```
;*********************************************************
;*                      DOS9133R.ASM                     *
;*               FLOPPY CBIOS FOR CP/M 2.XX              *
;*                       BRUCE JONES                     *
;*                        JUNE 21,1985                   *
;*********************************************************

      .PABS
      .PHEX
      .XSYM
      .XLINK
      .Z80



CPVERS      ==    22H         ;CP/M VERSION NUMBER
BIVERS      ==    0H          ;BIOS VERSION NUMBER



TRUE  ==    -1          ;TRUE AND FALSE VALUES
FALSE ==    #TRUE

Z3S   ==    TRUE        ;THIS VERSION FOR DMA OR PIO BOARD

ZCPR  ==    0     ;\    \ ENTER 1 FOR ZCPR, 0 FOR NOT SO \

SLOW  ==    0     ;\    \ENTER 1 FOR SLOW CONSOLE I/O, 0 IF NORMAL \


;*********************************************************
;          ASK FOR CP/M SYSTEM SIZE ETC. NEXT
;*********************************************************


STDNR =\    \0 FOR 56K TPA, 1 FOR 55K           \

      .IFE  STDNR,[

      NODSK ==    2
      ][
      NODSK ==    4
      ]

      .IFE  STDNR,[
      STEP  ==    FALSE ;STEP RATE ALLOWED FOR 5" DRIVES
      ][
      STEP  ==    TRUE
      ]


STEPS   ==  0           ;0=3MS

HSTRW ==    1           ;1 FOR 1024 BYTE SCTORS
```

```
HLDOPT      ==    0           ;0 FOR STANDARD HEAD LOAD, 1 FOR DUAL


MSIZ  ==    64          ;MEMORY SIZE


      .IFE  STDNR,[
MSIZE ==    MSIZ-3
            ][
MSIZE ==    MSIZ-4
            ]

BIAS  ==    (MSIZE-20)*1024
CCP   ==    3400H+BIAS  ;BASE OF CCP
BDOS  ==    CCP+806H    ;BASE OF BDOS
BIOS  ==    CCP+1600H   ;BASE OF BIOS
NSECTS      ==    (BIOS-CCP)/128    ;WARM START SECTOR CNT
NSBIOS      ==    70-NSECTS   ;MAX SECTORS FOR CBIOS
CPMDSK      ==    0004H       ;CURRENT DISK NUMBER ADDR
IOBYTE      ==    0003H       ;INTEL I/O BYTE ADDR


;MEMORY AVAILABLE FUNCTION (* 1024 BYTES)  NEXT


      .IFE  STDNR,[
MEMFCT      ==    3
            ][
MEMFCT      ==    4
            ]


      .IFE  STEP,[

STEPR ==    83H         ;5" STEP RATE
      ][
STEPR ==    STEPS       ;NO ITS 8" NOW
      ]

CR    ==    0DH
LF    ==    0AH
BELL  ==    07H
CLEAR ==    1AH


;*****************************************************
;*          SYSTEM DATA I/O PORT CONSTANTS NEXT
;*****************************************************

;               KEYBOARD & SCREEN

TTYI  ==    1           ;TTY INPUT PORT ADDRESS
```

```
TTYO  ==    1           ;TTY OUTPUT PORT ADDRESS
TTYS  ==    0           ;TTY I/O STATUS PORT ADDRESS
TTYDA ==    2           ;TTY DATA AVAILABLE MASK
TTYBE ==    4           ;TTY XMIT BUFFER EMPTY MASK

;                   MODEM PORT

CRTI  ==    5           ;CRT INPUT PORT ADDRESS (MODEM)
CRTO  ==    5           ;CRT OUTPUT PORT ADDRESS (MODEM)
CRTS  ==    4           ;CRT I/O STATUS PORT ADDRESS
CRTDA ==    1           ;CRT DATA AVAILABLE MASK
CRTBE ==    80H         ;CRT XMIT BUFFER EMPTY MASK



;    PRINTER PORTS FOLLOW , PARALLEL PRINTER

LSTST ==    6           ;LIST DEVICE STATUS PORT
LSTDAT      ==    7           ;LIST DEVICE DATA PORT
CEN   ==    9           ;TURN ON CENTRONICS PORT
SPIN  ==    8           ;TURN ON SPINWRITER PORT


;          SERIAL PRINTER PORT NEXT

SERST ==    2           ;SERIAL PRINTER STATUS
SERDT ==    3           ;SERIAL PRINTER DATA PORT


CMSK  ==    00000011B   ;IOBYTE MASK FOR CONSOLE
LMSK  ==    11000000B   ;IOBYTE MASK FOR LIST


;*********************************************************
;                  FLOPPY DISK EQUATES.
;*********************************************************


;*********************************************************
;               DMA & PIO CONTROLLER PORTS
;*********************************************************

DCMD  ==    63H         ;Z3S COMMAND/CONTROL PORT
W     ==    64H         ;WDC 179X ADDRESS
WCMD  ==    W+0         ;COMMAND PORT
WSTAT ==    W+0         ;STATUS PORT
WTRACK      ==    W+1         ;TRACK REG
WSECT ==    W+2         ;SECTOR REG
WDATA ==    W+3         ;DATA I/O REG

;*********************************************************
;             8257 DMA CONTROLLER PORTS
;*********************************************************
```

```
CMND  ==    078H        ;DMA COMMAND PORT
WCTO  ==    071H        ;DMA WORD COUNT PORT
ADRO  ==    070H        ;DMA ADDRESS PORT
DMACHK      ==    062H        ;DMA STATUS PORT


;*******************************************************
;               8257 DMA CONTROLLER COMMANDS
;*******************************************************

RD    ==    04H         ;DMA WRITE INTO MEMORY
WR    ==    08H         ;DMA READ FROM MEMORY


;*******************************************************
;             COMMON 179X CONTROLLER COMMANDS
;*******************************************************

WHOME ==    00001000B   ;HOME COMMAND
WREAD ==    10001000B   ;READ SECTOR COMMAND
WWRITE      ==    10101000B   ;WRITE SECTOR COMMAND
WSEEK ==    00011000B   ;SEEK TO GIVEN TRACK COMMAND
WUNLD ==    00010000B   ;SEEK AND UNLOAD HEAD COMMAND
WLOAD ==    00011000B   ;SEEK AND LOAD HEAD COMMAND


;*******************************************************
;              COMMON 179X CONTROLLER STATUS
;*******************************************************

WBBUSY      ==    0           ;179X BUSY STATUS BIT
WBSID1      ==    1           ;SIDE SELECT FLAG COMMAND BIT
WBDEL ==    2           ;HEAD SETTLE DELAY COMMAND BIT
WBWRIT      ==    5           ;READ/WRITE DISTINGUISHING BIT
WBRNF ==    4           ;RECORD NOT FOUND STATUS BIT
WSREAD      ==    10011100B   ;READ SECTOR STATUS MASK
WSWRIT      ==    11111100B   ;WRITE SECTOR STATUS MASK
WSSEEK      ==    10011000B   ;SEEK STATUS MASK
WFCINT      ==    11010000B   ;FORCE INTERRUPT COMMAND


;*******************************************************
; PRIMARY JUMP TABLE. ALL CALLS FROM CP/M TO THE CBIOS
; COME THROUGH THIS TABLE.
; THE FIRST INSTRUCTION IN THE COLD BOOT POINTS
; TO XTABLE, THE EXTERNAL BIOS DISK TABLE.
;*******************************************************

      .LOC  BIOS


      JMP   CBOOT       ;COLD BOOT
WBOOTE:     JMP   WBOOT       ;WARM BOOT
      JMP   CONST       ;CONSOLE STATUS
```

```
      JMP   CONIN       ;CONSOLE CHARACTER IN
BIOOUT:     JMP   CONOUT            ;CONSOLE CHARACTER OUT,
      JMP   LIST        ;LIST CHARACTER OUT
      JMP   PUNCH       ;PUNCH CHARACTER OUT
      JMP   READER            ;READER CHARACTER IN
      JMP   HOME        ;MOVE HEAD TO HOME POSITION
      JMP   SELDSK            ;SELECT DISK
      JMP   SETTRK            ;SET TRACK NUMBER
      JMP   SETSEC            ;SET SECTOR NUMBER
      JMP   SETDMA            ;SET DMA ADDRESS
      JMP   READ        ;READ DISK
      JMP   WRITE       ;WRITE DISK
      JMP   LISTST            ;RETURN LIST STATUS
      JMP   SECTRA            ;SECTOR TRANSLATE



;*********************************************************
; STEP SPEED TABLE. THIS TABLE TELLS THE BIOS WHAT SIZE
; OF DRIVE IS AT EACH ADDRESS AND WHAT TRACK TO TRACK
; STEPPING SPEED TO USE FOR THAT DRIVE. THE BITS MEAN
; THE FOLLOWING:
;*********************************************************

;     1000 0011
;     ^      ^
;     :      :- THE 179X STEP SPEED BITS. THESE ARE:
;     :           VALUE 8"    5"
;     :           00    3 MS  6 MS
;     :           01    6 MS  12 MS
;     :           10    10 MS 20 MS
;     :           11    15 MS 30 MS
;     :
;     :-------- 0=8 INCH DISK AT THIS ADDRESS,
;              1=5 INCH DISK AT THIS ADDRESS.

SPDTAB:
      .BYTE STEPS       ;DRIVE A
        .BYTE     STEPS       ;DRIVE B
      .BYTE STEPR       ;DRIVE C
      .BYTE STEPR       ;DRIVE D

;*********************************************************
; THIS TABLE PROVIDES INFORMATION TO EXTERNAL PROGRAMS.
;*********************************************************

XTABLE:     .BYTE 0FDH,0DDH   ;IDENTIFICATION BYTES
      .BYTE 0           ;COMPATIBILITY BYTE
      .BYTE 0H          ;FLAG BYTE FOR Z3S CBIOS
      .BYTE 0           ;FLAG BYTE
      .BYTE CPVERS            ;VERSION IDENTIFICATION
      .BYTE BIVERS
```

```
; OK TO CHANGE THE NEXT TWO BYTES.

INITIO:     .BYTE 01010100B   ;INITIAL I/O BYTE AND
      .BYTE 0H          ;INITIAL DISK AFTER COLD BOOT


HOMER:      .BYTE 4           ;ALLOW 3 USER DISK FIXES
RETRYIT:.BYTE     10          ;RETRY 10 TIMES EACH
FSOFT:      .BYTE FALSE       ;RESERVED FOR INTERNAL USE
                        ;WHEN SET TRUE SOFT ERRORS
                        ;ARE DISPLAYED ON SCREEN

      .WORD DSKRST            ;DISK RESET ENTRY ADDRESS
      .WORD HEADLD            ;HEAD LOAD ENTRY ADDRESS


;*******************************************************
; Z3S OPERATING CONTROL TABLE. THIS TABLE CONTAINS
; THOSE BITS IN THE Z3S CONTROL BYTE WHICH SELECT THE
; OPERATING MODE. THESE BITS ARE COMBINED WITH ADDRESS
; CONTROL BITS TO FORM A COMPLETE Z3S CONTROL BYTE. THE
; BITS ARE DEFINED AS FOLLOWS:
;*******************************************************

;     1111 0000
;     ^^^^ ^
;     :::: :- THESE BITS IN ADDRESS CONTROL TABLE
;     ::::
;     ::::--- 0=SIDE 1, 1=SIDE 0
;       :::---- 0=8 INCH DISK, 1=5 INCH DISK
;     ::----- 0=ENABLE HARDWARE WAIT, 1=DISABLE
;     :------ 0=SINGLE DENSITY (FM), 1=DOUBLE (MFM)

OPRTAB:

; SIZE &
; DENSITY

C8S:  .BYTE 01010000B   ;SIDE 0, NO WAIT
      .BYTE 00010000B   ;SIDE 0, WAIT
      .BYTE 01000000B   ;SIDE 1, NO WAIT
      .BYTE 00000000B   ;SIDE 1, WAIT

C5S:  .BYTE 01110000B
      .BYTE 00110000B
      .BYTE 01100000B
      .BYTE 00100000B

C8D:  .BYTE 11010000B
      .BYTE 10010000B
      .BYTE 11000000B
      .BYTE 10000000B

C5D:  .BYTE 11110000B
```

```
      .BYTE 10110000B
      .BYTE 11100000B
      .BYTE 10100000B

;*******************************************************
; Z3S ADDRESS CONTROL TABLE. THIS TABLE CONTAINS THSOE
; BITS IN THE Z3S BYTE WHICH SELECT THE ACTIVE DISK.
