```
;*****************************************************
;* XM8.ASM
;* 5-1/4" WINCHESTER XIOS
;* FOR DTC SCSI & MDRIVE
;* FLOPPY DRIVES ADDED
;* BRUCE JONES
;* APRIL 6, 1989
;* 16:00
;*****************************************************


      .PABS
      .PHEX
      .Z80
      .XLINK
      .XSYM



TRUE        ==    0FFH
FALSE       ==    0
JUMP        ==    0C3H


;.INCLUDE   LOCAT


LOC   =\    \ENTER 0 FOR .LOC 0 OR 1 FOR .LOC 100H         \

DATCAC      ==    0;\   \0 FOR DATA CACHE, 1 IF NOT                  \

QEXIT ==    1;\   \0 FOR QUIET EXIT, 1 IF NOT                    \

SCACHE      ==    0;\   \0 FOR CACHE, 1 IF NOT                       \

FILLER      ==    1;\   \NUMBER OF FILLER BYTES                      \

BANKED      ==    0;\   \ENTER 0 COMMON BUFFER, 1 IF BANKED          \

DPOL  ==    0;\   \ENTER 0 FOR DISK POLL, 1 IF NONE              \

CLOCK ==    1;\   \0 FOR 60 TICKS, 1 FOR 150 TICKS               \


CONNR ==    7

BNKUNT      ==    8

CONLST      ==    0;\   \ENTER 0 FOR CON 2 = CON, 1 IF LIST DEV       \

MODCON      =\    \ENTER 0 FOR MODEM USE, 1 FOR CONSOLE USE      \
```

```
FLOPPY      ==    0     ;ACTUALLY NO FLOPPY DRIVES

FLOP  ==    1     ;NO FLOPPY FOR UNITXIOS


;*****************************************************
;*        DTC HARD DISK SUBSYSTEM EQUATES.
;*****************************************************

DIO   ==    0A0H        ;DISK I/O PORT

;     BUS STATUS         7   6   5   4   3   2   1   0
;                        |   |   |   |   |   |   |
REQ   ==    080H  ;<-------+   |   |   |   |   |   |
DIROUT      ==    040H  ;<-----------+   |   |   |   |   |
MSSG  ==    020H    ;<---------------+   |   |   |   |
WCMND ==    010H  ;<--------------------+   |   |   |
BUSY  ==    008H  ;<------------------------+   |   |
PTERR ==    004H  ;<----------------------------+   |
AVINT ==    002H  ;<--------------------------------+


;     CONTROL REGISTER   7   6   5   4   3   2   1   0
;                            |       |   |   |   |   |
SLCT  ==    040H  ;<-----------+       |   |   |   |   |
;INTERRUPT ENABLE  <--------------------+   |   |   |   |
;REQ INTERRUPT ENABLE    <------------------------+   |   |   |
;DMA INTERRUPT ENABLE    <----------------------------+   |   |
DODTA ==    002H  ;<--------------------------------+   |
EDMA  ==    001H  ;<------------------------------------+




;     CONTROLLER STATUS BYTE MASKS.

CERR  ==    2           ;CONTROLLER ERROR
PERR  ==    1           ;PARITY ERROR ON BUS
FERR  ==    PERR ! CERR       ;EITHER ERROR


;     CONTROLLER COMMANDS.

RCCMD ==    001H        ;RECALIBRATE DISK CONTROLLER.
STCMD ==    000H        ;STATUS COMMAND
ESCMD ==    003H        ;ERROR SENSE COMMAND
FDCMD ==    004H        ;FORMAT DISK COMMAND
FTCMD ==    006H        ;FORMAT TRACK COMMAND
RDCMD ==    008H        ;READ DISK COMMAND
WTCMD ==    00AH        ;WRITE DISK COMMAND

RDDMA ==    005H        ;READ DMA CHIP REQUEST
WRDMA ==    001H        ;WRITE DMA CHIP REQUEST
```

```
;       MP/M TO HOST DISK CONSTANTS FOR BUFFER BLOCKING/DEBLOCKING


HSTSIZ      ==    512         ;HOST DISK SECTOR SIZE
HSTSPT      ==    17

HSTBLK      ==    HSTSIZ/128  ;MP/M SECTS/HOST BUFF
SECMSK      ==    HSTBLK-1    ;SECTOR MASK


NOFDD ==    0           ;NUMBER OF FLOPPY DISK DRIVES
NOHRD ==    2           ;NUMBER OF LOGICAL HARD DISKS
MAXHD ==    NOHRD
MAXDSK      ==    MAXHD+NOFDD ;MAXIMUM NUMBER OF DRIVES
NDSK  ==    MAXDSK          ;ACTUAL NUMBER OF DRIVES
NODSK ==    NOFDD

CLEAR ==    26
CR    ==    0DH
LF    ==    0AH
K     ==    1024


;     BDOS CONSTANTS ON ENTRY TO WRITE

WRALL ==    0           ;WRITE TO ALLOCATED
WRDIR ==    1           ;WRITE TO DIRECTORY
WRUAL ==    2           ;WRITE TO UNALLOCATED


;     DTC HARD DISK SUBSYSTEM EQUATES.

DIO   ==    0A0H        ;DISK I/O PORT

;     BUS STATUS         7   6   5   4   3   2   1   0
;                        |   |   |   |   |   |   |
REQ   ==    080H  ;<-------+   |   |   |   |   |   |
DIROUT      ==    040H  ;<-----------+   |   |   |   |   |
MSSG  ==    020H    ;<---------------+   |   |   |   |
CMND  ==    010H  ;<-------------------+   |   |   |
BUSY  ==    008H  ;<-----------------------+   |   |
PTERR ==    004H  ;<---------------------------+   |
AVINT ==    002H  ;<-------------------------------+


;     CONTROL REGISTER   7   6   5   4   3   2   1   0
;                            |       |   |   |   |   |
SLCT  ==    040H  ;<-----------+       |   |   |   |   |
;INTERRUPT ENABLE  <-------------------+   |   |   |   |
;REQ INTERRUPT ENABLE    <----------------------+   |   |   |
;DMA INTERRUPT ENABLE    <---------------------------+   |   |
DODTA ==    002H  ;<-------------------------------+   |
EDMA  ==    001H  ;<-----------------------------------+
```

```
;     CONTROLLER STATUS BYTE MASKS.

CERR  ==    2           ;CONTROLLER ERROR
PERR  ==    1           ;PARITY ERROR ON BUS
FERR  ==    PERR ! CERR       ;EITHER ERROR


;     CONTROLLER COMMANDS.

RCCMD ==    001H        ;RECALIBRATE DISK CONTROLLER.
STCMD ==    000H        ;STATUS COMMAND
ESCMD ==    003H        ;ERROR SENSE COMMAND
FDCMD ==    004H        ;FORMAT DISK COMMAND
FTCMD ==    006H        ;FORMAT TRACK COMMAND
RDCMD ==    008H        ;READ DISK COMMAND
WTCMD ==    00AH        ;WRITE DISK COMMAND
FSCMD ==    0C0H        ;FDD FORMAT SELECTION

RDDMA ==    005H        ;READ DMA CHIP REQUEST
WRDMA ==    001H        ;WRITE DMA CHIP REQUEST


      .IFE LOC,[

      .LOC  0
            ][
      .LOC  100H

      ]


START:                  ; FOR MP/M 1.1 COMPATABILITY

      JMP   CLDST ; COLD START.
      JMP   WRMST ; WARM START.
      JMP   CONST ; CONSOLE STATUS.
      JMP   CONIN ; CONSOLE READ.
BIOOUT:     JMP   CONOUT      ; CONSOLE WRITE.
      JMP   LIST  ; PRINTER.
      JMP   RNTEM ; PUNCH  (NOT IMPLEMENTED).
      JMP   RNTEM ; READER (NOT IMPLEMENTED).
      JMP   HOME  ; MOVE HEAD OF SELECTED DRIVE TO TRACK ZERO.
      JMP   SELDSK      ; SELECT DRIVE.
      JMP   SETTRK      ; SEEK TRACK NUMBER.
      JMP   SETSEC      ; SET SECTOR NUMBER.
      JMP   SETDMA      ; SET DIRECT MEMORY ADDRESS.
      JMP   READ  ; READ SELECTED SECTOR.
      JMP   WRITE ; WRITE SELECTED SECTOR.
      JMP   POLLPT      ; PRINTER STATUS.
      JMP   SECTRN      ; SECTOR TRANSLATE.

      JMP   SELMEM      ; SELECT MEMORY.
```

```
      JMP   POLDEV      ; POLL DEVICE.
      JMP   STRCLK      ; START CLOCK.
      JMP   STPCLK      ; STOP CLOCK.
      JMP   EXIREG      ; EXIT REGION.
      JMP   MAXCON      ; MAXIMUM CONSOLE.
      JMP   SYSNIT      ; SYSTEM INITIALIZATION.
      JMP   IDLE  ; SYSTEM IDLE CALL VECTOR.


;     FILLER


      .BLKB FILLER



      .IFE  SCACHE,[

;*********************************************************
;             XIOS DIRECTORY CACHE ROUTINES
;*********************************************************

HOME:
      LDA   ACCEPT
      CPI   TRUE
      JNZ   XHOME
      LHLD  CURTAB
      INX   H
      MOV   A,M
      CPI   TRUE
      RZ
      JMP   XHOME


SELDSK:
      MOV   A,C         ;GET SELECTED DRIVE
      STA   CURDRIVE
      CPI   'M'-'A'
      JNZ   HDN1        ;XSELDK
      LXI   H,RAMDPH
      RET


HDN1: CPI   2
      JNC   XSELDK

      STA   CURDRIVE    ;SAVE HERE
      PUSH  D           ;SAVE SELECT FLAG
      CPI   0
DNBR9:      CPI   6
      JNC   NOTFOUND
      LXI   H,DRTAB
      MVI   D,0
      ADD   A
```

```
      ADD   A
      ADD   A
      MOV   E,A
      DAD   D
      SHLD  CURTAB

;     POINT FIRST BANK

      INX   H
      INX   H
      INX   H
      INX   H
      INX   H
      INX   H
      MOV   A,M
      STA   BNKNUM

      MVI   A,TRUE
      STA   ACCEPT
      POP   D
      MOV   A,E
      ORA   A
      JNZ   XSELDK
      LHLD  CURTAB
      INX   H
      MVI   M,FALSE
      JMP   XSELDK


NOTFOUND:
      MVI   A,FALSE           ;NO SUCH DRIVE
      STA   ACCEPT
      POP   D           ;FIX STACK, LET XIOS DO IT
      JMP   XSELDK

SETDMA:
      LXI   H,CDMAAD
      MOV   M,C
      INX   H
      MOV   M,B
      JMP   XDMAAD

READ:
      LDA   CURDRIVE          ;SEKDSK
      CPI   'M'-'A'
      JZ    REDRAM            ;XREAD
HDN2: CPI   2
      JNC   XREAD

      LDA   ACCEPT
      CPI   TRUE
      JNZ   XREAD
      MOV   A,B         ;GET HIGH RECORD #
      ORA   A           ;SEE IF ZERO
```

```
      JNZ   XREAD       ;SEE IF TOO HIGH, THEN GO XIOS


      LHLD  CURTAB            ;ELSE GET CURRENT TABLE POINTER
      INX   H           ;POINT 5 LOCATIONS UP
      INX   H           ;TO GET MIDDLE RECORD
      INX   H
      INX   H
      INX   H

      MOV   A,M         ;GET MID RECORD COUNT
      CMP   D           ;TEST WITH MPM MID RECORD #
      JC    XREAD       ;IF TOO HIGH GO XIOS READ
      JNZ   HOP         ;IF NOT SAME DO HOP
      DCX   H           ;IF SAME GET LOW RECORD #
      MOV   A,M         ;INTO ACC
      CMP   E           ;SEE IF IN RANGE WITH MPM RECORD #
      JC    XREAD       ;IF NOT GO XIOS READ
      INX   H           ;POINT MID AGAIN

HOP:
      DCX   H           ;POINT BACK TO BUFFER FLAG
      DCX   H
      DCX   H
      DCX   H
      MOV   A,M         ;GET BUFFER FLAG
      CPI   TRUE        ;SEE IF SET 0FFH
      JZ    NOXIOR            ;IF SO GET FROM BUFFER
      INX   H           ;GO BACK TO MID RECORD TO READ
      INX   H
      INX   H
      INX   H
      MOV   A,M         ;GET INTO A
      CMP   D           ;TEST WITH MID
      JNZ   RWCOMI
      DCX   H           ;POINT TO LOW RECORD
      MOV   A,M
      CMP   E           ;TEST LOW
      JNZ   RWCOMI            ;IF NOT IN BUFFER
      DCX   H
      DCX   H
      DCX   H
      MVI   M,TRUE            ;SHOW IN BUFFER

RWCOMI:
      PUSH  D
      CALL  XREAD
      JMP   RWCOM

NOXIOR:
      INX   H
      PUSH  D
      MOV   E,M
      INX   H
```

```
      MOV   D,M
      XCHG
      POP   D
      CALL  SHIFT
      CALL  BANKER

      LHLD  CDMAAD
      XCHG
      LXI   B,128

      CALL  BREAD

      XRA   A
      MOV   B,A
      MOV   C,A
      RET

WRITE:
      LDA   CURDRIVE          ;SEKDSK
      CPI   'M'-'A'
