE 0          ;HARD DISK TRANSLATION ROUTINE
                TRANHD
E488 69      MOV      H,B
E489 23      MOV      L,C
E48A C9      INX      H
                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.
*
*****
    
```

```

E48B 79      SETDRV  MOV      A,C          ;SAVE THE DRIVE #
E48C 3244EE  STA      CPMDRV
E48F FE07    CPI      MAXFLOP+(MAXHD*LOGDSK) ;CHECK FOR A VALID DRIVE #
E491 D293E5  JNC     ZRET          ;ILLEGAL DRIVE #
E494 7B      MOV      A,E          ;TEST IF DRIVE EVER LOGGED IN BEFORE
E495 E601    ANI      1
E497 C27AE5  JNZ     SETDRV1      ;BIT 0 OF E = 0 -> NEVER SELECTED BEFORE
    
```

```

E49A 3A44EE  IF      (MAXHD NE 0) AND (MAXFLOP NE 0) ;BOTH TYPES ?
                LDA     CPMDRV          ;GET THE DRIVE NUMBER
    
```

```

E49D FE03    IF      FIRST
E49F DA4DE5  CPI      MAXHD*LOGDSK          ;OVER THE # OF HARD DISKS ?
E4A2 D603    JC      DRVHD
                SUI     MAXHD*LOGDSK
                ELSE
E4A2 D603    CPI      MAXFLOP          ;OVER THE # OF FLOPPIES ?
                JNC     SUBFP
                ENDIF
                ENDIF
    
```

```

E4A4 4F      IF      (MAXFLOP NE 0) AND FIRST
E4A5 3E00    MOV      C,A          ;SAVE DRIVE #
E4A6 =      MVI      A,0          ;HAVE THE FLOPPIES BEEN ACCESSED YET ?
                FLOPFLG EQU $-1
    
```

```

E4A7 A7 ANA A
E4A8 C2F8E4 JNZ FLOPOK
E4AB 0611 MVI B,17 ;FLOPPIES HAVN'T BEEN ACCESSED
E4AD 2100F8 LXI H,DJBOOT ;CHECK IF 2D CONTROLLER IS INSTALLED
E4B0 3EC3 MVI A,(JMP)
E4B2 BE CLOPP CMP M
E4B3 C293E5 JNZ ZRET
E4B6 23 INX H
E4B7 23 INX H
E4B8 23 INX H
E4B9 05 DCR B
E4BA C2B2E4 JNZ CLOPP
E4BD 11D5E4 LXI D,DJINIT ;INITIALIZATION SEQUENCE
E4C0 21E2FF LXI H,ORIGIN+7E2H ;LOAD ADDRESS
E4C3 061E MVI B,30 ;BYTE COUNT
E4C5 CD06E7 CALL MOVLOP
E4C8 3EFF MVI A,0FFH ;START 1791
E4CA 32F9FB STA DREG
E4CD 3ED0 MVI A,CLRCMD ;1791 RESET
E4CF 32FCFB STA CMDREG
E4D2 C3F3E4 JMP DJNEXT

E4D5 0000001800DJINIT DB 0, 0, 0, 18H, 0, 0, 8, 0, 7EH, 0, 8, 0, 9, 0FFH, 9, 0FFH
E4E5 09FF09FF09 DB 9, 0FFH, 9, 0FFH, 9, 0, 1, 0, 0, 0, 0, 0, 0, 0

E4F3 3E01 DJNEXT MVI A,1 ;SAVE 2D INITIALIZED FLAG
E4F5 32A6E4 STA FLOPFLG

ENDIF
IF MAXFLOP NE 0
E4F8 210100 FLOPOK LXI H,1 ;SELECT SECTOR 1 OF TRACK 1
E4FB 2246EE SHLD TRUESEC
E4FE 3E01 MVI A,1
E500 3245EE STA CPMTRK
E503 CDD0E6 CALL FILL ;FLUSH BUFFER AND REFILL
E506 DA93E5 JC ZRET ;TEST FOR ERROR RETURN
E509 CD27F8 CALL DJSTAT ;GET STATUS ON CURRENT DRIVE
E50C E60C ANI 0CH ;STRIP OFF UNWANTED BITS
E50E F5 PUSH PSW ;USED TO SELECT A DPB
E50F 1F RAR

E510 21BBE5 LXI H,XLTS ;TABLE OF XLT ADDRESSES
E513 5F MOV E,A
E514 1600 MVI D,0
E516 19 DAD D
E517 E5 PUSH H ;SAVE POINTER TO PROPER XLT
E518 CDA2E5 CALL GETDPB ;GET DPH POINTER INTO DE
E51B EB XCHG ;
E51C D1 POP D
E51D 0602 MVI B,2 ;NUMBER OF BYTES TO MOVE
E51F CD06E7 CALL MOVLOP ;MOVE THE ADDRESS OF XLT
E522 110800 LXI D,8 ;OFFSET TO DPB POINTER
E525 19 DAD D ;HL <- &DPH.DPB
E526 E5 PUSH H
E527 2A07F8 LHLD ORIGIN+7 ;GET ADDRESS OF DJ TERMINAL OUT ROUTINE
E52A 23 INX H ;BUMP TO LOOK AT ADDRESS OF
; UART STATUS LOCATION

E52B 7E MOV A,M

```

VR-1412

VS-1-READ

```

E52C EE03      XRI      3          ;ADJUST FOR PROPER REV DJ
E52E 6F        MOV      L,A
E52F 26FB      MVI      H,(ORIGIN+300H)/100H
E531 7E        MOV      A,M
E532 E608      ANI      DBLSID      ;CHECK DOUBLE SIDED BIT
E534 1122E9    LXI      D,DPB128S ;BASE FOR SINGLE SIDED DPB'S
E537 C23DE5    JNZ      SIDEOK
E53A 1162E9    LXI      D,DPB128D      ;BASE OF DOUBLE SIDED DPB'S
E53D EB        SIDEOK  XCHG      ;HL <- DBP BASE, DE <- &DPH.DPB
E53E D1        POP      D          ;RESTORE DE (POINTER INTO DPH)
E53F F1        POP      PSW       ;OFFSET TO CORRECT DPB
E540 17        RAL
E541 17        RAL
E542 4F        MOV      C,A
E543 0600      MVI      B,0
E545 09        DAD      B
E546 EB        XCHG      ;PUT DPB ADDRESS IN DPH
E547 73        MOV      M,E
E548 23        INX      H
E549 72        MOV      M,D
                ENDIF

E54A C37AE5    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

E54D CD99E5    DRVHD  CALL     DIVLOG ;DIVIDE BY LOGICAL DISKS PER DRIVE
E550 79        MOV      A,C
E551 324FE8    STA      HDDISK
E554 CD3DE8    CALL     DRVPTR
E557 7E        MOV      A,M
E558 3C        INR      A
E559 C27AE5    JNZ      SETDRV1
E55C F6FC      ORI      NULL ;SELECT DRIVE
E55E D352      OUT     HDFUNC
E560 3E05      MVI      A,SCENBL ;ENABLE THE CONTROLLER
E562 D350      OUT     HDCNTL
E564 0EEF      MVI      C,239 ;WAIT APPROX 2 MINUTES FOR DISK TO READY
E566 210000    LXI      H,0
E569 2B        TDELAY DCX      H
E56A 7C        MOV      A,H
E56B B5        ORA      L
E56C CC97E5    CZ      DCRC
E56F C8        RZ
E570 DB50      IN      HDSTAT ;TEST IF READY YET
E572 E620      ANI      DRVRDY
E574 C269E5    JNZ      TDELAY

                IF      NOT FUJITSU
                LXI      H,0 ;TIME ONE REVOLUTION OF THE DRIVE
                MVI      C,INDEX
                IN      HDSTAT
                ANA      C
    
```

VR-1412

VS1-READ

```

      MOV      B,A          ;SAVE CURRENT INDEX LEVEL IN B
      INDX1   IN      HDSTAT
      ANA     C
      CMP     B          ;LOOP UTIL INDEX LEVEL CHANGES
      JZ      INDX1
      INDX2   INX     H
      IN      HDSTAT     ;START COUNTING UNTIL INDEX RETURNS TO
      ANA     C          ;
      CMP     B
      JNZ     INDX2
      IF      M10        ;MEMOREX M10'S HAVE 40 MS HEAD SETTLE
      DAD     H
      ENDIF
      IF      M26        ;SHUGART M26'S HAVE 30 MS HEAD SETTLE
      XRA     A
      MOV     A,H
      RAR
      MOV     D,A
      MOV     A,L
      RAR
      MOV     E,A
      DAD     D
      ENDIF
      SHLD    SETTLE     ;SAVE THE COUNT FOR TIMEOUT DELAY
      ENDIF
E577 CD2CE7      CALL    HDHOME
      ENDIF
E57A CDA2E5      SETDRV1 CALL    GETDPB          ;GET ADDRESS OF DPB IN HL
E57D 010F00      LXI     B,15          ;OFFSET TO SECTOR SIZE
E580 09          DAD     B
E581 7E          MOV     A,M          ;GET SECTOR SIZE
E582 E607        ANI     7H
E584 32CFE5      STA     SECSIZ
E587 7E          MOV     A,M
E588 1F          RAR
E589 1F          RAR
E58A 1F          RAR
E58B 1F          RAR
E58C E60F        ANI     0FH
E58E 320DE6      STA     SECPSEC
E591 EB          XCHG          ;HL <- DPH
E592 C9          RET
E593 210000      ZRET    LXI     H,0          ;SELDRV ERROR EXIT
E596 C9          RET
      IF      MAXHD NE 0
E597 0D          DCRC    DCR     C          ;CONDITIONAL DECREMENT C ROUTINE
E598 C9          RET
E599 0E00        DIVLOG MVI     C,0
E59B D603        DIVLOGX SUI    LOGDSK

```