; THESE BITS ARE COMBINED WITH OPERATING CONTROL BITS
; TO FORM A COMPLETE Z3S CONTROL BYTE.
;*******************************************************

ADRTAB:     .BYTE 1110B       ;DRIVE A
      .BYTE 1101B       ;DRIVE B
      .BYTE 1011B       ;DRIVE C
      .BYTE 0111B       ;DRIVE D


;*******************************************************
; INTERRUPT DISABLE AND ENABLE. DISINT IS CALLED BY
; THE FLOPPY DISK PHYSICAL I/O ROUTINE TO DISABLE
; INTERRUPTS BEFORE READING OR WRITING A SECTOR.
; ENAINT IS CALLED TO ENABLE INTERRUPTS AFTER
; COMPLETION OF THE I/O OPERATION.
; IF DMA CONTROLLER NO DI OR EI IS PERFORMED
;*******************************************************

DISINT:
      LDA   DMAFLG            ;SEE IF DMA CONTROLLER
      ORA   A
      RNZ               ;IF YES DON'T DI
      DI
      RET


;*******************************************************
; THE SYSTEM CONFIGURATION OPTION BYTES FOLLOW
; THESE BYTES DETERMINE:
; LETTER PRINTER SHIFT SUPPRESSION
; INITIAL ERROR MESSAGE PRINT SUPPRESSION
; NUMBER OF DRIVES IN SYSTEM
; DEFAULT LIST DEVICE
;*******************************************************

CFGOPT:


      SFTSUP:     .BYTE 0     ;SHIFT CHARS TO SPINWRITER
                        ;0 = NOT SUPPRESSED

      ERRSUP:     .BYTE 0     ;INITIAL DISK ERROR MESSAGE
                        ;0 = NOT SUPPRESSED

      ACTDSK:     .BYTE NODSK ;NUMBER OF DISKS SUPPORTED
                        ;NORMALLY 2 DRIVES
```

```
      LTYPE:      .BYTE 'L'   ;DEFAULT LIST DEVICE
                        ;NORMALLY THE LINE PRINTER

      HLOPT:      .BYTE HLDOPT      ;SINGLE OR DUAL HEAD LOAD

      KBIT: .BYTE 4     ;AUTO-RECAL COUNTER


;******************************************************
; COLD BOOT ENTRY POINT. THE FIRST FOUR INSTRUCTIONS
; SHOULD NOT BE CHANGED.
;******************************************************

CBOOT:

      LXI   H,XTABLE    ;POINT TO BIOS EXTERNAL TABLE
      LXI   SP,80H            ;VALIDATE THE STACK POINTER
      LHLD  INITIO            ;SET INITIAL I/O BYTE, DISK
      SHLD  IOBYTE

      .IFN  ZCPR,[

      LXI   H,CMDSET    ;MULTIPLE COMMAND LINE BUFFER
      LXI   D,CLBASE
      LXI   B,128
      LDIR


      LXI   H,PATH            ;MOVE COLD BOOT PATH
      LXI   D,PATHBASE
      LXI   B,9
      LDIR

      ]

      LDÁ   LTYPÅ       ;GEÔ DEFAULÔ LISÔ TYPE
      STÁ   LLOCÂ       ;PUÔ IÎ MESSAGE
      LDÁ   ACTDSK
      ADÉ   30H
      STÁ   DLOCB
      CALÌ  MSÇ         ;SIGÎ ON
      .BYTE CLEAR

      .ASCII      'DOS 9.133 R '

      .IFE  STDNR,[

      .ASCII      \56K TPA \
            ][
      .ASCII      \55K TPA \
            ]

      .IFN  ZCPR,[
```

```
      .ASCII      ' ZCPR '
      ]

      .BYTE ' '+80H

      LDA   SFTSUP
      ORA   A
      JZ    NORAW
      CALL  MSG

      .ASCIS      \ read after write \


NORAW:
      CALL  MSG


LLOCB:      .BYTE 'L',' '
DLOCB:      .BYTE '2',' '
      .BYTE (MSIZE+MEMFCT)/10+'0',(MSIZE+MEMFCT) @10+'0'
      .ASCII      'K CP/M '
      .BYTE '2','.','2'

      .BYTE CR,LF+80H

      CALL  BCOMM       ;DO COMMON STUFF
      JMPR  GOCPM       ;PREPARE TO GO TO CCP



;******************************************************
; WARM BOOT ENTRY POINT. IN ORDER TO WARM BOOT FROM
; A FLOPPY DISK, CP/M MUST BE READ FROM TRACKS 0 AND 1.
; THE PHYSICAL SECTORS ON THE SYSTEM TRACKS ARE ALWAYS
; 512 BYTES LONG, BUT CP/M DOES NOT ALWAYS OCCUPY THE
; ENTIRE SECTOR. THE SECTOR LAYOUT FOR BOTH 8" AND 5"
; DISKS IS:

;     TRACK 0, SECTOR 1, BYTES 0 THRU 127 - COLD
;        START LOADER
;     TRACK 0, SECTOR 1, BYTES 128 THRU 511 - CP/M
;     TRACK 0, SECTOR 2 THRU SECTOR 7 - CP/M
;     TRACK 0, SECTOR 8, BYTES 0 THRU 383 - CP/M
;     TRACK 0, SECTOR 8, BYTES 384 THRU 511 - DDB
;     TRACK 1, SECTOR 1 THRU SECTOR 9 - CP/M,
;        FOLLOWED BY THE CBIOS

; THE SYSTEM TRACKS ON 8" DISKS ARE RECORDED IN SINGLE
; DENSITY, WHILE THE SYSTEM TRACKS ON 5" DISKS ARE
; RECORDED IN DOUBLE DENSITY. ONLY SIDE 0 OF A
; DISK IS USED FOR THE SYSTEM.
;******************************************************

WBOOT:      LXI   SP,80H            ;VALIDATE THE STACK POINTER
```

```
      CALL  CLOSE
EEXIT:
      CALL  BCOMN       ;DO COMMON STUFF


..JOG ==    CCP+(30*128)      ;LAST GOOD ADDRESS
                        ;= 32 -(LOADER + DDB)


      XRA   A           ;READ FROM TRACK 0
      LXI   B,9<8+1           ;READ 9 SECTORS STARTING AT #1
      LXI   H,CCP-128   ;START HERE TO SKIP LOADER
      CALL  READM


;NOW COPY DOSWN SECTOR #9

      LXI   H,CCP+(31*128)    ;LAST GOOD SECTOR STARTS HERE
      LXI   D,CCP+(30*128)    ;AND GOES OVER DDB HERE
      LXI   B,4*128           ;4 CPM RECORDS
      LDIR              ;MOVE IT NOW



..LEFT      ==    NSECTS-34
..PART      ==    ..LEFT@4
..FULL      ==    ..LEFT/4
..ADDR      ==    CCP+(128*(NSECTS-..LEFT))
..LAST      ==    ..ADDR+(..FULL*512)

      MVI   A,1         ;READ FROM TRACK 1
      LXI   B,..FULL <8+1     ;READ FULL SECTORS
      LXI   H,CCP+(34*128)    ;STARTING AFTER 34 CP/M RECORDS
      CALL  READM

      .IFN  ..PART,[

      MVI   A,1         ;READ FROM TRACK 1
      LXI   B,1<8+(1+..FULL);READ ONE LAST SECTOR
      LXI   H,RDBUFF    ;INTO READ BUFFER FOR NOW
      CALL  READM

      LXI   B,..PART*128
      LXI   D,..LAST    ;SECTOR GOES HERE
      LXI   H,RDBUFF    ;SECTOR COMES FROM READ BUFFER
      LDIR              ;MOVE THE SECTOR

      ]


;*****************************************************
;THE FOLLOWING RESETS THE CCP BUFFER ON WARMBOOT
;TO KILL AUTO-START ROUTINES
;*****************************************************
```

```
      XRA   A           ;CLEAR ACC.
      STA   CCP+7       ;STORE A CHAR. COUNTER LOCATION


;*********************************************************
;       COMMON CODE BEFORE ENTERING CP/M.
;*********************************************************

GOCPM:

      MVI   A,JMP       ;PATCH WARM START JUMP
      STA   0
      LXI   H,WBOOTE
      SHLD  1

      STA   5           ;PATCH JUMP TO BDOS
      LXI   H,BDOS
      SHLD  6

      LXI   H,80H       ;SET DEFAULT DMA ADDR
      SHLD  DMAADD

      LDA   CPMDSK            ;PASS CURRENT DISK
      MOV   C,A         ;NUMBER TO THE CCP
      JMP   CCP         ;GO TO THE CCP



;*********************************************************
; READ MULTIPLE SECTORS. USED BY THE FLOPPY DISK WARM
; BOOT.
;*********************************************************

READM:      STA   PTRACK            ;SET TRACK

LOOP: PUSH  B           ;SAVE COUNT, SECTOR
      MOV   A,C
      STA   PSECT       ;SET SECTOR
      PUSH  H           ;SAVE DMA ADDRESS
      SHLD  PDMA        ;SET DMA ADDRESS
      CALL  PREAD       ;READ THE SECTOR
      JC    FATERR            ;FATAL ERROR
      POP   H
      INR   H           ;POINT TO NEXT DMA ADDRESS
      INR   H
      POP   B
      INR   C           ;POINT TO NEXT PHYSICAL SECTOR
      DJNZ  LOOP        ;REPEAT UNTIL DONE
      RET


;*********************************************************
; BOOT COMMON ROUTINE. USED BY COLD BOOT AND WARM BOOT.
```

```
;
; FIRST TEST TO SEE WHICH CONTROLLER IS IN SYSTEM
;                DMA OR PIO
;*******************************************************

BCOMM:

      IN    078H        ;READ DMA PORT AND TEST DATA
      CPI   0FFH        ;IF < 0FFH THEN IT'S A DMA
      JRZ   NDMA        ;ELSE SET DMA FLAG FALSE
      MVI   A,0FFH            ;OR TRUE
      STA   DMAFLG
      JMPR  BCOMN
NDMA: XRA   A
      STA   DMAFLG            ;SET FLAG ZERO FOR PIO

BCOMN:      CALL  DSKRST            ;RESET DISK SYSTEM
      MVI   C,0         ;SELECT DISK 0, GET DDB
      CALL  SELDSK
      MOV   A,H         ;TEST FOR SUCCESSFUL SELECT
      ORA   L
      RNZ               ;SUCCESSFUL
      JMP   QUIT        ;FATAL ERROR


;*******************************************************
; RESET DISK SYSTEM. INVALIDATE CERTAIN FLOPPY DISK
; TABLES AND BYTES TO ALLOW CHANGING DISKS. CALLED BY
; COLD BOOT, WARM BOOT, AND SOME EXTERNAL ROUTINES.
;*******************************************************

DSKRST:     XRA   A
      LXI   H,UNACNT    ;INVALIDATE UNALLOCATED COUNT
      MVI   B,UNALEN
LOP:  MOV   M,A
      INX   H
      DJNZ  LOP
      LXI   H,ACTDSK    ;POINT TO # DISKS
      MOV   B,M         ;GET NUMBER INTO B
      LXI   D,APBDIS    ;GET DISTANCE BETWEEN FD APBS

      LXI   H,APB0+(FLAG-ATABLE) ;POINT TO FLAG

LOOP1:      MOV   M,A         ;INVALIDATE ALL FLOPPY DISK
      DAD   D           ;APBS BY CLEARING FLAGS
      DJNZ  LOOP1

      DCR   A
      STA   OLDFLO            ;FORCE HEAD UNLOAD/LOAD
      STA   ADISK       ;INVALIDATE ATABLE
      RET


      .IFN  ZCPR,[
```

```
CLBASE      ==    0FF4EH            ;ALLOW 50 FOR STACK, 128
BUFLEN      ==    128         ;FOR BUFFER

CMDSET:

      .WORD CLBASE+4
      .BYTE BUFLEN
      .BYTE 0
      .BYTE 0,0,0,0,0,0,0
      .BYTE 0


PATHBASE    ==    40H

PATH:
      .BYTE '$','$'
      .BYTE '$',0
      .BYTE 1,'$'
      .BYTE 1,0
      .BYTE 0

      ]


;*********************************************************
;            LOAD HEAD ON CURRENT DISK
;*********************************************************

HEADLD:     RET



;*********************************************************
; LOGICALLY SELECT THE DISK DRIVE FOR FUTURE READS AND
; WRITES TO THAT PASSED IN REGISTER C. IF THE DDB FOR
; THE DRIVE HAS NOT YET BEEN READ, THEN READ IT IN FROM
; THE DISK. OTHERWISE, DON'T PERFORM A PHYSICAL SELECT
; UNTIL A READ OR WRITE SECTOR CALL IS MADE. NOTE THAT
; THE DPH, APB, DPB, AND TRANSLATE TABLE FOR THE DRIVE
; ARE ALL VALID AT THE COMPLETION OF THIS CALL.