      JZ    WRTRAM            ;XWRITE
HDN3: CPI   2
      JNC   XWRITE

      LDA   ACCEPT
      CPI   TRUE
      JNZ   XWRITE
      MOV   A,B
      ORA   A
      JNZ   XWRITE
      LHLD  CURTAB
      INX   H
      INX   H
      INX   H
      INX   H
      INX   H
      MOV   A,M
      CMP   D
      JC    XWRITE
      JNZ   WHOP
      DCX   H
      MOV   A,M
      CMP   E
      JC    XWRITE

WHOP:
      PUSH  D
      CALL  XWRITE

RWCOM:
      LHLD  CURTAB
      INX   H
      INX   H
      MOV   E,M
```

```
      INX   H
      MOV   D,M
      XCHG
      POP   D
      ORA   A
      RNZ
      CALL  SHIFT
      CALL  BANKER

      LHLD  CDMAAD
      LXI   B,128

      CALL  BREAD

      XRA   A
      RET

;     SHIFT DE 7 BITS LEFT

SHIFT:
      MVI   C,7

SLOOP:
      XRA   A
      MOV   A,E
      RAL
      MOV   E,A
      MOV   A,D
      RAL
      MOV   D,A
      DCR   C
      JNZ   SLOOP
      RET


BANKER:
      DAD   D
      MOV   A,H
      CPI   0C0H
      LDA   BNKNUM
      JC    BNKOK
      MOV   A,H
      SUI   0C0H
      MOV   H,A
      LDA   BNKNUM
      INR   A
BNKOK:
      XCHG
      RET


;     DATA TABLES

CDMAAD:
```

```
      .WORD 0
CURTAB:
      .WORD 0
CURDRIVE:
      .BYTE 0
ACCEPT:
      .BYTE 0
NDRIVES:
      .BYTE 4

BNKNUM:     .BYTE 3


DRTAB:

      .BYTE 0     ;DRIVE ID
      .BYTE 0     ;DRIVE NIT FLAG

      .IFN  DATCAC,[

      .WORD 0
      .WORD 95
      .BYTE 3
      ][
ADD0: .WORD 0     ;BUFFER ADDRESS
SEC0: .WORD 95    ;LAST SECTOR
      .BYTE 3     ;INITIAL BANK
      ]

      .BYTE 0     ;FILLER


      .BYTE 0     ;DRIVE ID
      .BYTE 0     ;DRIVE NIT FLAG

.IFN  DATCAC,[

      .WORD 3000H
      .WORD 95
      .BYTE 3
      ][

ADD1: .WORD 3000H ;BUFFER ADDRESS
SEC1: .WORD 95    ;LAST SECTOR
      .BYTE 3     ;INITIAL BANK
      ]

      .BYTE 0     ;FILLER

      .BYTE 0     ;DRIVE ID
      .BYTE 0     ;DRIVE NIT FLAG
ADD2: .WORD 6000H ;BUFFER ADDRESS
SEC2: .WORD 95    ;LAST SECTOR
      .BYTE 3     ;INITIAL BANK
```

```
      .BYTE 0     ;FILLER


      .BYTE 0     ;DRIVE ID
      .BYTE 0     ;DRIVE NIT FLAG
ADD3: .WORD 9000H ;BUFFER ADDRESS
SEC3: .WORD 95    ;LAST SECTOR
      .BYTE 3     ;INITIAL BANK
      .BYTE 0     ;FILLER


            ]

.IFE  DATCAC,[

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;           CACHE STACK FOR DISK SECTORS
;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::

.ASCII      \STACK1\

STACK1:

.WORD 0FFFFH,0100H
.WORD 0FFFFH,0300H
.WORD 0FFFFH,0500H
.WORD 0FFFFH,0700H
.WORD 0FFFFH,0900H
.WORD 0FFFFH,0B00H
.WORD 0FFFFH,0D00H
.WORD 0FFFFH,0F00H

.WORD 0FFFFH,1100H
.WORD 0FFFFH,1300H
.WORD 0FFFFH,1500H
.WORD 0FFFFH,1700H
.WORD 0FFFFH,1900H
.WORD 0FFFFH,1B00H
.WORD 0FFFFH,1D00H
.WORD 0FFFFH,1F00H

.WORD 0FFFFH,2100H
.WORD 0FFFFH,2300H
.WORD 0FFFFH,2500H
.WORD 0FFFFH,2700H
.WORD 0FFFFH,2900H
.WORD 0FFFFH,2B00H
.WORD 0FFFFH,2D00H
.WORD 0FFFFH,2F00H

.WORD 0FFFFH,3100H
.WORD 0FFFFH,3300H
.WORD 0FFFFH,3500H
.WORD 0FFFFH,3700H
```

```
.WORD 0FFFFH,3900H
.WORD 0FFFFH,3B00H
.WORD 0FFFFH,3D00H
.WORD 0FFFFH,3F00H

.WORD 0FFFFH,4100H
.WORD 0FFFFH,4300H
.WORD 0FFFFH,4500H
.WORD 0FFFFH,4700H
.WORD 0FFFFH,4900H
.WORD 0FFFFH,4B00H
.WORD 0FFFFH,4D00H
.WORD 0FFFFH,4F00H


.WORD 0FFFFH,5100H
.WORD 0FFFFH,5300H
.WORD 0FFFFH,5500H
.WORD 0FFFFH,5700H
.WORD 0FFFFH,5900H
.WORD 0FFFFH,5B00H
.WORD 0FFFFH,5D00H
.WORD 0FFFFH,5F00H

;.WORD      0FFFFH,6100H
;.WORD      0FFFFH,6300H
;.WORD      0FFFFH,6500H
;.WORD      0FFFFH,6700H
;.WORD      0FFFFH,6900H
;.WORD      0FFFFH,6B00H
;.WORD      0FFFFH,6D00H
;.WORD      0FFFFH,6F00H

;.WORD      0FFFFH,7100H
;.WORD      0FFFFH,7300H
;.WORD      0FFFFH,7500H
;.WORD      0FFFFH,7700H
;.WORD      0FFFFH,7900H
;.WORD      0FFFFH,7B00H
;.WORD      0FFFFH,7D00H


BLOCKS      ==    (.-STACK1)/4


.WORD 0
PEND1:
.WORD 5F00H             ;WAS 7D00H  NEXT FREE BUFFER

.ASCII      \P STACK END\

            ]
```

```
.IFE  SCACHE,[

XDMAAD:
      ][
SETDMA:
      ]

      SBCD  DMAADD
      MOV   H,B
      MOV   L,C
      INX   H
      MOV   A,L
      ORA   H
      JRZ   UNDEFR
      RET

UNDEFR:
      LXI   H,HSTWRT
      MOV   A,M
      MVI   M,0
      ORA   A
      RZ
      JMP   WRTHST


WINFLG:     .BYTE 0


.IFE  SCACHE,[

XSELDK:
            ][
SELDSK:
            ]

      MOV   A,C   ;GET DRIVE
      CPI   'M'-'A'
      JNZ   NOTRAM
      STA   SEKDSK
      LXI   H,RAMDPH
      RET

NOTRAM:
      LXI   H,0
DNBR1:      CPI   4
      RNC         ;BAD DRIVE #
DNBR2:      CPI   4     ;MAK HARD DISK
      JNC   SELFLP


      STA   SEKDSK
      STA   DSKSAV      ;GOOD #

      MOV   E,A   ;INDEX FOR ALLOCATION SIZE
```

```
      MVI   A,0FFH
      STA   WINFLG

      MVI   D,0
      LXI   H,ALOCSZ
      DAD   D
      MOV   A,M
      STA   ALOCA

      LDA   DSKSAV
      MVI   H,0
      MOV   L,A   ;DISK # INTO L
      DAD   H     ;*2
      DAD   H     ;*4
      DAD   H     ;*8
      DAD   H     ;*16
      LXI   D,DPBASE
      DAD   D
      LDA   HDSPT
      STA   ULRPS ;SECTORS/TRACK
      RET

.IFE  SCACHE,[

XHOME:
            ][
HOME:
            ]

      LDA   WINFLG
      INR   A
      CNZ   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
      LDA   WINFLG
      INR   A
      RNZ
      SBCD  NEWSEC
      RET

;*****************************************************
;            SECTOR TRANSLATION ROUTINE.
;*****************************************************

SECTRN:
      MOV   A,D   ;SEE IF FLOPPY OR WINCH
      ORA   E
      JRNZ  YUP1
      MOV   H,B
      MOV   L,C
      RET

YUP1:
      XCHG
      MOV   A,C
      SBCD  NEWSEC
      DAD   B
      MOV   L,M
      MVI   H,0
      RET


;*******************************************************
;           READ 128 BYTE LOGICAL SECTOR
;*******************************************************

.IFE  SCACHE,[

XREAD:
            ][
READ:
            ]

      LDA   SEKDSK
      CPI   'M'-'A'
      JZ    REDRAM

      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


;******************************************************
;          WRITE 128 BYTE LOGICAL SECTOR
;******************************************************

.IFE  SCACHE,[

XWRITE:
            ][
WRITE:
            ]

      LDA   SEKDSK
      CPI   'M'-'A'
      JZ    WRTRAM

      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 MP/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
      INX   H
      LDA   SEKTRK+1
      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:
      LDA   WINFLG
      INR   A
      JNZ   AFLOP
      XRA   A
      STA   ERFLAG

      LDA   SEKSEC

      ORA   A           ;MAKE CORRECT HOST SECTOR TO R/W
      RAR
SEC256:
      ORA   A
      RAR

      MVI   D,0
      MOV   E,A
      LXI   H,SKWTAB
      DAD   D
      MOV   A,M
      STA   SEKHST
      JMP   RWCM

AFLOP:      CALL  GETTRK

;     CALL  MSG

;     .ASCIS      \ GOT TRACK STUFF \

RWCM:
      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 TO R/W IS CORRECT
;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
```

```
      LDA   WINFLG
      INR   A
      JRZ   WNTRK
      LXI   H,HSTTRK
      LDA   CTRACK
      CMP   M
      JRNZ  NOMAT
      JMPR  SECT

WNTRK:
      LXI   H,HSTTRK
      LDA   SEKTRK
      CMP   M
      JRNZ  NOMAT
      INX   H
      LDA   SEKTRK+1
      CMP   M
      JRNZ  NOMAT

;SEE IF SECTORS ARE SAME

SECT:
      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   WINFLG
      INR   A
      JZ    WNOMAT
      LDA   HSTWRT
      ORA   A
      CNZ   FLUSH


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

FILHST:
      LDA   WINFLG
      INR   A
      JZ    FILWIN

;     CALL  MSG

;     .ASCIS      \DOING FLOPPY FILHST \

      LDA   SEKDSK
      STA   HSTDSK
      LHLD  CTRACK
```

```
      SHLD  HSTTRK
      LDA   SEKHST
      STA   HSTSEC

      LDA   RSFLAG
      ORA   A
      CNZ   RDHST
      XRA   A
      STA   HSTWRT


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

MATCH:
      LDA   WINFLG
      INR   A
      JZ    WINMAT

      LDA   CREC
      MOV   B,A
      MVI   C,0
      SRLR  B
      RARR  C
      JMP   MOVREC


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   WINFLG
      INR   A
      JZ    WINWRT
      LDA   HSTDSK

;     JMP   FLPWRT


DNBR4:      CPI   4
      JC    WINWRT
      JMP   FLPWRT



;*********************************************************
;*         WINCHESTER NO MATCH ROUTINES
;*********************************************************

WNOMAT:
```

```
      LDA   HSTWRT
      ORA   A
      CNZ   WINWRT
FILWIN:
      LXI   H,SEKDSK
      LXI   D,HSTDSK
      LXI   B,3
      LDIR
      LDA   SEKHST
      STAX  D
      LDA   RSFLAG
      ORA   A
      CNZ   WINRED
      XRA   A
      STA   HSTWRT

WINMAT:
      LDA   SEKSEC            ;GET SECTOR
      ANI   SECMSK            ;MASK FOR HOST SECTOR SIZE
      MOV   L,A         ;INTO L FOR INDEX
      MVI   H,0         ;CLEAR H
      DAD   H           ;*2
      DAD   H           ;*4
      DAD   H           ;*8
      DAD   H           ;*16
      DAD   H           ;*32
      DAD   H           ;*64
      DAD   H           ;*128

      MOV   B,H
      MOV   C,L

;     BC HAS RELATIVE HOST ADDRESS


MOVREC:
      LXI   H,HSTBUF
      DAD   B     ;HL IS NOW HOST BUFFER
      LDED  DMAADD      ;DE HAS DMA ADDRESS
      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:
      CALL  RWMOV


;NOW CHECK WRITE TYPE FOR DIRECTORY UPDATE

      LDA   WRTYPE      ;GET TYPE OF WRITE
```

```
      DCR   A     ;IS IT TO THE DIRECTORY
      JRNZ  GOODOP      ;IF NOT SHOW A SUCCESSFUL R/W OPERATION

;CLEAR HOST BUFFER FOR DIRECTORY WRITE

WRITIT:
      STA   HSTWRT
      CALL  WRTHST
GOODOP:
      LDA   WINFLG
      INR   A
      JZ    WEXIT
      LDA   MRML
      CPI   0
      MVI   A,1
      RZ
      XRA   A
      RET

WEXIT:
      LDA   ERFLAG


      RET


;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;        WINCHESTER READ AND WRITE ROUTINES HERE
;
;     ENTRY HSTTRK, HSTDSK, HSTSEC ARE SET
;     EXIT  ERROR FLAG SET IN RAM
;           A SET FOR BDOS 0 IF GOOD 1 IF BAD
;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