```

E59D D8      RC
E59E 0C      INR      C
E59F C39BE5  JMP      DIVLOGX
                ENDIF

```

```

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

```

```

E5A2 3A44EE  GETDPB  LDA      CPMDRV
E5A5 6F      MOV      L,A          ;FORM OFFSET
E5A6 2600    MVI      H,0
E5A8 29      DAD      H
E5A9 29      DAD      H
E5AA 29      DAD      H
E5AB 29      DAD      H
E5AC 11D2E9  LXI      D,DPBASE   ;BASE OF DPH'S
E5AF 19      DAD      D
E5B0 E5      PUSH     H          ;SAVE ADDRESS OF DPH
E5B1 110A00  LXI      D,10       ;OFFSET TO DPB
E5B4 19      DAD      D
E5B5 7E      MOV      A,M          ;GET LOW BYTE OF DPB ADDRESS
E5B6 23      INX      H
E5B7 66      MOV      H,M          ;GET LOW BYTE OF DPB
E5B8 6F      MOV      L,A
E5B9 D1      POP      D
E5BA C9      RET

```

```

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

```

```

                IF      MAXFLOP NE 0
E5BB 54E8    XLTS      DW      XLT128   ;XLT FOR 128 BYTE SECTORS
E5BD 6FE8    DW      XLT256   ;XLT FOR 256 BYTE SECTORS
E5BF A4E8    DW      XLT512   ;XLT FOR 512 BYTE SECTORS
E5C1 E1E8    DW      XLT124   ;XLT FOR 1024 BYTE SECTORS
                ENDIF

```

```

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

```

VR-1412

MSI READ


```

E5C3 79      WRITE MOV      A,C      ;SAVE WRITE COMMAND TYPE
E5C4 3235E6      STA      WRITYP
E5C7 3E01      MVI      A,1      ;SET WRITE COMMAND
E5C9 06      DB      (MVI) OR (B*8) ;THIS "MVI B" INSTRUCTION CAUSES
;          THE FOLLOWING "XRA A" TO
;          BE SKIPPED OVER.
    
```

```

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

```

E5CA AF      READ      XRA      A      ;SET THE COMMAND TYPE TO READ
E5CB 3221E6      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.
*
*****
    
```

```

E5CE 0600      REDWRT MVI      B,0      ;THE 0 IS MODIFIED TO CONTAIN THE LOG2
E5CF =          SECSIZ EQU      $-1      ; OF THE PHYSICAL SECTOR SIZE/128
;          ON THE CURRENTLY SELECTED DISK.
E5D0 2A42EE      LHLD     CPMSEC      ;GET THE DESIRED CP/M SECTOR #
E5D3 7C          MOV      A,H
E5D4 E680      ANI      80H      ;SAVE ONLY THE SIDE BIT
E5D6 4F          MOV      C,A      ;REMEMBER THE SIDE
E5D7 7C          MOV      A,H
E5D8 E67F      ANI      7FH      ;FORGET THE SIDE BIT
E5DA 67          MOV      H,A
E5DB 2B          DCX      H      ;TEMPORARY ADJUSTMENT
E5DC 05          DIVLOOP DCR      B      ;UPDATE REPEAT COUNT
E5DD CAEAE5      JZ      DIVDONE
E5E0 B7          ORA      A
E5E1 7C          MOV      A,H
E5E2 1F          RAR
E5E3 67          MOV      H,A
E5E4 7D          MOV      A,L
E5E5 1F          RAR      ;DIVIDE THE CP/M SECTOR # BY THE SIZE
;          OF THE PHYSICAL SECTORS
E5E6 6F          MOV      L,A
E5E7 C3DCE5      JMP      DIVLOOP
E5EA 23          DIVDONE INX      H
    
```

VR-1412

VISI-READ

```

E5EB 7C      MOV      A,H
E5EC B1      ORA      C          ;RESTORE THE SIDE BIT
E5ED 67      MOV      H,A
E5EE 2246EE  SHLD    TRUESEC      ;SAVE THE PHYSICAL SECTOR NUMBER
E5F1 2144EE  LXI     H,CPMDRV      ;POINTER TO DESIRED DRIVE,TRACK, AND SECTOR
E5F4 1148EE  LXI     D,BUFDRV      ;POINTER TO BUFFER DRIVE,TRACK, AND SECTOR
E5F7 0605      MVI     B,5          ;COUNT LOOP
E5F9 05      DTSLOP DCR      B          ;TEST IF DONE WITH COMPARE
E5FA CA0BE6  JZ      MOVE         ;YES, MATCH. GO MOVE THE DATA
E5FD 1A      LDAX   D          ;GET A BYTE TO COMPARE
E5FE BE      CMP     M          ;TEST FOR MATCH
E5FF 23      INX    H          ;BUMP POINTERS TO NEXT DATA ITEM
E600 13      INX    D
E601 CAF9E5  JZ      DTSLOP      ;MATCH, CONTINUE TESTING
    
```

```

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

PE LIMIT EQU \$

```

E604 CDD0E6  CALL   FILL          ;FILL THE BUFFER WITH CORRECT PHYSICAL SECTOR
E607 D8      RC          ;NO GOOD, RETURN WITH ERROR INDICATION
    
```

```

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

```

E608 3A42EE  MOVE   LDA      CPMSEC      ;GET THE CP/M SECTOR TO TRANSFER
E60B 3D      DCR     A          ;ADJUST TO PROPER SECTOR IN BUFFER
E60C E600    ANI     0          ;STRIP OFF HIGH ORDERED BITS
E60D =      SECPSEC EQU    $-1      ;THE 0 IS MODIFIED TO REPRESENT THE # OF
; CP/M SECTORS PER PHYSICAL SECTORS
E60E 6F      MOV     L,A          ;PUT INTO HL
E60F 2600    MVI     H,0
E611 29      DAD     H          ;FORM OFFSET INTO BUFFER
E612 29      DAD     H
E613 29      DAD     H
E614 29      DAD     H
E615 29      DAD     H
E616 29      DAD     H
E617 29      DAD     H
E618 1142EA LXI     D,BUFFER      ;BEGINNING ADDRESS OF BUFFER
E61B 19      DAD     D          ;FORM BEGINNING ADDRESS OF SECTGR TO TRANSFER
E61C EB      XCHG
E61D 210000 LXI     H,0          ;GET DMA ADDRESS, THE 0 IS MODIFIED T/
; CONTAIN THE DMA ADDRESS
E61E =      CPMDMA EQU    $-2
E620 3E00    MVI     A,0          ;THE ZERO GETS MODIFIED TO CONTAIN
; A ZERO IF A READ, OR A 1 IF WRITE
E621 =      RDWR   EQU    $-1
E622 A7      ANA     A          ;TEST WHICH KIND OF OPERATION
    
```

VR-1412
VISI READ

```

E623 C22BE6 JNZ INTO ;TRANSFER DATA INTO THE BUFFER
E626 CD04E7 OUTOF CALL MOVER
E629 AF XRA A
E62A C9 RET
    
```

```

E62B EB INTO XCHG ;
E62C CD04E7 CALL MOVER ;MOVE THE DATA, HL = DESTINATION
; DE = SOURCE
    
```

```

E62F 3E01 MVI A,1
E631 323EE6 STA BUFWRN ;SET BUFFER WRITTEN INTO FLAG
E634 3E00 MVI A,0 ;CHECK FOR DIRECTORY WRITE
E635 = WRITTP EQU $-1
    
```

```

E636 3D DCR A
E637 3E00 MVI A,0
E639 3235E6 STA WRITTP ;SET NO DIRECTORY WRITE
E63C C0 RNZ ;NO ERROR EXIT
    
```

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

```

E63D 3E00 FLUSH MVI A,0 ;THE 0 IS MODIFIED TO REFLECT IF
; THE BUFFER HAS BEEN WRITTEN INTO
    
```

```

E63E = BUFWRN EQU $-1
E63F A7 ANA A ;TEST IF WRITTEN INTO
E640 C8 RZ ;NOT WRITTEN, ALL DONE
    
```

```

E641 2118F8 IF (MAXHD NE 0) AND (MAXFLOP NE 0)
E644 11BFE7 LXI H,DJWRITE ;WRITE OPERATION FOR DISK JOCKEY
E647 CD13E7 LXI D,HDWRITE ;WRITE OPERATION FOR HARD DISK
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.
*

```

E64A F3 PREP DI ;RESET INTERRUPTS
E64B AF XRA A ;RESET BUFFER WRITTEN FLAG
E64C 323EE6 STA BUFWRN
E64F 22B1E6 SHLD RETRYOP ;SET UP THE READ/WRITE OPERATION
E652 060A MVI B,RETRIES ;MAXIMUM NUMBER OF RETRIES TO ATTEMPT
    
```

VR-1412

VISI-READ

```

E654 C5      RETRYLP PUSH      B          ;SAVE THE RETRY COUNT
E655 3A48EE  LDA              BUFDRV    ;GET DRIVE NUMBER INVOLVED IN THE OPERATION

```

```

E658 FE03   IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E65A DA5FE6 IF      FIRST
E65D D603   CPI      MAXHD*LOGDSK
            JC      NOADJST
            SUI    MAXHD*LOGDSK
            ELSE

```

```

            CPI    MAXFLOP
            JC      NOADJST
            SUI    MAXFLOP
            ENDIF

```

```

E65F 4F      NOADJST MOV      C,A
E660 2133E3 LXI      H,DJDRV    ;SELECT DRIVE
E663 111BE7 LXI      D,HDRV
E666 CD0FE7 CALL     DECIDGO

```

```

            ELSE
            MOV      C,A
            IF      MAXHD NE 0

```

```

            CALL    HDDRV
            ENDIF

```

```

            IF      MAXFLOP NE 0
            CALL    DJDRV    ;SELECT THE DRIVE
            ENDIF
            ENDIF

```

```

E669 3A49EE LDA      BUFTRK
E66C A7      ANA      A          ;TEST FOR TRACK ZERO
E66D 4F      MOV      C,A
E66E C5      PUSH     B

```

```

E66F 2109F8 IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E672 112CE7 LXI      H,DJHOME
E675 CC0FE7 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

```