;*********************************************************

SELDSK:
      MOV   A,C
      LXI   H,ACTDSK    ;POINT TO ACTUAL DISK #
      CMP   M           ;IN RANGE(y/n)
      JNC   BAD         ;NO
      STA   SEKDSK            ;D.R. HOST DISK
      STA   PDISK

      PUSH  D           ;SAVE DISK RESET FLAG

      MOV   L,C         ;GET APB, DPH ADDRESSES
```

```
      CALL  GETDPH            ;GET RAM LOCATION FOR DPH
      SDED  APBADR            ;SAVE APB ADDRESS
      SHLD  DPHADR            ;SAVE DPH ADDRESS
      CALL  GETAPB            ;GET ATABLE FOR THIS DISK

      POP   D
      BIT   0,E
      JRZ   FRST        ;MAY HAVE BEEN RESET

      LDA   FLAG
      ORA   A           ;DDB PROCESSED(y/n)
      JNZ   OK          ;YES

FRST:
      CALL  CLOSE       ;ELSE CLEAR ANY PENDING WRITE
                        ;AND READ DDB FROM DISK

      XRA   A           ;WE WILL READ FROM TRACK 0
      STA   PTRACK
      MVI   A,8         ;AND SECTOR 8
      STA   PSECT
      LXI   H,RDBUFF    ;INTO THE READ BUFFER
      SHLD  PDMA

      LDA   PDISK       ;GET DISK NUMBER
      LXI   H,SPDTAB    ;POINT TO STEP SPEED TABLE
      MOV   E,A
      MVI   D,0
      DAD   D
      MOV   A,M         ;GET SPEED BYTE FOR THIS DISK
      MOV   B,A         ;SAVE IN B
      ANI   3           ;ISOLATE SPEED BITS
      MOV   H,A         ;PUT IN H FOR DOUBLE STORE NEXT
      MVI   L,0FFH            ;CURRENT TRACK - UNKNOWN
      SHLD  TRACK       ;UPDATE ATABLE TRACK AND SPEED


;*****************************************************
;          ASSUME 8" S.S.S.D. DISK NEXT
;*****************************************************

      MVI   A,00000001B       ;INITIAL FLAG FOR 8" DRIVE
      LXI   H,STDDDB    ;STANDARD 8" DDB ADDRESS


      .IFN  STDNR,[

;TEST IF 5" OR 8" DISK IN SPEED TABLE

      BIT   7,B         ;8" DISK(y/n)
      JRZ   EIG         ;IF YES GO DO 8"
      MVI   A,00010101B       ;INITIAL 5" FLAG
      LXI   H,ALTDDB
```

```
      ]

EIG:  STA   FLAG
      SHLD  SAVADR

      LXI   H,1         ;AT LEAST 1 SYSTEM TRACK
      SHLD  OFF
      LXI   H,2+32 < 8  ;512 BYTE SECTOR & 32 LOGICAL SECTORS
      SHLD  SSLEN

      CALL  GETD3S            ;GET Z3S CTRL BYTES INTO ATABLE
      CALL  PREAD       ;GET THE DDB
      JRNC  YUP         ;WE GOT SOMETHING


;*******************************************************
; IF ERROR IS R.N.F. THEN SHOW DISK READ FAILED
; ELSE TEST IT FOR GOOD DISK TYPE
;*******************************************************

      BIT   WBRNF,A           ;RNF ERROR(y/n)
      JRZ   ERR         ;NOPE, GIVE UP


;*******************************************************
;    SET UP FOR STANDARD 8" S.S.S.D.  DISK HERE
;*******************************************************

NOV:

      LHLD  SAVADR      ;POINT TO STANDARD DDB
      LXI   B,128-10
      LXI   D,RDBUFF+384+10
      LDIR              ;FIX RD BUFFER TO BE A STD DDB
      JMPR  COMP

YUP:  LHLD  RDBUFF+384  ;GET VALIDITY BYTES FROM DDB
      LXI   D,0DDH+0FDH < 8 ;EXPECTED VALUE OF BYTES
      ORA   A
      DSBC  D           ;DDB VALID(y/n)
      JRNZ  NOV         ;NOPE
      LHLD  RDBUFF+384+2      ;MORE VALIDITY BYTES
      DSBC  D           ;DDB VALID(y/n)
      JRNZ  NOV         ;NOPE

      LDA   RDBUFF+384+4      ;TEST FOR COMPATIBILITY
      ANI   11111110B
      JRNZ  BAD         ;GIVE UP ON THIS DISK

COMP: CALL  PUTAPB            ;UPDATE TRACK, SPEED IN APB

      LHLD  APBADR            ;GET APB ADDR FOR THIS DISK
      LXI   D,FLAG-ATABLE     ;POINT TO WHERE FLAG GOES
      DAD   D
```

```
      XCHG              ;MAKE THIS THE DESTINATION ADDR
      LXI   H,RDBUFF+384+10 ;FROM FLAG IN RD BUFFER

      LXI   B,ALEN-(FLAG-ATABLE)+DPBLEN+TRALEN

      LDIR              ;MOVE DDB, DPB, TRANS INTO APB


;*****************************************************
;*           NOW SET UP ALLOCATION SIZE
;*           FOR THE DISK JUST SELECTED
;* GET THE ALLOCATION SIZE FROM THE DSM VALUE IN THE DPB
;*****************************************************

GALV:
      LDA   RDBUFF+384+19     ;GET DSM
      MOV   C,A         ;SAVE VALUE
      LDA   SEKDSK
      MOV   E,A
      MVI   D,0
      LXI   H,ALOCSZ
      DAD   D
      MOV   M,C         ;SAVE FOR THIS DISK

      MVI   A,0FFH            ;UPDATE ATABLE FROM APB
      STA   ADISK
      CALL  GETAPB
      CALL  GETD3S            ;PUT VALID Z3S BYTES IN ATABLE
      CALL  PUTAPB            ;UPDATE APB FROM FULLY VALID ATABLE

OK:   LHLD  DPHADR            ;RETURN DPH ADDRESS
      RET

ERR:
      CALL  EPRINT            ;PRINT ERROR IN READING DDB

BAD:  XRA   A           ;DESELECT INVALID DRIVE
      STA   CPMDSK
      MOV   H,A         ;ERROR RETURN CODE
      MOV   L,A
      RET



;*****************************************************
; SET TRACK FOR FUTURE READS OR WRITES TO TRACK 0. ALSO
; PARTIALLY RESET THE DISK SYSTEM TO ALLOW FOR CHANGED
; DISKS.
;*****************************************************

HOME:
      CALL  CLOSE
      LDA   HSTWRT            ;TEST FOR PENDING WRITE
      ORA   A
```

```
      JRNZ  HOMED
      STA   HSTACT            ;CLEAR HOST ACTIVE FLAG
HOMED:
      LXI   B,0         ;DROP THRU TO SET TRACK TO 0


;*********************************************************
; SET TRACK FOR FUTURE READS OR WRITES TO THAT PASSED
; IN REGISTER PAIR BC.
;*********************************************************

SETTRK:     SBCD  SEKTRK
      RET



;*********************************************************
; SET SECTOR FOR FUTURE READS OR WRITES TO THAT PASSED
; IN REGISTER PAIR BC.
;*********************************************************

SETSEC:     SBCD  SEKSEC
      RET



;*********************************************************
; SET DMA ADDRESS FOR FUTURE READS OR WRITES TO THAT
; PASSED IN REGISTER PAIR BC.
;*********************************************************

SETDMA:     SBCD  DMAADD
      RET



;*********************************************************
; SECTOR TRANSLATION ROUTINE. THE ROUTINE ONLY
; TRANSLATES SECTORS ON THE USER TRACKS, SINCE CP/M
; ACCESSES THE SYSTEM TRACKS WITHOUT CALLING FOR
; TRANSLATION.
;*********************************************************

SECTRA:

YUP1: XCHG              ;HL GETS TRANS TABLE ADDRESS
                        ;CP/M PASSED IT IN DE

      MOV   A,C         ;GET SECTOR #
      SBCD  NEWSEC            ;SAVE FOR UNALLOC TEST


      DAD   B           ;INDEX INTO TABLE, LOGICAL SECTOR
                        ;IS PASSED IN BC
```

```
      MOV   L,M         ;GET THE TRANSLATED BYTE
      MVI   H,0
      RET



;*********************************************************
; CP/M ENTRY POINT FOR SECTOR READS. BUFFERED SECTOR
; READS ARE DONE HERE. BUFFERED READ OPERATIONS REQUIRE
; READING THE SECTOR FROM DISK INTO THE READ BUFFER,
; AND THEN PROVIDING 128 BYTE LOGICAL SECTORS TO THE
; CALLING PROGRAM ON REQUEST.
;*********************************************************

READ:

      XRA   A
      STA   UNACNT            ;UNACNT=0, WE WON'T WRITE WITHOUT
                        ;PRE-READS FOR NOW
      INR   A
      STA   READOP            ;SHOW WE ARE DOING A READ OPERATION
      STA   RSFLAG            ;MUST READ DATA
      MVI   A,2
      STA   WRTYPE            ;TREAT AS UNALLOCATED

      JMP   RWOPER            ;DO THE READ


;*********************************************************
; CP/M ENTRY POINT FOR SECTOR WRITES. BUFFERED SECTOR
; WRITES ARE DONE HERE. BUFFERED WRITE OPERATIONS
; REQUIRE ACCEPTING 128 BYTE LOGICAL SECTORS FROM THE
; CALLING PROGRAM, ACCUMULATING THEM IN A WRITE BUFFER,
; THEN WRITING THE BUFFER WHEN IT BECOMES FULL. THE
; BUFFER IS IMMEDIATELY WRITTEN OUT IF THE LOGICAL
; SECTOR IS PART OF THE DISK DIRECTORY.