WINWRT:
      LDA   HSTDSK
DNBR7:      CPI   MAXHD
      JNC   FLPWRT

      MVI   A,1
      STA   HDPOP
      MVI   A,WTCMD
      JMPR  HDRDWR
WINRED:
      XRA   A
      STA   HDPOP
      MVI   A,RDCMD
HDRDWR:
      STA   HDOPER            ;SAVE R/W CODE HERE
      LDA   HSTDSK            ;GET HOST DISK
      STA   HDISK       ;MAKE LOGICAL WINCHESTER

      MVI   A,4
      STA   WRTRY
```

```
      XRA   A
      STA   ERFLAG


;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;                    S E T U P P
;     SET UP PHYSICAL PARAMETERS FOR READ OR WRITE
;
;     ENTRY HDOPER = COMMAND
;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::

SETUPP:
      LDA   HDOPER            ;GET READ OR WRITE
      STA   CIOPB       ;SET COMMAND IN C.D.B.

      MVI   D,0         ;PREPARE TO GET LOGICAL OFFSET
      LDA   HSTDSK            ;LOGICAL DRIVE (0 OR 1)
      ADD   A
      ADD   A
      MOV   E,A
      LXI   H,DSKOFF    ;OFFSET TABLE INDEX
      DAD   D
      MOV   E,M         ;GET FIRST OFFSET VALUE
      INX   H
      MOV   D,M         ;GET SECOND OFFSET VALUE
      INX   H
      MOV   A,M         ;GET UNIT VALUE
      MOV   C,A

SETP1:
      PUSH  B           ;SAVE UNIT SELECTION
      LHLD  HSTTRK            ;GET TRACK NUMBER
      MOV   B,H
      MOV   C,L
      DAD   H           ;*2
      DAD   H           ;*4
      DAD   H           ;*8
      DAD   H           ;*16

      DAD   B           ;*17

      LDA   HSTSEC            ;GET HOST SECTOR
      MVI   B,0         ;CLEAR B
      MOV   C,A         ;GET SECTOR INTO C
      DAD   B           ;HL = SPTK*TRK+SEC
      POP   B           ;RESTORE BC = UNIT SELECTION
      XRA   A           ;A = 0
      DAD   D           ;ADD DRIVE OFFSET
      ADC   C           ;GET UNIT # INTO A WITH CY
      XCHG              ;'SAVE' HL IN DE

;     NOW FILL UP NEXT PART OF COMMAND DESCRIPTOR WITH LOGICAL RECORD
INFO
```

```
      LXI   H,CIOPB+1   ;POINT TO UNIT & HI ADDRESS
      MOV   M,A         ;FILL IT
      INX   H           ;POINT TO MID ADDRESS
      MOV   M,D         ;FILL IT
      INX   H           ;POINT TO LOW ADDRESS
      MOV   M,E         ;FILL IT

;     NOW SET SECTORS TO READ

      INX   H           ;# OF SECTORS
      MVI   M,1         ;READ ONE SECTOR
      INX   H           ;PARMS NEXT

;     NOW SET STEP RATE & ERROR CORRECTION


BUFOP:
      MVI   M,00000000B

;     POINT TO UPDATED COMMAND DESCRIPTOR FIELDS


FINAL:

      .IFE  DATCAC,[

      LBCD  CIOPB+2               ;GET USEFUL SECTOR
      LXI   H,STACK1          ;START OF BUFFER LIST
      MVI   D,0               ;LOCATE BUFFER COUNTER
      MVI   E,BLOCKS          ; WAS 63 ,MAX BUFFER COUNTER

CHKCAC:
      PUSH  H                 ;SAVE LIST ENTRY
      CALL  COMP2             ;SEE IF A MATCH
      POP   H                 ;RECOVER LIST ENTRY
      JZ    HAVEIT                ;GET FROM BUFFER
      INR   D
      DCR   E                 ;ELSE TRY NEXT
      JZ    NOCACH                ;UNLESS LIST EXHAUSTED
      INX   H                 ;IF NOT POINT NEXT ENTRY
      INX   H
      INX   H
      INX   H
      JMP   CHKCAC                ;AND CHECK THAT ONE

THISBF:     .WORD 0,0

HAVEIT:
      LDA   HDPOP             ;SEE IF READ OR WRITE
      ORA   A
      JZ    RDCACH                ;GET BUFFER IF READ
      INX   H                 ;ELSE FIND BUFFER FOR WRITE
      INX   H
      MOV   E,M               ;MAKE IT DESTINATION
```

```
        INX   H
        MOV   D,M
        LXI   H,HSTBUF          ;SOURCE IS HOST
        LXI   B,512
        LDIR                    ;UPDATE CACHE BUFFER
        JMP   NOCACH

RDCACH:
        PUSH  H                 ;SAVE ENTRY
        PUSH  D                 ;SAVE COUNTERS
        LXI   D,THISBF          ;SAVE CONTENTS HERE
        LXI   B,4
        LDIR
        POP   D                 ;RECOVER COUNTERS
        MOV   A,D               ;GET LOCATE COUNTER
        CPI   0                 ;SEE IF HEAD OF LIST
        JZ    HAVHED                  ;IF SO SKIP NEXT

;       MOVE LOWER LIST DOWN

        INR   A
        MVI   B,0
        ADD   A                 ;*2
        ADD   A                 ;*4
        MOV   C,A

        POP   D                 ;GET HL
        PUSH  D                 ;SAVE IT
        INX   D
        INX   D
        INX   D
        POP   H
        DCX   H
        LDDR
        LXI   H,THISBF
        LXI   D,STACK1
        LXI   B,4
        LDIR
        LXI   H,STACK1

        PUSH  H

HAVHED:
        POP   H                 ;GET ENTRY
        INX   H                 ;POINT TO BUFFER ADDRESS
        INX   H
        MOV   A,M               ;LOW BUFFER ADDRESS
        INX   H                 ;POINT NEXT
        MOV   H,M               ;HI BUFFER ADDRESS
        MOV   L,A               ;LOW TO L
        LXI   D,HSTBUF          ;HOST BUFFER IS DESTINATION
        LXI   B,512             ;512 BYTES
        LDIR                    ;MOVE IT
        XRA   A
```

```
      STA   ERFLAG
      RET


COMP2:
      MOV   A,M
      CMP   C
      RNZ
      INX   H
      MOV   A,M
      CMP   B
      RET

NOCACH:
      ]

      LXI   H,CIOPB                 ;NOW SET COMMAND AREA ADDRESS
      CALL  EXEC              ;SEND COMMAND TO CONTROLLER
      CALL  HDRWE             ;DO R/W AND GET COMPLETION STATUS


;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;                     T E S T E R
;     TEST FOR ANY CONTROLLER/DISK ERRORS
;     ENTRY C HAS COMPLETION STATUS
;           B HAS NULL BYTE
;           A HAS MESSAGE BYTE
;
;     EXIT  SET ERROR FLAG, PRINT ANY ERRORS
;           RETURN
;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::

;           TEST FOR ERROR CONDITIONS

TESTER:
      ORA   A
      MVI   A,80H
      JNZ   FNL3        ;IF MESSAGE BYTE IS NON-ZERO
      MOV   A,B
      ANI   MSSG
      MVI   A,80H
      JZ    FNL3        ;IF MESSAGE BIT ZERO

      MOV   A,C
      ANI   FERR        ;MASK FOR ERRORS
      STA   ERFLAG
      JNZ   WCONT       ;IF ERRORS

.IFN  DATCAC,[

      RET
            ][

;     UPDATE BUFFER STACK BY MOVING ALL DOWN ONE
```

```
;     THEN MOVE HOST BUFFER TO NEXT FREE CACHE BUFFER

      LDA   HDPOP       ;GET READ/WRITE OP.
      ORA   A           ;TEST IT
      RNZ               ;RETURN IF WRITE

      LXI   H,PEND1-3
      LXI   D,PEND1+1
      LXI   B,BLOCKS*4  ;32         ;AS 252
      LDDR
      LHLD  CIOPB+2           ;SECTOR JUST READ IS AT HEAD
;     LDA   CIOPB+1
;     ANI   00100000B
;     JZ    END0
;     SET   7,H         ;SHOW UNIT 1
END0:
      SHLD  STACK1
      LHLD  PEND1       ;FREE BUFFER GOES TO HEAD
      SHLD  STACK1+2
      XCHG              ;BUFFER IS DESTINATION
      LXI   H,HSTBUF          ;FOR HOST DATA
      LXI   B,512
      LDIR
      RET


      ]


WCONT:
      XRA   A
      STA   CIOER+1

      LXI   H,CIOER
      CALL  EXEC        ;ASK FOR ERROR BYTES

      LXI   H,TEMPBF    ;USE TEMPBF FOR ERROR BYTE STORRAGE
      CALL  INBUFF            ;READ IT IN

      LDA   TEMPBF            ;GET ERROR BYTE
      ANI   00111111B   ;MASK FOR GOOD INFO

FNL3: STA   ERFLAG            ;SET ERFLAG WITH BITS

      LDA   WRTRY
      DCR   A
      STA   WRTRY
      JNZ   FINAL

      LDA   ERFLAG
      JMP   EMSROT
```

```
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;                        E X E C
;           SEND COMMAND TO CONTROLLER
;
;     ENTRY:      HL HAS RAM ADDRESS OF C.D.B.
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::

EXEC:
;     NOP
;     NOP               ;TEST 'TILL NOT BUSY
      IN    DIO+2
      ANI   BUSY
      JRNZ  EXEC
      MVI   A,SLCT            ;SET SELECT
      OUT   DIO+1

EXEC2:
;     NOP
;     NOP               ;WAIT FOR BUSY
      IN    DIO+2
      ANI   BUSY
      JRZ   EXEC2
      MVI   A,DODTA           ;ENABLE DATA OUT
      OUT   DIO+1

EXEC3:
;     NOP
;     NOP
      IN    DIO+2
      XRI   0FFH
      JM    EXEC3
      MVI   B,6         ;6 COMMAND BYTES
EXEC4:
;     NOP
;     NOP
      IN    DIO+2
      XRI   0FFH
      JM    EXEC4
      MOV   A,M
      INX   H
      OUT   DIO
      DCR   B
      JZ    TPOLL
      JMPR  EXEC4

TPOLL:
      CALL  CHDPOL
      RET



CIOPB:      .BLKB 10          ;DISK COMMAND BUFFER
TEMPBF:     .BLKB 10          ;RESULT STATUS CELLS
CIOER:      .BYTE ESCMD,0,0,0,0,0,0,0,0,0     ;REQUEST ERROR CODE
```

```
HDPOP:      .BYTE 0                     ;WINCH PHYSICAL R/W FLAG

DSKSAV:     .BYTE 0                 ;DISK TO R/W
ACTDSK:     .BYTE 0                 ;SAME
CMDSAV:     .BLKW 1                 ;ADDRESS OF C.D.B. FOR DISK I/O
CTRL: .BYTE 0FFH              ;TYPE OF CONTROLER
NITFLG:     .BYTE 0FFH


;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;                       H D R W E
;     WINCH READ/WRITE ENTRY POINT
;
;     ENTRY HDPOP FLAG SET 0 = READ, 1 = WRITE
;     EXIT  JUMP TO CORRECT ROUTINE
;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::

HDRWE:
      LDA   HDPOP       ;SEE IF READ OR WRITE
      ORA   A
      JZ    WPREAD


;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;                     W P W R I T E
;       WRITE HOST BUFFER TO CONTROLLER BUFFER
;
;     ENTRY NO PARMS
;     EXIT  GET COMPLETION STATUS AND RETURN
;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::

;     WRITE SECTOR TO DISK

WPWRITE:
      LXI   H,HSTBUF          ;GET FWA OF BUFFER
      MVI   C,DIO
      MVI   D,WCMND
      JMPR  OBF2

OBF1: ANA   D                 ;MASK COMMAND BIT
      JNZ   GCMPS             ;IF DONE WITH TRANSFER
      OUTI
OBF2: IN    DIO+2             ;READ STATUS
      ORA   A
      JM    OBF1              ;IF REQUEST IS PRESENT
      CALL  BUFPOL
      JMPR  OBF2


;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;                       W P R E A D
;       READ CONTROLLER BUFFER TO HOST BUFFER
;
;     ENTRY NO PARMS
```

```
;       EXIT  GET COMPLETION STATUS AND RETURN
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::

;       READ SECTOR FROM DISK

WPREAD:
        LXI   H,HSTBUF           ;GET FWA OF BUFFER
INBUFF:       MVI   C,DIO
        MVI   D,WCMND                  ;SET COMMAND BIT MASK