```

E678 C1      POP      B          ;RESTORE TRACK #

```

```

E679 210CF8 IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E67C 113AE7 LXI      H,DJTRK
E67F CD0FE7 LXI      D,HDTRK
            CALL    DECIDGO

```

```

            ELSE
            IF      MAXHD NE 0

```

```

            CALL    HDTRK
            ENDIF

```

```

IF      MAXFLOP NE 0
CALL    DJTRK          ;SEEK TO PROPER TRACK
ENDIF

```

```

ENDIF

```

```

E682 2A4AEE  LHL D  BUFSEC
E685 7C      MOV   A,H          ;GET SECTOR INVOLVED IN OPERATION
E686 07      RLC           ;BIT 0 OF A EQUALS SIDE #
E687 E601    ANI   1          ;STRIP OFF UNNECESSARY BITS
E689 4F      MOV   C,A          ;C <- SIDE #

```

```

IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E68A 2130F8  LXI   H,DJSIDE
E68D 1163E7  LXI   D,HDSIDE
E690 CD0FE7  CALL  DECIDGO

```

```

ELSE
IF      MAXHD NE 0
CALL    HDSIDE
ENDIF

```

```

IF      MAXFLOP NE 0
CALL    DJSIDE          ;SELECT THE SIDE
ENDIF

```

```

ENDIF

```

```

E693 2A4AEE  LHL D  BUFSEC
E696 7C      MOV   A,H
E697 E67F    ANI   7FH          ;STRIP OFF SIDE BIT
E699 47      MOV   B,A          ;C <- SECTOR #
E69A 4D      MOV   C,L

```

```

IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E69B 210FF8  LXI   H,DJSEC
E69E 116CE7  LXI   D,HDSEC
E6A1 CD0FE7  CALL  DECIDGO

```

```

ELSE
IF      MAXHD NE 0
CALL    HDSEC
ENDIF

```

```

IF      MAXFLOP NE 0
CALL    DJSEC          ;SELECT THE SIDE
ENDIF

```

```

ENDIF

```

```

E6A4 0142EA  LXI   B,BUFFER      ;SET THE DMA ADDRESS

```

```

IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E6A7 2112F8  LXI   H,DJDMA
E6AA 115EE7  LXI   D,HDDMA
E6AD CD0FE7  CALL  DECIDGO

```

```

ELSE
IF      MAXHD NE 0
CALL    HDDMA
ENDIF

```

```

IF      MAXFLOP NE 0
CALL    DJDMA          ;SELECT THE SIDE
ENDIF

```

```

ENDIF

```

ENDIF

```

E6B0 CD0000 CALL 0 ;GET OPERATION ADDRESS
E6B1 = RETRYOP EQU $-2
E6B3 C1 POP B ;RESTORE THE RETRY COUNTER
E6B4 3E00 MVI A,0 ;NO ERROR EXIT STATUS
E6B6 D0 RNC ;RETURN NO ERROR
E6B7 05 DCR B ;UPDATE THE RETRY COUNTER
E6B8 37 STC ;ASSUME RETRY COUNT EXPIRED
E6B9 3EFF MVI A,0FFH ;ERROR RETURN
E6BB C8 RZ ;RETURN SAD NEWS
E6BC 78 MOV A,E
E6BD FE05 CPI RETRIES/2 ;RESEEK AFTER HALF RETRIES DONE
E6BF C254E6 JNZ RETRYLP ;TRY AGAIN
E6C2 C5 PUSH B

```

```

E6C3 2109F8 IF (MAXHD NE 0) AND (MAXFLOP NE 0)
E6C6 112CE7 LXI H,DJHOME
E6C9 CC0FE7 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

```

```

E6CC C1 POP B
E6CD C354E6 JMP RETRYLP ;TRY AGAIN

```

```

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

```

```

E6D0 CD3DE6 FILL CALL FLUSH ;FLUSH BUFFER FIRST
E6D3 D8 RC ;CHECK FOR ERROR
E6D4 1144EE LXI D,CPMDRV ;UPDATE THE DRIVE, TRACK, AND SECTOR
E6D7 2148EE LXI H,BUFDRV
E6DA 0604 MVI B,4 ;NUMBER OF BYTES TO MOVE
E6DC CD06E7 CALL MOVLOP ;COPY THE DATA

```

```

E6DF 3A21E6 LDA RDWR
E6E2 A7 ANA A
E6E3 CAF8E6 JZ FREAD
E6E6 3A35E6 LDA WRITTP
E6E9 3D DCR A
E6EA 3D DCR A
E6EB C8 RZ
E6EC CDA2E5 CALL GETDPB
E6EF 110F00 LXI D,15
E6F2 19 DAD D
E6F3 7E MOV A,M

```

VR-1412

VIS-READ

E6F4 E603 ANI 3
 E6F6 3D DCR A
 E6F7 C8 RZ

E6F8 = FREAD EQU \$
 IF (MAXHD NE 0) AND (MAXFLOP NE 0)
 E6F8 2115F8 LXI H,DJREAD
 E6FB 118AE7 LXI D,HDREAD
 E6FE CD13E7 CALL DECIDE

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

E701 C34AE6 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.

E704 0680 MOVER MVI B,128 ;LENGTH OF TRANSFER
 E706 1A MOVLOP LDAX D ;GET A BTE OF SOURCE
 E707 77 MOV M,A ;MOVE IT
 E708 13 INX D ;BUMP POINTERS
 E709 23 INX H
 E70A 05 DCR B ;UPDATE COUNTER
 E70B C206E7 JNZ MOVLOP ;CONTINUE MOVING UNTIL DONE
 E70E C9 RET

 * ROUTINES TO DECIDE WHICH CONTROLLER TO USE.

E70F CD13E7 DECIDGO CALL DECIDE ;WHICH CONTROLLER ?
 E712 E9 PCHL
 ENDIF

E713 3A48EE DECIDE IF (MAXHD NE 0) AND (MAXFLOP NE 0)
 LDA BUFDRV ;GET PROPER ROUTINE INTO H&L, BASED
 IF FIRST ; ON CURRENTLY SELECTED DRIVE
 E716 FE03 CPI MAXHD*LOGDSK
 E718 D0 RNC
 ELSE
 CPI MAXFLOP
 RC
 ENDIF

VR-1412
 VISI-READ

E719 EB XCHG
 E71A C9 RET
 ENDIF

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

		IF	MAXHD NE 0	
E71B 79	HDDRV	MOV	A,C	;SELECT HARD DISK DRIVE
E71C CD99E5		CALL	DIVLOG	;GET THE PHYSICAL DRIVE #
E71F 79		MOV	A,C	
E720 324FE8		STA	HDDISK	;SELECT THE DRIVE
E723 F6FC		ORI	NULL	
E725 D352		OUT	HDFUNC	
E727 3E0F		MVI	A,WENABL	
E729 D350		OUT	HDCNTL	
E72B C9		RET		

E72C CD3DE8	HDHOME	CALL	DRVPTR	
E72F 3600		MVI	M,0	;SET TRACK TO ZERO
E731 DB50		IN	HDSTAT	;TEST STATUS
E733 E601		ANI	TKZERO	;AT TRACK ZERO ?
E735 C8		RZ		;YES

		IF	NOT FUJITSU	
	STEPO	IN	HDSTAT	;TEST STATUS
		ANI	TKZERO	;AT TRACK ZERO ?
		JZ	DELAY	
		MVI	A,1	

		STC		
		CALL	ACCOK	;TAKE ONE STEP OUT
		JMP	STEPO	

ELSE

E736 AF		XRA	A	
E737 C34BE7		JMP	ACCOK	
		ENDIF		

		IF	NOT FUJITSU	
	DELAY	LXI	H,0	;GET DELAY
	SETTLE	EQU	\$-2	
	DELOOP	DCX	H	;WAIT 20MS
		MOV	A,H	

		ORA	L	
		INX	H	
		DCX	H	
		JNZ	DELOOP	
		RET		
		ENDIF		

E73A CD3DE8	HDTRK	CALL	DRVPTR	;GET POINTER TO CURRENT TRACK
-------------	-------	------	--------	-------------------------------

VR-1412

VISI-READ

E73D 5E MOV E,M ;GET CURRENT TRACK
 E73E 71 MOV M,C ;UPDATE THE TRACK
 E73F 7B MOV A,E ;NEED TO SEEK AT ALL ?

E740 91 SUB C
 E741 C8 RZ
 E742 3F CMC ;GET CARRY INTO DIRECTION

E743 DA48E7 JC HDTRK2
 E746 2F CMA
 E747 3C INR A

E748 C34BE7 HDTRK2 IF FUJITSU
 JMP ACCOK
 ELSE
 HDTRK2 CALL ACCOK
 JMP DELAY
 ENDIF

E74B 47 ACCOK MOV B,A ;PREP FOR BUILD
 E74C CD48E8 CALL BUILD

E74F E6FB SLOOP ANI NSTEP ;GET STEP PULSE LOW
 E751 D352 OUT HDFUNC ;OUTPUT LOW STEP LINE
 E753 F604 ORI PSTEP ;SET STEP LINE HIGH
 E755 D352 OUT HDFUNC ;OUTPUT HIGH STEP LINE
 E757 05 DCR B ;UPDATE REPEAT COUNT
 E758 C24FE7 JNZ SLOOP ;KEEP GOING THE REQUIRED # OF TRACKS
 E75B C364E7 JMP WSDONE

E75E 60 HDDMA MOV H,B ;SAVE THE DMA ADDRESS

E75F 69 MOV L,C
 E760 22A4E7 SHLD HDADD
 E763 = HDSIDE EQU \$
 E763 C9 RET

E764 DB50 WSDONE IN HDSTAT ;WAIT FOR SEEK COMPLETE TO FINISH
 E766 E604 ANI COMPLT
 E768 CA64E7 JZ WSDONE
 E76B C9 RET

HDSEC IF M26
 MVI A,01FH ;FOR COMPATIBILITY WITH CBIOS REV 2.3, 2.4

ANA C
 CZ GETSPT
 STA HDSECTR
 MVI A,0E0H
 ANA C
 RLC

RLC
 RLC
 STA HEAD
 GETSPT MVI A,HDSPT
 RET

ELSE
 E76C 79 HDSEC MOV A,C
 E76D CD81E7 CALL DIVSPT
 E770 C615 ADI HDSPT

VR-1412