;*********************************************************


WRITE:

      XRA   A
      STA   READOP            ;SET TO WRITE

      MOV   A,C
      STA   WRTYPE            ;SAVE TYPE OF WRITE

      CPI   2           ;WRITE UNALLOCATED(y/n)
      JRNZ  CKUN        ;GO SEE IF O.K. ANYWAY

      LDA   ALOCA       ;GET MAXIMUM UNALLOCATED RECORD COUNT
      STA   UNACNT            ;AND PUT HERE FOR WRITING
      LDA   SEKDSK            ;GET CURRENT DISK
```

```
      STA   UNADSK
      LHLD  SEKTRK            ;GET CURRENT TRACK
      SHLD  UNATRK
      LDA   NEWSEC            ;GET CURRENT CP/M SECTOR
      STA   UNASEC

CKUN:                 ;SEE IF UNALLOCATED RECORDS REMAIN

      LDA   UNACNT            ;GET UNALLOCATED RECORDS LEFT
      ORA   A
      JZ    ALOC        ;NO UNALLOCATED LEFT


;WE STILL HAVE UNALLOCATED RECORDS LEFT

      DCR   A           ;UPDATE UNALLOCATED RECORD COUNT
      STA   UNACNT


;NOW CHECK FOR CORRECT DISK, TRACK & SECTOR

      LDA   SEKDSK            ;COMPARE DISKS FIRST
      LXI   H,UNADSK
      CMP   M
      JNZ   ALOC


;DISKS ARE SAME, NOW CHECK FOR TRACK

      LXI   H,UNATRK
      LDA   SEKTRK
      CMP   M
      JRNZ  ALOC


;TRACKS ARE SAME, NOW TEST FOR SECTOR

      LDA   NEWSEC            ;COMPARE SECTORS NOW
      LXI   H,UNASEC
      CMP   M
      JRNZ  ALOC


;SECTORS ARE SAME, NOW UPDATE PARAMETERS

      INR   M           ;MAKE NEXT EXPECTED SECTOR
      MOV   A,M         ;GET NEXT EXPECTED SECTOR
      LXI   H,ULRPS           ;POINT TO SECTORS/USER TRACK
      CMP   M           ;TEST FOR END OF TRACK
      JRC   NOVR        ;NO OVERFLOW


;HERE WE ALLOW FOR NEXT UNALLOCATED RECORD ON A NEW TRACK
```

```
      XRA   A           ;SET SECTOR AS FIRST
      STA   UNASEC
      LXI   H,UNATRK    ;POINT TO UNALLOCATED TRACK #
      INR   M           ;MAKE IT NEXT ONE


;WRITE PARAMETERS MATCH, DON'T PRE-READ

NOVR:
      XRA   A
      STA   RSFLAG            ;SHOW WE DON'T READ A SECTOR
      JMPR  RWOPER


;NOT AN UNALLOCATED RECORD, DO A PRE-READ

ALOC:
      XRA   A
      STA   UNACNT            ;SET UNALLOCATED = 0
      INR   A
      STA   RSFLAG            ;RSFLAG = 1, WE MUST READ THE SECTOR


;DO READ OR WRITE OPERATION NEXT

RWOPER:

      CALL  GETTRK            ;COMPUTE PHYSICAL TRACK & SECTOR

      LXI   H,HSTACT    ;GET HOST ACTIVE FLAG
      MOV   A,M
      MVI   M,1         ;SET IT ACTIVE FOR SURE
      ORA   A           ;SEE IF IT WAS ACTIVE
      JZ    FILHST            ;IF NOT FILL IT


;*********************************************************
;* CHECK TO SEE IF SECTOR IN HOST BUFFER IS CORRECT ONE
;* IF NOT WRITE TO HOST BUFFER IF NEEDED & PREPARE FOR
;* CORRECT HOST BUFFER
;*********************************************************

;SEE IF DISKS ARE SAME

      LDA   SEKDSK            ;COMPARE DISKS FIRST
      LXI   H,HSTDSK
      CMP   M
      JRNZ  NOMAT


;SEE IF TRACKS ARE SAME

      LXI   H,HSTTRK
      LDA   CTRACK
```

```
      CMP   M
      JRNZ  NOMAT


;SEE IF SECTORS ARE SAME

      LDA   SEKHST            ;COMPARE SECTORS NOW
      LXI   H,HSTSEC
      CMP   M
      JRZ   MATCH


;HOST PARAMETERS DO NOT MATCH CURRENT R/W PARAMETERS
;SEE IF WE HAVE TO FLUSH THE HOST BUFFER

NOMAT:
      LDA   HSTWRT            ;SEE IF HOST BUFFER WAS WRITTEN
      ORA   A
      CNZ   FLUSH       ;WRITE OUT THE BUFFER IF NOT


;MAY HAVE TO FILL HOST BUFFER
;SEET UP NEW PARAMETERS

FILHST:

      LDA   SEKDSK
      STA   HSTDSK
      LHLD  CTRACK
      SHLD  HSTTRK
      LDA   SEKHST            ;THE PHYSICAL SECTOR
      STA   HSTSEC

      LDA   RSFLAG            ;SEE IF WE NEED TO READ
      ORA   A
      CNZ   RDHST       ;IF SO READ IT
      XRA   A
      STA   HSTWRT            ;SHOW NO PENDING WRITE

;WE HAVE CORRECT SECTOR SO COPY DATA TO/FROM DMA BUFFER

MATCH:

      LDA   CREC        ;GET RECORD #
      MOV   B,A         ;COMPUTE RECORD IN HOST BUFFER
      MVI   C,0
      SRLR  B
      RARR  C
      LXI   H,HSTBUF
      DAD   B           ;HL IS NOW HOST BUFFER
      LDED  DMAADD            ;DE HAS DMA ADDRESS
      LXI   B,128       ;BYTES TO MOVE
      LDA   READOP
      ORA   A           ;SEE IF WE ARE READING OR WRITING
```

```
      JRNZ  RWMOVE            ;SKIP ON READ
      MVI   A,1         ;IF A WRITE THEN MARK & COPY TO BUFFER
      STA   HSTWRT            ;HSTWRT = 1
      XCHG              ;DE IS NOW DESTINATION,= HOST ON WRITE
      LHLD  DMAADD            ;HL IS SOURCE, = USER AREA TO GET

RWMOVE:

      LDIR              ;MOVE DATA


;NOW CHECK WRITE TYPE FOR DIRECTORY UPDATE

      LDA   WRTYPE            ;GET TYPE OF WRITE
      DCR   A           ;IS IT TO THE DIRECTORY
      JRZ   WRITIT            ;IF SO WRITE IT OUT

      JMPR  GOODOP            ;IF NOT SHOW A SUCCESSFUL R/W OPERATION


;CLEAR HOST BUFFER FOR DIRECTORY WRITE

WRITIT:

      STA   HSTWRT
      CALL  WRTHST

GOODOP:
      LDA   MRML        ;GET RETRY BYTE
      ORA   A           ;TEST FOR ZERO
      MVI   A,1         ;JUST IN CASE OF FAILURE
      RZ                ;IF FAILED SHOW IT TO BDOS
      XRA   A           ;ELSE SET AS O.K.
      RET               ;SHOW BDOS SUCCESSFUL READ/WRITE




CLOSE:
      LDA   HSTWRT            ;SEE IF WE HAVE A PENDING WRITE
      ORA   A
      RZ                ;IF NOT RETURN NOW

;WRITE FROM THE HOST BUFFER

FLUSH:
WRTHST:
      LDA   HSTDSK            ;GET ACTUAL WRITE DISK
      STA   PDISK       ;MAKE IT THE PHYSICAL DISK
      CALL  GETAPB            ;GET APB FOR THE DISK
      LDA   HSTSEC
      STA   PSECT
      LDA   HSTTRK
      STA   PTRACK
```

```
      LXI   H,WRBUFF    ;POINT TO WRITE BUFFER
      SHLD  PDMA        ;MAKE IT THE PHYSICAL DMA ADDRESS

      CALL  PWRITE            ;WRITE BACK THE COMBINED SECTOR
      PUSH  PSW
      LDA   SFTSUP            ;SEE IF READ AFTER WRITE
      ORA   A
      JNZ   CHKSEC
      POP   PSW

      RET


RDHST:
      LDA   HSTDSK
      STA   PDISK
      LXI   H,RDBUFF    ;POINT TO READ BUFFER
      SHLD  PDMA        ;MAKE IT PHYSICAL DMA ADDRESS
      LDA   CTRACK
      STA   PTRACK
      LDA   CSECT
      STA   PSECT

      CALL  PREAD       ;READ SECTOR INTO READ BUFFER
      RET


;GET ACTUAL TRACK TO SEEK

GETTRK:
      LDA   SEKTRK            ;GET CP/M TRACK NUMBER
      LXI   H,OFF       ;GET NUMBER OF SYSTEM TRACKS
      MOV   E,M
      MVI   C,0         ;ASSUME SINGLE SIDED DISK
      LXI   H,SSLEN           ;POINT TO SYSTEM SECTOR LENGTH
      CMP   E
      JRC   SYST        ;IT WAS A SYSTEM TRACK
      LXI   H,FLAG            ;POINT TO FLAG BYTE
      BIT   1,M         ;TEST SIDES BIT
      JRZ   SING        ;SINGLE SIDED DISK
      ADD   E           ;ADD IN OFFSET
      SRLR  A           ;COMPUTE PHYSICAL TRACK NUMBER
      RARR  C           ;GET SIDE NUMBER BIT
SING: LXI   H,USLEN
SYST:
      STA   CTRACK            ;SAVE ACTUAL TRACK NUMBER


;GET ACTUAL SECTOR TO READ/WRITE

PYSEC:
      MOV   A,M         ;GET LENGTH BYTE
      MOV   B,A
      CPI   3
```

```
      JRNZ  LRG
      MVI   H,7
      JMPR  GSEC
LRG:  CPI   2           ;512 BYTE SECTOR(y/n)
      JRNZ  NO          ;NOPE, ACC HAS RECORD MASK
      INR   A           ;FIND MASK FOR 512 BYTE SECTOR
NO:   MOV   H,A         ;SAVE RECORD MASK

GSEC: LDA   SEKSEC            ;GET CP/M SECTOR NUMBER
      DCR   A           ;ADJUST DOWN TO START AT ZERO
      MOV   L,A         ;SAVE FOR LATER
      INR   B           ;ADJUST FOR EASY LOOP
      JMPR  JOIN
LOOP2:      SRLR  A           ;PLACE SECTOR NUMBER IN LSB'S
JOIN: DJNZ  LOOP2       ;REPEAT UNTIL ALIGNED IN LSB'S
      INR   A           ;ADJUST TO MAKE PHYSICAL SECTOR
      ORA   C           ;GET SIDE BIT
      STA   SEKHST            ;HOST SECTOR
      STA   CSECT       ;SAVE COMBINED SECTOR
      MOV   A,L         ;GET CP/M SECTOR NUMBER
      ANA   H           ;MASK ALL BUT RECORD NUMBER
      STA   CREC        ;SAVE RECORD NUMBER
      RET


;*****************************************************
; FLOPPY DISK PHYSICAL READ AND WRITE ROUTINE. ALL
; FLOPPY DISK I/O IS PERFORMED BY CALLS TO THIS
; ROUTINE. ON ENTRY PDISK, PTRACK, PSECT, AND PDMA
; MUST BE VALID.
;*****************************************************

PWRITE:     MVI   A,WWRITE    ;SET WRITE COMMAND
      JMPR  PCOM        ;JOIN COMMON CODE

PREAD:      MVI   A,WREAD           ;SET READ COMMAND

PCOM: STA   PCMD        ;REMEMBER WHETHER READ OR WRITE
      CALL  GETAPB            ;MAKE SURE ATABLE IS RIGHT ONE


;*****************************************************
;     IF LAST I/O WAS ON DIFFERENT DISK, TELL THE
;     179X TO UNLOAD ITS HEAD. THE HEAD LOAD ONE-SHOT
;     WILL THEN BE RETRIGERED ON THE NEXT COMMAND.
;*****************************************************

      LXI   H,OLDFLO    ;POINT TO OLD FLOPPY NUMBER
      LDA   PDISK       ;GET NEW NUMBER
      CMP   M           ;SAME DISK(y/n)
      JRZ   SADS        ;YES

      MOV   M,A         ;NO, UPDATE OLD NUMBER TO NEW
      CALL  TRIMWT            ;WAIT FOR TRIM ERASE TO END
```

```
      CALL  SETD3S            ;SET Z3S CONTROL BYTE
      LDA   HLOPT       ;SEE IF DUAL HEAD LOAD
      ORA   A
      JRNZ  SADS        ;DON'T UNLOAD HEAD
      CALL  FUNLD       ;UNLOAD HEAD
      CALL  FDONE       ;INSURE FDC IS DONE


;*******************************************************
;     INITIALIZE RETRY LIMITS. INSURE Z3S BYTE IS
;     SET. SEEK TO CORRECT TRACK.
;*******************************************************

SADS:
      LDA   HOMER       ;NUMBER OF HOME OPERATIONS
      STA   MRML
      STA   KBIT
MAC:
      LDA   RETRYIT     ;RETRIES BETWEEN HOME OPERATIONS
      STA   RMACRO

      CALL  SETD3S            ;SET Z3S CONTROL BYTE
      LXI   H,TRACK     ;GET OLD TRACK NUMBER
      MOV   A,M
      OUT   WTRACK            ;UPDATE 179X TRACK REG
      LDA   PTRACK            ;GET DESIRED TRACK NUMBER
      CMP   M           ;SAME AS BEFORE(y/n)
      JRZ   SATR        ;YES

      MOV   M,A         ;UPDATE TRACK NUMBER
      CALL  TRIMWT            ;WAIT FOR TRIM ERASE TO END
      LDA   PTRACK            ;GET DESIRED TRACK
      ORA   A           ;TRACK 0 DESIRED(y/n)
      JRNZ  NOZE        ;NOPE

      CALL  FHOME       ;SEEK TO TRACK 0 BY HOME CMD
      JC    FATERR
      JMPR  ENDS        ;DONE SEEKING

NOZE: CALL  FSEEK       ;NORMAL SEEK TO DESIRED TRACK
      JC    FATERR

ENDS: CALL  PUTAPB            ;UPDATE APB FROM ATABLE
      LDA   PCMD        ;GET READ/WRITE COMMAND
      SET   WBDEL,A           ;INSURE HEAD IS SETTLED
NOMIL:
      STA   PCMD        ;BY SETTING DELAY BIT IN CMD


;*******************************************************
;     SET UP DMA ADDRESS, SECTOR REGISTER. ISSUE
;     THE READ OR WRITE COMMAND. SET HARDWARE WAIT.