        JMPR  IBF2


IBF1:   ANA   D                  ;MASK FOR COMMAND BIT
        JNZ   GCMPS              ;IF DONE WITH TRANSFER
        INI
IBF2:   IN    DIO+2              ;READ STATUS
        ORA   A
        JM    IBF1               ;IF REQUEST IS PRESENT
        CALL  BUFPOL
        JMPR  IBF2



;::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;                       G C M P S
;       GET COMPLETION STATUS OF CONTROLLER I/O
;
;       ENTRY NO PARMS
;       EXIT  C HAS COMPLETION STATUS
;             B HAS NULL BYTE
;             A HAS MESSAGE BYTE
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::

;             GET COMPLETION STATUS OF R/W

GCMPS:
        NOP
        NOP
        NOP
        NOP

        IN    DIO          ;GET COMPLETION STATUS
        MOV   C,A

GCMP1:
        NOP
        NOP
        NOP
        IN    DIO+2
        ORA   A
        JP    GCMP1        ;IF REQ NOT SET

        MOV   B,A
        NOP
```

```
      NOP
      NOP
      IN    DIO         ;GET MESSAGE BYTE
      RET


;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;                  E M S R O T
;     PRINT ERROR MESSAGE ON SCREEN
;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::

EPRINT:
      PUSH  PSW
      IN    WTRACK
      STA   ITRACK
      IN    WSECT
      STA   ISECT
      POP   PSW

EMSROT:
      PUSH  PSW   ;FOR HARD DISK ENTRY

      CALL  MSG
      .ASCII      'Err'
      .BYTE ' '+80H
      LDA   HSTDSK

      ADI   'A'

      MOV   C,A
      CALL  BIOOUT
      CALL  MSG
      .BYTE ':',' '+80H

      POP   PSW

      CALL  LBYTE

      MVI   C,' '
      CALL  BIOOUT

      LDA   HSTTRK+1    ;GET TRACK NUMBER
      CALL  LBYTE
      LDA   HSTTRK
      CALL  LBYTE
      MVI   C,' '
      CALL  BIOOUT

      LDA   HSTSEC            ;GET SECTOR NUMBER
      CALL  LBYTE
      MVI   C,' '
      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


;************************************************
; 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:
      LXI   D,0   ;CONSOLE 0
..LOOP:
      MOV   C,M   ;PUT CHARACTER INTO C
      PUSH  D     ;SAVE POLLED CON. #
      PUSH  H     ;SAVE MESSAGE POINTER
      CALL  XBIOOUT     ;PRINT IT
      POP   H     ;RECOVER MESSAGE
      POP   D     ;RECOVER CON. #
      MOV   A,M   ;GET LAST BYTE
      INX   H     ;POINT TO NEXT
      ORA   A     ;TEST BIT 7 SET
      JP    ..LOOP      ;IF NOT KEEP PRINTING
      RET


XBIOOUT:
      PUSH  B
      MVI   C,153
      CALL  XDOS
      MOV   D,A
      POP   B
```

```
      CALL  BIOOUT
      RET

;*********************************************************
;                   FLOPPY DISK STUFF
;*********************************************************


STEP  ==    1
STEPS   ==  0
IDLY  ==    20
HLDOPT      ==    0


STDNR ==    1



NOFLP ==    3           ;STDNR+1

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

EEXIT:
      JMP   0



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

DCMD  ==    63H   ;FDC 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


;*********************************************************
;             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



;*******************************************************
; 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   0 ;DRIVE A
      .BYTE 0H    ;DRIVE B
      .BYTE 0H
      .BYTE 0H


;*******************************************************
; FMICRO & FMACRO SET TO GIVE 3 USER TRIES AT
; CORRECTING PROBLEM ON DISK
;*******************************************************
```

```
HOMER:      .BYTE 4     ;ALLOW 3 USER DISK FIXES
RETRYIT:.BYTE     5     ;RETRY 5 TIMES EACH
FSOFT:      .BYTE FALSE ;RESERVED FOR INTERNAL USE
                  ;WHEN SET TRUE SOFT ERRORS
                  ;ARE DISPLAYED ON SCREEN
KBIT: .BYTE 4     ;KEYBOARD RETRYS



;*******************************************************
; FDC OPERATING CONTROL TABLE. THIS TABLE CONTAINS
; THOSE BITS IN THE FDC CONTROL BYTE WHICH SELECT THE
; OPERATING MODE. THESE BITS ARE COMBINED WITH ADDRESS
; CONTROL BITS TO FORM A COMPLETE FDC 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 &     SIDE 0        SIDE 0    SIDE 1    SIDE 1
; DENSITY    NO WAIT    WAIT     NO WAIT    WAIT

C8S:  .BYTE 01010000B,00010000B,01000000B,00000000B
C5S:  .BYTE 01110000B,00110000B,01100000B,00100000B
C8D:  .BYTE 11010000B,10010000B,11000000B,10000000B
C5D:  .BYTE 11110000B,10110000B,11100000B,10100000B


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

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


;*******************************************************
; THE SYSTEM CONFIGURATION OPTION BYTES FOLLOW
;*******************************************************
```

```
CFGOPT:
      HLOPT:      .BYTE 0     ;SINGLE OR DUAL HEAD LOAD



;*****************************************************
; 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
      MVI   B,3    ;GET NUMBER OF FLOPPIES 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

;*********************************************************
; FLOPPY DISK SELECT
;*********************************************************

SELFLP:
      MOV   A,C
      STA   SEKDSK


DNBR3:      SUI   MAXHD
      MOV   C,A   ;DON'T FORGET TO CORRECT C

      STA   PDISK

      XRA   A
      STA   WINFLG

      PUSH  D     ;FIRST SELECT 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

      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


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 FDC CTRL BYTES INTO ATABLE
      CALL  PREAD ;GET THE DDB
      PUSH  PSW
```

```
;     CALL  MSG

;.ASCIS     \FIRST READ DONE \

      POP   PSW

      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 FDC BYTES IN ATABLE
      CALL  PUTAPB      ;UPDATE APB FROM FULLY VALID ATABLE

OK:   LHLD  DPHADR      ;RETURN DPH ADDRESS
      RET

ERR:  MVI   C,2
      CALL  EPRINT      ;PRINT ERROR IN READING DDB

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


FLPWRT:
      LDA   HSTDSK
DNBR5:      CPI   MAXHD
      CC    CLOSE
      LDA   SEKDSK

DNBR8:      SUI   MAXHD
      STA   PDISK ;MAKE IT THE PHYSICAL DISK

;     CALL  MSG

;     .ASCIS      \ FLOPPY WRITE \


      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
      RET


RDHST:
;     CALL  MSG

;     .ASCIS      \ FLOPPY READ HOST \

      LDA   HSTDSK

DNBR6:      SUI   MAXHD

      LDA   PDISK
      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:
      DI
      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 FDC 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 FDC BYTE IS
;     SET. SEEK TO CORRECT TRACK.
;*************************************************

SADS:
      LDA   HOMER ;NUMBER OF HOME OPS.
      STA   MRML
      STA   KBIT

MAC:
      LDA   RETRYIT     ;RETRIES BETWEEN HOME OPS.
      STA   RMACRO

      CALL  SETD3S      ;SET FDC 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
      RC
      JMPR  ENDS  ;DONE SEEKING

NOZE: CALL  FSEEK ;NORMAL SEEK TO DESIRED TRACK
      RC

ENDS: CALL  PUTAPB      ;UPDATE APB FROM ATABLE
      LDA   PCMD  ;GET READ/WRITE COMMAND
      BIT   5,A   ;SEE IF READ OR WRITE
```

```
      JRNZ  NOMIL ;NO DELAY ON READ

      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:
      DI
      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   PCMD  ;GET READ OR WRITE COMMAND

      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 FDC 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

      LXI   H,HSTBUF

      BIT   WBWRIT,D ;ARE WE WRITING(y/n)
      JRNZ  WRIT  ;YES, GO WRITE


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

PIRD:
      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:
PIWR:
      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 FDC 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

      LXI   H,KBIT
      DCR   M
      JZ    KEYIT

      CALL  FHOME
      RC
      XRA   A
      STA   TRACK

      JMP   MAC


;IF WE CAME TO HERE IT IS TIME TO RECAL. THE DRIVE

KEYIT:
      MVI   A,4
      STA   KBIT
      LDA   ISTAT
      ANA   C
      CALL  EPRINT

NORE:

;*************************************************
;THE FOLLOWING ROUTINE RINGS BELL & FLASHES LED ON
```

```
;DRIVE WITH COUNTED DOWN MICRO RETRY
;************************************************

      CALL  FHOME
      RC
      XRA   A
      STA   TRACK

      CALL  RING
      MVI   C,153
      CALL  XDOS
      STA   XCON
CRIN:
      CALL  XCONST      ;SEE IF KEY PRESSED
      JRZ   CRIN  ;IF NOT KEEP RINGING
      CALL  XCONIN      ;GET CHAR
      CPI   3     ;A ^C (y/n)
      JZ    EEXIT ;DO A SPECIAL WARM BOOT IF SO
      CPI   4     ;A ^D (y/n)
      JRZ   FXER  ;IGNORE ERROR
      CPI   0DH   ;SEE IF 'CR'
      JRNZ  CRIN  ;IF NOT RING AGAIN
      CALL  BS    ;ELSE 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


;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  ;17 CHARS TO ERASE
BS1:  CALL  XCONOUT
      DCR   B
      JRNZ  BS1
      RET

;PRINT 17 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

XCONIN:
      LDA   XCON
      MOV   A,D
      CALL  CONIN
      RET

XCONOUT:
      LDA   XCON
      MOV   D,A
      CALL  CONOUT
      RET

XCONST:
      LDA   XCON
      MOV   D,A
      CALL  CONST
      RET

XCON: .BYTE 0

;******************************************************
; SET FDC CONTROL BYTE. GET 2 FDC 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 FDC CONTROL BYTES. CREATE THE EIGHT FDC 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 FDC 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 FDC BYTES

      LDA   FLAG  ;GET THE FLAG
      ANI   00001100B ;ISOLATE USER DENSITY, SIZE
                  ;AND GET USER TRACK FDC 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
```

```
;*************************************************
; 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 REGS
      LDA   PCMD
```

```
      BIT   5,A   ;SEE IF READ OR WRITE
      JRZ   NDLY  ;IF READ DON'T DELAY

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