```

E772 A7      ANA      A
E773 CC7DE7  CZ        GETSPT
E776 322DE8  STA        HDSECTR
E779 79      MOV        A,C
E77A 3249E8  STA        HEAD
E77D 3E15    GETSPT  MVI        A,HDSPT
E77F 0D      DCR        C
E780 C9      RET

```

```

E781 0E00    DIVSPT  MVI        C,0
E783 D615    DIVSPTX SUI        HDSPT
E785 D8      RC
E786 0C      INR        C
E787 C383E7  JMP        DIVSPTX
                ENDIF

```

```

E78A CD08E8  HDREAD  CALL        HDPREP
E78D D8      RC
E78E AF      XRA        A
E78F D351    OUT        HDCMND
E791 2F      CMA

```

```

E792 D353    OUT        HDDATA
E794 D353    OUT        HDDATA
E796 3E01    MVI        A,RSECT      ;READ SECTOR COMMAND
E798 D351    OUT        HDCMND
E79A CDEEE7  CALL        PROCESS
E79D D8      RC

```

```

E79E AF      XRA        A
E79F D351    OUT        HDCMND
E7A1 0680    MVI        B,SECLN/4
E7A3 210000  LXI        H,0

```

```

E7A4 =      HDADD  EQU        $-2
E7A6 DB53    IN         HDDATA
E7A8 DB53    IN         HDDATA
E7AA DB53    RTLOOP   IN         HDDATA      ;MOVE FOUR BYTES
E7AC 77      MOV        M,A

```

```

E7AD 23      INX        H
E7AE DB53    IN         HDDATA
E7B0 77      MOV        M,A
E7B1 23      INX        H
E7B2 DB53    IN         HDDATA
E7B4 77      MOV        M,A

```

```

E7B5 23      INX        H
E7B6 DB53    IN         HDDATA
E7B8 77      MOV        M,A
E7B9 23      INX        H
E7BA 05      DCR        B
E7BB C2AAE7  JNZ        RTLOOP
E7BE C9      RET

```

```

E7BF CD08E8  HDWRITE CALL        HDPREP      ;PREPARE HEADER
E7C2 D8      RC
E7C3 AF      XRA        A
E7C4 D351    OUT        HDCMND
E7C6 2AA4E7  LHL        HDADD
E7C9 0680    MVI        B,SECLN/4

```

```

E7CB 7E          WTLOOP MOV      A,M          ;MOVE 4 BYTES
E7CC D353        OUT      HDDATA
E7CE 23          INX      H
E7CF 7E          MOV      A,M
E7D0 D353        OUT      HDDATA
E7D2 23          INX      H
E7D3 7E          MOV      A,M
E7D4 D353        OUT      HDDATA
E7D6 23          INX      H
E7D7 7E          MOV      A,M
E7D8 D353        OUT      HDDATA
E7DA 23          INX      H
E7DB 05          DCR      B
E7DC C2CBE7     JNZ      WTLOOP
E7DF 3E05        MVI      A,WSECT    ;ISSUE WRITE SECTOR COMMAND
E7E1 D351        OUT      HDCMND
E7E3 CDEEE7     CALL     PROCESS
E7E6 D8          RC
E7E7 3E10        MVI      A,WFAULT
E7E9 A0          ANA      B
E7EA 37          STC
E7EB C8          RZ
E7EC AF          XRA      A
E7ED C9          RET
    
```

```

E7EE DB50        PROCESS IN      HDSTAT    ;WAIT FOR COMMAND TO FINISH
E7F0 47          MOV      B,A
E7F1 E602        ANI      OPDONE
E7F3 CAEEE7     JZ       PROCESS
E7F6 3E07        MVI      A,DSKCLK
E7F8 D350        OUT      HDCNTL
E7FA DB50        IN       HDSTAT
E7FC E608        ANI      TMOUT    ;TIMED OUT ?
E7FE 37          STC
E7FF C0          RNZ
E800 DB51        IN       HDRESLT
E802 E602        ANI      RETRY    ;ANY RETRIES ?
E804 37          STC
E805 C0          RNZ
E806 AF          XRA      A
E807 C9          RET
    
```

```

E808 DB50        HDPREP IN      HDSTAT
E80A E620        ANI      DRVRDY
E80C 37          STC
E80D C0          RNZ
E80E 3E08        MVI      A,ISBUFF  ;INITIALIZE POINTER
E810 D351        OUT      HDCMND
E812 CD48E8     CALL     BUILD
E815 F60C        ORI      0CH
E817 D352        OUT      HDFUNC
E819 3A49E8     LDA      HEAD
E81C D353        OUT      HDDATA    ;FORM HEAD BYTE
E81E CD3DE8     CALL     DRVPTR
E821 7E          MOV      A,M          ;FORM TRACK BYTE
E822 D353        OUT      HDDATA
    
```

VR-1412

VSI-READ

```

E824 A7 ANA A
E825 0680 MVI B,80H
E827 CA2CE8 JZ ZKEY
E82A 0600 MVI B,0
E82C 3E00 ZKEY MVI A,0 ;FORM SECTOR BYTE
E82D = HDSECTR EQU $-1
E82E D353 OUT HDDATA
E830 78 MOV A,B
E831 D353 OUT HDDATA
E833 3E07 MVI A,DSKCLK
E835 D350 OUT HDCNTL
E837 3E0F MVI A,WENABL
E839 D350 OUT HDCNTL
E83B AF XRA A
E83C C9 RET
    
```

```

E83D 2A4FE8 DRVPTR LHLD HDDISK
E840 EB XCHG
    
```

```

E841 1600 MVI D,0
E843 2153E8 LXI H,DRIVES
E846 19 DAD D
E847 C9 RET
    
```

```

E848 3E00 BUILD MVI A,0
E849 = HEAD EQU $-1
    
```

```

E84A 17 RAL
E84B 17 RAL
E84C 17 RAL
E84D 17 RAL
E84E F600 ORI 0
    
```

```

E84F = HDDISK EQU $-1
E850 EEF0 XRI 0F0H
E852 C9 RET
    
```

```

E853 = DRIVES EQU $
REPT MAXHD
DB 0FFH
ENDM
E853+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.
*
*****
    
```

```

E854 00 XLT128 IF MAXFLOP NE 0
E855 01070D1319 DB 0
E85A 050B1117 DB 1,7,13,19,25
DB 5,11,17,23
    
```

VR-1412

VISI-READ

```

E85E 03090F15      DB      3,9,15,21
E862 02080E141A    DB      2,8,14,20,26
E867 060C1218      DB      6,12,18,24
E86B 040A1016      DB      4,10,16,22

E86F 00           XLT256 DB      0
E870 0102131425    DB      1,2,19,20,37,38
E876 0304151627    DB      3,4,21,22,39,40
E87C 0506171829    DB      5,6,23,24,41,42
E882 0708191A2B    DB      7,8,25,26,43,44
E888 090A1B1C2D    DB      9,10,27,28,45,46
E88E 0B0C1D1E2F    DB      11,12,29,30,47,48
E894 0D0E1F2031    DB      13,14,31,32,49,50
E89A 0F10212233    DB      15,16,33,34,51,52
E8A0 11122324      DB      17,18,35,36
    
```

```

E8A4 00           XLT512 DB      0
E8A5 0102030411    DB      1,2,3,4,17,18,19,20
E8AD 2122232431    DB      33,34,35,36,49,50,51,52
E8B5 0506070815    DB      5,6,7,8,21,22,23,24
E8BD 2526272835    DB      37,38,39,40,53,54,55,56
E8C5 090A0B0C19    DB      9,10,11,12,25,26,27,28
E8CD 292A2B2C39    DB      41,42,43,44,57,58,59,60
E8D5 0D0E0F101D    DB      13,14,15,16,29,30,31,32
E8DD 2D2E2F30      DB      45,46,47,48
    
```

```

E8E1 00           XLT124 DB      0
E8E2 0102030405    DB      1,2,3,4,5,6,7,8
E8EA 191A1B1C1D    DB      25,26,27,28,29,30,31,32
E8F2 3132333435    DB      49,50,51,52,53,54,55,56
E8FA 090A0B0C0D    DB      9,10,11,12,13,14,15,16
E902 2122232425    DB      33,34,35,36,37,38,39,40
E90A 393A3B3C3D    DB      57,58,59,60,61,62,63,64
E912 1112131415    DB      17,18,19,20,21,22,23,24
E91A 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.
*
*****
    
```

```

E922 1A00      DPB128S DW      26      ;CP/M SECTORS/TRACK
E924 03        DB          3          ;BSH
E925 07        DB          7          ;BLM
E926 00        DB          0          ;EXM
E927 F200      DW      242      ;DSM
E929 3F00      DW          63      ;DRM
    
```

VR-1412

VISI-READ

E92B C0	DB	0C0H	;AL0
E92C 00	DB	0	;AL1
E92D 1000	DW	16	;CKS
E92F 0200	DW	2	;OFF
E931 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.
 *

E932 3400	DPB256S	DW	52	;CP/M SECTORS/TRACK
E934 04		DB	4	;BSH
E935 0F		DB	15	;BLM
E936 00		DB	0	;EXM
E937 F200		DW	242	;DSM
E939 7F00		DW	127	;DRM
E93B C0		DB	0C0H	;AL0
E93C 00		DB	0	;AL1
E93D 2000		DW	32	;CKS
E93F 0200		DW	2	;OFF
E941 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.
 *

E942 3C00	DPB512S	DW	60	;CP/M SECTORS/TRACK
E944 04		DB	4	;BSH
E945 0F		DB	15	;BLM
E946 00		DB	0	;EXM
E947 1801		DW	280	;DSM
E949 7F00		DW	127	;DRM
E94B C0		DB	0C0H	;AL0
E94C 00		DB	0	;AL1
E94D 2000		DW	32	;CKS
E94F 0200		DW	2	;OFF
E951 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.
 *

VR-1412

VISI READ