;*******************************************************
```

```
SATR:
MIC:
      CALL  DISINT            ;DISABLE INTERRUPTS
      LDA   PSECT       ;GET DESIRED SECTOR NUMBER
      MOV   B,A         ;SAVE SIDE BIT
      BIT   7,A         ;TEST SIDE BIT
      MVI   A,'0'
      JRZ   SIDEZERO
      MVI   A,'1'
SIDEZERO:
      STA   SIDEID
      MOV   A,B         ;GET BACK SECTOR
      ANI   07FH        ;DROP SIDE BIT
      OUT   WSECT       ;UPDATE 179X SECTOR REGISTER
      LDA   DMAFLG
      ORA   A

      LDA   PCMD        ;GET READ OR WRITE COMMAND
      JRNZ  SID2

      BIT   7,B         ;ARE WE ON SIDE 1(y/n)
      JRZ   SID01       ;NO, LEAVE SSO BIT AS 0
      SET   WBSID1,A    ;YES,UPDATE SSO BIT TO 1


SID01:
      OUT   WCMD        ;FOR PIO 179X COMMAND
SID2: STA   OCMD
      STA   ORWCMD            ;SAVE LAST READ OR WRITE CMD
      MOV   D,A         ;SAVE THE COMMAND

      LDA   D3SWT       ;GET WAIT ACTIVE Z3S BYTE
      OUT     DCMD

      STA   OD3S

      LXI   B,128 < 8+WDATA ;SET PORT AND LENGTH
      LDA   PTRACK            ;GET CURRENT TRACK
      LXI   H,OFF
      CMP   M           ;IS IT A USER TRACK(y/n)
      LDA   USLEN       ;GET USER SECTOR LENGTH
      JRNC  ULEN
      LDA   SSLEN       ;GET SYSTEM SECTOR LENGTH

ULEN: LHLD  PDMA        ;GET DMA ADDRESS
      BIT   WBWRIT,D    ;ARE WE WRITING(y/n)
      JRNZ  WRIT        ;YES, GO WRITE


;*******************************************************
;                   READ THE SECTOR.
;*******************************************************

      PUSH  PSW
```

```
      LDA   DMAFLG
      ORA   A
      JRZ   PIRD
      POP   PSW

      ORA   A           ;IS SECTOR 128 BYTES(y/n)
      JRZ   R128        ;YES
      DCR   A           ;IS SECTOR 256 BYTES(y/n)
      JRZ   R256        ;YES
      DCR   A           ;IS SECTOR 512 BYTES(y/n)
      JRZ   R512
      LXI   D,RD < 12+1024-1
      JMPR  RDMA
R512: LXI   D,RD < 12+512-1   ;MERGE READ BIT WITH BYTE COUNT
      JMPR  RDMA        ;BYTE COUNT IS OFF BY ONE

R256: LXI   D,RD < 12+256-1
      JMPR  RDMA

R128: LXI   D,RD < 12+128-1

RDMA: CALL  DMARW

      JMPR  RDFN

PIRD:

      POP   PSW
      ORA   A
      JRZ   PR12
      MVI   B,0H
      DCR   A
      JRZ   PR25
      DCR   A
      JRZ   PR51
      INIR
      INIR
PR51: INIR
PR25:
PR12: INIR

RDFN:

      MVI   C,WSREAD    ;STATUS BITS TO TEST
      JMPR  CHEK


;*******************************************************
;                 WRITE THE SECTOR.
;*******************************************************

WRIT:

      PUSH  PSW
```

```
      LDA   DMAFLG
      ORA   A
      JRZ   PIWR

      POP   PSW

      ORA   A           ;IS SECTOR 128 BYTES(y/n)
      JRZ   W128        ;YES
      DCR   A           ;IS SECTOR 256 BYTES(y/n)
      JRZ   W256
      DCR   A
      JRZ   W512
      LXI   D,WR <12+1024-1
      JMPR  WDMA

W512: LXI   D,WR <12+512-1
      JMPR  WDMA

W256: LXI   D,WR <12+256-1
      JMPR  WDMA

W128: LXI   D,WR <12+128-1

WDMA: CALL  DMARW

      JMPR  WRFN

PIWR:
      POP   PSW

      ORA   A
      JRZ   PW12
      MVI   B,0H
      DCR   A
      JRZ   PW25
      DCR   A
      JRZ   PW51
      OUTIR
      OUTIR

PW51: OUTIR
PW25:
PW12: OUTIR


WRFN:
      MVI   C,WSWRIT    ;STATUS BITS TO TEST


;*******************************************************
;     WAIT FOR COMPLETION OF DISK OPERATION.
;     TEST FOR ERRORS.
;*******************************************************
```

```
CHEK:
      EI                ;REENABLE INTERRUPTS
      LDA   D3SNO       ;GET NO WAIT Z3S BYTE
      OUT     DCMD
      STA   OD3S
      CALL  FQDONE            ;WAIT FOR 179X DONE
      ANA   C           ;ANY ERROR BITS(y/n)
      RZ                ;NO, RETURN - SUCCESSFUL


;*****************************************************
;         RETRY THE I/O IF AN ERROR OCCURED.
;*****************************************************

NOPRNT:
      LXI   H,RMACRO    ;POINT TO MACRO RETRY COUNT
      DCR   M           ;DECREMENT IT
      JNZ   MIC         ;RE-DO READ/WRITE

      LXI   H,KBIT
      DCR   M
      JZ    KEYIT

      CALL  FHOME
      RC
      XRA   A
      STA   TRACK
      JMP   MAC


;     IF WE CAME HERE IT IS TIME TO RE-CALIBRATE THE DRIVE

KEYIT:
      MVI   A,4
      STA   KBIT
      LDA   ISTAT       ;GET BACK LAST STATUS
      ANA   C
      CALL  EPRINT

NORE:

;*****************************************************
;THE FOLLOWING ROUTINE RINGS BELL AND WAITS FOR USER
;TO RETRY OPERATION ON DRIVE WITH ERROR
;*****************************************************

      CALL  FHOME
      JC    FATERR
      XRA   A
      STA   TRACK

      CALL  RING
CRIN:
      CALL  CONST       ;SEE IF KEY PRESSED
```

```
      JRZ   CRIN        ;IF NOT KEEP RINGING
      CALL  CONIN       ;GET CHAR
      CPI   3           ;A ^C (y/n)
      JZ    EEXIT       ;DO A WARM BOOT IF SO
      CPI   4           ;A ^D (y/n)
      JRZ   FXER        ;IGNORE ERROR
      CPI   0DH         ;SEE IF 'CR'
      JRNZ  CRIN        ;IF NOT TRY AGAIN
      CALL  BS          ;BACK UP CURSOR
      CALL  PSP         ;NOW 'ERASE' PRINTED ERROR MESSAGE
      CALL  BS          ;AND BACK UP CURSOR AGAIN

      LXI   H,MRML
      DCR   M
      STC
      RZ

      JMP   MAC         ;DO ANOTHER RE-TRY


;IGNORE DISK I/O ERROR AND CONTINUE WITH STATUS SET O.K.

FXER:
      STC               ;FIX UP CARRY
      CMC
      XRA   A           ;CLEAR ERROR FLAG
      RET


RING:
      CALL  MSG         ;RING THE BELL AND DELAY 1 SECOND
      .BYTE 87H
      RET


;BACK UP CURSOR 8 PLACES TO GET TO START OF ERROR MESSAGE

BS:

      MVI   C,8         ;BACKSPACE CHARACTER IN C
BS0:  MVI   B,17
BS1:  CALL  CONOUT
      DCR   B
      JRNZ  BS1
      RET

;PRINT 8 SPACES ON CONSOLE TO 'ERASE' LAST ERROR MESSAGE
;ENTER BACK SPACE ROUTINE AT BS0: TO EXECUTE 8 SPACES

PSP:
      MVI   C,20H       ;SPACE CHARACTER IN C
      JMPR  BS0
```

```
;*******************************************************
; SET Z3S CONTROL BYTE. GET 2 Z3S CONTROL BYTES, ONE
; WITH THE HARDWARE WAIT BIT ACTIVE AND ONE WITHOUT,
; FROM THE 8 POSSIBLE CONTROL BYTES. OUTPUT THE NO WAIT
; BYTE. SAVE BOTH FOR LATER USE.
;*******************************************************

SETD3S:     LDA   PTRACK            ;GET DESIRED TRACK
      LXI   H,OFF
      CMP   M           ;USER TRACK(y/n)
      LXI   H,US0N            ;POINT TO USER TRACK BYTES
      JRNC  USER
      LXI   H,SS0N            ;POINT TO SYSTEM TRACK BYTES

USER: LDA   PSECT       ;TEST IF ON SIDE 1
      BIT   7,A
      JRZ   SID0        ;ON SIDE 0
      INX   H           ;POINT TO SIDE 1 BYTES
      INX   H

SID0: MOV   E,M         ;GET THE NO WAIT BYTE
      INX   H
      MOV   D,M         ;GET THE WAIT BYTE
      SDED  D3SNO       ;SAVE BOTH
      MOV   A,E
      OUT     DCMD            ;OUTPUT THE NO WAIT BYTE
      STA   OD3S
      RET


;*******************************************************
; GET Z3S CONTROL BYTES. CREATE THE EIGHT Z3S CONTROL
; BYTES IN ATABLE. THESE BYTES SET THE DENSITY BIT FOR
; SYSTEM TRACKS AND USER TRACKS, SELECT SIDE 0 OR SIDE
; 1, AND ENABLE OR DISABLE THE HARDWARE DATA WAIT.
; CREATE THE BYTES BY MERGING THE APPROPRIATE HARDWARE
; CONTROL BITS IN OPRTAB WITH THE CORRECT ADDRESS
; CONTROL BITS IN ADRTAB.
;*******************************************************

GETD3S:     LDA   ADISK       ;GET CURRENT DISK NUMBER
      LXI   H,ADRTAB    ;POINT TO ADDRESS
      MOV   E,A         ;CONTROL BITS TABLE
      MVI   D,0
      DAD   D
      MOV   C,M         ;SAVE ADDR BITS FOR Z3S BYTES

      LDA   FLAG        ;GET THE FLAG
      ANI   00010100B   ;ISOLATE SYSTEM DENSITY, SIZE
      MOV   B,A
      RRC               ;ALIGN THE DENSITY BIT
      ORA   B           ;COMBINE SIZE AND DENSITY
      ANI   00001100B   ;DROP UNALIGNED BITS
      LXI   D,SS0N            ;POINT TO BEGINNING OF DEST
```

```
      CALL  GET         ;GET SYSTEM TRACK Z3S BYTES

      LDA   FLAG        ;GET THE FLAG
      ANI   00001100B   ;ISOLATE USER DENSITY, SIZE
                        ;AND GET USER TRACK Z3S BYTES

GET:  LXI   H,OPRTAB    ;POINT TO CONTROL BITS TABLE
      PUSH  D
      MOV   E,A         ;GET CONTROL BITS START ADDRESS
      MVI   D,0
      DAD   D           ;START ADDRESS NOW IN HL
      POP   D           ;DESTINATION RESTORED TO DE
      MVI   B,4
LOOP3:      MOV   A,M         ;GET A BYTE OF CONTROL BITS
      INX   H
      ORA   C           ;COMBINE WITH ADDRESS BITS
      STAX  D           ;STORE COMBINED BYTE
      INX   D
      DJNZ  LOOP3       ;DO ALL BYTES
      RET


;*******************************************************
; DMA READ/WRITE ROUTINE
; SET UP THE DMA CONTROLLER CHIP WITH THE PROPER
; COMMAND,DMA ADDRESS AND BYTE COUNT.
; THE FINAL COMMAND TO THE DISC CONTROLLER CHIP
; STARTS THE DMA PROCESS.