```
;*************************************************
; 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   SELDSK      ;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


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
      .ASCIS      \ STP\
      JMP   EEXIT ;GO SPECIAL WARM BOOT




;*************************************************
;             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 ;STANDARD
      .BYTE 3,9,15,21,2,8,14,20,26  ;TRANSLATE
      .BYTE 6,12,18,24,4,10,16,22   ;TABLE



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

```
; 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:     .BLKW 1     ;PHYSICAL TRACK FOR NEXT I/O
PSECT:      .BLKW 1     ;PHYSICAL SECTOR FOR NEXT I/O
;PDMA:      .BLKW 1     ;PHYSICAL BUFFER ADDR, NEXT I/O

CTRACK:     .BLKW 1
CSECT:      .BLKW 1


;*****************************************************
;    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:     .BLKW 1     ;DPH ADDRESS FOR CURRENT DISK
APBADR:     .BLKW 1     ;APB ADDRESS FOR CURRENT DISK
WRTYPE:     .BLKB 1     ;TYPE OF WRITE
D3SNO:      .BLKB 1     ;CURRENT NO WAIT FDC BYTE
D3SWT:      .BLKB 1     ;CURRENT WAIT FDC BYTE
SAVADR:     .BLKW 1     ;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 FDC
; CONTROLLER AND THE 179X CHIP. IF A PERMANENT DISK
; ERROR OCCURS, THE VARIABLES ARE PRINTED OUT AS AN AID
; IN ERROR DIAGNOSIS.
;*****************************************************

EINFO:
ISTAT:      .BLKB 1     ;LAST INPUT FROM STATUS REG
ACTIVE:     .BLKB 1     ;SHOWS WHICH ROUTINE HAD ERROR
OD3S: .BLKB 1     ;LAST OUTPUT TO FDC CTRL BYTE
OCMD: .BLKB 1     ;LAST OUTPUT TO CMD REG

EILEN ==    .-EINFO
```

```
;*******************************************************
; 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 FDC CONTROL BYTES
SS0W: .BLKB 1
SS1N: .BLKB 1
SS1W: .BLKB 1
US0N: .BLKB 1     ;USER TRACK FDC 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:  .BLKW 1     ;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




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


ALV1: .BLKB 86    ;ALLOCATION VECTOR
CSV1: .BLKB 64    ;CHECKSUM VECTOR

ALV2: .BLKB 86    ;ALLOCATION VECTOR
CSV2: .BLKB 64    ;CHECKSUM VECTOR



;*******************************************************
;                     WORKING STORAGE
;*******************************************************


HDCSV:
;     DIRECTORY CHECK BUFFER (NONE REQUIRED)
```

```
; COMPUTE WORST CASE ALLOCATION MAP SIZE FOR DRIVE A AND B

WDSMA ==    (((306*68*3)/(1024*4/128))-1)/8


HDALL0:
HDALL1      ==    HDALL0+WDSMA
HDALL2      ==    HDALL1+WDSMA
HDALL3      ==    HDALL2+WDSMA
HDALL4      ==    HDALL3+WDSMA


;::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       READ TRACK 0, SECTOR 0 TO GET
;       DDBS, DSKOFF, DTYP & CTYP
;       SEND DRIVE CHARACTERISTICS TO CONTROLLER
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::

      BCON  ==    DIO+1
      BSTAT ==    BCON+1


NIT1:
      IN    43H         ;CLEAR M DRIVE

      MVI   A,0C9H
      STA   RETIT

      CALL  DSKRST

      LXI   H,0
      SHLD  HSTSEC
      SHLD  HSTTRK
      XRA   A
      STA   HSTDSK

      CALL  WINRED


      .IFE  DPOL,[

      XRA   A
      STA   NITFLG
            ]

      LXI   H,HSTBUF
      LXI   D,128       ;OFFSET TO DDBS
      DAD   D
      LXI   D,DDB0            ;FILL DDBS
      LXI   B,DDBLEN
      LDIR

      LXI   D,DSKOFF    ;NOW DO PHYSICAL DISK PARAMATERS
      LXI   B,OFFLEN
```

```
        LDIR


        LDA   STEPR       ;GET STEP RATE
        STA   BUFOP+1

;             GET ALLOCATION SIZE FOR EACH DRIVE

        LDA   DDB0+2            ;GET BSH
        CALL  GALLO
        STA   ALOCSZ

        LDA   DDB1+2
        CALL  GALLO
        STA   ALOCSZ+1

        LDA   DDB2+2
        CALL  GALLO

        STA   ALOCSZ+2

        LDA   DDB3+2
        CALL  GALLO
        STA   ALOCSZ+3


;       SET DIRECTORY CHECK VECTORS NEXT

        LXI   H,8000H
        SHLD  DDB0+11
        SHLD  DDB1+11
        SHLD  DDB2+11
        SHLD  DDB3+11


        .IFE  SCACHE,[

;       SET CACHE TABLES

        LDA   DRIVES
        STA   DNBR1+1
        STA   DNBR9+1
              ]

        LDA   DRIVES
        ADI   3           ;3 FLOPPIES ON TO DRIVE #
        STA   DNBR1+1

        LDA   DRIVES            ;NUMBER OF WINCH DRIVES FROM DDB

        STA   HDN1+1
        STA   HDN2+1
        STA   HDN3+1
```

```
        STA   DNBR2+1
        STA   DNBR3+1
        STA   DNBR4+1
        STA   DNBR5+1
        STA   DNBR6+1
        STA   DNBR7+1
        STA   DNBR8+1

        DI

;       CLEAR SCREENS

        MVI   A,25
        INR   A
        OUT   3
        OUT   5
        OUT   13H
        OUT   15H
        OUT   17H
        OUT   1FH
        OUT   11H

        CALL  MSG


        .BYTE 10,13

        .ASCII      \Xios 989HE Aug 31, 1987 \

        .IFN  CONLST,[

        .ASCII      \serial list 1st \
              ][
        .ASCII      \parallel list 1st \
              ]

        .IFE  DATCAC,[
        .ASCIS      \ dir/dat cache\
                    ][
        .ASCIS      \ dir cache\
              ]

        OUT   VIRTC


        XRA   A
        STA   RETIT
        RET

GALLO:
        SUI   3
        LXI   H,ALOREC
        MOV   E,A
        DAD   D
```

```
      MOV   A,M
      RET


      .LOC  HDALL4+WDSMA



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

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


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

SEKSEC:     .BLKW 1     ;LOGICAL SECTOR TO R/W
CREC: .BLKB 1     ;CURRENT RECORD WITHIN SECTOR


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

ALOCSZ:     .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

ALOCA:      .BLKB 1     ;RECORDS/BLOCK FOR SELECTED DRIVE

;     UNALLOCATED PARAMETERS FOLLOW


UNACNT:     .BLKB 1     ;UNALLOCATED RECORD COUNT

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

NEWSEC:     .BLKW 1     ;MP/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:     .BLKW 1
HSTSEC:     .BLKW 1
```

```
SEKDSK:     .BLKB 1
SEKTRK:     .BLKW 1
SEKHST:     .BLKW 1

      UNALEN      ==    .-UNACNT


ITRACK:     .BLKB 2
ISECT:      .BLKB 2
WRTRY:      .BLKB 1
SIDEID:     .BYTE 0



;     HARD DISK AND BLOCKING/DEBLOCKING SPECIFIC DATA AREA

HDOPER:     .BLKB 1           ;HARD DISK OPERATION FLAG
HDISK:      .BLKB 1           ;HARD DISK DRIVE NUMBER
HTRACK:     .BLKB 2           ;HARD DISK TRACK NUMBER
HSECT:      .BLKB 1           ;HARD DISK SECTOR NUMBER

ERFLAG:     .BLKB 1           ;ERROR REPORTING


;     DRIVE OFFSETS

DSKOFF:     .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

STEPR:      .BYTE 1           ;0 = BUFFER STEP, 1 = 3MS STEP
DRIVES:     .BYTE 2           ;NUMBER OF WINCH DRIVES
CTYP: .BYTE 0           ;0 - XEBEC, 1 = OMTI

;     SECTOR SKEW TABLE

SKWTAB:     .BYTE 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16

CNIT: .BYTE 0CH         ;0CH = XEBEC, 0C2H = OMTI

DPT:  .BYTE 0,0,0,0,0,0,0,0,0,0

OFFLEN      ==    .-DSKOFF


;*****************************************************
;*                MP/M 2.1 COMMONBASE
;*****************************************************

CLDST:
      JMP   COLD

SWTUSR:     JMP   0
SWTSYS:     JMP   0
PDISP:      JMP   0
```

```
XDOS: JMP   0

SYSDTA:     .BLKW 1

COLD:
WRMST:
      MVI   C,0
      JMP   XDOS



;*****************************************************
; 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      ;LOGICAL TO PHYSICAL XLATE TAB
      .WORD 0     ;SCRATCH
      .WORD 0
      .WORD 0
      .WORD DIRBUF      ;DIRECTORY BUFFER
      .WORD DPB1  ;DISK PARAMETER BLOCK
      .WORD CSV1  ;CHECKSUM VECTOR
      .WORD ALV1  ;ALLOCATION VECTOR



      .WORD APB2
      .WORD TRANS2      ;LOGICAL TO PHYSICAL XLATE TAB
      .WORD 0     ;SCRATCH
      .WORD 0
      .WORD 0
      .WORD DIRBUF      ;DIRECTORY BUFFER
      .WORD DPB2  ;DISK PARAMETER BLOCK
      .WORD CSV2  ;CHECKSUM VECTOR
      .WORD ALV2  ;ALLOCATION VECTOR
```