```

E952 4000      DP1024S DW      64      ;CP/M SECTORS/TRACK
E954 04        DB          4          ;BSH
E955 0F        DB          15         ;BLM
E956 00        DB          0          ;EXM
E957 2B01      DW          299        ;DSM
E959 7F00      DW          127        ;DRM
E95B C0        DB          0C0H       ;AL0
E95C 00        DB          0          ;AL1
E95D 2000      DW          32         ;CKS
E95F 0200      DW          2          ;OFF
E961 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.
*
*****
    
```

```

E962 3400      DPB128D DW      52      ;CP/M SECTORS/TRACK
E964 04        DB          4          ;BSH
E965 0F        DB          15         ;BLM
E966 01        DB          1          ;EXM
E967 F200      DW          242        ;DSM
E969 7F00      DW          127        ;DRM
E96B C0        DB          0C0H       ;AL0
E96C 00        DB          0          ;AL1
E96D 2000      DW          32         ;CKS
E96F 0200      DW          2          ;OFF
E971 09        DB          9H         ;
    
```

```

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

```

E972 6800      DPB256D DW     104      ;CP/M SECTORS/TRACK
E974 04        DB          4          ;BSH
E975 0F        DB          15         ;BLM
E976 00        DB          0          ;EXM
E977 E601      DW          486        ;DSM
E979 FF00      DW          255        ;DRM
E97B F0        DB          0F0H       ;AL0
E97C 00        DB          0          ;AL1
E97D 4000      DW          64         ;CKS
E97F 0200      DW          2          ;OFF
E981 1A        DB          1AH        ;
    
```

```

*****
*
* THE FOLLOWING DPB DEFINES A DISKETTE AS 512 BYTE SECTORS,
*
*****
    
```

VR-1412

VISI-RFAD

* DOUBLE DENSITY, AND DOUBLE SIDED. *
 * *

```

E982 7800 DPB512D DW 120 ;CP/M SECTORS/TRACK
E984 04 DB 4 ;BSH
E985 0F DB 15 ;BLM
E986 00 DB 0 ;EXM
E987 3102 DW 561 ;DSM
E989 FF00 DW 255 ;DRM
E98B F0 DB 0F0H ;AL0
E98C 00 DB 0 ;AL1
E98D 4000 DW 64 ;CKS
E98F 0200 DW 2 ;OFF
E991 3B DB 3BH
    
```

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

```

E992 8000 DP1024D DW 128 ;CP/M SECTORS/TRACK
E994 04 DB 4 ;BSH
E995 0F DB 15 ;BLM
E996 00 DB 0 ;EXM
E997 5702 DW 599 ;DSM
E999 FF00 DW 255 ;DRM
E99B F0 DB 0F0H ;AL0
E99C 00 DB 0 ;AL1
E99D 4000 DW 64 ;CKS
E99F 0200 DW 2 ;OFF
E9A1 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
A IF MAXHD NE 0
B IF M26 NE 0
DPBHD1 DW 1024 ;CP/M SECTORS/TRACK
DB 5 ;BSH
DB 31 ;BLM
DB 1 ;EXM
DW 1973 ;DSM
DW 511 ;DRM
DB 0FFH ;AL0
DB 0FFH ;AL1
DW 0 ;CKS
    
```

VR-1412

VR-1412


```

DW      1      ;OFF
DB      33H    ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
                ;LOG2(#BYTES PER SECTOR/128) + 1 +
                ;8 IF DOUBLE SIDED.
DPBHD2  DW      1024 ;CP/M SECTORS/TRACK
        DB      5    ;BSH
        DB      31   ;BLM
        DB      1    ;EXM
        DW      1973 ;DSM
        DW      511  ;DRM
        DB      0FFH ;AL0
        DB      0FFH ;AL1
        DW      0    ;CKS
        DW      64   ;OFF
        DB      33H  ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
                ;LOG2(#BYTES PER SECTOR/128) + 1 +
                ;8 IF DOUBLE SIDED.

```

```

DPBHD3  DW      1024 ;CP/M SECTORS/TRACK
        DB      5    ;BSH
        DB      31   ;BLM
        DB      1    ;EXM
        DW      1973 ;DSM
        DW      511  ;DRM
        DB      0FFH ;AL0
        DB      0FFH ;AL1
        DW      0    ;CKS
        DW      127  ;OFF
        DB      33H  ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
                ;LOG2(#BYTES PER SECTOR/128) + 1 +
                ;8 IF DOUBLE SIDED.

```

```
ENDIF
```

```

IF      M10 NE 0
DPBHD1  DW      336  ;CP/M SECTORS/TRACK
        DB      5    ;BSH
        DB      31   ;BLM
        DB      1    ;EXM
        DW      1269 ;DSM
        DW      511  ;DRM
        DB      0FFH ;AL0
        DB      0FFH ;AL1
        DW      0    ;CKS
        DW      1    ;OFF
        DB      33H  ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
                ;LOG2(#BYTES PER SECTOR/128) + 1 +
                ;8 IF DOUBLE SIDED.

```

```

DPBHD2  DW      336  ;CP/M SECTORS/TRACK
        DB      5    ;BSH
        DB      31   ;BLM
        DB      1    ;EXM
        DW      1280 ;DSM
        DW      511  ;DRM
        DB      0FFH ;AL0
        DB      0FFH ;AL1
        DW      0    ;CKS
        DW      122  ;OFF

```

DB 33H ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
 ;LOG2(#BYTES PER SECTOR/128) + 1 +
 ;8 IF DOUBLE SIDED.

ENDIF

IF M20 NE 0

E9A2 A002 DPBHD1 DW 672 ;CP/M SECTORS/TRACK

E9A4 05 DB 5 ;BSH

E9A5 1F DB 31 ;BLM

E9A6 01 DB 1 ;EXM

E9A7 DF07 DW 2015 ;DSM

E9A9 FF01 DW 511 ;DRM

E9AB FF DB 0FFH ;AL0

E9AC FF DB 0FFH ;AL1

E9AD 0000 DW 0 ;CKS

E9AF 0100 DW 1 ;OFF

E9B1 33 DB 33H ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
 ;LOG2(#BYTES PER SECTOR/128) + 1 +
 ;8 IF DOUBLE SIDED.

E9B2 A002 DPBHD2 DW 672 ;CP/M SECTORS/TRACK

E9B4 05 DB 5 ;BSH

E9B5 1F DB 31 ;BLM

E9B6 01 DB 1 ;EXM

E9B7 DF07 DW 2015 ;DSM

E9B9 FF01 DW 511 ;DRM

E9BB FF DB 0FFH ;AL0

E9BC FF DB 0FFH ;AL1

E9BD 0000 DW 0 ;CKS

E9BF 6200 DW 98 ;OFF

E9C1 33 DB 33H ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
 ;LOG2(#BYTES PER SECTOR/128) + 1 +
 ;8 IF DOUBLE SIDED.

E9C2 A002 DPBHD3 DW 672 ;CP/M SECTORS/TRACK

E9C4 05 DB 5 ;BSH

E9C5 1F DB 31 ;BLM

E9C6 01 DB 1 ;EXM

E9C7 0404 DW 1028 ;DSM

E9C9 FF01 DW 511 ;DRM

E9CB FF DB 0FFH ;AL0

E9CC FF DB 0FFH ;AL1

E9CD 0000 DW 0 ;CKS

E9CF C300 DW 195 ;OFF

E9D1 33 DB 33H ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
 ;LOG2(#BYTES PER SECTOR/128) + 1 +
 ;8 IF DOUBLE SIDED.

ENDIF

ENDIF

ENDIF

B A

 *
 * 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. *

VR-1412

VSI-READ

*

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 M10 NE 0
SECPT EQU 336 ;SECTORS PER TRACK
TOTBLS EQU 2562 ;TOTAL BLOCKS (4096 BYTE)
TRACKS EQU 244 ;TOTAL TRACKS
ENDIF

IF M20 NE 0
SECPT EQU 672
TOTBLS EQU 5124
TRACKS EQU 244
ENDIF

IF M26 NE 0
SECPT EQU 1024
TOTBLS EQU 6464
TRACKS EQU 202
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
```

```
E9D2 = 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 (M26 NE 0) OR (M20 NE 0)
HEADER %DN,DPBHD3
DN SET DN+1
ENDIF
ENDM
```

```
E9D2+0000 DW 0 ;TRANSLATION TABLE FILLED IN LATER
E9D4+0000000000 DW 0,0,0 ;SCRATCH
E9DA+4CEE DW DIRBUF ;DIRECTORY BUFFER
E9DC+A2E9 DW DPBHD1 ;DPB FILLED IN LATER
E9DE+C8EF DW CSV0 ;DIRECTORY CHECK VECTOR
E9E0+CCEE DW ALV0 ;ALLOCATION VECTOR
```

VR-1412
VISI-READ

ENDIF

IF STDLOG NE 0

```

DN IF FIRST
   SET MAXFLOP
   REPT MAXHD
   REPT STDLOG ;GENERATE HARD DISK DPH'S FOLLOWED
   HEADER %DN,DPBHD,%(DN-MAXFLOP) ;BY FLOPPY DPH'S
DN SET DN+1
    
```

```

DN ENDM
   ENDM
   SET 0 ;FLOPPIES ALWAYS START AT ZERO
   REPT MAXFLOP
   HEADER %DN,0,0
DN SET DN+1
   ENDM
    
```

```

DN ELSE ;GENERATE FLOPPIES BEFORE HARD DISK
   SET 0
   REPT MAXFLOP
   HEADER %DN,0,0
DN SET DN+1
   ENDM
    
```

```

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
    
```

EA42 = BUFFER EQU \$

```

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

```

EA42 801A PROMPT DB 80H, CLEAR ;CLEAN BUFFER AND SCREEN
EA44 0A0D0A0D0A DB ACR,ALF,ACR,ALF,ACR,ALF
EA4A 4D6F72726F DB 'Morrow Designs '
EA59 36 DB '0'+MSIZE/10 ;CP/M MEMORY SIZE
EA5A 32 DB '0'+(MSIZE MOD 10)
EA5B 4B2043502F DB 'K CP/M ' ;CP/M VERSION NUMBER
EA62 32 DB CPMREV/10+'0'
EA63 2E DB '.'
EA64 32 DB (CPMREV MOD 10)+'0'
EA65 2C20436269 DB ', Cbios rev '
EA71 322E DB REVNUM/10+'0', '.' ;CBIOS REVISION NUMBER
EA73 39 DB REVNUM MOD 10+'0'
    
```

IF MAXHD NE 0

VR-1412

VISI-READ

EA74 2E
EA75 32
EA76 30

DB .
DB MREV/10+'0'
DB MREV MOD 10+'0'

ENDIF

IF M10
IF FUJITSU
DB 'F'
ELSE

DB 'M'
ENDIF
ENDIF

EA77 0A0D
EA79 466F7220

DB ACR,ALF
DB 'For '

EA7D 6120446973

IF MAXFLOP NE 0
DB 'a Disk Jockey 2D/B'

ENDIF

EABF 20616E6420

IF (MAXHD NE 0) AND (MAXFLOP NE 0)

DB ' and '

ELSE

DB .