;*******************************************************

DMARW:
      SDED  DEDMA

      MVI   A,WFCINT    ;FORCE INTERRUPT COMMAND TO 179X
      OUT   WCMD

      MVI   A,41H       ;CHANNEL 0 REQUEST
      OUT   CMND

      MOV   A,E         ;BYTE COUNT LSB
      OUT   WCTO
      MOV   A,D         ;BYTE COUNT MSB, ALSO DETERMINES READ OR WRITE
      OUT   WCTO

      MOV   A,L         ;DMA ADDRESS BYTES
      OUT   ADRO
      MOV   A,H
      OUT   ADRO

      LDA   OCMD        ;GET DISK CONTROLLER COMMAND
      OUT   WCMD        ;START DMA PROCESS

      RET
```

```
;*******************************************************
; WAIT FOR TRIM ERASE TO END. WAIT ONLY IF THE LAST
; FLOPPY DISK COMMAND WAS A WRITE. CALLED ONLY IF
; PHYSICAL HEAD MOTION IS NEEDED TO ACCESS THE NEXT
; SECTOR. THE WAIT IS ABOUT 500 USEC AT 4 MHZ. THIS
; ALLOWS TRIM ERASE TO COMPLETE BEFORE THE DRIVE IS
; DESELECTED OR THE HEAD IS MOVED.
;*******************************************************

TRIMWT:     LDA   ORWCMD            ;GET LAST READ OR WRITE COMMAND
      BIT   WBWRIT,A    ;TEST WRITE BIT
      RZ                ;IT WAS A READ, DON'T WAIT
      MVI   B,150       ;WAIT
      DJNZ  .
      RET


;*******************************************************
;* MULTI-PURPOSE 179X SEEK SUBROUTINE. THE ENTRY
;* POINTS ARE:
;*
;*    FHOME - RESTORE HEAD TO TRACK 0 POSITION
;*    FSEEK - SEEK WITH HEAD LOAD TO DEST. IN ACC
;*
;*    FUNLD - SEEK SAME TRACK TO UNLOAD HEAD
;*    FLOAD - SEEK SAME TRACK TO LOAD HEAD
;*
;* FOR THESE LAST TWO FUNCTIONS THE COMMAND IS STILL IN
;* PROGRESS WHEN RETURN IS MADE. THE CALLING PROGRAM
;* MUST WAIT FOR THE COMMAND TO COMPLETE.
;*******************************************************

FHOME:      MVI   B,WHOME     ;SET UP HOME COMMAND
      JMPR  FH

FSEEK:      OUT   WDATA       ;OUTPUT SEEK DESTINATION
      MVI   B,WSEEK           ;SET UP SEEK, DIFFERENT TRACK

FH:   LDA   SPEED       ;GET SEEK SPEED
      ORA   B           ;MERGE WITH SEEK COMMAND
      OUT   WCMD        ;OUTPUT TO 179X
      STA   OCMD
      CALL  FDONE
      ANI   WSSEEK            ;ELIMINATE UNWANTED STATUS BITS


;*******************************************************
;    NOW GO IN A WAIT LOOP FOR HEAD SETTLE TIME
;*******************************************************

SETL:   PUSH      PSW         ;SAVE FLAGS

      LDA   PCMD
```

```
      BIT   5,A         ;SEE IF READ OR WRITE
      JRZ   NDLY

      MVI   A,20        ;WAIT 20 MS
SETL1:  DJNZ      SETL1       ;LOOP ON B REG
        DCR     A             ;LOOP ON SETTLE VALUE
        JRNZ    SETL1
NDLY:
        POP PSW         ;GET BACK REGS
      RZ                ;RETURN - SUCCESSFUL
      STC
      RET               ;RETURN WITH CARRY SET - ERROR

FUNLD:      MVI   B,WUNLD           ;SET UP UNLOAD COMMAND
      JMPR  FU

FLOAD:      MVI   B,WLOAD           ;SET UP LOAD COMMAND

FU:   IN    WTRACK            ;GET CURRENT TRACK
      OUT   WDATA       ;OUTPUT SEEK DESTINATION

      MOV   A,B         ;OUTPUT SEEK COMMAND
      OUT   WCMD
      RET


;*******************************************************
; 179X NOT BUSY SUBROUTINE. WAIT FOR 179X NOT BUSY.
; THEN RETURN THE LAST STATUS READ FROM THE CHIP. THERE
; ARE TWO ENTRY POINTS. FDONE DELAYS A SHORT WHILE TO
; ALLOW THE 179X TO SET ITS BUSY BIT.
; THIS ROUTINE TESTS FOR TYPE 1 AND TYPE 2
; COMMAND COMPLETION.
;*******************************************************

FDONE:

      MVI   B,10        ;DELAY
      DJNZ  .


FQDONE:
      LDA   DMAFLG
      ORA   A
      JRZ   PIODON


      IN    DMACHK            ;WAIT FOR DMA TO FINISH
      RLC
      JRC   FQDONE

DONE:
      IN    WSTAT       ;GET STATUS FROM DISC CONTROLLER
```

```
      BIT   WBBUSY,A    ;TEST BUSY BIT
      JRNZ  DONE
      STA   ISTAT
      RET

PIODON:
      IN    WSTAT
      STA   ISTAT
      BIT   WBBUSY,A
      JRNZ  PIODON
      RET



;*********************************************************
; FLOPPY DISK ERROR PRINT SUBROUTINE. ANY NON ZERO BITS
; IN THE ACCUMULATOR ARE ERRORS. THE FIRST ONE FOUND IS
; PRINTED OUT.
;*********************************************************

EPRINT:
      PUSH  PSW         ;SAVE ERROR BITS
      IN    WTRACK            ;GET BAD TRACK
      STA   ITRACK
      IN    WSECT       ;GET BAD SECTOR
      STA   ISECT

EMSROT:
      CALL  MSG
      .ASCII      'Err'
      .BYTE ' '+80H
      LDA   SEKDSK
      ADI   'A'
      MOV   C,A
      CALL  BIOOUT
      CALL  MSG
      .BYTE ':',' '+80H

      POP   PSW

      CALL  LBYTE

      MVI   C,' '
      CALL  BIOOUT
      LDA   ITRACK
      CALL  LBYTE
      MVI   C,' '
      CALL  BIOOUT
      LDA   ISECT
      CALL  LBYTE
      MVI   C,' '
      CALL  BIOOUT
      LDA   SIDEID           ;GET SIDE NUMBER
```

```
      MOV   C,A
      JMP   BIOOUT

;*********************************************************
;             PRINT HEX BYTE ON CONSOLE.
;*********************************************************

LBYTE:      PUSH  PSW
      RRC
      RRC
      RRC
      RRC
      CALL  P2
      POP   PSW
P2:   ANI   0FH
      ADI   90H
      DAA
      ACI   40H
      DAA
      MOV   C,A
      JMP   BIOOUT


;*********************************************************
; MOVE ATABLE INTO THE CORRECT APB. MOVE THE CORRECT
; APB INTO ATABLE.
;*********************************************************

PUTAPB:     LDA   ADISK       ;GET CURRENT ATABLE NUMBER
      CALL  ASET        ;SET UP FOR MOVE
      LDIR              ;COPY ATABLE INTO APB
      RET

GETAPB:     LXI   H,ADISK           ;POINT TO ATABLE DRIVE NUMBER
      LDA   PDISK       ;GET NEW NUMBER
      CMP   M           ;THE SAME(y/n)
      RZ                ;YES, ATABLE ALREADY VALID
      MOV   M,A         ;UPDATE ATABLE NUMBER
      CALL  ASET        ;SET UP FOR MOVE
      XCHG              ;APB ADDRESS NOW SOURCE
      LDIR              ;COPY APB INTO ATABLE

;NOW GET ALLOCATION SIZE FOR SELECTED DISK

      LDA   PDISK       ;GET DISK #
      MOV   E,A         ;MAKE INDEX
      MVI   D,0
      LXI   H,ALOCSZ    ;POINT TO SAVED BSH TABLE
      DAD   D
      MOV   A,M         ;GET BSH VALUE
      LXI   H,ALOREC    ;POINT TO RECORDS/ALLOCATION SIZE
      SUI   3           ;SET FOR INDEX
      MOV   E,A         ;MAKE INDEX
      DAD   D
```

```
      MOV   A,M         ;GET RECORDS/ALLOCATION
      STA   ALOCA       ;SAVE # FOR SELECTED DRIVE

      RET

;TABLE FOR # OF RECORDS FOR ALLOCATION SIZES FROM 1K TO 16K

ALOREC:
      .BYTE 8,16,32,64,128


ASET: MOV   L,A         ;GET DISK NUMBER INTO L
      CALL  GETDPH
      LXI   H,ATABLE
      LXI   B,ALEN
      RET


;*******************************************************
; GET DPH ADDRESS AND APB ADDRESS. RETURN THE DPH
; ADDRESS IN HL, THE CORRESPONDING APB ADDRESS IN DE.
; DISK NUMBER MUST BE IN L AT CALL.
;*******************************************************

GETDPH:     MVI   H,0
      DAD   H           ;GET NUMBER * 2
      MOV   E,L
      MOV   D,H
      DAD   H
      DAD   H
      DAD   H           ;GET NUMBER * 16
      DAD   D           ;GET NUMBER * 18
      LXI   D,APBBEG
      DAD   D           ;NOW WE HAVE ADDR OF APB ADDR
      MOV   E,M         ;GET APB ADDRESS INTO DE
      INX   H
      MOV   D,M
      INX   H           ;HL NOW HAS DPH ADDRESS
      RET


;*******************************************************
;                  FATAL ERROR.
;*******************************************************

FATERR:
QUIT: CALL  MSG
      .ASCII      ' ST'
      .BYTE 'P'+80H
      JMP   EEXIT       ;GO WARM BOOT


;*******************************************************
; MESSAGE OUTPUT SUBROUTINE. THERE ARE TWO ENTRY
```

```
; POINTS. FOR MSG A MESSAGE FOLLOWS INLINE AFTER THE
; CALL. FOR MSGHL THE MESSAGE ADDRESS IS IN HL. FOR
; BOTH ENTRY POINTS THE MESSAGE IS PRINTED OUT UP TO
; AND INCLUDING A CHARACTER WITH ITS HI ORDER BIT SET.
;*****************************************************

MSG:  XTHL              ;GET MESSAGE ADDRESS
      CALL  MSGHL       ;PRINT THE MESSAGE
      XTHL              ;RESTORE NEW RETURN ADDRESS
      RET

MSGHL:      MOV   C,M         ;GET NEXT CHAR OF MESSAGE
      INX   H
      CALL  BIOOUT            ;OUTPUT THE CHAR
      ORA   A           ;TEST FOR HI BIT SET
      JP    MSGHL       ;NOT SET, KEEP GOING
      RET



;*****************************************************
; I/O VECTORING ROUTINES. THESE ROUTINES USE THE IOBYTE
; TO SELECT THE PHYSICAL DEVICE WHICH WILL CORRESPOND
; TO THE CONSOLE OR TO THE LIST OUTPUT. THE JUMPS TO
; THE PHYSICAL DEVICES SHOULD BE CHANGED HERE TO USE
; EXTERNAL I/O. THIS RETAINS IOBYTE CONTROL OVER THE
; ROUTINES.
;*****************************************************


;*****************************************************
;    I/O ROUTINES FOR THE TTY PHYSICAL DEVICE.
;*****************************************************

TTYIS:      IN    TTYS        ;GET STATUS
      ANI   TTYDA       ;TEST FOR READ DATA AVAILABLE
      RZ                ;RETURN FALSE IF UNAVAILABLE
      MVI   A,0FFH
      RET               ;RETURN TRUE

TTYIN:      CALL  TTYIS       ;GET STATUS
      JRZ   TTYIN       ;WAIT UNTIL DATA AVAILABLE
      IN    TTYI        ;GET THE DATA
      ANI   07FH        ;DROP PARITY
      RET

TTYOS:      IN    TTYS        ;GET STATUS
      ANI   TTYBE       ;TEST FOR TRANSMIT BUFFER EMPTY
      RZ                ;RETURN FALSE IF NOT EMPTY
      MVI   A,0FFH
      RET               ;RETURN TRUE

TTYOUT:     CALL  TTYOS       ;GET STATUS
      JRZ   TTYOUT            ;WAIT UNTIL BUFFER EMPTY
```

```
      MOV   A,C
      OUT   TTYO        ;OUTPUT THE DATA
      RET



;*******************************************************
; I/O ROUTINES FOR THE CRT PHYSICAL DEVICE.