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

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


;*********************************************************
; RAM DISK I/O
;*********************************************************


REDRAM:
      XRA   A
      STA   RAMOP
      JMP   RWRAM

WRTRAM:
      MVI   A,1
      STA   RAMOP
RWRAM:
      LHLD  SEKTRK            ;SET TRACK
      MOV   A,L
      OUT   42H
      LDA   SEKSEC            ;SET SECTOR
      OUT   43H
      XRA   A
      OUT   41H         ;BYTE TO 0

      CALL  SWTUSR

      MVI   B,128
      MVI   C,40H
```

```
      LHLD  DMAADD
      LDA   RAMOP
      ORA   A
      JZ    RRAM
      OUTIR
      JMP   RDONE

RRAM:
      INIR
RDONE:
      CALL  SWTSYS
      XRA   A
      RET



RAMOP:      .BYTE 0

DMAADR:
DMAADD:     .BLKW 1     ;LOGICAL & ACTUAL MP/M DMA ADDR


RDBUFF:
WRBUFF:
HSTBUF:


;;::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;;         SYSTEM INITIALIZATION ROUTINES
;;::::::::::::::::::::::::::::::::::::::::::::::::::::::::

SYSNIT:
      DI
      SHLD  BIOSAD      ; SAVE SYSTEM BIOS ADDR.
      XCHG
      SHLD  DEBUGA      ; SAVE DDT/RDT RST ENTRY ADDR.
      MOV   A,C
      RAL
      RAL
      RAL
      MVI   H,0
      MOV   L,A
      SHLD  DEBUGR      ; SAVE DDT/RDT RST LOC.

;     PLACE SYSTEM INFO INTO MEMORY BANKS

SYSNT2:

      LHLD  SYSDTA
      LXI   D,15  ; OFFSET TO MEMORY CONTROL SEGS.
      DAD   D     ; HL => MAXIMUM SEG. COUNT.
      MOV   C,M   ; TOTAL NUMBER OF CONTROL SEGS. IN TABLE.
      INX   H     ; HL => FIRST CONTROL SEG.
      PUSH  H     ; SAVE ADDRESS OF FIRST CONTROL SEG.
```

```
      PUSH  H
      PUSH  B

;     LXI   D,BANKMS
;     MVI   C,9
;     CALL  XDOS
;     MVI   C,1
;     CALL  XDOS
;     CPI   '1'
;     JZ    PBINI
;     MVI   A,0
;     STA   BSWT
;     JMP   PBINI

;BANKMS:
;     .BYTE 10,13
;     .ASCII      \ENTER 1 IF STATIC RAM, 0 IF DYNAMIC   $\

PBINI:
      POP   B
      POP   H

BINI: PUSH  B     ; SAVE COUNT OF BANKS TO INIT.
      MOV   B,H   ; BC => SELECTED CONTROL SEG.
      MOV   C,L
      PUSH  H     ; SAVE CURRENT TABLE POS.
      CALL  SELMEM      ; SELECT BANK (CONTROL SEG. [BC]).


;:::::::::::::::::::::::::::::::::::::::::::::::::::::
;:  ROUTINE TO PLACE SYSTEM INFO IN CURRENT SELECTED BANK
;:  EACH BANK IS TESTED TO SEE IF IT CAN READ/WRITE O.K.
;:::::::::::::::::::::::::::::::::::::::::::::::::::::

SETUP:
      MVI   A,0C3H
      STA   0     ; JMP INST AT WARMSTART.
      STA   1*8   ; JMP INST AT RST#1.
      LHLD  DEBUGR
      MOV   M,A   ; JMP INST AT DEBUG RST.
      INX   H
      XCHG
      LHLD  DEBUGA
      XCHG
      MOV   M,E   ; JMP ADDR AT DEBUG RST.
      INX   H
      MOV   M,D
      LHLD  BIOSAD
      SHLD  1     ; JMP ADDR AT WARMSTART.
      LXI   H,INT1HD
      SHLD  1*8+1 ; JMP ADDR AT RST#1.
```

```
NXTONE:
      POP   H     ; RECOVER TABLE POS.
      LXI   D,4   ; OFFSET TO NEXT CONTROL SEG.
      DAD   D     ; HL => NEXT CONTROL SEG.
      POP   B     ; RECOVER REMAINING BANK COUNT.
      DCR   C
      JRNZ  BINI  ; INIT ALL BANKS IN SEGMENT TABLE.
      POP   B     ; BC => FIRST CONTROL SEG. (RESELECT).
      CALL  SELMEM      ; REQUEST INITIAL BANK.