ENDIF

EA94 6120

IF MAXHD NE 0

IF MAXHD EQ 1

DB 'a'

ENDIF

IF MAXHD EQ 2

DB 'two'

ENDIF

IF MAXHD EQ 3

DB 'three'

ENDIF

IF MAXHD EQ 4

DB 'four'

ENDIF

IF MREV EQ 10

IF FUJITSU

DB 'Fujitsu'

ELSE

DB 'Memorex'

ENDIF

DB 'M10'

ENDIF

EA96 46756A6974

IF MREV EQ 20

DB 'Fujitsu M20'

ENDIF

IF MREV EQ 26

DB 'Shugart M26'

ENDIF

EAA2 6861726420

DB 'hard disk'

IF MAXHD NE 1

DB 's'

EAAE 2E

ENDIF
DB
ENDIF

EAAE 0A0D

DB ACR,ALF
IF (CONTYP EQ 0) OR (CONTYP EQ 1)
DB 'Nothing', ACR, ALF
ENDIF

IF CONTYP EQ 2
IF MULTR3
DB 'Multi I/O'

EAAE 4465636973

ELSE
DB 'Decision 1'
ENDIF

ENDIF
IF CONTYP EQ 3
DB '2D/B'
ENDIF

EAB8 2061732063

DB ' as console'

EAC3 2C20

IF LSTTYP GE 2
DB ' , '

IF LSTTYP EQ 2
DB 'serial'

EAC5 4354532070

ENDIF
IF LSTTYP EQ 3
DB 'CTS protocol serial'

ENDIF
IF LSTTYP EQ 4
DB 'DSR protocol serial'

ENDIF
IF LSTTYP EQ 5
DB 'Xon/Xoff protocol serial'

ENDIF
IF LSTTYP EQ 6
DB 'Centronics parallel'

ENDIF
IF LSTTYP EQ 7
DB 'Diablo HyType II parallel'

ENDIF

EAD8 207072696E

DB ' printer as LST:'

ELSE
DB ' , '

ENDIF

EAE8 0A0D

DB ACR, ALF

EAEA 00

DB 0 ;END OF MESSAGE

*

*

VR-1412

VISI/READ

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

```
EAEB 7E MESSAGE MOV A,M ;GET A CHARACTER OF THE MESSAGE
EAEC 23 INX H ;BUMP TEXT POINTER
EAED B7 ORA A ;TEST FOR END
EAEE C8 RZ ;RETURN IF DONE
EAEF E5 PUSH H ;SAVE POINTER TO TEXT
EAF0 4F MOV C,A ;OUTPUT CHARACTER IN C
EAF1 CD0CE3 CALL COUT ;OUTPUT THE CHARACTER
EAF4 E1 POP H ;RESTORE THE POINTER
EAF5 C3EBEA 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. *
 *

```
EAF8 310001 CBOOT LXI SP,TPA ;SET UP STACK
EAFB AF XRA A ;CLEAR COLD BOOT FLAG
EAFD 32FAE3 STA CWFLG
EAFD 323AE3 STA GROUP ;CLEAR GROUP SELECT BYTE
EB02 2100FC IF MAXFLOP NE 0 ;IF 2D/B IS THERE THEN MAKE RAM COPY
EB02 2100FC LXI H,DJRAM ; OF THE JUMP TABLE.
EB05 1100F8 LXI D,ORIGIN
EB08 0633 MVI B,33H ;SIZE OF JUMP TABLE
EB0A CD06E7 CALL MOVLOP ;COPY TABLE
EB0A CD06E7 ENDF
EB0D 3E00 MVI A,INTIOBY
EB0F 320300 STA IOBYTE
EB12 CD31EB IF CONTYP NE 0 ;DO NOT CALL TINIT FOR PROM'S
EB12 CD31EB CALL TINIT ;INITIALIZE THE TERMINAL
EB12 CD31EB ENDF
EB15 CDD2EB IF LSTTYP NE 0 ;DO NOT CALL LINIT FOR PROM'S
EB15 CDD2EB CALL LINIT ;INITIALIZE THE LIST DEVICE
EB15 CDD2EB ENDF
EB18 2142EA LXI H,PROMPT ;PREP FOR SENDING SIGNON MESSAGE
EB1B CDEBEA CALL MESSAGE ;SEND THE PROMPT
EB1E AF XRA A ;SELECT DISK A
EB1F 3244EE STA CPMDRV
EB22 320400 STA CDISK
EB25 32A6E4 IF (MAXFLOP NE 0) AND FIRST
EB25 32A6E4 STA FLOPFLG
EB25 32A6E4 ENDF
EB28 2103E3 LXI H,BIOS+3 ;PATCH COLD BOOT TO WARM CODE
```

*LXI H, TERMSET
CALL MESSAGE*

VR-1412

EB2B 2201E3 SHLD BIOS+1
 EB2E C3A8E3 JMP GOCPM

1950 DB 7BH, 3FH, 121'

IF CONTYP EQ 2 ;MULTI I/O, DECISION 1

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

EB31 3A3AE3 TINIT LDA GROUP ;GET GROUP BYTE
 EB34 F601 ORI CONGRP ;SELECT CONSOLE DEVICE
 EB36 D34F OUT GRPSEL

EB38 DB48 IN RBR ;CLEAR RECIEVER BUFFERS
 EB3A DB48 IN RBR
 EB3C AF XRA A
 EB3D D34D OUT LSR ;CLEAR STATUS
 EB3F D349 OUT IER ;SET NO INTERRUPTS

~~IF BAUD TEST~~ ~~IF NOT MULTRS~~ ~~FORCE DEFCON BAND RATE~~
 EB41 3A3AE3 LDA GROUP ;GET GROUP BYTE
 EB44 D34F OUT GRPSEL ;SELECT GROUP ZERO TO READ SENSE SWITCH
 EB46 DB49 IN SENSESW ;GET SENSE SWITCH.
 EB48 E6E0 ANI 0E0H ;MASK IN UPPER THREE BITS

EB4A 07 RLC ;MOVE INTO LOWER 3 BITS
 EB4B 07 RLC
 EB4C 07 RLC
 EB4D FE07 CPI 7 ;CHECK FOR SENSE = 7
 EB4F F5 PUSH PSW ;SAVE VALUE
 EB50 3A3AE3 LDA GROUP ;GET GROUP BYTE
 EB53 F601 ORI CONGRP ;RESELECT SERIAL PORT GROUP
 EB55 D34F OUT GRPSEL
 EB57 F1 POP PSW

EB58 CA97EB JZ VALID ;DO DEFAULT RATE

EB5B 2169EB LXI H, BTAB ;POINTER TO BAUD RATE TABLE
 EB5E 87 ADD A ;TABLE OF WORDS SO DOUBLE
 EB5F 5F MOV E, A ;MAKE A 16 BIT NUMBER INTO (DE)

EB60 1600 MVI D, 0
 EB62 19 DAD D ;GET A POINTER INTO BAUD RATE TABLE
 EB63 5E MOV E, M ;GET LOWER BYTE OF WORD
 EB64 23 INX H ;POINT TO HIGH BYTE OF WORD
 EB65 56 MOV D, M ;GET UPPER BYTE. (DE) NOW HAS DIVISOR
 EB66 C3C0EB JMP SETIT ;SET BAUD RATE.

EB69 1704 BTAB DW 1047 ;110 BAUD 000
 EB6B 8001 DW 384 ;300 001
 EB6D 6000 DW 96 ;1200 010
 EB6F 3000 DW 48 ;2400 011
 EB71 1800 DW 24 ;4800 100
 EB73 0C00 DW 12 ;9600 101
 EB75 0600 DW 6 ;19200 110

*LXI H, 1950
CALL MESSAGE*

Related

#2

VR-1412

VISI-READ

ENDIF

 * THE FOLLOWING IS A LIST OF VALID BAUD RATES. THE CURRENT
 * BAUD RATE IS CHECKED ON COLD BOOT. IF IT IS NOT IN THE
 * VTAB TABLE THEN THE BAUD RATE WILL BE SET FROM THE DEFCON
 * WORD FOUND BELOW THE CBIOS JUMP TABLE. IF THE USER
 * HAPPENS TO HAVE A WEIRD BAUD RATE THAT IS NOT IN THIS
 * TABLE OR IS LOOKING FOR A WAY TO SAVE SPACE THEN ENTRIES
 * CAN BE ADDED OR DELETED FROM THE TABLE.

EB77	0009	<u>VTAB</u>	DW	2304	;50 BAUD
EB79	0006		DW	1536	;75
EB7B	1704		DW	1047	;110
EB7D	5903		DW	857	;134.5
EB7F	0003		DW	768	;150
EB81	8001		DW	384	;300
EB83	C000		DW	192	;600
EB85	6000		DW	96	;1200
EB87	4000		DW	64	;1800
EB89	3A00		DW	58	;2000
EB8B	3000		DW	48	;2400
EB8D	2000		DW	32	;3600
EB8F	1800		DW	24	;4800
EB91	1000		DW	16	;7200
EB93	0C00		DW	12	;9600
EB95	0600		DW	6	;19200

0010 = SVTAB EQU (\$-VTAB)/2 ;LENGTH OF THE VTAB TABLE

 * VALID CHECKS IF THE DIVISOR LATCH IS A REASONABLE VALUE.
 * IF THE VALUE SEEMS OFF THEN IT WILL GET THE DEFAULT BAUD
 * RATE FROM DEFCON AND JUMP TO SETIT.

EB97	3E87	VALID	MVI	A,DLAB+WLS0+WLS1+STB	
EB99	D34B		OUT	LCR	;ACCESS DIVISOR LATCH
EB9B	DB48		IN	DLL	;GET LOWER DIVISOR VALUE
EB9D	5F		MOV	E,A	
EB9E	DB49		IN	DLM	;GET UPPER DIVISOR VALUE
EBA0	57		MOV	D,A	
EBA1	3E07		MVI	A,WLS1+WLS0+STB	
EBA3	D34B		OUT	LCR	
EBA5	2177EB		LXI	H, <u>VTAB</u>	;VALID BAUD RATE TABLE
EBA8	0E10		MVI	C, <u>SVTAB</u>	;LENGTH OF THE BAUD RATE TABLE
EBA A	7B	VLOOP	MOV	A,E	
EBAB	BE		CMP	M	;CHECK LOW BYTE
EBAC	C2B6EB		JNZ	VSKIP	;FIRST BYTE IS BAD
EBAF	23		INX	H	
EBB0	7A		MOV	A,D	

VR-1412

VISI-READ