; THIS IS OUR MODEM PORT .SETTING THE CON: TO CTR:
; (MODEM) PUTS THE I/O IN A LOOP TO ENABLE CONTROL
; OF THE SYSTEM FROM THE DEVICE CALLING ON THE MODEM
; ALL CHARACTERS RECEIVED ARE ECHOD BACK TO THE CALLING
; DEVICE AND ONTO THE SCREEN OF THE CAPTIVE SYSTEM
; A NULL CHARACTER PUT ON THE LOCAL KEYBOARD WILL RESET
; THE IOBYTE TO TTY AND CAUSE A WARM BOOT
;*******************************************************

CRTIS:      IN    TTYI        ;TEST LOCAL KEYBOARD
      ANI   7EH         ;MASK IT FOR 'CONTROL-A'
      JZ    FIXER       ;IF NULL GO RESET IOBYTE
                        ;AND DO A WARM BOOT
      IN    CRTS        ;GET MODEM STATUS
      ANI   CRTDA       ;TEST FOR READ DATA AVAILABLE
      RZ                ;RETURN FALSE IF UNAVAILABLE
      MVI   A,0FFH
      RET               ;RETURN TRUE

READER:
CRTIN:      CALL  CRTIS       ;GET STATUS
      JRZ   CRTIN       ;WAIT UNTIL DATA AVAILABLE
      IN    CRTI        ;GET THE DATA
      ANI   07FH        ;DROP PARITY
      RET

CRTOS:      IN    CRTS        ;GET STATUS
      ANI   CRTBE       ;TEST FOR TRANSMIT BUFFER EMPTY
      RZ                ;RETURN FALSE IF NOT EMPTY
      MVI   A,0FFH
      RET               ;RETURN TRUE
PUNCH:
CRTOUT:     CALL  CRTOS       ;GET STATUS
      JRZ   CRTOUT            ;WAIT UNTIL BUFFER EMPTY
      MOV   A,C
      OUT   CRTO        ;OUTPUT THE DATA
TUBE: IN      0         ;NOW TEST THE LOCAL SCREEN
      ANI   4           ;FOR A CLEAR STATUS
      JRZ   TUBE
      MOV   A,C         ;AND SEND THE CHARACTER
      OUT   1           ;TO IT BEFORE EXITING
            RET


;*******************************************************
; IO ROUTINES FOR THE OTHER IOBYTE VECTORED DEVICES
```

```
;******************************************************

CONST:      LDA   IOBYTE            ;GET THE CURRENT I/O BYTE
      ANI   CMSK        ;DROP IRRELEVANT BITS
      JRZ   TTYIS       ;TTY SELECTED
      CPI   1           ;TEST CRT (MODEM)
      JRZ   CRTIS       ;CRT SELECTED(MODEM)
      CPI   2           ;TEST CONTROL REMOTE
      JRZ   CRTIS       ;THIS WILL DO
      JMP   TTYIS       ;DO UC1:

CONIN:      LDA   IOBYTE
      ANI   CMSK
      JRZ   TTYIN
      CPI   1           ;TEST CRT(MODEM)
      JRZ   CRTIN
      CPI   2           ;TEST REMOTE
      JRZ   CRTIN       ;THIS WILL DO
      JMPR  TTYIN       ;DO UC1:

CONOUT:
      .IFN  SLOW,[

      XRA   A
SLOWIT:
      DCR   A
      JNZ   SLOWIT
      ]

      LDA   IOBYTE
      ANI   CMSK
      JRZ   TTYOUT
      CPI   1           ;TEST CRT(MODEM)
      JRZ   CRTOUT
      CPI   2           ;TEST REMOTE
      JRZ   CRTOUT      ;THIS WILL DO
      JMPR  CRTOUT            ;DO UC1:


;******************************************************
;       USUAL PRINTER I/O HANDLED NEXT
;******************************************************

LISTST:     LDA   IOBYTE
      ANI   LMSK
      JRZ   CRTOS       ;CHECK MODEM OUT STAT
      CPI   40H
      JRZ   CENST       ;CHECK L.P. STAT
      CPI   80H
      JRZ   SPINST            ;CHECK SPIN STAT
      JMPR  SERPST            ;CHECK SERIAL STAT


;******************************************************
```

```
; FIND OUT WHICH PRINTER IS SET IN IOBYTE AND OUTPUT
; TO IT
;*******************************************************

LIST: LDA   IOBYTE
      ANI   LMSK
      JRZ   CRTOT1            ;IS IT MODEM (y/n)
      CPI   40H         ;IS IT CENT (y/n)
      JRZ   LISTC
      CPI   80H         ;IS IT CENTRONICS (y/n)
      JRZ   SPIN0
      JMPR  SERP        ;IS IT SERIAL (y/n)


;*******************************************************
;            OUTPUT TO CENTRONICS
;*******************************************************

LISTC:      IN    LSTST       ;SEE IF CPU 'C'
      ANI   00110000B
      JRZ   LIST1
      OUT   9           ;TURN ON CENT MUX
LIST1:
      IN      LSTST     ;GET PRINTER STATUS
      RAR               ;PUT IT IN CARRY
        JRC     LIST1         ;BUSY ON C SO RETRY
        MOV     A,C           ;GET OUTBYTE
        OUT     LSTDAT        ;SEND TO PRINTER
        RET


;*******************************************************
;        SERIAL PRINTER I/O HANDLED HERE
;*******************************************************

SERP: IN    SERST       ;GET SERIAL STATUS
      RAL               ;PUT IN CARRY
      JRNC  SERP        ;BUSY SO RETRY
      MOV   A,C         ;GET OUTBYTE
      OUT   SERDT       ;SEND TO SERIAL PRINTER
      RET


;*******************************************************
;      LETTER QUALITY PRINTER I/O HANDLED HERE
;*******************************************************

SPIN0:
      IN    LSTST       ;GET STATUS
      RAL               ;PUT BIT 7 IN CARRY
      JRC   SPIN0       ;BUSY IF CARRY SET

;NOW CHECK C.P.U. TYPE AND OUTPUT TO CORRECT PORT
```

```
      IN    LSTST       ;SEE IF CPU 'C'
      ANI   00110000B   ;TEST BIT 6 , 0 = CPU 'C'
      MOV   A,C         ;GET LIST BYTE
      JRZ   SPIN2       ;GO HERE FOR CPU 'C'
      OUT   SPIN        ;TURN ON SPIN MUX
      OUT   LSTDAT            ;SEND TO SPIN WRITER
SPIN2:      OUT   LSTST       ;PORT 6 FOR CPU 'C'
      RET


;*********************************************************
;       MODEM AS LIST DEVICE FROM HERE
;*********************************************************

CRTOT1:
      CALL  CRTOS       ;OUTPUT TO MODEM
      JRZ   CRTOT1
      MOV   A,C
      OUT   CRTO
      RET



;*********************************************************
;      GET STATUS FOR VARIOUS LIST DEVICES
;*********************************************************

;SPIN WRITER STATUS

SPINST:     IN    LSTST       ;GET STAT
      CMA
      ANI   80H
      RZ                ;SHOW IT IS BUSY
      MVI   A,0FFH            ;SHOW IT IS READY
      RET

;LINE PRINTER STATUS

CENST:      IN    LSTST
      CMA
      ANI   1
      RZ
      MVI   A,0FFH
      RET

;SERIAL PRINTER STATUS

SERPST:     IN    SERST
      ANI   80H
      RZ
      MVI   A,0FFH
      RET

;GET OUT OF SLAVE MODE AND WARM BOOT
```

```
FIXER:      LDA   3           ;GET I/O BYTE
      ANI   0FCH        ;SET TO TTY
      STA   3           ;FIX IT UP THERE
      JMP   0           ;GO WARM BOOT

;*****************************************************
; FLOPPY DISK DPH TABLE. FOR THE CONVENIENCE OF THE
; CBIOS THE APB ADDRESS FOR A DISK PRECEDES THE DPH
; FOR THE DISK.
;*****************************************************

APBBEG:
      .WORD APB0
      .WORD TRANS0            ;LOGICAL TO PHYSICAL XLATE TAB
      .WORD 0           ;SCRATCH
      .WORD 0
      .WORD 0
      .WORD DIRBUF            ;DIRECTORY BUFFER
      .WORD DPB0        ;DISK PARAMETER BLOCK
      .WORD CSV0        ;CHECKSUM VECTOR
      .WORD ALV0        ;ALLOCATION VECTOR

      .WORD APB1
      .WORD TRANS1
      .WORD 0
      .WORD 0
      .WORD 0
      .WORD DIRBUF
      .WORD DPB1
      .WORD CSV1
      .WORD ALV1

      .IFN  STDNR,[

      .WORD APB2
      .WORD TRANS2
      .WORD 0
      .WORD 0
      .WORD 0
      .WORD DIRBUF
      .WORD DPB2
      .WORD CSV2
      .WORD ALV2

      .WORD APB3
      .WORD TRANS3
      .WORD 0
      .WORD 0
      .WORD 0
      .WORD DIRBUF
      .WORD DPB3
      .WORD CSV3
      .WORD ALV3
```

```
      ]

;*********************************************************
; STANDARD APBS, DPBS AND TRANSLATE TABLES. THESE
; TABLES ARE USED BY SELDSK WHEN IT CANNOT FIND A VALID
; DDB ON THE DISK. THE TABLES ARE FOR STANDARD 8" OR 5"
; SINGLE DENSITY FLOPPY DISKS. THIS ALLOWS PROGRAM
; INTERCHANGE WITH OTHER CP/M BASED SYSTEMS WITHOUT
; REQUIRING A DDB TO BE WRITTEN ON EACH DISK.
;*********************************************************



;*********************************************************
;             8" FLOPPY DISK TABLES.
;*********************************************************

STDDDB:
      .BYTE 00000001B   ;FLAG
      .WORD 2           ;OFF
      .BYTE 0           ;SSLEN
      .BYTE 26          ;SLRPS
      .BYTE 0           ;USLEN
      .BYTE 26          ;ULRPS

      .WORD 26          ;STANDARD DPB
      .BYTE 3,7,0
      .WORD 242
      .WORD 63
      .BYTE 192,0
      .WORD 16
      .WORD 2

      .BYTE 1,7,13,19,25,5,11,17,23
      .BYTE 3,9,15,21,2,8,14,20,26
      .BYTE 6,12,18,24,4,10,16,22



;*********************************************************
;             5" FLOPPY DISK TABLES.
;*********************************************************

.IFN  STDNR,[

ALTDDB:     .BYTE 00000101B   ;FLAG
      .WORD 3           ;OFF
      .BYTE 0           ;SSLEN
      .BYTE 18          ;SLRPS
      .BYTE 0           ;USLEN
      .BYTE 18          ;ULRPS

      .WORD 18          ;STANDARD DPB
```

```
      .BYTE 3,7,0
      .WORD 71
      .WORD 63
      .BYTE 192,0
      .WORD 16
      .WORD 3

      .BYTE 1,5,9,13,17,3,7,11,15
      .BYTE 2,6,10,14,18,4,8,12,16

      ]

;     TEST FOR GOOD DATA ON LAST WRITE

CHKSEC:
      POP   PSW   ;FIX STACK
      MVI   A,10  ;10 READ RETRIES
      STA   CHKRDS

RECHK:
      LXI   H,TRACK     ;GET OLD TRACK
      MOV   A,M
      OUT   WTRACK      ;SET IT
      LDA   PTRACK      ;GET PHYSICAL TRACK
      CMP   M     ;TEST WITH OLD
      JRZ   TRKOK ;IF SAME O.K.