DOINT:
      DI          ;AGAIN PLEASE
      IM0         ; FORCE INTERRUPT MODE ZERO.
      JMP   NIT1  ;GET DISK PARAMETER BLOCK


      .LOC  HSTBUF+1024


DIRBUF:     .BLKB 128


;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;        BANKED CACHE DATA AND SWITCH ROUTINE
;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

BADDR:      .BYTE 00,0C0H,00,06H

BREAD:
      DI
      PUSH  H
      PUSH  D
      LXI   B,BADDR
      STA   BADDR+3
      CALL  SELMEM
      POP   D
      POP   H
      LXI   B,128
      LDIR
      CALL  SWTSYS
      JMP   EXIREG


;           MOVE RECORD INTO/OUT OF HOST BUFFER AREA

RWMOV:


      PUSH  D           ;SAVE DESTINATION ADDRESS
      PUSH  H           ;SAVE SOURCE ADDRESS


      .IFN  BANKED,[
```

```
      JZ    WRSWP       ;DO A SWAP FOR WRITE
      POP   H           ;RECOVER SOURCE ADDRESS
      LXI   D,SWPBUF    ;POINT TO SWAP BUFFER
      LXI   B,128       ;128 BYTES TO MOVE
      LDIR              ;MOVE THEM

      CALL  SWTUSR            ;GET USER BANK IN


      POP   D           ;GET DESTINATION ADDRESS
      LXI   H,SWPBUF    ;DATA FROM SWAP BUFFER
      LXI   B,128
      LDIR

      CALL  SWTSYS
      RET


WRSWP:
      CALL  SWTUSR            ;GET USER BANK IN

      POP   H           ;GET SOURCE ADDRESS INTO HL
      LXI   D,SWPBUF    ;DESTINATION IN SWAP BUFFER
      LXI   B,128
      LDIR

      CALL  SWTSYS

      POP   D           ;GET DESTINATION ADDRESS
      LXI   H,SWPBUF
      LXI   B,128
      LDIR

      RET

            ][

      CALL  SWTUSR

      POP   H
      POP   D
      LXI   B,128
      LDIR
      CALL  SWTSYS
      RET

      ]

CHDPOL:
      MVI   C,POLL
      MVI   E,PLBUS
      CALL  XDOS
      RET
```

```
;*******************************************************
;*                   UNITXIOS.ASM                      *
;*    MP/M VERSION 1.1 & II UNIT RECORD XIOS           *
;*              BRUCE JONES                 *
;*                   11/02/83                          *
;*******************************************************

;*******************************************************
;*            SYSTEM SWITCHED HARDWARE
;*******************************************************

; MEMORY BANK CONTROL.

MULTIB      ==    0FFH  ; BASE ADDRESS PORT FOR START OF
                  ; MULITIPLE BANK MEMORY BOARDS.

; INTERRUPT CONTROL.

VIRTC ==    0DH   ; INTERRUPT START PORT.
INTC  ==    0CH   ; INTERRUPT CLEAR & STOP PORT.


;*******************************************************
;*       MP/M XDOS FUNCTION DEFINITIONS.
;*******************************************************

POLL  ==    131   ; XDOS POLL REQ FUNCTION.
FLAGST      ==    133   ; XDOS FLAG SET FUNCTION.
DSPTCH      ==    142   ; XDOS DISPATCH FUNCTION.
SYSDAT      ==    154   ; XDOS SYS-ADDR FUNCTION.
GETCON      ==    99H


;*******************************************************
;*          MP/M ROUTINE DEVICE ASSIGNMENTS.
;*******************************************************


APOL  ==    0     ;FIRST POLL DEVICE ASSIGNMENT

;     CONSOLE I/O PORTS ARE FIRST

PLCO0 ==    APOL +0           ; POLL CONSOLE OUT #0.
PLCO1 ==    APOL +1           ;        "         #1.
PLCO2 ==    APOL +2           ;        "         #2.
PLCO3 ==    APOL +3           ;        "         #3.
PLCO4 ==    APOL +4           ;        "         #4.
PLCO5 ==    APOL +5           ;        "         #5.
PLCO6 ==    APOL +6           ;        "         #6
PLCO7 ==    APOL +7           ;        "         #7.
```

```
PLCI0 ==    APOL +8          ; POLL CONSOLE IN  #0.
PLCI1 ==    APOL +9          ;        "         #1.
PLCI2 ==    APOL +10    ;        "         #2.
PLCI3 ==    APOL +11    ;        "         #3.
PLCI4 ==    APOL +12    ;        "         #4.
PLCI5 ==    APOL +13    ;        "         #5.
PLCI6 ==    APOL +14    ;        "         #6.
PLCI7 ==    APOL +15    ;        "         #7.


;     DISK DRIVE PORTS NEXT

PLDSK ==    APOL +16    ;POLL FLOPPY DISK
PLDMA ==    APOL +17    ;POLL FLOPPY DMA DONE

;HARD DISK POLL VALUES FOR MP/M POLL DEVICE ROUTINE


PLBUS ==    APOL +18    ;POLL FOR DISK BUSY
FINDSK      ==    PLBUS       ;LAST POLL VALUE FOR DISK
PLBUF ==    30


;     PRINTER POLL VALUES


PRN0  ==    FINDSK+1    ; POLL PRINTER    #0
PRN1  ==    PRN0+1           ; POLL PRINTER  #1
PRN2  ==    PRN0+2           ; POLL PRINTER  #2
PRN3  ==    PRN0+3      ; POLL PRINTER  #3
PRN4  ==    PRN0+4      ; POLL PRINTER  #4


POLDEV:
      MOV   A,C
      CPI   PLBUS
      JZ    SCSIPL
      CPI   PLBUF
      JZ    IOPOL

      .IFN  CONLST,[

      CPI   20
      JRC   POLING
      CPI   22
      JRNC  POLING
      CALL  POLING
      CMA
      RET
            ][
      CPI   19
      JRC   POLING
      CPI   21
```

```
      JRNC  POLING
      CALL  POLING
      CMA
      RET
            ]

POLLPT:
      MOV   A,D
      ADI   19

      .IFN  CONLST,[

      CPI   20
      JRC   POLING
      CPI   22
      JRNC  POLING
      CALL  POLING
      CMA
      RET
      ][
      CPI   21
      JRNC  POLING
      CALL  POLING
      CMA
      RET
      ]

CONST:
      MOV   A,D         ;GET CONSOLE #
      CPI   7
      JNZ   NOT7
      XRA   A
      MVI   C,0FFH
      RET

NOT7:
      MOV   B,A         ;SAVE IN B
      ADI   8           ;SET FOR INDEX TABLE ENTRY

POLING:
      MVI   D,0         ;CLEAR FOR INDEX
      ADD   A           ;*2
      ADD   A           ;*4
      MOV   E,A         ;MAKE DE INDEX VALUE
      LXI   H,POLTAB    ;POINT HL TO CONSOLE INPUT TABLE
      DAD   D           ;POINT TO CORRECT DEVICE #
      MOV   E,M         ;SAVE IN E FOR FUTURE
      INX   H           ;POINT TO STAT PORT VALUE
      MOV   C,M         ;PUT PORT VALUE IN C
      MOV   A,M         ;GET IT TO ACC
      CPI   0FFH        ;SEE IF DUMMY
      MVI   A,0
      RZ
      INX   H           ;POINT TO READY MASK
```

```
      INP   A           ;GET I/O FROM STATUS PORT
      ANA   M           ;TEST WITH READY MASK
      RZ                ;RETURN HERE BUSY
      MVI   A,0FFH            ;ELSE SET READY
      RET


;     DEVICE INPUT

CONIN:
      CALL  CONST


;     C HAS STATUS PORT, H POINTS TO READY MASK, A = 00 BUSY

      INX   H           ;POINT TO I/O PORT
      MOV   C,M         ;PUT IN C
      ORA   A           ;SEE IF BUSY
      JNZ   ..RDY

      PUSH  B           ;SAVE PORT AND CONSOLE #
      MVI   C,POLL
      CALL  XDOS
      POP   B           ;RECOVER PORT AND CONSOLE #

..RDY:
      INP   A
      ANI   7FH
      CPI   1EH
      RNZ

      MVI   C,147
      CALL  XDOS

      MVI   A,0
      RET




;           PRINTER OUTPUT

LIST:
      MOV   A,D
      ADI   19

      .IFN  CONLST,[
      CPI   19
      JRZ   ENTOUT
      CPI   22
      JRNC  ENTOUT
      MOV   B,C
      CALL  POLING
      CMA
```

```
      JMPR  ENTOU2
            ][
      CPI   21
      JRNC  ENTOUT
      MOV   B,C
      CALL  POLING
      CMA
      JMPR  ENTOU2
            ]

;                 CONSOLE OUTPUT

CONOUT:
      MOV   A,D
      CPI   7
      RZ

ENTOUT:
      MOV   B,C         ;SAVE OUT BYTE
      CALL  POLING

;     E HAS DEVICE #, H POINTS MASK VALUE, A = 00 BUSY

ENTOU2:
      INX   H           ;POINT I/O PORT
      MOV   C,M         ;INTO C
      ORA   A           ;TEST BUSY
      JRNZ  ..RDY       ;IF READY GO OUTPUT

      PUSH  B           ;ELSE SAVE PORT & DATA
      MVI   C,POLL            ;SET UP XDOS POLL
      CALL  XDOS        ;DO POLL
      POP   B           ;BACK FROM POLL

..RDY:
      OUTP  B           ;XMIT BYTE


      RET               ;DONE


;;::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;;             ALL DEVICE I/O TABLES NEXT
;;::::::::::::::::::::::::::::::::::::::::::::::::::::::::

POLTAB:

;     CONSOLE OUTPUT SEGMENT
;     DEV # / STAT PORT / MASK VALUE / OUT PORT


      .BYTE 0,00H,04H,01H

      .IFE  MODCON,[
```

```
        .BYTE 1,02H,80H,03H
              ][
        .BYTE 1,04H,80H,05H
              ]


        .IFN  MODCON,[


        .IFE  CONLST,[

        .BYTE 2,02H,80H,03H           ;IF CONSOLE #2 FOR THIS PORT
              ][
        .BYTE 2,10H,80H,11H           ;ELSE USE THIS PORT
              ]