```

EBB1 BE          CMP      M          ;CHECK HIGH BYTE
EBB2 CACEEB     JZ       DONE       ;BAUD RATE IS OK... DO CLEANUP
EBB5 2B         DCX      H
EBB6 23         VSKIP   INX      H          ;SKIP TO NEXT ENTRY
EBB7 23         INX      H
EBB8 0D         DCR      C          ;BUMP ENTRY COUNTER
EBB9 C2AAEB     JNZ      VLOOP

```

```

EBBC 2A36E3     MOV LHLD   DEFCON   ;GET DEFAULT BAUD RATE

```

```

EBBF EB         XCHG

```

```

EBC0 3E87     MOV MVI    A,DLAB+WLS1+WLS0+STB ;ENABLE DIVISOR ACCESS LATCH

```

```

EBC2 D34B     OUT     LCR          ;SET THE BAUD RATE IN (DE)

```

```

EBC4 7A       MOV     A,D
EBC5 D349     OUT     DLM          ;SET UPPER DIVISOR

```

```

EBC7 7B       MOV     A,E
EBC8 D348     OUT     DLL          ;SET LOWER DIVISOR

```

```

EBCA 3E07     MVI    A,WLS1+WLS0+STB

```

```

EBCC D34B     OUT     LCR

```

```

EBCE AF       DONE    XRA     A          ;CLEAR STATUS REGISTER

```

```

EBCF D34D     OUT     LSR

```

```

EBD1 C9       RET

```

```

ENDIF ;MULTI I/O, DECISION I

```

```

IF     CONTYP EQ 3 ;2D/B CONSOLE INITIALIZATION

```

```

TINIT CALL DJTSTAT ;CLEAN INPUT BUFFER

```

```

RNZ ;ALL EMPTY

```

```

CALL DJCIN

```

```

JMP TINIT

```

```

ENDIF ;2D/B CONSOLE

```

```

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

```

```

EBD2 3A3AE3     LINIT  LDA     GROUP   ;GET GROUP BYTE

```

```

EBD5 F603     ORI     LSTGRP   ;SELECT LIST DEVICE

```

```

EBD7 D34F     OUT     GRPSEL

```

```

EBD9 3E80     MVI    A,DLAB   ;ACCESS DIVISOR LATCH

```

```

EBDB D34B     OUT     LCR

```

```

EBDD 2A38E3     LHLD   DEFLST   ;GET LST: BAUD RATE DIVISOR

```

```

EBE0 7C       MOV     A,H

```

```

EBE1 D349     OUT     DLM          ;SET UPPER BAUD RATE

```

```

EBE3 7D       MOV     A,L

```

```

EBE4 D348     OUT     DLL

```

```

EBE6 3E07     MVI    A,STB+WLS0+WLS1

```

```

EBE8 D34B     OUT     LCR

```

```

EBEA DB48     IN     RBR          ;CLEAR INPUT BUFFER

```

```

EBEC AF       XRA     A

```

```

EBED D349     OUT     IER        ;NO INTERRUPTS

```

```

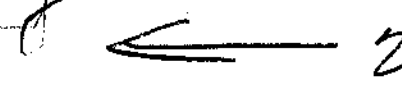
EBEF C9       RET

```

```

ENDIF

```



VR-1412

VISI-READ

IF LSTTYP EQ 6 ;MULTI I/O PARALLEL, CENTRONICS

```

LIMIT LDA GROUP ;GET GROUP BYTE
      ORI DENABLE ;SET DRIVER ENABLE BIT
      STA GROUP
      OUT GRPSEL ;SELECT GROUP ZERO WITH DRIVERS ENABLED
      XRA A
      OUT DAISI1 ;ZERO OUT DATA
      MVI A,D9+D10 ;SET STROBE HIGH, INIT LOW
      OUT DAISI0
DLOOP MVI A,10 ;WAIT ABOUT 50US FOR PRINTER TO INITILIZE
      DCR A
      JNZ DLOOP
      MVI A,D11+D9+D10
      OUT DAISI0
      RET
      ENDIF
    
```

IF LSTTYP EQ 7 ;DIABLO HYTYPE II

```

*****
*
* INITIALIZE DIABLO HYTYPE PRINTER. IF THE PRINTER FAILS
* TO INITIALIZE THEN THE OUTPUT DRIVERS WILL BE TURNED OFF
* AND ANY ATTEMPTS TO PRINT WILL RESULT IN REDIRECTION TO
* THE CONSOLE.
*
*****
    
```

```

LIMIT IF MULTR3 ;MULTI I/O INITIALIZATION
      LDA GROUP ;GET GROUP BYTE
      ORI DENABLE ;ADD DRIVER ENABLE BIT
      OUT GRPSEL
      ORI RESTORE ;TOGGLE RESTORE HIGH
      OUT GRPSEL
DLOOP MVI A,10 ;HOLD LINE UP FOR 50US
      DCR A
      JNZ DLOOP
      LDA GROUP
      OUT GRPSEL ;TURN DENABLE AND RESTORE OFF
      ELSE ;MOTHER BOARD INITIALIZATION
    
```

```

LIMIT LDA GROUP ;GET GROUP BYTE
      OUT GRPSEL ;SELECT GROUP ZERO
      MVI A,PSELECT+RLIFT ;SET SELECT LINE ACTIVE, RLIFT NOT ACTIVE
      OUT CLK
      MVI A,0FFH
      OUT DAISI0
      MVI A,0FFH-RESTORE ;STROBE RESTORE BIT LOW
      OUT DAISI0
DLOOP MVI A,10 ;WAIT ABOUT 50US
      DCR A
      JNZ DLOOP
      MVI A,0FFH ;RAISE RESTORE BACK UP
    
```

VR-1412

VISI-READ

OUT
ENDIF

DAISI0

XRA A
OUT DAISI1 ;CLEAR DATA BUFFERS

IF MULTR3 ;LIFT RIBBON

LDA GROUP
ORI DENABLE

OUT GRPSEL ;RE-ENABLE THE DRIVERS
MVI A,0FFH-RESTORE ;PULL -RIBBON LIFT DOWN
OUT DAISI0

ELSE
MVI A,PSELECT ;RE-ENABLE DRIVERS AND LIFT RIBBON
OUT CLK

ENDIF

LIMIT9 LXI H,HINC/CPERI
SHLD HMI ;SAVE HMI = 120/(CHARACTERS PER INCH)

LXI H,VINC/LPERI
SHLD VMI ;SAVE VMI = 48/(LINES PER INCH)

LXI H,0 ;OTHER VARIABLES DEFAULT TO ZERO

SHLD VPOS
SHLD DLVPOS
SHLD HPOS

SHLD DLHPOS
SHLD LMAR

CALL CLRALL ;CLEAR THE TAB ARRAY

XRA A
STA KLUDGE ;RESET TAB CLEAR BYTE
STA DIRFLG
STA GRHFLG

RET

ENDIF

EBF0 00FF00 DB 0,0FFH,0

EBF3 DS 512-(\$-BUFFER) ;MAXIMUM SIZE BUFFER FOR 512 BYTE SECTORS

EC42 IF MAXFLOP NE 0
DS 512 ;ADDITIONAL SPACE FOR FLOPPIES 1K SECTORS
ENDIF

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

EE42 0000 CPMSEC DW 0 ;CP/M SECTOR #

VR-1412

VISI-READ

```

EE44 00      CPMDRV DB      0      ;CP/M DRIVE #
EE45 00      CPMTRK DB      0      ;CP/M TRACK #
EE46 0000    TRUESEC DW     0      ;DISK JOCKEY SECTOR THAT CONTAINS CP/M SECTOR
EE48 00      BUFDRV DB      0      ;DRIVE THAT BUFFER BELONGS TO
EE49 00      BUFTRK DB      0      ;TRACK THAT BUFFER BELONGS TO
EE4A 0000    BUFSEC DW     0      ;SECTOR THAT BUFFER BELONGS TO
    
```

```

EE4C      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
REPT    MAXFLOP
ALLOC   %DN,75,64
DN      SET      DN+1
    