      MOV   M,A   ;ELSE UPDATE
      CALL  FSEEK ;SEEK TO IT
      RC          ;REAL BAD

TRKOK:
      DI
      LDA   PSECT ;GET SECTOR
      MOV   B,A
      ANI   7FH   ;DROP SIDE BIT
      OUT   WSECT ;SET IT
      LDA   D3SWT
      OUT   DCMD

      LDA   DMAFLG
      ORA   A
      JNZ   DMARAW
      MVI   A,WREAD     ;READ COMMAND

      BIT   7,B   ;TEST SIDE
      JRZ   SID011      ;IF SIDE 0
      SET   WBSID1,A;SET SIDE 1 COMPARE
SID011:
      OUT   WCMD  ;SET PIO FCD IN MOTION


      LXI   B,128<8+WDATA ;SET SECTOR LENGTH AND PORT

      LDA   PTRACK      ;GET DESIRED TRACK
```

```
      LXI   H,OFF ;POINT TO OFFSET
      CMP   M     ;TEST IT
      LDA   USLEN ;ASSUME USER TRACK
      JRNC  ULEN1 ;IF IT IS USER
      LDA   SSLEN ;ELSE GET SYSTEM SCTOR LENGTH

ULEN1:
      ORA   A     ;TEST IT FOR 128 BYTES
      JRZ   R128C ;IF SO READ 128 BYTES
      MVI   B,0   ;MUST BE 256, 512 OR 1024 BYTES
      DCR   A     ;TEST FOR 256
      JRZ   R256C ;IF SO
      DCR   A     ;TEST FOR 512
      JRZ   R512C ;IF SO
TEN24:
      INP   A
      DCR   B
      JRNZ  TEN24
TEN241:
      INP   A
      DCR   B
      JRNZ  TEN241
R512C:
      INP   A
      DCR   B
      JRNZ  R512C
R128C:
R256C:
      INP   A
      DCR   B
      JRNZ  R256C

RDFNC:
      EI
      LDA   D3SNO
      OUT   DCMD
      STA   OD3S
      CALL  FQDONE

      ANI   WSREAD
      RZ
      LXI   H,CHKRDS
      DCR   M
      JNZ   RECHK

      JMP   NOPRNT

DMARAW:
      MVI   A,10
      OUT   0FFH
      LDED  DEDMA       ;GET LAST DE VALUE FOR DMA
      MOV   A,D         ;GET COMMAND & HI WORD COUNT
      ANI   00111111B   ;STRIP COMMAND
      ORI   01000000B   ;MEGRE IN RD COMMAND
```

```
      MOV   D,A         ;MAKE NEW D VALUE

      MVI   A,WFCINT
      OUT   WCMD
      MVI   A,41H
      OUT   CMND
      MOV   A,E
      OUT   WCTO
      MOV   A,D
      OUT   WCTO
      XRA   A
      OUT   ADRO
      OUT   ADRO
      MVI   A,WREAD
      OUT   WCMD
      XRA   A
DMALP:
      DCR   A
      JRNZ  DMALP
DMACK:
      IN    DMACHK
      ANI   80H
      JNZ   DMACK
      XRA   A
      OUT   0FFH
      JMPR  RDFNC


;*********************************************************
;        SHOW HOW MUCH DISK SPACE IS LEFT
;*********************************************************

      .IFG  (.-BIOS)-(NSBIOS*128),[
      .PRNTX      \BIOS EXCEEDS DISK SPACE \
      ]

      .DEFINE     SBOND[XX]=[.PRNTX /XX BYTES OF DISK LEFT/
      ]
      .DEFINE SBOD[XX]=[.PRNTX /XX BYTES BEYOND DISK SPACE/
      ]
      .IF1,[
      .IFG  (NSBIOS*128)-(.-BIOS),[
      SBOND \(NSBIOS*128)-(.-BIOS)
      ][
      SBOD  \(.-BIOS)-(NSBIOS*128)
      ]
      ]



;*********************************************************
; END OF BIOS INSTRUCTIONS AND CONSTANTS. BEGINNING
; OF WORK AREA.
;*********************************************************
```

```
;*********************************************************
; CP/M CALL PARAMETER STORAGE. CP/M SETS THE LOGICAL
; PARAMETERS BY PRELIMINARY CALLS BEFORE CALLING THE
; READ OR WRITE ROUTINES. THE CP/M I/O ROUTINES SET
; AND USE THE REMAINING VARIABLES.
;*********************************************************

DMAFLG:     .BLKB 1           ;CONTROLLER TYPE FLAG
SEKSEC:     .BLKB 2           ;LOGICAL SECTOR TO R/W
DMAADD:     .BLKB 2           ;LOGICAL & ACTUAL CP/M DMA ADDR
CREC: .BLKB 1           ;CURRENT RECORD WITHIN SECTOR


;*********************************************************
; PHYSICAL DISK I/O PARAMETER STORAGE. THESE PARAMETERS
; MUST BE SET BEFORE CALLING THE FLOPPY DISK PHYSICAL
; READ OR WRITE ROUTINES.
;*********************************************************

PDISK:      .BLKB 1           ;PHYSICAL DISK FOR NEXT I/O
PTRACK:     .BLKB 2           ;PHYSICAL TRACK FOR NEXT I/O
PSECT:      .BLKB 2           ;PHYSICAL SECTOR FOR NEXT I/O
PDMA: .BLKB 2           ;PHYSICAL BUFFER ADDR, NEXT I/O

CTRACK:     .BLKB 2
CSECT:      .BLKB 2



;*********************************************************
;    GENERAL PURPOSE VARIABLES ARE STORED HERE.
;*********************************************************

RETRY:
RMICRO:     .BLKB 1           ;MICRO FDC RETRY COUNT
RMACRO:     .BLKB 1           ;MACRO FDC RETRY COUNT
ADISK:      .BLKB 1           ;DISK NUMBER OF DISK AT ATABLE
OLDFLO:     .BLKB 1           ;OLD FLOPPY DRIVE NUMBER
PCMD: .BLKB 1           ;ACTUAL FLOPPY READ/WRITE CMD
ORWCMD:     .BLKB 1           ;LAST R/W OUTPUT TO CMD REG
DPHADR:     .BLKB 2           ;DPH ADDRESS FOR CURRENT DISK
APBADR:     .BLKB 2           ;APB ADDRESS FOR CURRENT DISK
WRTYPE:     .BLKB 1           ;TYPE OF WRITE
D3SNO:      .BLKB 1           ;CURRENT NO WAIT Z3S BYTE
D3SWT:      .BLKB 1           ;CURRENT WAIT Z3S BYTE
SAVADR:     .BLKB 2           ;STD OR ALT DDB ADDRESS
MRML: .BLKB 1           ;MACRO RETRY MAJOR LOOP


;*********************************************************
; ERROR INFORMATION STORAGE. VARIOUS ROUTINES USE THIS
; AREA TO STORE KEY VARIABLES RELATING TO THE Z3S
```

```
; CONTROLLER AND THE 179X CHIP.
;*****************************************************

EINFO:
ITRACK:     .BLKB 1           ;LAST INPUT FROM TRACK REG
ISECT:      .BLKB 1           ;LAST INPUT FROM SECTOR REG
ISTAT:      .BLKB 1           ;LAST INPUT FROM STATUS REG
ACTIVE:     .BLKB 1           ;SHOWS WHICH ROUTINE HAD ERROR
OD3S: .BLKB 1           ;LAST OUTPUT TO Z3S CTRL BYTE
OCMD: .BLKB 1           ;LAST OUTPUT TO CMD REG
SIDEID:     .BLKB 1           ;LASE SIDE SELECTED

EILEN ==    .-EINFO



;*****************************************************
;*   ALLOCATION VALUES FOR SELECTED DISKS ARE NEXT
;*****************************************************

ALOCSZ:     .BLKB 4           ;BSH VALUE FOR 4 DRIVES
ALOCA:      .BLKB 1           ;RECORDS/BLOCK FOR SELECTED DRIVE

      ;UNALLOCATED PARAMETERS FOLLOW


UNACNT:     .BLKB 1           ;UNALLOCATED RECORD COUNT

UNADSK:     .BLKB 1           ;      "     DISK #
UNATRK:     .BLKB 2           ;      "     TRACK #
UNASEC:     .BLKB 2           ;      "     SECTOR #

NEWSEC:     .BLKB 2           ;CP/M SET SECTOR

RSFLAG:     .BLKB 1           ;READ SECTOR FLAG
READOP:     .BLKB 1           ;READ = 0, WRITE = 1
HSTACT:     .BLKB 1           ;0 = HOST NOT ACTIVE
HSTWRT:     .BLKB 1           ;HOST WRITTEN FLAG


HSTDSK:     .BLKB 1
HSTTRK:     .BLKB 2
HSTSEC:     .BLKB 2

SEKDSK:     .BLKB 1
SEKTRK:     .BLKB 2
SEKHST:     .BLKB 2

      UNALEN      ==    .-UNACNT


;*****************************************************
; APB COPY FOR CURRENT DISK. THE APB CONTAINS SEVERAL
; IMPORTANT BYTES NEEDED TO CONTROL THE ACTIVE DISK
```

```
; DRIVE. THE EXACT LENGTH AND ORDERING OF THESE ENTRIES
; IS CRITICAL, SO CHANGES MUST BE MADE WITH CARE. ONLY
; THE FLAG BYTE HAS ANY MEANING FOR A HARD DISK.
;*********************************************************

ATABLE:
SS0N: .BLKB 1           ;SYSTEM TRACK Z3S CONTROL BYTES
SS0W: .BLKB 1
SS1N: .BLKB 1
SS1W: .BLKB 1
US0N: .BLKB 1           ;USER TRACK Z3S CONTROL BYTES
US0W: .BLKB 1
US1N: .BLKB 1
US1W: .BLKB 1
TRACK:      .BLKB 1           ;CURRENT TRACK
SPEED:      .BLKB 1           ;DRIVE SEEK SPEED
FLAG: .BLKB 1           ;FLAG BYTE
OFF:  .BLKB 2           ;NUMBER OF SYSTEM TRACKS
SSLEN:      .BLKB 1           ;SECTOR LENGTH, SYSTEM
SLRPS:      .BLKB 1           ;RECORDS PER SIDE, SYS TRACKS
USLEN:      .BLKB 1           ;SECTOR LENGTH, USER TRACKS
ULRPS:      .BLKB 1           ;RECORDS PER SIDE, USER TRACKS

ALEN  ==    .-ATABLE    ;ATABLE LENGTH

CHKRDS:     .BLKB 1           ;READ CHECK RETRIES
DEDMA:      .BLKB 2           ;DE FOR DMA OP.


;*********************************************************
; FLOPPY DISK APBS, DPBS AND TRANSLATE TABLES. THE APB
; FOR EACH DRIVE IS HERE, FOLLOWED BY THE DPB, FOLLOWED
; BY THE TRANSLATE TABLE. EXTERNAL ROUTINES NEEDING TO
; ACCESS THE APB ASSUME THAT IT IMMEDIATELY PRECEDES
; THE DPB.
;*********************************************************

APB0: .BLKB ALEN        ;AUXILIARY PARAMETER BLOCK
DPB0: .BLKB 15          ;DISK PARAMETER BLOCK

DPBLEN      ==    .-DPB0

TRANS0:     .BLKB 72          ;TRANSLATE TABLE

TRALEN      ==    .-TRANS0
APBDIS      ==    .-APB0


APB1: .BLKB ALEN
DPB1: .BLKB 15
TRANS1:     .BLKB 72


      .IFN  STDNR,[
```

```
APB2: .BLKB ALEN
DPB2: .BLKB 15
TRANS2:     .BLKB 72


APB3: .BLKB ALEN
DPB3: .BLKB 15
TRANS3:     .BLKB 72

      ]



;*********************************************************
;       SECTOR READ AND WRITE BUFFERS.
;*********************************************************

HSTBUF:
RDBUFF:
WRBUFF:
      .BLKB 1024


;*********************************************************
; CP/M WORK AREA. USED BY CP/M FOR DIRECTORY
; OPERATIONS, FLOPPY DISK ALLOCATION VECTORS,
; AND FLOPPY DISK CHANGED DISK CHECKSUMS.
;*********************************************************

DIRBUF:     .BLKB 128         ;DIRECTORY OPERATION BUFFER

ALV0: .BLKB 86          ;ALLOCATION VECTOR
CSV0: .BLKB 64          ;CHECKSUM VECTOR

ALV1: .BLKB 86
CSV1: .BLKB 64


      .IFN  STDNR,[

ALV2: .BLKB 86
CSV2: .BLKB 64

ALV3: .BLKB 86
CSV3: .BLKB 64
      ]


; END OF WORK AREA.


      .IF1,[
```

```
;//MODIFIED TO PRINT HOW FAR OUT YOU ARE.
      .DEFINE BADMEM[XX]=[
      .PRNTX      /BIOS EXCEEDS MEMORY SIZE BY XX BYTES/    ]
      .DEFINE OKMEM[XX]=[
      .PRNTX      /XX BYTES OF MEMORY LEFT BY BIOS/   ]
      .DEFINE LEFMEM[XX]=[
      .PRNTX      /XX IS 1st FREE ADDRESS AFTER BIOS/ ]
      .IFG  (.-BIOS)-(6*256+MEMFCT*1024),[
      BADMEM      \((.-BIOS)-(6*256+MEMFCT*1024))
][
      OKMEM \((6*256+MEMFCT*1024)-(.-BIOS))
      .RADIX      16
      LEFMEM      \.
]
      ]



      .END
```