              ][
        .IFE  CONLST,[

        .BYTE 2,0FFH,0FFH,0FFH        ;IF CONSOLE #2 FOR THIS PORT
              ][
        .BYTE 2,10H,80H,11H           ;ELSE USE THIS PORT
              ]

              ]


        .BYTE 3,12H,80H,13H
        .BYTE 4,14H,80H,15H
        .BYTE 5,16H,80H,17H
        .BYTE 6,1EH,80H,1FH
        .BYTE 7,10H,80H,11H


;       CONSOLE INPUT SEGMENT   DEV #/ STAT PORT / MASK VALUE / IN PORT


        .BYTE 8,00H,02H,01H

        .IFE  MODCON,[

        .BYTE 9,02H,01H,03H
              ][
        .BYTE 9,04H,01H,05H
              ]


        .IFN  MODCON,[


        .IFE  CONLST,[

        .BYTE 10,02H,01H,03H          ;IF CONSOLE # 2 THIS PORT
```

```
            ][
      .BYTE 10,10H,01H,11H          ;ELSE USE THIS PORT
            ]
            ][

      .IFE  CONLST,[

      .BYTE 10,0FFH,0FFH,0FFH       ;IF CONSOLE # 2 THIS PORT
            ][
      .BYTE 10,10H,01H,11H          ;ELSE USE THIS PORT
            ]
            ]


      .BYTE 11,12H,01H,13H
      .BYTE 12,14H,01H,15H
      .BYTE 13,16H,01H,17H
      .BYTE 14,1EH,01H,1FH
      .BYTE 15,10H,01H,11H


;           DISK DRIVE TABLE NEXT
;           DEV#/ STAT PORT/ MASK VALUE/ FILLER

;           FLOPPY DISK

      .BYTE 16,78H,01H,00H
      .BYTE 17,78H,01H,00H

;           WINCHESTER

      .BYTE 18,DIO+2,80H,00H

;           PRINTER PORTS NEXT
;           DEV #/ STAT PORT/ MASK VALUE/ OUT PORT


      .IFN  CONLST,[

      .BYTE 19,02H,80H,03H
      .BYTE 20,06H,01H,07H
      .BYTE 21,06H,80H,06H
      .BYTE 22,22H,80H,23H
;     .BYTE 23,24H,80H,25H

            ][

      .BYTE 19,06H,01H,07H
      .BYTE 20,06H,80H,06H
      .BYTE 21,22H,80H,23H
      .BYTE 22,24H,80H,25H

            ]
```

```
POLLEN      ==    .-POLTAB

NMBDEV      ==    POLLEN/4




BUFPOL:
      PUSH  H
      PUSH  D
      PUSH  B
      MVI   C,POLL
      MVI   E,PLBUF
      CALL  XDOS
      POP   B
      POP   D
      POP   H
      RET



SCSIPL:
      IN    DIO+2       ;GET SCSI STATUS
      XRI   0FFH
      ANI   50H
      MVI   A,0FFH
      RNZ
      XRA   A
      RET

IOPOL:
      IN    DIO+2
      ORA   A
      MVI   A,0FFH
      RM
      XRA   A
      RET




;*****************************************************
;*             MP/M MEMORY HANDLER
;*****************************************************

; REG BC = ADDR OF MEM DESCRIPTOR.
; BC ->     BASE  1 BYTE.
;     SIZE  1 BYTE.
;     ATTRIB      1 BYTE.
```

```
;     BANK  1 BYTE.

SELMEM:
RETIT:
      NOP

      LXI   H,3         ; TABLE SEGMENT => BANK.
      DAD   B           ; POINT TO BANK BYTE
      MOV   A,M         ; BANK NUMBER INTO B.

;BSWT:      INR   A

      OUT   MULTIB            ; SWITCH BANK IN
      RET               ; RETURN FROM BANK SELECT.


;*********************************************************
;*          SYSTEM CLOCK ENABLE ROUTINE
;*********************************************************

; WILL CAUSE FLAG #1 TO BE SET AT EACH SYSTEM TIME UNIT TICK.

STRCLK:
      MVI   A,0FFH
      STA   TICKN
      RET


;*********************************************************
;*        SYSTEM CLOCK DISABLE ROUTINE
;*********************************************************

; WILL STOP FLAG #1 SETTING AT SYSTEM TIME UNIT TICK.

STPCLK:
      XRA   A
      STA   TICKN
      RET


;*****************************************************
;*         SYSTEM REGIONAL EXIT ENABLE ROUTINE
;*****************************************************

EXIREG:
      LDA   PREEMP            ;EI IF NOT PRE-EMPTED
      ORA   A
      RNZ
      EI
      RET


;*********************************************************
;*        IDENTIFY MAXIMUM CONSOLE ABILITY
```

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

MAXCON:
      MVI   A,16        ;GET MAX NUMBER
      RET


;*********************************************************
;                    RETURN EMPTY
;*********************************************************

RNTEM:
      XRA   A
      RET


;*********************************************************
;*         MP/M INTERRUPT ONE HANDLER
;*********************************************************

INT1HD:
      PUSH  PSW
      LDA   SLICE
      DCR   A
      STA   SLICE
      JZ    STIME
      IN    4
      ANI   01H
      JNZ   STIME
      POP   PSW
      OUT   INTC
      OUT   VIRTC
      EI
      RET

STIME:
      MVI   A,4
      STA   SLICE

      POP   PSW
      SHLD  HLSAV
      POP   H           ;PULL USER RETURN
      SHLD  RTSAV       ;SAVE IT
      PUSH  PSW         ;RESET STACK POINTER
      SSPD  SPSAV
      LXI   SP,INTSTK
      PUSH  D
      PUSH  B

      MVI   A,0FFH            ;PREEMPT SET
      STA   PREEMP

      OUT   INTC        ;RESET INT. F.F.
      OUT   VIRTC       ;RE-ENABLE F.F.
```

```
      LDA   TICKN       ;GET SYSTEM TICK
      ORA   A
      JRZ   NOTICK            ;IF NOT TICK DON'T SET IT
      MVI   C,FLAGST
      MVI   E,1
      CALL  XDOS        ;GO SET TICK FLAG

NOTICK:
      LXI   H,SYSTIC          ;GET 1/300 SECOND
      DCR   M
      JNZ   NOTSEC
      MVI   M,60

      MVI   C,FLAGST    ;AND DO XDOS CALL
      MVI   E,2         ;TO SET SECOND FLAG
      CALL  XDOS

NOTSEC:
      XRA   A           ;CLEAR PSW
      STA   PREEMP            ;CLEAR PREEMPT

      POP   B
      POP   D
      LSPD  SPSAV
      POP   PSW
      LHLD  RTSAV
      PUSH  H
      LHLD  HLSAV


;     FORCE ROUND ROBIN SCHEDUING NEXT BY GOING TO DISPATCHER


      JMP   PDISP       ;JUMP TO DISPATCHER


;*******************************************************
;*        MP/M IDLE PROCESS HANDLER ROUTINE
;*******************************************************

IDLE:
      MVI   C,DSPTCH    ; DISPATCH CALL
      JMP   XDOS


DPBASE:
      .WORD 0,0,0,0,DIRBUF,DDB0,HDCSV,HDALL0
      .WORD 0,0,0,0,DIRBUF,DDB1,HDCSV,HDALL1
      .WORD 0,0,0,0,DIRBUF,DDB2,HDCSV,HDALL2
      .WORD 0,0,0,0,DIRBUF,DDB3,HDCSV,HDALL3
RAMDPH:     .WORD 0,0,0,0,DIRBUF,RAMDPB,HDCSV,HDALL4
```

```
DBF0  ==    DPBASE+8
DBF1  ==    DBF0+16
DBF2  ==    DBF1+16
DBF3  ==    DBF2+16


;           DDBS ETC READ INTO HERE

HDSPT:                      ;HARD DISK SECTORS PER TRACK

DDB0: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
DDB1: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
DDB2: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
DDB3: .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
DDBLEN      ==    .-DDB0

RAMDPB:     .WORD 16
      .BYTE 4,15,1
      .WORD 250,255
      .BYTE 11110000B,00000000B
      .WORD 0,0




;*********************************************************
;*      XIOS INFORMATION STORAGE AREA.
;*********************************************************

BIOSAD:     .WORD 0           ;BIOS ADDRESS
DEBUGA:     .WORD 0           ;
DEBUGR:     .WORD 0           ;DE-BUG ADDRESS

SYSTIC:     .BYTE 150         ;SYSTEM CLOCK LOCATION
SLICE:      .BYTE 4

SPSAV:      .WORD 0           ;STACK SAVE
RTSAV:      .WORD 0           ;RETURN ADDRESS SAVE
HLSAV:      .WORD 0

TICKN:      .BYTE 0           ;SYSTEM TICK FLAG
PREEMP:     .BYTE 0           ;PRE-EMPT FLAG
LBANK:      .BYTE 0           ;LAST BANK SELECTED
      .BYTE '*'


      .BLKB 32

INTSTK:
ENDIOS:
      .BYTE '*'
```

```
ENDPAG      ==    (ENDIOS+256)&0FF00H


;XSIZEXSIZE                   SHOW SIZE OF XIOS HERE


.PRNTX      \         \


      .IF2,[


;SHOW TOTAL RAM USED


      .DEFINE     SIZMEM[XX]=[
      .PRNTX      /XIOS CONSUMES XX BYTES/]
      SIZMEM \.-(START)


;SHOW COMMON AREA USED

      .DEFINE     COMMEM[XX]=[
      .PRNTX      /COMMON AREA IS XX BYTES LONG/]
      COMMEM      \.-(CLDST)



;SHOW USER TPA

      .DEFINE     COMLOC[X]=[
      .PRNTX      /USER BANK SIZE IS  X  BYTES/]

      COMLOC      \(206-((.-CLDST)/256))*256



      .RADIX      16

      .DEFINE     CMLOC[XX]=[
      .PRNTX      /COMMON BASE STARTS AT XX/]

      CMLOC \((0CE00H)-(.-CLDST))


      .DEFINE     SWTPT[X]=[
      .PRNTX      /MEMORY HARDWARE SWITCHED AT PAGE  X/]


      SWTPT \(((0CE00H)-(.-CLDST))/100H)
```

```
.DEFINE     COMSIZ[XX]=[
.PRNTX      /START TO COMMONBASE IS XX/]

COMSIZ      \CLDST

.RADIX      10

.DEFINE FREMEM[XX]=[
.PRNTX      /MEMORY FREE IN XIOS IS XX/]

FREMEM      \ENDPAG-ENDIOS

]

.END
```