```

ENDM

```

REPT    MAXHD
IF      M26 NE 0
    
```

```

DN      ALLOC   %DN,247,0
        SET      DN+1
    
```

```

DN      ALLOC   %DN,247,0
        SET      DN+1
    
```

```

DN      ALLOC   %DN,247,0
        SET      DN+1
    
```

```

DN      ALLOC   %DN,247,0
        SET      DN+1
    
```

ENDIF

```

IF      M10 NE 0
ALLOC   %DN,159,0
    
```

```

DN      SET      DN+1
    
```

```

DN      ALLOC   %DN,161,0
        SET      DN+1
    
```

ENDIF

```

IF      M20 NE 0
ALLOC   %DN,252,0
    
```

```

DN      SET      DN+1
    
```

```

DN      ALLOC   %DN,252,0
        SET      DN+1
    
```

```

DN      ALLOC   %DN,129,0
        SET      DN+1
    
```

ENDIF

ENDM

ELSE

```

REPT    MAXHD
IF      M26 NE 0
    
```

```

DN      ALLOC   %DN,247,0
        SET      DN+1
    
```

```

DN      ALLOC   %DN,247,0
        SET      DN+1
    
```

```

DN      ALLOC   %DN,247,0
        SET      DN+1
    
```

ENDIF

ENDM

VR-1412

VISI-READ

DN SET DN+1

ENDIF

IF M10 NE 0

DN ALLOC %DN,159,0

SET DN+1

ALLOC %DN,161,0

DN SET DN+1

ENDIF

IF M20 NE 0

DN ALLOC %DN,252,0

SET DN+1

ALLOC %DN,252,0

DN SET DN+1

DN ALLOC %DN,129,0

SET DN+1

ENDIF

ENDM

EECC+ ALV0 DS 252

EFC8+ CSV0 DS 0

EFC8+ ALV1 DS 252

F0C4+ CSV1 DS 0

F0C4+ ALV2 DS 129

F145+ CSV2 DS 0

REPT MAXFLOP

ALLOC %DN,75,64

DN SET DN+1

ENDM

F145+ ALV3 DS 75

F190+ CSV3 DS 64

F1D0+ ALV4 DS 75

F21B+ CSV4 DS 64

F25B+ ALV5 DS 75

F2A6+ CSV5 DS 64

F2E6+ ALV6 DS 75

F331+ CSV6 DS 64

ENDIF

ENDIF

IF STDLOG NE 0

IF MAXHD NE 0

DN SET MAXFLOP

REPT MAXHD

REPT STDLOG

ALLOC %DN,((TOTBLS/LOGDSK)/8)+1,0

DN SET DN+1

ENDM

ENDM

ENDIF

DN IF MAXFLOP NE 0

SET 0

REPT MAXFLOP

ALLOC %DN,75,64

DN SET DN+1

ENDM

ENDIF

;MAKE UP HARD DISK ALLOCATION VECTORS

;HARD DISKS ALWAYS START AFTER FLOPPIES

;MAKE UP FLOPPY ALLOCATION VECTORS

;FLOPPIES FIRST

ENDIF

F371 END

0006	AACK	0007	ABEL	0008	ABS	E74B	ACCOK	000A	ACR
007F	ADEL	001B	AESC	0003	AETX	000C	AFF	0009	AHT
000D	ALF	EECC	ALV0	EFC8	ALV1	F0C4	ALV2	F145	ALV3
F1D0	ALV4	F25B	ALV5	F2E6	ALV6	0000	ANUL	001E	ARS
0020	ASP	001F	AUS	E3FB	AUTOFLG	0000	AUTOLF	000B	AVT
0004	BANK	D500	BDOS	A800	BIAS	E300	BIOS	EB69	BTAB
EE48	BUFDRV	0080	BUFF	EA42	BUFFER	EE4A	BUFSEC	EE49	BUFTRK
E63E	BUFWRN	E848	BUILD	EAF8	CBOOT	CD00	CCP	0004	CDISK
0040	CHECK	E33B	CIFLSH	E309	CIN	E341	CITTY	E3F3	CLDBOT
E3DE	CLDCMND	001A	CLEAR	004A	CLK	E4B2	CLOPP	00D0	CLRCMD
FBFC	CMDREG	E3FC	COLDBEG	E3FC	COLDEND	0004	COMPLT	0001	CONGRP
E348	CONIN1	E35B	CONOUT1	E306	CONST	0002	CONTYP	E354	COTTY
E30C	COUT	0004	COVER	000A	CPERI	E61E	CPMDMA	EE44	CPMDRV
0016	CPMREV	EE42	CPMSEC	EE45	CPMTRK	0010	CRRDY	0020	CRSTB
I020	CRSTRD	E366	CSTTY	EFC8	CSV0	F0C4	CSV1	F145	CSV2
F190	CSV3	F21B	CSV4	F2A6	CSV5	F331	CSV6	0010	CTS
E3FA	CWFLG	0002	D10	0004	D11	0008	D12	0001	D9
0048	DAISI0	0049	DAISI1	0048	DAISY0	0049	DAISY1	0008	DBLSID
E597	DCRC	E713	DECIDE	E70F	DECIDGO	E336	DEFCON	E338	DEPLST
0020	DENABLE	006E	DFRMLN	EE4C	DIRBUF	E5EA	DIVDONE	E599	DIVLOG
E59B	DIVLOGX	E5DC	DIVLOOP	E781	DIVSPT	E783	DIVSPTX	F800	DJBOOT
F803	DJCIN	F806	DJCOUT	F82D	DJDEN	F812	DJDMA	E333	DJDRV
F82A	DJERR	F809	DJHOME	E4D5	DJINIT	E4F3	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	EBCE	DONE	E442	DONOP	E992	DP1024D	E952	DP1024S
E962	DPB128D	E922	DPB128S	E972	DPB256D	E932	DPB256S	E982	DPB512D
E942	DPB512S	E9D2	DPBASE	E9A2	DPBHD1	E9B2	DPBHD2	E9C2	DPBHD3
FBF9	DREG	0001	DR	E853	DRIVES	E54D	DRVHD	E83D	DRVPTR
0020	DRVRDY	0007	DSKCLK	0020	DSR	E5F9	DTSLOP	0008	ENINT
0005	ENTRY	E6D0	FILL	0001	FIRST	E4A6	FLOPFLG	E4F8	FLOPOK
E63D	FLUSH	E6F8	FREAD	0001	FUJITSU	E5A2	GETDPB	E77D	GETSPT
E3A8	GQCPM	E33A	GROUP	004F	GRPSEL	0000	GZERO	E7A4	HDADD
0051	HDCMND	0050	HDCNTL	0053	HDDATA	E84F	HDDISK	E75E	HDDMA
E71B	HDDRV	0052	HDFUNC	E72C	HDHOME	0050	HDORG	E808	HDPREP
E78A	HDREAD	0051	HDRESLT	0004	HDRLEN	E76C	HDSEC	E82D	HDSECTR
E763	HDSIDE	0015	HDSPT	0050	HDSTAT	E73A	HDTRK	E748	HDTRK2
E7BF	HDWRITE	E849	HEAD	0078	HINC	E449	HOME	0000	IDBUFF
0049	IER	0040	INDEX	0000	INTIOBY	E62B	INTO	FBF8	IO
0003	IOBYTE	0008	ISBUFF	004B	LCR	EBD2	LINIT	E375	LIST
E38E	LISTST	E37C	LL	0003	LOGDSK	0006	LPERI	004D	LSR
0003	LSTGRP	0003	LSTTYP	0000	M10F	0000	M10M	0000	M10
0001	M20	0000	M26	0400	MAXCHRS	0004	MAXFLOP	0001	MAXHD
0630	MAXRGT	0048	MBASE	00F7	MDIR	EAEB	MESSAGE	E608	MOVE
E704	MOVER	E706	MOVLOP	0014	MREV	003E	MSIZE	004E	MSR
FFFF	MULTIO	0000	MULTR3	E65F	NOADJST	00FB	NSTEP	00FC	NULL
00A0	NUMTABS	EBBC	NVALID	3E00	OFFSETC	0002	OPDONE	F800	ORIGIN
E626	OUTOF	0002	PAPER	0008	PFRDY	0010	PFSTB	0810	PFSTRD
E64A	PREP	E7EE	PROCESS	EA42	PROMPT	0080	PSELECT	0004	PSTEP
E3A2	PUNCH	0020	PWRDY	0040	PWSTB	2040	PWSTRD	0048	RBR
E621	RDWR	E3A5	READER	E5CA	READ	0080	READY	E5CE	REDWRT
0080	RESTORE	0010	RESTOR	000A	RETRIES	0002	RETRY	E654	RETRYLP
E6B1	RETRYOP	001D	REVNUM	0001	RIBBON	0040	RLIFT	0001	RSECT
E7AA	RTL00P	0001	S0	0002	S1	0005	SCENBL	0200	SECLEN
E60D	SECPSEC	E5CF	SECSIZ	E450	SECTRAN	0049	SENSESW	E443	SETDMA
E48B	SETDRV	E57A	SETDRV1	EBC0	SETIT	E43D	SETSEC	E44B	SETTRK
E466	SIDEA	E53D	SIDEOK	E469	SIDEONE	E46F	SIDETWO	E74F	SLOOP
0003	SMASK	0004	STB	0000	STDLOG	0010	SVTAB	E569	TDELAY
0048	THR	0020	THRE	EB31	TINIT	0001	TKZERO	0008	TMOUT
0100	TPA	E458	TRANFP	E487	TRANHD	EE46	TRUESEC	EB97	VALID

VR-1412

VISI-READ

0030 VINC	EBA4 VLOOP	EBB6 VSKIP	EB77 VTAB	E3FD WARMBEG
E3FD WARMEND	E41A WARMLOD	E42C WARMRD	E303 WBOOTE	E3FE WBOOT
0000 WBOT	000F WENABL	0010 WFAULT	0001 WLS0	0002 WLS1
000B WRESET	E5C3 WRITE	E635 WRITTYP	E42F WRMREAD	E764 WSDONE
0005 WSECT	E7CB WTLOOP	E8E1 XLT124	E854 XLT128	E86F XLT256
E8A4 XLT512	E5BB XLTS	0013 XOFF	0011 XON	E82C ZKEY

E593 ZRET

VR-1412

VISI-READ